Oracle: Java logger API

return to the de Facto Standards

: **Note**: The following is an excerpt from the official TTTTT site. It is provided here as a connivence and is not authoritative. Refer to the original document as the authoritative reference.

Table 1: Data sheet for Java logger API,

Title	Java logger API (Java Platform SE-8)	
Acronym	Logger	
Version	Java logger API	
Operating Systems	Java VM	
Downloads	https://www.oracle.com/technetwork/java/javase/downloads/index.html	
Supported Languages		
License	[https://download.oracle.com/otndocs/jcp/java_se-8-mrel-spec/license.html	Oracle Proprietary
Reference	[https://docs.oracle.com/en/java/javase/12/docs/api/java.logging/module-summary.html]]	

Overview

Note: The following is an excerpt from the official Oracle site. It is provided here as a connivence and is not authoritative. Refer to the original document(s) as the authoritative reference.
Provides the classes and interfaces of the Java™ 2 platform's core logging facilities. The central goal of the logging APIs is to support maintaining and servicing software at customer sites. There are four main target uses of the logs:

1. *Problem diagnosis by end users and system administrators. This consists of simple logging of common problems that can be fixed or tracked locally, such as running out of resources, security failures, and simple configuration errors.*
2. *Problem diagnosis by field service engineers. The logging information used by field service engineers may be considerably more complex and verbose than that required by system administrators. Typically such information will require extra logging within particular subsystems.*
3. *Problem diagnosis by the development organization. When a problem occurs in the field, it may be necessary to return the captured logging information to the original development team for diagnosis. This logging information may be extremely detailed and fairly inscrutable. Such information might include detailed tracing on the internal execution of particular subsystems.*
4. *Problem diagnosis by developers. The Logging APIs may also be used to help debug an application under development. This may include logging information generated by the target application as well as logging information generated by lower-level libraries. Note however that while this use is perfectly reasonable, the logging APIs are not intended to replace the normal debugging and*

profiling tools that may already exist in the development environment.

The key elements of this package include:

- ***Logger*** *The main entity on which applications make logging calls. A Logger object is used to log messages for a specific system or application component.*
- ***LogRecord*** *Used to pass logging requests between the logging framework and individual log handlers.*
- ***Handler*** *Exports LogRecord objects to a variety of destinations including memory, output streams, consoles, files, and sockets. A variety of Handler subclasses exist for this purpose. Additional Handlers may be developed by third parties and delivered on top of the core platform.*
- ***Level*** *Defines a set of standard logging levels that can be used to control logging output. Programs can be configured to output logging for some levels while ignoring output for others. Filter: Provides fine-grained control over what gets logged, beyond the control provided by log levels. The logging APIs support a general-purpose filter mechanism that allows application code to attach arbitrary filters to control logging output.*
- ***Formatter*** *Provides support for formatting LogRecord objects. This package includes two formatters, SimpleFormatter and XMLFormatter, for formatting log records in plain text or XML respectively. As with Handlers, additional Formatters may be developed by third parties.*

  The Logging APIs offer both static and dynamic configuration control. Static control enables field service staff to set up a particular configuration and then re-launch the application with the new logging settings. Dynamic control allows for updates to the logging configuration within a currently running program. The APIs also allow for logging to be enabled or disabled for different functional areas of the system. For example, a field service engineer might be interested in tracing all AWT events, but might have no interest in socket events or memory management.

From:
https://omgwiki.org/dido/ - **DIDO Wiki**

Permanent link:
https://omgwiki.org/dido/doku.php?id=dido:public:ra:xapend:xapend.b_stds:defact:orcle:java_logger_api&rev=1559928514

Last update: **2019/06/07 13:28**