

# Quarkus - Configuring Logging

This guide explains logging and how to configure it.

## Runtime configuration

Run time logging is configured in the `application.properties` file, for example to set everything to **INFO** logging except Hibernate:

```
quarkus.log.level=INFO
quarkus.log.category."org.hibernate".level=DEBUG
```

All possible properties are listed in [the logging configuration reference](#).



If you are adding these properties via command line make sure `"` is escaped. For example `-Dquarkus.log.category.\"org.hibernate\".level=DEBUG`.

## Logging categories

Logging is done on a per-category basis. Each category can be independently configured. A configuration which applies to a category will also apply to all sub-categories of that category, unless there is a more specific matching sub-category configuration. For every category the same settings that are configured on ( console / file / syslog ) apply. These can also be overridden by attaching a one or more named handlers to a category. See example in [Named handlers attached to a category](#)

Property Name	Default	Description
<code>quarkus.log.category."&lt;category-name&gt;".level</code>	<b>INFO</b> <sup>[1]</sup>	The level to use to configure the category named <code>&lt;category-name&gt;</code> . The quotes are necessary.
<code>quarkus.log.category."&lt;category-name&gt;".use-parent-handlers</code>	<b>true</b>	Specify whether or not this logger should send its output to its parent logger.
<code>quarkus.log.category."&lt;category-name&gt;".handlers=[&lt;handler&gt;]</code>	<b>empty</b> <sup>[2]</sup>	The names of the handlers that you want to attach to a specific category.



The quotes shown in the property name are required as categories normally contain `'` which must be escaped. An example is shown in [File TRACE Logging Configuration](#).

# Root logger configuration

The root logger category is handled separately, and is configured via the following properties:

Property Name	Default	Description
<code>quarkus.log.level</code>	<code>INFO</code>	The default minimum log level for every log category.

## Log levels

There are several log levels you can use:

Level	Description
<code>FATAL</code>	A critical service failure/complete inability to service requests of any kind
<code>ERROR</code>	A significant disruption in a request or the inability to service a request
<code>WARN</code>	A non-critical service error or problem that may not require immediate correction
<code>INFO</code>	Service lifecycle events or important related very-low-frequency information
<code>DEBUG</code>	Messages that convey extra information regarding lifecycle or non-request-bound events which may be helpful for debugging
<code>TRACE</code>	Messages that convey extra per-request debugging information that may be very high frequency

## Format String

The logging format string supports the following symbols:

Symbol	Summary	Description
<code>%%</code>	<code>%</code>	Renders a simple <code>%</code> character.
<code>%c</code>	Category	Renders the category name.
<code>%C</code>	Source class	Renders the source class name. <sup>[3]</sup>
<code>%d{xxx}</code>	Date	Renders a date with the given date format string, which uses the syntax defined by <code>java.text.SimpleDateFormat</code> .
<code>%e</code>	Exception	Renders the thrown exception, if any.
<code>%F</code>	Source file	Renders the source file name. <sup>[3]</sup>
<code>%h</code>	Host name	Renders the system simple host name.

Symbol	Summary	Description
%H	Qualified host name	Renders the system's fully qualified host name, which may be the same as the simple host name, depending on OS configuration.
%i	Process ID	Render the current process PID.
%l	Source location	Renders the source location information, which includes source file name, line number, class name, and method name. <sup>[3]</sup>
%L	Source line	Renders the source line number. <sup>[3]</sup>
%m	Full Message	Renders the log message plus exception (if any).
%M	Source method	Renders the source method name. <sup>[3]</sup>
%n	Newline	Renders the platform-specific line separator string.
%N	Process name	Render the name of the current process.
%p	Level	Render the log level of the message.
%r	Relative time	Render the time in milliseconds since the start of the application log.
%s	Simple message	Renders just the log message, with no exception trace.
%t	Thread name	Render the thread name.
%t{id}	Thread ID	Render the thread ID.
%z{<zone name>}	Time zone	Set the time zone of the output to <zone name>.
%X{<MDC property name>}	Mapped Diagnostics Context Value	Renders the value from Mapped Diagnostics Context
%X	Mapped Diagnostics Context Values	Renders all the values from Mapped Diagnostics Context in format {property.key=property.value}
%x	Nested Diagnostics context values	Renders all the values from Nested Diagnostics Context in format {value1.value2}

## Alternative Console Logging Formats

It is possible to change the output format of the console log. This can be useful in environments where the output of the Quarkus application is captured by a service which can, for example, process and store the log information for later analysis.

## JSON Logging

In order to configure JSON logging, the `quarkus-logging-json` extension may be employed. Add this extension to your application POM as the following snippet illustrates.

*Modifications to POM file to add the JSON logging extension*

```
<dependencies>
  <!-- ... your other dependencies are here ... -->
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-logging-json</artifactId>
  </dependency>
</dependencies>
```

The presence of this extension will, by default, replace the output format configuration from the console configuration. This means that the format string and the color settings (if any) will be ignored. The other console configuration items (including those controlling asynchronous logging and the log level) will continue to be applied.


For some, it will make sense to use logging that is humanly readable (unstructured) in dev mode and JSON logging (structured) in production mode. This can be achieved using different profiles, as shown in the following configuration.

*Disable JSON logging in application.properties for dev and test mode*

```
%dev.quarkus.log.console.json=false
%test.quarkus.log.console.json=false
```

## Configuration

The JSON logging extension can be configured in various ways. The following properties are supported:

 Configuration property fixed at build time - All other configuration properties are overridable at runtime

Configuration property	Type	Default
<code>quarkus.log.console.json</code> Determine whether to enable the JSON console formatting extension, which disables "normal" console formatting.	boolean	<code>true</code>
<code>quarkus.log.console.json.pretty-print</code> Enable "pretty printing" of the JSON record. Note that some JSON parsers will fail to read pretty printed output.	boolean	<code>false</code>

<code>quarkus.log.console.json.date-format</code>		
The date format to use. The special string "default" indicates that the default format should be used.	string	default
<code>quarkus.log.console.json.record-delimiter</code>		
The special end-of-record delimiter to be used. By default, no delimiter is used.	string	
<code>quarkus.log.console.json.zone-id</code>		
The zone ID to use. The special string "default" indicates that the default zone should be used.	string	default
<code>quarkus.log.console.json.exception-output-type</code>		
The exception output type to specify.	detailed, formatted, detailed-and-formatted	detailed
<code>quarkus.log.console.json.print-details</code>		
Enable printing of more details in the log. Printing the details can be expensive as the values are retrieved from the caller. The details include the source class name, source file name, source method name and source line number.	boolean	false



Enabling pretty printing might cause certain processors and JSON parsers to fail.



Printing the details can be expensive as the values are retrieved from the caller. The details include the source class name, source file name, source method name and source line number.

## Examples

*Console DEBUG Logging, No color, Shortened Time, Shortened Category Prefixes*

```
quarkus.log.console.enable=true
quarkus.log.console.format=%d{HH:mm:ss} %-5p [%c{2.}] (%t) %s%e%n
quarkus.log.console.level=DEBUG
quarkus.log.console.color=false

quarkus.log.category."io.quarkus".level=DEBUG
```



If you are adding these properties via command line make sure " is escaped. For example `-Dquarkus.log.category.\"io.quarkus\".level=DEBUG`.

#### File TRACE Logging Configuration

```
quarkus.log.file.enable=true
# Send output to a trace.log file under the /tmp directory
quarkus.log.file.path=/tmp/trace.log
quarkus.log.file.level=TRACE
quarkus.log.file.format=%d{HH:mm:ss} %-5p [%c{2.}] (%t) %s%n
# Set 2 categories (io.quarkus.smallrye.jwt,
io.undertow.request.security) to TRACE level
quarkus.log.category."io.quarkus.smallrye.jwt".level=TRACE
quarkus.log.category."io.undertow.request.security".level=TRACE
```

#### Named handlers attached to a category

```
# Send output to the console
quarkus.log.file.path=/tmp/trace.log
quarkus.log.console.format=%d{HH:mm:ss} %-5p [%c{2.}] (%t) %s%n
# Configure a named handler that logs to console
quarkus.log.handler.console."STRUCTURED_LOGGING".format=%e%n
# Configure a named handler that logs to file
quarkus.log.handler.file."STRUCTURED_LOGGING_FILE".enable=true
quarkus.log.handler.file."STRUCTURED_LOGGING_FILE".format=%e%n
# Configure the category and link the two named handlers to it
quarkus.log.category."io.quarkus.category".level=INFO
quarkus.log.category."io.quarkus.category".handlers=STRUCTURED_LOGG
ING,STRUCTURED_LOGGING_FILE
```

## Supported Logging APIs

Applications and components may use any of the following APIs for logging, and the logs will be merged:

- JDK `java.util.logging`
- [JBoss Logging](#)
- [SLF4J](#)
- [Apache Commons Logging](#)

## Centralized Log Management

If you want to send your logs to a centralized tool like Graylog, Logstash or Fluentd, you can follow the [Centralized log management guide](#).

# How to Configure Logging for @QuarkusTest

If you want to configure logging for your `@QuarkusTest`, don't forget to set up the `maven-surefire-plugin` accordingly. In particular, you need to set the appropriate `LogManager` using the `java.util.logging.manager` system property.

*Example Configuration*

```
<build>
  <plugins>
    <plugin>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>${surefire-plugin.version}</version>
      <configuration>
        <systemPropertyVariables>

<java.util.logging.manager>org.jboss.logmanager.LogManager</java.ut
il.logging.manager> ①
          <quarkus.log.level>DEBUG</quarkus.log.level> ②
          <maven.home>${maven.home}</maven.home>
        </systemPropertyVariables>
      </configuration>
    </plugin>
  </plugins>
</build>
```

① Make sure the `org.jboss.logmanager.LogManager` is used.


② Enable debug logging for all logging categories.

If you are using Gradle, add this to your `build.gradle`:

```
test {
    systemProperty "java.util.logging.manager",
    "org.jboss.logmanager.LogManager"
}
```

See also: [Running @QuarkusTest from an IDE](#)

## Logging configuration reference

 Configuration property fixed at build time - All other configuration properties are overridable at runtime

Configuration property	Type	Default
------------------------	------	---------

<code>quarkus.log.level</code>  The log level of the root category, which is used as the default log level for all categories. In addition to the standard JDK log level JBoss Logging also adds the following: <code>org.jboss.logmanager.Level#FATAL</code> <code>org.jboss.logmanager.Level#ERROR</code> <code>org.jboss.logmanager.Level#WARN</code> <code>org.jboss.logmanager.Level#INFO</code> <code>org.jboss.logmanager.Level#DEBUG</code> <code>org.jboss.logmanager.Level#TRACE</code>	Level	INFO
<code>quarkus.log.min-level</code>  The default minimum log level	Level	INFO
<b>Console logging</b>	<b>Type</b>	<b>Default</b>
<code>quarkus.log.console.enable</code>  If console logging should be enabled	boolean	true
<code>quarkus.log.console.format</code>  The log format. Note that this value will be ignored if an extension is present that takes control of console formatting (e.g. an XML or JSON-format extension).	string	<code>%d{yyy y-MM- dd HH:mm: ss,SSS } %-5p [%c{3. }] (%t) %s%n</code>
<code>quarkus.log.console.level</code>  The console log level.	Level	ALL
<code>quarkus.log.console.color</code>  If the console logging should be in color. If undefined quarkus takes best guess based on operating system and environment. Note that this value will be ignored if an extension is present that takes control of console formatting (e.g. an XML or JSON-format extension).	boolean	
<code>quarkus.log.console.darken</code>  Specify how much the colors should be darkened. Note that this value will be ignored if an extension is present that takes control of console formatting (e.g. an XML or JSON-format extension).	int	0

<code>quarkus.log.console.async</code>		
Indicates whether to log asynchronously	boolean	false
<code>quarkus.log.console.async.queue-length</code>		
The queue length to use before flushing writing	int	512
<code>quarkus.log.console.async.overflow</code>		
Determine whether to block the publisher (rather than drop the message) when the queue is full	block, discarded	block
<b>File logging</b>	<b>Type</b>	<b>Default</b>
<code>quarkus.log.file.enable</code>		
If file logging should be enabled	boolean	false
<code>quarkus.log.file.format</code>		
The log format	string	%d{yyy y-MM- dd HH:mm: ss,SSS } %h %N[%i] %-5p [%c{3. }] (%t) %%e%n
<code>quarkus.log.file.level</code>		
The level of logs to be written into the file.	Level	ALL
<code>quarkus.log.file.path</code>		
The name of the file in which logs will be written.	File	quarkus.log
<code>quarkus.log.file.async</code>		
Indicates whether to log asynchronously	boolean	false
<code>quarkus.log.file.async.queue-length</code>		
The queue length to use before flushing writing	int	512

<code>quarkus.log.file.async.overflow</code>	block, discarded	block
Determine whether to block the publisher (rather than drop the message) when the queue is full		
<code>quarkus.log.file.rotation.max-file-size</code>	Memory Size 	
The maximum file size of the log file after which a rotation is executed.		
<code>quarkus.log.file.rotation.max-backup-index</code>	int	1
The maximum number of backups to keep.		
<code>quarkus.log.file.rotation.file-suffix</code>	string	
File handler rotation file suffix. Example fileSuffix: .yyyy-MM-dd		
<code>quarkus.log.file.rotation.rotate-on-boot</code>	boolean	true
Indicates whether to rotate log files on server initialization. You need to either set a <code>max-file-size</code> or configure a <code>file-suffix</code> for it to work.		
<b>Syslog logging</b>	<b>Type</b>	<b>Default</b>
<code>quarkus.log.syslog.enable</code>	boolean	false
If syslog logging should be enabled		
<code>quarkus.log.syslog.endpoint</code>	host:port	localhost:514
The IP address and port of the syslog server		
<code>quarkus.log.syslog.app-name</code>	string	
The app name used when formatting the message in RFC5424 format		
<code>quarkus.log.syslog.hostname</code>	string	
The name of the host the messages are being sent from		

<p><code>quarkus.log.syslog.facility</code></p> <p>Sets the facility used when calculating the priority of the message as defined by RFC-5424 and RFC-3164</p>	<p>kernel, user- level, mail- system, system- daemo- ns, securi- ty, syslog d, line- printe- r, networ- k-news, uucp, clock- daemon, securi- ty2, ftp- daemon, ntp, log- audit, log- alert, clock- daemon 2, local- use-0, local- use-1, local- use-2, local- use-3, local- use-4, local- use-5, local- use-6, local- use-7</p>	<p>user- level</p>
--	---	------------------------

<code>quarkus.log.syslog.syslog-type</code>		
Set the <code>SyslogType</code> <code>syslog type</code> this handler should use to format the message sent	<code>rfc5424, rfc3164</code>	<code>rfc5424</code>
<code>quarkus.log.syslog.protocol</code>		
Sets the protocol used to connect to the syslog server	<code>tcp, udp, ssl-tcp</code>	<code>tcp</code>
<code>quarkus.log.syslog.use-counting-framing</code>		
Set to <code>true</code> if the message being sent should be prefixed with the size of the message	<code>boolean</code>	<code>false</code>
<code>quarkus.log.syslog.truncate</code>		
Set to <code>true</code> if the message should be truncated	<code>boolean</code>	<code>true</code>
<code>quarkus.log.syslog.block-on-reconnect</code>		
Enables or disables blocking when attempting to reconnect a <code>org.jboss.logmanager.handlers.SyslogHandler.Protocol#TCP</code> or <code>org.jboss.logmanager.handlers.SyslogHandler.Protocol#SSL_TCP</code> protocol	<code>boolean</code>	<code>false</code>
<code>quarkus.log.syslog.format</code>		
The log message format	<code>string</code>	<code>%d{yyy y-MM-dd HH:mm:ss,SSS} %-5p [%c{3.}] (%t) %s%n</code>
<code>quarkus.log.syslog.level</code>		
The log level specifying, which message levels will be logged by syslog logger	<code>Level</code>	<code>ALL</code>
<code>quarkus.log.syslog.async</code>		
Indicates whether to log asynchronously	<code>boolean</code>	<code>false</code>
<code>quarkus.log.syslog.async.queue-length</code>		
The queue length to use before flushing writing	<code>int</code>	<code>512</code>

<code>quarkus.log.syslog.async.overflow</code>		
Determine whether to block the publisher (rather than drop the message) when the queue is full	<code>block, discard</code>	<code>block</code>
<b>Logging categories</b>	<b>Type</b>	<b>Default</b>
<code>quarkus.log.category."categories".level</code>	InheritableLevel	<code>inherit</code>
The log level level for this category		
<code>quarkus.log.category."categories".handlers</code>	list of string	
The names of the handlers to link to this category.		
<code>quarkus.log.category."categories".use-parent-handlers</code>	boolean	<code>true</code>
Specify whether or not this logger should send its output to its parent Logger		
<b>Console handlers</b>	<b>Type</b>	<b>Default</b>
<code>quarkus.log.handler.console."console-handlers".enable</code>	boolean	<code>true</code>
If console logging should be enabled		
<code>quarkus.log.handler.console."console-handlers".format</code>	string	<code>%d{yyy y-MM- dd HH:mm: ss,SSS } %-5p [%c{3. }] (%t) %s%n</code>
The log format. Note that this value will be ignored if an extension is present that takes control of console formatting (e.g. an XML or JSON-format extension).		
<code>quarkus.log.handler.console."console-handlers".level</code>	Level	<code>ALL</code>
The console log level.		
<code>quarkus.log.handler.console."console-handlers".color</code>	boolean	
If the console logging should be in color. If undefined quarkus takes best guess based on operating system and environment. Note that this value will be ignored if an extension is present that takes control of console formatting (e.g. an XML or JSON-format extension).		

<code>quarkus.log.handler.console."console-handlers".darken</code> Specify how much the colors should be darkened. Note that this value will be ignored if an extension is present that takes control of console formatting (e.g. an XML or JSON-format extension).	int	0
<code>quarkus.log.handler.console."console-handlers".async</code> Indicates whether to log asynchronously	boolean	false
<code>quarkus.log.handler.console."console-handlers".async.queue-length</code> The queue length to use before flushing writing	int	512
<code>quarkus.log.handler.console."console-handlers".async.overflow</code> Determine whether to block the publisher (rather than drop the message) when the queue is full	block, discard	block
<b>File handlers</b>	<b>Type</b>	<b>Default</b>
<code>quarkus.log.handler.file."file-handlers".enable</code> If file logging should be enabled	boolean	false
<code>quarkus.log.handler.file."file-handlers".format</code> The log format	string	%d{yyy y-MM- dd HH:mm: ss,SSS } %h %N[%i] %-5p [%c{3. }] (%t) %s%n
<code>quarkus.log.handler.file."file-handlers".level</code> The level of logs to be written into the file.	Level	ALL
<code>quarkus.log.handler.file."file-handlers".path</code> The name of the file in which logs will be written.	File	quarkus.log

<code>quarkus.log.handler.file."file-handlers".async</code> Indicates whether to log asynchronously	boolean	<code>false</code>
<code>quarkus.log.handler.file."file-handlers".async.queue-length</code> The queue length to use before flushing writing	int	<code>512</code>
<code>quarkus.log.handler.file."file-handlers".async.overflow</code> Determine whether to block the publisher (rather than drop the message) when the queue is full	block, discard	<code>block</code>
<code>quarkus.log.handler.file."file-handlers".rotation.max-file-size</code> The maximum file size of the log file after which a rotation is executed.	Memory Size 	
<code>quarkus.log.handler.file."file-handlers".rotation.max-backup-index</code> The maximum number of backups to keep.	int	<code>1</code>
<code>quarkus.log.handler.file."file-handlers".rotation.file-suffix</code> File handler rotation file suffix. Example fileSuffix: .yyyy-MM-dd	string	
<code>quarkus.log.handler.file."file-handlers".rotation.rotate-on-boot</code> Indicates whether to rotate log files on server initialization. You need to either set a <code>max-file-size</code> or configure a <code>file-suffix</code> for it to work.	boolean	<code>true</code>
<b>Syslog handlers</b>	<b>Type</b>	<b>Default</b>
<code>quarkus.log.handler.syslog."syslog-handlers".enable</code> If syslog logging should be enabled	boolean	<code>false</code>
<code>quarkus.log.handler.syslog."syslog-handlers".endpoint</code> The IP address and port of the syslog server	host:port	<code>localhost:514</code>
<code>quarkus.log.handler.syslog."syslog-handlers".app-name</code> The app name used when formatting the message in RFC5424 format	string	

<code>quarkus.log.handler.syslog."syslog-handlers".hostname</code> The name of the host the messages are being sent from	string	
---	--------	--

<p><code>quarkus.log.handler.syslog."syslog-handlers".facility</code></p> <p>Sets the facility used when calculating the priority of the message as defined by RFC-5424 and RFC-3164</p>	<p>kernel, user- level, mail- system, system- daemo- ns, securi- ty, syslog d, line- printe- r, networ- k-news, uucp, clock- daemon, securi- ty2, ftp- daemon, ntp, log- audit, log- alert, clock- daemon 2, local- use-0, local- use-1, local- use-2, local- use-3, local- use-4, local- use-5, local- use-6, local- use-7</p>	<p>user- level</p>
--	---	------------------------

<code>quarkus.log.handler.syslog."syslog-handlers".syslog-type</code> Set the <code>LogLevel</code> <code>syslog</code> type this handler should use to format the message sent	<code>rfc5424, rfc3164</code>	<code>rfc5424</code>
<code>quarkus.log.handler.syslog."syslog-handlers".protocol</code> Sets the protocol used to connect to the syslog server	<code>tcp, udp, ssl-tcp</code>	<code>tcp</code>
<code>quarkus.log.handler.syslog."syslog-handlers".use-counting-framing</code> Set to <code>true</code> if the message being sent should be prefixed with the size of the message	boolean	<code>false</code>
<code>quarkus.log.handler.syslog."syslog-handlers".truncate</code> Set to <code>true</code> if the message should be truncated	boolean	<code>true</code>
<code>quarkus.log.handler.syslog."syslog-handlers".block-on-reconnect</code> Enables or disables blocking when attempting to reconnect a <code>org.jboss.logmanager.handlers.SyslogHandler.Protocol#TCP</code> or <code>org.jboss.logmanager.handlers.SyslogHandler.Protocol#SSL_TCP</code> protocol	boolean	<code>false</code>
<code>quarkus.log.handler.syslog."syslog-handlers".format</code> The log message format	string	<code>%d{yyy-MM-dd HH:mm:ss,SSS} %-5p [%c{3.}] (%t) %s%n</code>
<code>quarkus.log.handler.syslog."syslog-handlers".level</code> The log level specifying, which message levels will be logged by syslog logger	Level	<code>ALL</code>
<code>quarkus.log.handler.syslog."syslog-handlers".async</code> Indicates whether to log asynchronously	boolean	<code>false</code>

<code>quarkus.log.handler.syslog."syslog-handlers".async.queue-length</code> The queue length to use before flushing writing	int	512
<code>quarkus.log.handler.syslog."syslog-handlers".async.overflow</code> Determine whether to block the publisher (rather than drop the message) when the queue is full	block, discard	block
<b>Log cleanup filters - internal use</b>	<b>Type</b>	<b>Default</b>
<code>quarkus.log.filter."filters".if-starts-with</code> The message starts to match	list of string	inherit
<code>quarkus.log.filter."filters".target-level</code> The new log level for the filtered message, defaults to DEBUG	Level	DEBUG



#### About the MemorySize format

A size configuration option recognises string in this format (shown as a regular expression): `[0-9]+[KkMmGgTtPpEeZzYy ]?`. If no suffix is given, assume bytes.

[1] Some extensions may define customized default log levels for certain categories, in order to reduce log noise by default. Setting the log level in configuration will override any extension-defined log levels.

[2] By default the configured category gets the same handlers attached as the one on the root logger.

[3] Format sequences which examine caller information may affect performance