| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/LogManager.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/LoggingPermission.html)   [**NEXT CLASS**](http://docs.google.com/java/util/logging/LogRecord.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/logging/LogManager.html)    [**NO FRAMES**](http://docs.google.com/LogManager.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#2s8eyo1) | [METHOD](#3rdcrjn) |

## **java.util.logging**

Class LogManager

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

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

There is a single global LogManager object that is used to maintain a set of shared state about Loggers and log services.

This LogManager object:

* Manages a hierarchical namespace of Logger objects. All named Loggers are stored in this namespace.
* Manages a set of logging control properties. These are simple key-value pairs that can be used by Handlers and other logging objects to configure themselves.

The global LogManager object can be retrieved using LogManager.getLogManager(). The LogManager object is created during class initialization and cannot subsequently be changed.

At startup the LogManager class is located using the java.util.logging.manager system property.

By default, the LogManager reads its initial configuration from a properties file "lib/logging.properties" in the JRE directory. If you edit that property file you can change the default logging configuration for all uses of that JRE.

In addition, the LogManager uses two optional system properties that allow more control over reading the initial configuration:

* "java.util.logging.config.class"
* "java.util.logging.config.file"

These two properties may be set via the Preferences API, or as command line property definitions to the "java" command, or as system property definitions passed to JNI\_CreateJavaVM.

If the "java.util.logging.config.class" property is set, then the property value is treated as a class name. The given class will be loaded, an object will be instantiated, and that object's constructor is responsible for reading in the initial configuration. (That object may use other system properties to control its configuration.) The alternate configuration class can use readConfiguration(InputStream) to define properties in the LogManager.

If "java.util.logging.config.class" property is **not** set, then the "java.util.logging.config.file" system property can be used to specify a properties file (in java.util.Properties format). The initial logging configuration will be read from this file.

If neither of these properties is defined then, as described above, the LogManager will read its initial configuration from a properties file "lib/logging.properties" in the JRE directory.

The properties for loggers and Handlers will have names starting with the dot-separated name for the handler or logger.

The global logging properties may include:

* A property "handlers". This defines a whitespace or comma separated list of class names for handler classes to load and register as handlers on the root Logger (the Logger named ""). Each class name must be for a Handler class which has a default constructor. Note that these Handlers may be created lazily, when they are first used.
* A property "<logger>.handlers". This defines a whitespace or comma separated list of class names for handlers classes to load and register as handlers to the specified logger. Each class name must be for a Handler class which has a default constructor. Note that these Handlers may be created lazily, when they are first used.
* A property "<logger>.useParentHandlers". This defines a boolean value. By default every logger calls its parent in addition to handling the logging message itself, this often result in messages being handled by the root logger as well. When setting this property to false a Handler needs to be configured for this logger otherwise no logging messages are delivered.
* A property "config". This property is intended to allow arbitrary configuration code to be run. The property defines a whitespace or comma separated list of class names. A new instance will be created for each named class. The default constructor of each class may execute arbitrary code to update the logging configuration, such as setting logger levels, adding handlers, adding filters, etc.

Note that all classes loaded during LogManager configuration are first searched on the system class path before any user class path. That includes the LogManager class, any config classes, and any handler classes.

Loggers are organized into a naming hierarchy based on their dot separated names. Thus "a.b.c" is a child of "a.b", but "a.b1" and a.b2" are peers.

All properties whose names end with ".level" are assumed to define log levels for Loggers. Thus "foo.level" defines a log level for the logger called "foo" and (recursively) for any of its children in the naming hierarchy. Log Levels are applied in the order they are defined in the properties file. Thus level settings for child nodes in the tree should come after settings for their parents. The property name ".level" can be used to set the level for the root of the tree.

All methods on the LogManager object are multi-thread safe.

**Since:** 1.4

| **Field Summary** | |
| --- | --- |
| static [String](http://docs.google.com/java/lang/String.html) | [**LOGGING\_MXBEAN\_NAME**](http://docs.google.com/java/util/logging/LogManager.html#LOGGING_MXBEAN_NAME)            String representation of the [ObjectName](http://docs.google.com/javax/management/ObjectName.html) for [LoggingMXBean](http://docs.google.com/java/util/logging/LoggingMXBean.html). |

| **Constructor Summary** | |
| --- | --- |
| protected | [**LogManager**](http://docs.google.com/java/util/logging/LogManager.html#LogManager())()            Protected constructor. |

| **Method Summary** | |
| --- | --- |
| boolean | [**addLogger**](http://docs.google.com/java/util/logging/LogManager.html#addLogger(java.util.logging.Logger))([Logger](http://docs.google.com/java/util/logging/Logger.html) logger)            Add a named logger. |
| void | [**addPropertyChangeListener**](http://docs.google.com/java/util/logging/LogManager.html#addPropertyChangeListener(java.beans.PropertyChangeListener))([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) l)            Adds an event listener to be invoked when the logging properties are re-read. |
| void | [**checkAccess**](http://docs.google.com/java/util/logging/LogManager.html#checkAccess())()            Check that the current context is trusted to modify the logging configuration. |
| [Logger](http://docs.google.com/java/util/logging/Logger.html) | [**getLogger**](http://docs.google.com/java/util/logging/LogManager.html#getLogger(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Method to find a named logger. |
| [Enumeration](http://docs.google.com/java/util/Enumeration.html)<[String](http://docs.google.com/java/lang/String.html)> | [**getLoggerNames**](http://docs.google.com/java/util/logging/LogManager.html#getLoggerNames())()            Get an enumeration of known logger names. |
| static [LoggingMXBean](http://docs.google.com/java/util/logging/LoggingMXBean.html) | [**getLoggingMXBean**](http://docs.google.com/java/util/logging/LogManager.html#getLoggingMXBean())()            Returns LoggingMXBean for managing loggers. |
| static [LogManager](http://docs.google.com/java/util/logging/LogManager.html) | [**getLogManager**](http://docs.google.com/java/util/logging/LogManager.html#getLogManager())()            Return the global LogManager object. |
| [String](http://docs.google.com/java/lang/String.html) | [**getProperty**](http://docs.google.com/java/util/logging/LogManager.html#getProperty(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Get the value of a logging property. |
| void | [**readConfiguration**](http://docs.google.com/java/util/logging/LogManager.html#readConfiguration())()            Reinitialize the logging properties and reread the logging configuration. |
| void | [**readConfiguration**](http://docs.google.com/java/util/logging/LogManager.html#readConfiguration(java.io.InputStream))([InputStream](http://docs.google.com/java/io/InputStream.html) ins)            Reinitialize the logging properties and reread the logging configuration from the given stream, which should be in java.util.Properties format. |
| void | [**removePropertyChangeListener**](http://docs.google.com/java/util/logging/LogManager.html#removePropertyChangeListener(java.beans.PropertyChangeListener))([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) l)            Removes an event listener for property change events. |
| void | [**reset**](http://docs.google.com/java/util/logging/LogManager.html#reset())()            Reset the logging configuration. |

| **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** |
| --- |

### LOGGING\_MXBEAN\_NAME

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

String representation of the [ObjectName](http://docs.google.com/javax/management/ObjectName.html) for [LoggingMXBean](http://docs.google.com/java/util/logging/LoggingMXBean.html).

**Since:** 1.5 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.util.logging.LogManager.LOGGING_MXBEAN_NAME)

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

### LogManager

protected **LogManager**()

Protected constructor. This is protected so that container applications (such as J2EE containers) can subclass the object. It is non-public as it is intended that there only be one LogManager object, whose value is retrieved by calling Logmanager.getLogManager.

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

### getLogManager

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

Return the global LogManager object.

### addPropertyChangeListener

public void **addPropertyChangeListener**([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) l)  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Adds an event listener to be invoked when the logging properties are re-read. Adding multiple instances of the same event Listener results in multiple entries in the property event listener table.

**Parameters:**l - event listener **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and if the caller does not have LoggingPermission("control"). [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the PropertyChangeListener is null.

### removePropertyChangeListener

public void **removePropertyChangeListener**([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) l)  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Removes an event listener for property change events. If the same listener instance has been added to the listener table through multiple invocations of addPropertyChangeListener, then an equivalent number of removePropertyChangeListener invocations are required to remove all instances of that listener from the listener table.

Returns silently if the given listener is not found.

**Parameters:**l - event listener (can 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").

### addLogger

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

Add a named logger. This does nothing and returns false if a logger with the same name is already registered.

The Logger factory methods call this method to register each newly created Logger.

The application should retain its own reference to the Logger object to avoid it being garbage collected. The LogManager may only retain a weak reference.

**Parameters:**logger - the new logger. **Returns:**true if the argument logger was registered successfully, false if a logger of that name already exists. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the logger name is null.

### getLogger

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

Method to find a named logger.

Note that since untrusted code may create loggers with arbitrary names this method should not be relied on to find Loggers for security sensitive logging.

**Parameters:**name - name of the logger **Returns:**matching logger or null if none is found

### getLoggerNames

public [Enumeration](http://docs.google.com/java/util/Enumeration.html)<[String](http://docs.google.com/java/lang/String.html)> **getLoggerNames**()

Get an enumeration of known logger names.

Note: Loggers may be added dynamically as new classes are loaded. This method only reports on the loggers that are currently registered.

**Returns:**enumeration of logger name strings

### readConfiguration

public void **readConfiguration**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Reinitialize the logging properties and reread the logging configuration.

The same rules are used for locating the configuration properties as are used at startup. So normally the logging properties will be re-read from the same file that was used at startup.

Any log level definitions in the new configuration file will be applied using Logger.setLevel(), if the target Logger exists.

A PropertyChangeEvent will be fired after the properties are read.

**Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and if the caller does not have LoggingPermission("control"). [IOException](http://docs.google.com/java/io/IOException.html) - if there are IO problems reading the configuration.

### reset

public void **reset**()  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Reset the logging configuration.

For all named loggers, the reset operation removes and closes all Handlers and (except for the root logger) sets the level to null. The root logger's level is set to Level.INFO.

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

### readConfiguration

public void **readConfiguration**([InputStream](http://docs.google.com/java/io/InputStream.html) ins)  
 throws [IOException](http://docs.google.com/java/io/IOException.html),  
 [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Reinitialize the logging properties and reread the logging configuration from the given stream, which should be in java.util.Properties format. A PropertyChangeEvent will be fired after the properties are read.

Any log level definitions in the new configuration file will be applied using Logger.setLevel(), if the target Logger exists.

**Parameters:**ins - stream to read properties from **Throws:** [SecurityException](http://docs.google.com/java/lang/SecurityException.html) - if a security manager exists and if the caller does not have LoggingPermission("control"). [IOException](http://docs.google.com/java/io/IOException.html) - if there are problems reading from the stream.

### getProperty

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

Get the value of a logging property. The method returns null if the property is not found.

**Parameters:**name - property name **Returns:**property value

### checkAccess

public void **checkAccess**()  
 throws [SecurityException](http://docs.google.com/java/lang/SecurityException.html)

Check that the current context is trusted to modify the logging configuration. This requires LoggingPermission("control").

If the check fails we throw a SecurityException, otherwise we return normally.

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

### getLoggingMXBean

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

Returns LoggingMXBean for managing loggers. The LoggingMXBean can also obtained from the [platform MBeanServer](http://docs.google.com/java/lang/management/ManagementFactory.html#getPlatformMBeanServer()) method.

**Returns:**a [LoggingMXBean](http://docs.google.com/java/util/logging/LoggingMXBean.html) object.**Since:** 1.5 **See Also:**[ManagementFactory](http://docs.google.com/java/lang/management/ManagementFactory.html)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/LogManager.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/LoggingPermission.html)   [**NEXT CLASS**](http://docs.google.com/java/util/logging/LogRecord.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/logging/LogManager.html)    [**NO FRAMES**](http://docs.google.com/LogManager.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#2s8eyo1) | [METHOD](#3rdcrjn) |

[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).