| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Logger.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/util/logging/Level.html)   [**NEXT CLASS**](http://docs.google.com/java/util/logging/LoggingMXBean.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/logging/Logger.html)    [**NO FRAMES**](http://docs.google.com/Logger.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#17dp8vu) | [METHOD](#26in1rg) |

## **java.util.logging**

Class Logger

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **java.util.logging.Logger**

public class **Logger**extends [Object](http://docs.google.com/java/lang/Object.html)

A Logger object is used to log messages for a specific system or application component. Loggers are normally named, using a hierarchical dot-separated namespace. Logger names can be arbitrary strings, but they should normally be based on the package name or class name of the logged component, such as java.net or javax.swing. In addition it is possible to create "anonymous" Loggers that are not stored in the Logger namespace.

Logger objects may be obtained by calls on one of the getLogger factory methods. These will either create a new Logger or return a suitable existing Logger.

Logging messages will be forwarded to registered Handler objects, which can forward the messages to a variety of destinations, including consoles, files, OS logs, etc.

Each Logger keeps track of a "parent" Logger, which is its nearest existing ancestor in the Logger namespace.

Each Logger has a "Level" associated with it. This reflects a minimum Level that this logger cares about. If a Logger's level is set to null, then its effective level is inherited from its parent, which may in turn obtain it recursively from its parent, and so on up the tree.

The log level can be configured based on the properties from the logging configuration file, as described in the description of the LogManager class. However it may also be dynamically changed by calls on the Logger.setLevel method. If a logger's level is changed the change may also affect child loggers, since any child logger that has null as its level will inherit its effective level from its parent.

On each logging call the Logger initially performs a cheap check of the request level (e.g. SEVERE or FINE) against the effective log level of the logger. If the request level is lower than the log level, the logging call returns immediately.

After passing this initial (cheap) test, the Logger will allocate a LogRecord to describe the logging message. It will then call a Filter (if present) to do a more detailed check on whether the record should be published. If that passes it will then publish the LogRecord to its output Handlers. By default, loggers also publish to their parent's Handlers, recursively up the tree.

Each Logger may have a ResourceBundle name associated with it. The named bundle will be used for localizing logging messages. If a Logger does not have its own ResourceBundle name, then it will inherit the ResourceBundle name from its parent, recursively up the tree.

Most of the logger output methods take a "msg" argument. This msg argument may be either a raw value or a localization key. During formatting, if the logger has (or inherits) a localization ResourceBundle and if the ResourceBundle has a mapping for the msg string, then the msg string is replaced by the localized value. Otherwise the original msg string is used. Typically, formatters use java.text.MessageFormat style formatting to format parameters, so for example a format string "{0} {1}" would format two parameters as strings.

When mapping ResourceBundle names to ResourceBundles, the Logger will first try to use the Thread's ContextClassLoader. If that is null it will try the SystemClassLoader instead. As a temporary transition feature in the initial implementation, if the Logger is unable to locate a ResourceBundle from the ContextClassLoader or SystemClassLoader the Logger will also search up the class stack and use successive calling ClassLoaders to try to locate a ResourceBundle. (This call stack search is to allow containers to transition to using ContextClassLoaders and is likely to be removed in future versions.)

Formatting (including localization) is the responsibility of the output Handler, which will typically call a Formatter.

Note that formatting need not occur synchronously. It may be delayed until a LogRecord is actually written to an external sink.

The logging methods are grouped in five main categories:

* There are a set of "log" methods that take a log level, a message string, and optionally some parameters to the message string.
* There are a set of "logp" methods (for "log precise") that are like the "log" methods, but also take an explicit source class name and method name.
* There are a set of "logrb" method (for "log with resource bundle") that are like the "logp" method, but also take an explicit resource bundle name for use in localizing the log message.
* There are convenience methods for tracing method entries (the "entering" methods), method returns (the "exiting" methods) and throwing exceptions (the "throwing" methods).
* Finally, there are a set of convenience methods for use in the very simplest cases, when a developer simply wants to log a simple string at a given log level. These methods are named after the standard Level names ("severe", "warning", "info", etc.) and take a single argument, a message string.

For the methods that do not take an explicit source name and method name, the Logging framework will make a "best effort" to determine which class and method called into the logging method. However, it is important to realize that this automatically inferred information may only be approximate (or may even be quite wrong!). Virtual machines are allowed to do extensive optimizations when JITing and may entirely remove stack frames, making it impossible to reliably locate the calling class and method.

All methods on Logger are multi-thread safe.

**Subclassing Information:** Note that a LogManager class may provide its own implementation of named Loggers for any point in the namespace. Therefore, any subclasses of Logger (unless they are implemented in conjunction with a new LogManager class) should take care to obtain a Logger instance from the LogManager class and should delegate operations such as "isLoggable" and "log(LogRecord)" to that instance. Note that in order to intercept all logging output, subclasses need only override the log(LogRecord) method. All the other logging methods are implemented as calls on this log(LogRecord) method.

**Since:** 1.4

| **Field Summary** | |
| --- | --- |
| static [Logger](http://docs.google.com/java/util/logging/Logger.html) | [**global**](http://docs.google.com/java/util/logging/Logger.html#global)  **Deprecated.** *Initialization of this field is prone to deadlocks. The field must be initialized by the Logger class initialization which may cause deadlocks with the LogManager class initialization. In such cases two class initialization wait for each other to complete. As of JDK version 1.6, the preferred way to get the global logger object is via the call Logger.getLogger(Logger.GLOBAL\_LOGGER\_NAME).* |
| static [String](http://docs.google.com/java/lang/String.html) | [**GLOBAL\_LOGGER\_NAME**](http://docs.google.com/java/util/logging/Logger.html#GLOBAL_LOGGER_NAME)            GLOBAL\_LOGGER\_NAME is a name for the global logger. |

| **Constructor Summary** | |
| --- | --- |
| protected | [**Logger**](http://docs.google.com/java/util/logging/Logger.html#Logger(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) name, [String](http://docs.google.com/java/lang/String.html) resourceBundleName)            Protected method to construct a logger for a named subsystem. |

| **Method Summary** | |
| --- | --- |
| void | [**addHandler**](http://docs.google.com/java/util/logging/Logger.html#addHandler(java.util.logging.Handler))([Handler](http://docs.google.com/java/util/logging/Handler.html) handler)            Add a log Handler to receive logging messages. |
| void | [**config**](http://docs.google.com/java/util/logging/Logger.html#config(java.lang.String))([String](http://docs.google.com/java/lang/String.html) msg)            Log a CONFIG message. |
| void | [**entering**](http://docs.google.com/java/util/logging/Logger.html#entering(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) sourceClass, [String](http://docs.google.com/java/lang/String.html) sourceMethod)            Log a method entry. |
| void | [**entering**](http://docs.google.com/java/util/logging/Logger.html#entering(java.lang.String,%20java.lang.String,%20java.lang.Object))([String](http://docs.google.com/java/lang/String.html) sourceClass, [String](http://docs.google.com/java/lang/String.html) sourceMethod, [Object](http://docs.google.com/java/lang/Object.html) param1)            Log a method entry, with one parameter. |
| void | [**entering**](http://docs.google.com/java/util/logging/Logger.html#entering(java.lang.String,%20java.lang.String,%20java.lang.Object%5B%5D))([String](http://docs.google.com/java/lang/String.html) sourceClass, [String](http://docs.google.com/java/lang/String.html) sourceMethod, [Object](http://docs.google.com/java/lang/Object.html)[] params)            Log a method entry, with an array of parameters. |
| void | [**exiting**](http://docs.google.com/java/util/logging/Logger.html#exiting(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) sourceClass, [String](http://docs.google.com/java/lang/String.html) sourceMethod)            Log a method return. |
| void | [**exiting**](http://docs.google.com/java/util/logging/Logger.html#exiting(java.lang.String,%20java.lang.String,%20java.lang.Object))([String](http://docs.google.com/java/lang/String.html) sourceClass, [String](http://docs.google.com/java/lang/String.html) sourceMethod, [Object](http://docs.google.com/java/lang/Object.html) result)            Log a method return, with result object. |
| void | [**fine**](http://docs.google.com/java/util/logging/Logger.html#fine(java.lang.String))([String](http://docs.google.com/java/lang/String.html) msg)            Log a FINE message. |
| void | [**finer**](http://docs.google.com/java/util/logging/Logger.html#finer(java.lang.String))([String](http://docs.google.com/java/lang/String.html) msg)            Log a FINER message. |
| void | [**finest**](http://docs.google.com/java/util/logging/Logger.html#finest(java.lang.String))([String](http://docs.google.com/java/lang/String.html) msg)            Log a FINEST message. |
| static [Logger](http://docs.google.com/java/util/logging/Logger.html) | [**getAnonymousLogger**](http://docs.google.com/java/util/logging/Logger.html#getAnonymousLogger())()            Create an anonymous Logger. |
| static [Logger](http://docs.google.com/java/util/logging/Logger.html) | [**getAnonymousLogger**](http://docs.google.com/java/util/logging/Logger.html#getAnonymousLogger(java.lang.String))([String](http://docs.google.com/java/lang/String.html) resourceBundleName)            Create an anonymous Logger. |
| [Filter](http://docs.google.com/java/util/logging/Filter.html) | [**getFilter**](http://docs.google.com/java/util/logging/Logger.html#getFilter())()            Get the current filter for this Logger. |
| [Handler](http://docs.google.com/java/util/logging/Handler.html)[] | [**getHandlers**](http://docs.google.com/java/util/logging/Logger.html#getHandlers())()            Get the Handlers associated with this logger. |
| [Level](http://docs.google.com/java/util/logging/Level.html) | [**getLevel**](http://docs.google.com/java/util/logging/Logger.html#getLevel())()            Get the log Level that has been specified for this Logger. |
| static [Logger](http://docs.google.com/java/util/logging/Logger.html) | [**getLogger**](http://docs.google.com/java/util/logging/Logger.html#getLogger(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Find or create a logger for a named subsystem. |
| static [Logger](http://docs.google.com/java/util/logging/Logger.html) | [**getLogger**](http://docs.google.com/java/util/logging/Logger.html#getLogger(java.lang.String,%20java.lang.String))([String](http://docs.google.com/java/lang/String.html) name, [String](http://docs.google.com/java/lang/String.html) resourceBundleName)            Find or create a logger for a named subsystem. |
| [String](http://docs.google.com/java/lang/String.html) | [**getName**](http://docs.google.com/java/util/logging/Logger.html#getName())()            Get the name for this logger. |
| [Logger](http://docs.google.com/java/util/logging/Logger.html) | [**getParent**](http://docs.google.com/java/util/logging/Logger.html#getParent())()            Return the parent for this Logger. |
| [ResourceBundle](http://docs.google.com/java/util/ResourceBundle.html) | [**getResourceBundle**](http://docs.google.com/java/util/logging/Logger.html#getResourceBundle())()            Retrieve the localization resource bundle for this logger for the current default locale. |
| [String](http://docs.google.com/java/lang/String.html) | [**getResourceBundleName**](http://docs.google.com/java/util/logging/Logger.html#getResourceBundleName())()            Retrieve the localization resource bundle name for this logger. |
| boolean | [**getUseParentHandlers**](http://docs.google.com/java/util/logging/Logger.html#getUseParentHandlers())()            Discover whether or not this logger is sending its output to its parent logger. |
| void | [**info**](http://docs.google.com/java/util/logging/Logger.html#info(java.lang.String))([String](http://docs.google.com/java/lang/String.html) msg)            Log an INFO message. |
| boolean | [**isLoggable**](http://docs.google.com/java/util/logging/Logger.html#isLoggable(java.util.logging.Level))([Level](http://docs.google.com/java/util/logging/Level.html) level)            Check if a message of the given level would actually be logged by this logger. |
| void | [**log**](http://docs.google.com/java/util/logging/Logger.html#log(java.util.logging.Level,%20java.lang.String))([Level](http://docs.google.com/java/util/logging/Level.html) level, [String](http://docs.google.com/java/lang/String.html) msg)            Log a message, with no arguments. |
| void | [**log**](http://docs.google.com/java/util/logging/Logger.html#log(java.util.logging.Level,%20java.lang.String,%20java.lang.Object))([Level](http://docs.google.com/java/util/logging/Level.html) level, [String](http://docs.google.com/java/lang/String.html) msg, [Object](http://docs.google.com/java/lang/Object.html) param1)            Log a message, with one object parameter. |
| void | [**log**](http://docs.google.com/java/util/logging/Logger.html#log(java.util.logging.Level,%20java.lang.String,%20java.lang.Object%5B%5D))([Level](http://docs.google.com/java/util/logging/Level.html) level, [String](http://docs.google.com/java/lang/String.html) msg, [Object](http://docs.google.com/java/lang/Object.html)[] params)            Log a message, with an array of object arguments. |
| void | [**log**](http://docs.google.com/java/util/logging/Logger.html#log(java.util.logging.Level,%20java.lang.String,%20java.lang.Throwable))([Level](http://docs.google.com/java/util/logging/Level.html) level, [String](http://docs.google.com/java/lang/String.html) msg, [Throwable](http://docs.google.com/java/lang/Throwable.html) thrown)            Log a message, with associated Throwable information. |
| void | [**log**](http://docs.google.com/java/util/logging/Logger.html#log(java.util.logging.LogRecord))([LogRecord](http://docs.google.com/java/util/logging/LogRecord.html) record)            Log a LogRecord. |
| void | [**logp**](http://docs.google.com/java/util/logging/Logger.html#logp(java.util.logging.Level,%20java.lang.String,%20java.lang.String,%20java.lang.String))([Level](http://docs.google.com/java/util/logging/Level.html) level, [String](http://docs.google.com/java/lang/String.html) sourceClass, [String](http://docs.google.com/java/lang/String.html) sourceMethod, [String](http://docs.google.com/java/lang/String.html) msg)            Log a message, specifying source class and method, with no arguments. |
| void | [**logp**](http://docs.google.com/java/util/logging/Logger.html#logp(java.util.logging.Level,%20java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.Object))([Level](http://docs.google.com/java/util/logging/Level.html) level, [String](http://docs.google.com/java/lang/String.html) sourceClass, [String](http://docs.google.com/java/lang/String.html) sourceMethod, [String](http://docs.google.com/java/lang/String.html) msg, [Object](http://docs.google.com/java/lang/Object.html) param1)            Log a message, specifying source class and method, with a single object parameter to the log message. |
| void | [**logp**](http://docs.google.com/java/util/logging/Logger.html#logp(java.util.logging.Level,%20java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.Object%5B%5D))([Level](http://docs.google.com/java/util/logging/Level.html) level, [String](http://docs.google.com/java/lang/String.html) sourceClass, [String](http://docs.google.com/java/lang/String.html) sourceMethod, [String](http://docs.google.com/java/lang/String.html) msg, [Object](http://docs.google.com/java/lang/Object.html)[] params)            Log a message, specifying source class and method, with an array of object arguments. |
| void | [**logp**](http://docs.google.com/java/util/logging/Logger.html#logp(java.util.logging.Level,%20java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.Throwable))([Level](http://docs.google.com/java/util/logging/Level.html) level, [String](http://docs.google.com/java/lang/String.html) sourceClass, [String](http://docs.google.com/java/lang/String.html) sourceMethod, [String](http://docs.google.com/java/lang/String.html) msg, [Throwable](http://docs.google.com/java/lang/Throwable.html) thrown)            Log a message, specifying source class and method, with associated Throwable information. |
| void | [**logrb**](http://docs.google.com/java/util/logging/Logger.html#logrb(java.util.logging.Level,%20java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.String))([Level](http://docs.google.com/java/util/logging/Level.html) level, [String](http://docs.google.com/java/lang/String.html) sourceClass, [String](http://docs.google.com/java/lang/String.html) sourceMethod, [String](http://docs.google.com/java/lang/String.html) bundleName, [String](http://docs.google.com/java/lang/String.html) msg)            Log a message, specifying source class, method, and resource bundle name with no arguments. |
| void | [**logrb**](http://docs.google.com/java/util/logging/Logger.html#logrb(java.util.logging.Level,%20java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.Object))([Level](http://docs.google.com/java/util/logging/Level.html) level, [String](http://docs.google.com/java/lang/String.html) sourceClass, [String](http://docs.google.com/java/lang/String.html) sourceMethod, [String](http://docs.google.com/java/lang/String.html) bundleName, [String](http://docs.google.com/java/lang/String.html) msg, [Object](http://docs.google.com/java/lang/Object.html) param1)            Log a message, specifying source class, method, and resource bundle name, with a single object parameter to the log message. |
| void | [**logrb**](http://docs.google.com/java/util/logging/Logger.html#logrb(java.util.logging.Level,%20java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.Object%5B%5D))([Level](http://docs.google.com/java/util/logging/Level.html) level, [String](http://docs.google.com/java/lang/String.html) sourceClass, [String](http://docs.google.com/java/lang/String.html) sourceMethod, [String](http://docs.google.com/java/lang/String.html) bundleName, [String](http://docs.google.com/java/lang/String.html) msg, [Object](http://docs.google.com/java/lang/Object.html)[] params)            Log a message, specifying source class, method, and resource bundle name, with an array of object arguments. |
| void | [**logrb**](http://docs.google.com/java/util/logging/Logger.html#logrb(java.util.logging.Level,%20java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.String,%20java.lang.Throwable))([Level](http://docs.google.com/java/util/logging/Level.html) level, [String](http://docs.google.com/java/lang/String.html) sourceClass, [String](http://docs.google.com/java/lang/String.html) sourceMethod, [String](http://docs.google.com/java/lang/String.html) bundleName, [String](http://docs.google.com/java/lang/String.html) msg, [Throwable](http://docs.google.com/java/lang/Throwable.html) thrown)            Log a message, specifying source class, method, and resource bundle name, with associated Throwable information. |
| void | [**removeHandler**](http://docs.google.com/java/util/logging/Logger.html#removeHandler(java.util.logging.Handler))([Handler](http://docs.google.com/java/util/logging/Handler.html) handler)            Remove a log Handler. |
| void | [**setFilter**](http://docs.google.com/java/util/logging/Logger.html#setFilter(java.util.logging.Filter))([Filter](http://docs.google.com/java/util/logging/Filter.html) newFilter)            Set a filter to control output on this Logger. |
| void | [**setLevel**](http://docs.google.com/java/util/logging/Logger.html#setLevel(java.util.logging.Level))([Level](http://docs.google.com/java/util/logging/Level.html) newLevel)            Set the log level specifying which message levels will be logged by this logger. |
| void | [**setParent**](http://docs.google.com/java/util/logging/Logger.html#setParent(java.util.logging.Logger))([Logger](http://docs.google.com/java/util/logging/Logger.html) parent)            Set the parent for this Logger. |
| void | [**setUseParentHandlers**](http://docs.google.com/java/util/logging/Logger.html#setUseParentHandlers(boolean))(boolean useParentHandlers)            Specify whether or not this logger should send its output to it's parent Logger. |
| void | [**severe**](http://docs.google.com/java/util/logging/Logger.html#severe(java.lang.String))([String](http://docs.google.com/java/lang/String.html) msg)            Log a SEVERE message. |
| void | [**throwing**](http://docs.google.com/java/util/logging/Logger.html#throwing(java.lang.String,%20java.lang.String,%20java.lang.Throwable))([String](http://docs.google.com/java/lang/String.html) sourceClass, [String](http://docs.google.com/java/lang/String.html) sourceMethod, [Throwable](http://docs.google.com/java/lang/Throwable.html) thrown)            Log throwing an exception. |
| void | [**warning**](http://docs.google.com/java/util/logging/Logger.html#warning(java.lang.String))([String](http://docs.google.com/java/lang/String.html) msg)            Log a WARNING message. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

### GLOBAL\_LOGGER\_NAME

public static final [String](http://docs.google.com/java/lang/String.html) **GLOBAL\_LOGGER\_NAME**

GLOBAL\_LOGGER\_NAME is a name for the global logger. This name is provided as a convenience to developers who are making casual use of the Logging package. Developers who are making serious use of the logging package (for example in products) should create and use their own Logger objects, with appropriate names, so that logging can be controlled on a suitable per-Logger granularity.

The preferred way to get the global logger object is via the call Logger.getLogger(Logger.GLOBAL\_LOGGER\_NAME).

**Since:** 1.6 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.util.logging.Logger.GLOBAL_LOGGER_NAME)

### global

[@Deprecated](http://docs.google.com/java/lang/Deprecated.html)  
public static final [Logger](http://docs.google.com/java/util/logging/Logger.html) **global**

**Deprecated.** *Initialization of this field is prone to deadlocks. The field must be initialized by the Logger class initialization which may cause deadlocks with the LogManager class initialization. In such cases two class initialization wait for each other to complete. As of JDK version 1.6, the preferred way to get the global logger object is via the call Logger.getLogger(Logger.GLOBAL\_LOGGER\_NAME).*The "global" Logger object is provided as a convenience to developers who are making casual use of the Logging package. Developers who are making serious use of the logging package (for example in products) should create and use their own Logger objects, with appropriate names, so that logging can be controlled on a suitable per-Logger granularity.

| **Constructor Detail** |
| --- |

### Logger

protected **Logger**([String](http://docs.google.com/java/lang/String.html) name,  
 [String](http://docs.google.com/java/lang/String.html) resourceBundleName)

Protected method to construct a logger for a named subsystem.

The logger will be initially configured with a null Level and with useParentHandlers true.

**Parameters:**name - A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as java.net or javax.swing. It may be null for anonymous Loggers.resourceBundleName - name of ResourceBundle to be used for localizing messages for this logger. May be null if none of the messages require localization. **Throws:** [MissingResourceException](http://docs.google.com/java/util/MissingResourceException.html) - if the ResourceBundleName is non-null and no corresponding resource can be found.

| **Method Detail** |
| --- |

### getLogger

public static [Logger](http://docs.google.com/java/util/logging/Logger.html) **getLogger**([String](http://docs.google.com/java/lang/String.html) name)

Find or create a logger for a named subsystem. If a logger has already been created with the given name it is returned. Otherwise a new logger is created.

If a new logger is created its log level will be configured based on the LogManager configuration and it will configured to also send logging output to its parent's handlers. It will be registered in the LogManager global namespace.

**Parameters:**name - A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as java.net or javax.swing **Returns:**a suitable Logger **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the name is null.

### getLogger

public static [Logger](http://docs.google.com/java/util/logging/Logger.html) **getLogger**([String](http://docs.google.com/java/lang/String.html) name,  
 [String](http://docs.google.com/java/lang/String.html) resourceBundleName)

Find or create a logger for a named subsystem. If a logger has already been created with the given name it is returned. Otherwise a new logger is created.

If a new logger is created its log level will be configured based on the LogManager and it will configured to also send logging output to its parent loggers Handlers. It will be registered in the LogManager global namespace.

If the named Logger already exists and does not yet have a localization resource bundle then the given resource bundle name is used. If the named Logger already exists and has a different resource bundle name then an IllegalArgumentException is thrown.

**Parameters:**name - A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as java.net or javax.swingresourceBundleName - name of ResourceBundle to be used for localizing messages for this logger. May be null if none of the messages require localization. **Returns:**a suitable Logger **Throws:** [MissingResourceException](http://docs.google.com/java/util/MissingResourceException.html) - if the named ResourceBundle cannot be found. [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the Logger already exists and uses a different resource bundle name. [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the name is null.

### getAnonymousLogger

public static [Logger](http://docs.google.com/java/util/logging/Logger.html) **getAnonymousLogger**()

Create an anonymous Logger. The newly created Logger is not registered in the LogManager namespace. There will be no access checks on updates to the logger.

This factory method is primarily intended for use from applets. Because the resulting Logger is anonymous it can be kept private by the creating class. This removes the need for normal security checks, which in turn allows untrusted applet code to update the control state of the Logger. For example an applet can do a setLevel or an addHandler on an anonymous Logger.

Even although the new logger is anonymous, it is configured to have the root logger ("") as its parent. This means that by default it inherits its effective level and handlers from the root logger.

**Returns:**a newly created private Logger

### getAnonymousLogger

public static [Logger](http://docs.google.com/java/util/logging/Logger.html) **getAnonymousLogger**([String](http://docs.google.com/java/lang/String.html) resourceBundleName)

Create an anonymous Logger. The newly created Logger is not registered in the LogManager namespace. There will be no access checks on updates to the logger.

This factory method is primarily intended for use from applets. Because the resulting Logger is anonymous it can be kept private by the creating class. This removes the need for normal security checks, which in turn allows untrusted applet code to update the control state of the Logger. For example an applet can do a setLevel or an addHandler on an anonymous Logger.

Even although the new logger is anonymous, it is configured to have the root logger ("") as its parent. This means that by default it inherits its effective level and handlers from the root logger.

**Parameters:**resourceBundleName - name of ResourceBundle to be used for localizing messages for this logger. May be null if none of the messages require localization. **Returns:**a newly created private Logger **Throws:** [MissingResourceException](http://docs.google.com/java/util/MissingResourceException.html) - if the named ResourceBundle cannot be found.

### getResourceBundle

public [ResourceBundle](http://docs.google.com/java/util/ResourceBundle.html) **getResourceBundle**()

Retrieve the localization resource bundle for this logger for the current default locale. Note that if the result is null, then the Logger will use a resource bundle inherited from its parent.

**Returns:**localization bundle (may be null)

### getResourceBundleName

public [String](http://docs.google.com/java/lang/String.html) **getResourceBundleName**()

Retrieve the localization resource bundle name for this logger. Note that if the result is null, then the Logger will use a resource bundle name inherited from its parent.

**Returns:**localization bundle name (may be null)

### setFilter

public void **setFilter**([Filter](http://docs.google.com/java/util/logging/Filter.html) newFilter)  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Set a filter to control output on this Logger.

After passing the initial "level" check, the Logger will call this Filter to check if a log record should really be published.

**Parameters:**newFilter - a filter object (may be null) **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and if the caller does not have LoggingPermission("control").

### getFilter

public [Filter](http://docs.google.com/java/util/logging/Filter.html) **getFilter**()

Get the current filter for this Logger.

**Returns:**a filter object (may be null)

### log

public void **log**([LogRecord](http://docs.google.com/java/util/logging/LogRecord.html) record)

Log a LogRecord.

All the other logging methods in this class call through this method to actually perform any logging. Subclasses can override this single method to capture all log activity.

**Parameters:**record - the LogRecord to be published

### log

public void **log**([Level](http://docs.google.com/java/util/logging/Level.html) level,  
 [String](http://docs.google.com/java/lang/String.html) msg)

Log a message, with no arguments.

If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output Handler objects.

**Parameters:**level - One of the message level identifiers, e.g. SEVEREmsg - The string message (or a key in the message catalog)

### log

public void **log**([Level](http://docs.google.com/java/util/logging/Level.html) level,  
 [String](http://docs.google.com/java/lang/String.html) msg,  
 [Object](http://docs.google.com/java/lang/Object.html) param1)

Log a message, with one object parameter.

If the logger is currently enabled for the given message level then a corresponding LogRecord is created and forwarded to all the registered output Handler objects.

**Parameters:**level - One of the message level identifiers, e.g. SEVEREmsg - The string message (or a key in the message catalog)param1 - parameter to the message

### log

public void **log**([Level](http://docs.google.com/java/util/logging/Level.html) level,  
 [String](http://docs.google.com/java/lang/String.html) msg,  
 [Object](http://docs.google.com/java/lang/Object.html)[] params)

Log a message, with an array of object arguments.

If the logger is currently enabled for the given message level then a corresponding LogRecord is created and forwarded to all the registered output Handler objects.

**Parameters:**level - One of the message level identifiers, e.g. SEVEREmsg - The string message (or a key in the message catalog)params - array of parameters to the message

### log

public void **log**([Level](http://docs.google.com/java/util/logging/Level.html) level,  
 [String](http://docs.google.com/java/lang/String.html) msg,  
 [Throwable](http://docs.google.com/java/lang/Throwable.html) thrown)

Log a message, with associated Throwable information.

If the logger is currently enabled for the given message level then the given arguments are stored in a LogRecord which is forwarded to all registered output handlers.

Note that the thrown argument is stored in the LogRecord thrown property, rather than the LogRecord parameters property. Thus is it processed specially by output Formatters and is not treated as a formatting parameter to the LogRecord message property.

**Parameters:**level - One of the message level identifiers, e.g. SEVEREmsg - The string message (or a key in the message catalog)thrown - Throwable associated with log message.

### logp

public void **logp**([Level](http://docs.google.com/java/util/logging/Level.html) level,  
 [String](http://docs.google.com/java/lang/String.html) sourceClass,  
 [String](http://docs.google.com/java/lang/String.html) sourceMethod,  
 [String](http://docs.google.com/java/lang/String.html) msg)

Log a message, specifying source class and method, with no arguments.

If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output Handler objects.

**Parameters:**level - One of the message level identifiers, e.g. SEVEREsourceClass - name of class that issued the logging requestsourceMethod - name of method that issued the logging requestmsg - The string message (or a key in the message catalog)

### logp

public void **logp**([Level](http://docs.google.com/java/util/logging/Level.html) level,  
 [String](http://docs.google.com/java/lang/String.html) sourceClass,  
 [String](http://docs.google.com/java/lang/String.html) sourceMethod,  
 [String](http://docs.google.com/java/lang/String.html) msg,  
 [Object](http://docs.google.com/java/lang/Object.html) param1)

Log a message, specifying source class and method, with a single object parameter to the log message.

If the logger is currently enabled for the given message level then a corresponding LogRecord is created and forwarded to all the registered output Handler objects.

**Parameters:**level - One of the message level identifiers, e.g. SEVEREsourceClass - name of class that issued the logging requestsourceMethod - name of method that issued the logging requestmsg - The string message (or a key in the message catalog)param1 - Parameter to the log message.

### logp

public void **logp**([Level](http://docs.google.com/java/util/logging/Level.html) level,  
 [String](http://docs.google.com/java/lang/String.html) sourceClass,  
 [String](http://docs.google.com/java/lang/String.html) sourceMethod,  
 [String](http://docs.google.com/java/lang/String.html) msg,  
 [Object](http://docs.google.com/java/lang/Object.html)[] params)

Log a message, specifying source class and method, with an array of object arguments.

If the logger is currently enabled for the given message level then a corresponding LogRecord is created and forwarded to all the registered output Handler objects.

**Parameters:**level - One of the message level identifiers, e.g. SEVEREsourceClass - name of class that issued the logging requestsourceMethod - name of method that issued the logging requestmsg - The string message (or a key in the message catalog)params - Array of parameters to the message

### logp

public void **logp**([Level](http://docs.google.com/java/util/logging/Level.html) level,  
 [String](http://docs.google.com/java/lang/String.html) sourceClass,  
 [String](http://docs.google.com/java/lang/String.html) sourceMethod,  
 [String](http://docs.google.com/java/lang/String.html) msg,  
 [Throwable](http://docs.google.com/java/lang/Throwable.html) thrown)

Log a message, specifying source class and method, with associated Throwable information.

If the logger is currently enabled for the given message level then the given arguments are stored in a LogRecord which is forwarded to all registered output handlers.

Note that the thrown argument is stored in the LogRecord thrown property, rather than the LogRecord parameters property. Thus is it processed specially by output Formatters and is not treated as a formatting parameter to the LogRecord message property.

**Parameters:**level - One of the message level identifiers, e.g. SEVEREsourceClass - name of class that issued the logging requestsourceMethod - name of method that issued the logging requestmsg - The string message (or a key in the message catalog)thrown - Throwable associated with log message.

### logrb

public void **logrb**([Level](http://docs.google.com/java/util/logging/Level.html) level,  
 [String](http://docs.google.com/java/lang/String.html) sourceClass,  
 [String](http://docs.google.com/java/lang/String.html) sourceMethod,  
 [String](http://docs.google.com/java/lang/String.html) bundleName,  
 [String](http://docs.google.com/java/lang/String.html) msg)

Log a message, specifying source class, method, and resource bundle name with no arguments.

If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output Handler objects.

The msg string is localized using the named resource bundle. If the resource bundle name is null, or an empty String or invalid then the msg string is not localized.

**Parameters:**level - One of the message level identifiers, e.g. SEVEREsourceClass - name of class that issued the logging requestsourceMethod - name of method that issued the logging requestbundleName - name of resource bundle to localize msg, can be nullmsg - The string message (or a key in the message catalog)

### logrb

public void **logrb**([Level](http://docs.google.com/java/util/logging/Level.html) level,  
 [String](http://docs.google.com/java/lang/String.html) sourceClass,  
 [String](http://docs.google.com/java/lang/String.html) sourceMethod,  
 [String](http://docs.google.com/java/lang/String.html) bundleName,  
 [String](http://docs.google.com/java/lang/String.html) msg,  
 [Object](http://docs.google.com/java/lang/Object.html) param1)

Log a message, specifying source class, method, and resource bundle name, with a single object parameter to the log message.

If the logger is currently enabled for the given message level then a corresponding LogRecord is created and forwarded to all the registered output Handler objects.

The msg string is localized using the named resource bundle. If the resource bundle name is null, or an empty String or invalid then the msg string is not localized.

**Parameters:**level - One of the message level identifiers, e.g. SEVEREsourceClass - name of class that issued the logging requestsourceMethod - name of method that issued the logging requestbundleName - name of resource bundle to localize msg, can be nullmsg - The string message (or a key in the message catalog)param1 - Parameter to the log message.

### logrb

public void **logrb**([Level](http://docs.google.com/java/util/logging/Level.html) level,  
 [String](http://docs.google.com/java/lang/String.html) sourceClass,  
 [String](http://docs.google.com/java/lang/String.html) sourceMethod,  
 [String](http://docs.google.com/java/lang/String.html) bundleName,  
 [String](http://docs.google.com/java/lang/String.html) msg,  
 [Object](http://docs.google.com/java/lang/Object.html)[] params)

Log a message, specifying source class, method, and resource bundle name, with an array of object arguments.

If the logger is currently enabled for the given message level then a corresponding LogRecord is created and forwarded to all the registered output Handler objects.

The msg string is localized using the named resource bundle. If the resource bundle name is null, or an empty String or invalid then the msg string is not localized.

**Parameters:**level - One of the message level identifiers, e.g. SEVEREsourceClass - name of class that issued the logging requestsourceMethod - name of method that issued the logging requestbundleName - name of resource bundle to localize msg, can be null.msg - The string message (or a key in the message catalog)params - Array of parameters to the message

### logrb

public void **logrb**([Level](http://docs.google.com/java/util/logging/Level.html) level,  
 [String](http://docs.google.com/java/lang/String.html) sourceClass,  
 [String](http://docs.google.com/java/lang/String.html) sourceMethod,  
 [String](http://docs.google.com/java/lang/String.html) bundleName,  
 [String](http://docs.google.com/java/lang/String.html) msg,  
 [Throwable](http://docs.google.com/java/lang/Throwable.html) thrown)

Log a message, specifying source class, method, and resource bundle name, with associated Throwable information.

If the logger is currently enabled for the given message level then the given arguments are stored in a LogRecord which is forwarded to all registered output handlers.

The msg string is localized using the named resource bundle. If the resource bundle name is null, or an empty String or invalid then the msg string is not localized.

Note that the thrown argument is stored in the LogRecord thrown property, rather than the LogRecord parameters property. Thus is it processed specially by output Formatters and is not treated as a formatting parameter to the LogRecord message property.

**Parameters:**level - One of the message level identifiers, e.g. SEVEREsourceClass - name of class that issued the logging requestsourceMethod - name of method that issued the logging requestbundleName - name of resource bundle to localize msg, can be nullmsg - The string message (or a key in the message catalog)thrown - Throwable associated with log message.

### entering

public void **entering**([String](http://docs.google.com/java/lang/String.html) sourceClass,  
 [String](http://docs.google.com/java/lang/String.html) sourceMethod)

Log a method entry.

This is a convenience method that can be used to log entry to a method. A LogRecord with message "ENTRY", log level FINER, and the given sourceMethod and sourceClass is logged.

**Parameters:**sourceClass - name of class that issued the logging requestsourceMethod - name of method that is being entered

### entering

public void **entering**([String](http://docs.google.com/java/lang/String.html) sourceClass,  
 [String](http://docs.google.com/java/lang/String.html) sourceMethod,  
 [Object](http://docs.google.com/java/lang/Object.html) param1)

Log a method entry, with one parameter.

This is a convenience method that can be used to log entry to a method. A LogRecord with message "ENTRY {0}", log level FINER, and the given sourceMethod, sourceClass, and parameter is logged.

**Parameters:**sourceClass - name of class that issued the logging requestsourceMethod - name of method that is being enteredparam1 - parameter to the method being entered

### entering

public void **entering**([String](http://docs.google.com/java/lang/String.html) sourceClass,  
 [String](http://docs.google.com/java/lang/String.html) sourceMethod,  
 [Object](http://docs.google.com/java/lang/Object.html)[] params)

Log a method entry, with an array of parameters.

This is a convenience method that can be used to log entry to a method. A LogRecord with message "ENTRY" (followed by a format {N} indicator for each entry in the parameter array), log level FINER, and the given sourceMethod, sourceClass, and parameters is logged.

**Parameters:**sourceClass - name of class that issued the logging requestsourceMethod - name of method that is being enteredparams - array of parameters to the method being entered

### exiting

public void **exiting**([String](http://docs.google.com/java/lang/String.html) sourceClass,  
 [String](http://docs.google.com/java/lang/String.html) sourceMethod)

Log a method return.

This is a convenience method that can be used to log returning from a method. A LogRecord with message "RETURN", log level FINER, and the given sourceMethod and sourceClass is logged.

**Parameters:**sourceClass - name of class that issued the logging requestsourceMethod - name of the method

### exiting

public void **exiting**([String](http://docs.google.com/java/lang/String.html) sourceClass,  
 [String](http://docs.google.com/java/lang/String.html) sourceMethod,  
 [Object](http://docs.google.com/java/lang/Object.html) result)

Log a method return, with result object.

This is a convenience method that can be used to log returning from a method. A LogRecord with message "RETURN {0}", log level FINER, and the gives sourceMethod, sourceClass, and result object is logged.

**Parameters:**sourceClass - name of class that issued the logging requestsourceMethod - name of the methodresult - Object that is being returned

### throwing

public void **throwing**([String](http://docs.google.com/java/lang/String.html) sourceClass,  
 [String](http://docs.google.com/java/lang/String.html) sourceMethod,  
 [Throwable](http://docs.google.com/java/lang/Throwable.html) thrown)

Log throwing an exception.

This is a convenience method to log that a method is terminating by throwing an exception. The logging is done using the FINER level.

If the logger is currently enabled for the given message level then the given arguments are stored in a LogRecord which is forwarded to all registered output handlers. The LogRecord's message is set to "THROW".

Note that the thrown argument is stored in the LogRecord thrown property, rather than the LogRecord parameters property. Thus is it processed specially by output Formatters and is not treated as a formatting parameter to the LogRecord message property.

**Parameters:**sourceClass - name of class that issued the logging requestsourceMethod - name of the method.thrown - The Throwable that is being thrown.

### severe

public void **severe**([String](http://docs.google.com/java/lang/String.html) msg)

Log a SEVERE message.

If the logger is currently enabled for the SEVERE message level then the given message is forwarded to all the registered output Handler objects.

**Parameters:**msg - The string message (or a key in the message catalog)

### warning

public void **warning**([String](http://docs.google.com/java/lang/String.html) msg)

Log a WARNING message.

If the logger is currently enabled for the WARNING message level then the given message is forwarded to all the registered output Handler objects.

**Parameters:**msg - The string message (or a key in the message catalog)

### info

public void **info**([String](http://docs.google.com/java/lang/String.html) msg)

Log an INFO message.

If the logger is currently enabled for the INFO message level then the given message is forwarded to all the registered output Handler objects.

**Parameters:**msg - The string message (or a key in the message catalog)

### config

public void **config**([String](http://docs.google.com/java/lang/String.html) msg)

Log a CONFIG message.

If the logger is currently enabled for the CONFIG message level then the given message is forwarded to all the registered output Handler objects.

**Parameters:**msg - The string message (or a key in the message catalog)

### fine

public void **fine**([String](http://docs.google.com/java/lang/String.html) msg)

Log a FINE message.

If the logger is currently enabled for the FINE message level then the given message is forwarded to all the registered output Handler objects.

**Parameters:**msg - The string message (or a key in the message catalog)

### finer

public void **finer**([String](http://docs.google.com/java/lang/String.html) msg)

Log a FINER message.

If the logger is currently enabled for the FINER message level then the given message is forwarded to all the registered output Handler objects.

**Parameters:**msg - The string message (or a key in the message catalog)

### finest

public void **finest**([String](http://docs.google.com/java/lang/String.html) msg)

Log a FINEST message.

If the logger is currently enabled for the FINEST message level then the given message is forwarded to all the registered output Handler objects.

**Parameters:**msg - The string message (or a key in the message catalog)

### setLevel

public void **setLevel**([Level](http://docs.google.com/java/util/logging/Level.html) newLevel)  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Set the log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded. The level value Level.OFF can be used to turn off logging.

If the new level is null, it means that this node should inherit its level from its nearest ancestor with a specific (non-null) level value.

**Parameters:**newLevel - the new value for the log level (may be null) **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and if the caller does not have LoggingPermission("control").

### getLevel

public [Level](http://docs.google.com/java/util/logging/Level.html) **getLevel**()

Get the log Level that has been specified for this Logger. The result may be null, which means that this logger's effective level will be inherited from its parent.

**Returns:**this Logger's level

### isLoggable

public boolean **isLoggable**([Level](http://docs.google.com/java/util/logging/Level.html) level)

Check if a message of the given level would actually be logged by this logger. This check is based on the Loggers effective level, which may be inherited from its parent.

**Parameters:**level - a message logging level **Returns:**true if the given message level is currently being logged.

### getName

public [String](http://docs.google.com/java/lang/String.html) **getName**()

Get the name for this logger.

**Returns:**logger name. Will be null for anonymous Loggers.

### addHandler

public void **addHandler**([Handler](http://docs.google.com/java/util/logging/Handler.html) handler)  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Add a log Handler to receive logging messages.

By default, Loggers also send their output to their parent logger. Typically the root Logger is configured with a set of Handlers that essentially act as default handlers for all loggers.

**Parameters:**handler - a logging Handler **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and if the caller does not have LoggingPermission("control").

### removeHandler

public void **removeHandler**([Handler](http://docs.google.com/java/util/logging/Handler.html) handler)  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Remove a log Handler.

Returns silently if the given Handler is not found or is null

**Parameters:**handler - a logging Handler **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and if the caller does not have LoggingPermission("control").

### getHandlers

public [Handler](http://docs.google.com/java/util/logging/Handler.html)[] **getHandlers**()

Get the Handlers associated with this logger.

**Returns:**an array of all registered Handlers

### setUseParentHandlers

public void **setUseParentHandlers**(boolean useParentHandlers)

Specify whether or not this logger should send its output to it's parent Logger. This means that any LogRecords will also be written to the parent's Handlers, and potentially to its parent, recursively up the namespace.

**Parameters:**useParentHandlers - true if output is to be sent to the logger's parent. **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and if the caller does not have LoggingPermission("control").

### getUseParentHandlers

public boolean **getUseParentHandlers**()

Discover whether or not this logger is sending its output to its parent logger.

**Returns:**true if output is to be sent to the logger's parent

### getParent

public [Logger](http://docs.google.com/java/util/logging/Logger.html) **getParent**()

Return the parent for this Logger.

This method returns the nearest extant parent in the namespace. Thus if a Logger is called "a.b.c.d", and a Logger called "a.b" has been created but no logger "a.b.c" exists, then a call of getParent on the Logger "a.b.c.d" will return the Logger "a.b".

The result will be null if it is called on the root Logger in the namespace.

**Returns:**nearest existing parent Logger

### setParent

public void **setParent**([Logger](http://docs.google.com/java/util/logging/Logger.html) parent)

Set the parent for this Logger. This method is used by the LogManager to update a Logger when the namespace changes.

It should not be called from application code.

**Parameters:**parent - the new parent logger **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and if the caller does not have LoggingPermission("control").

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Logger.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/util/logging/Level.html)   [**NEXT CLASS**](http://docs.google.com/java/util/logging/LoggingMXBean.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/logging/Logger.html)    [**NO FRAMES**](http://docs.google.com/Logger.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#17dp8vu) | [METHOD](#26in1rg) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).