Quarkus - Configuring Logging

This guide explains logging and how to configure it.

Run Time Configuration

Run time configuration of logging is done through the normal `application.properties` file.

▤ Configuration property fixed at build time - ⚙ Configuration property overridable at runtime

Configuration property	Type	Default
⚙ `quarkus.log.level` The default log level	Level	
⚙ `quarkus.log.min-level` The default minimum log level	Level	`INFO`
Logging categories	Type	Default
⚙ `quarkus.log.category."categories".level` The log level level for this category	string	`inherit`
⚙ `quarkus.log.category."categories".handlers` The names of the handlers to link to this category.	list of string	
⚙ `quarkus.log.category."categories".use-parent-handlers` Specify whether or not this logger should send its output to its parent Logger	boolean	`true`
Console handlers	Type	Default
⚙ `quarkus.log.handler.console."console-handlers".enable` If console logging should be enabled	boolean	`true`

	Type	Default
⚙ `quarkus.log.handler.console."console-handlers".format` 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	`%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c{3.}] (%t) %s%e%n`
⚙ `quarkus.log.handler.console."console-handlers".level` The console log level.	`Level`	`ALL`
⚙ `quarkus.log.handler.console."console-handlers".color` 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	
⚙ `quarkus.log.handler.console."console-handlers".darken` 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`
⚙ `quarkus.log.handler.console."console-handlers".async` Indicates whether to log asynchronously	boolean	`false`
⚙ `quarkus.log.handler.console."console-handlers".async.queue-length` The queue length to use before flushing writing	int	`512`
⚙ `quarkus.log.handler.console."console-handlers".async.overflow` Determine whether to block the publisher (rather than drop the message) when the queue is full	`block`, `discard`	`block`
File handlers	**Type**	**Default**
⚙ `quarkus.log.handler.file."file-handlers".enable` If file logging should be enabled	boolean	`false`

⚙ `quarkus.log.handler.file."file-handlers".format` The log format	string	`%d{yyyy-MM-dd HH:mm:ss,SSS} %h %N[%i] %-5p [%c{3.}] (%t) %s%e%n`
⚙ `quarkus.log.handler.file."file-handlers".level` The level of logs to be written into the file.	Level	`ALL`
⚙ `quarkus.log.handler.file."file-handlers".path` The name of the file in which logs will be written.	File	`quarkus.log`
⚙ `quarkus.log.handler.file."file-handlers".async` Indicates whether to log asynchronously	boolean	`false`
⚙ `quarkus.log.handler.file."file-handlers".async.queue-length` The queue length to use before flushing writing	int	`512`
⚙ `quarkus.log.handler.file."file-handlers".async.overflow` Determine whether to block the publisher (rather than drop the message) when the queue is full	`block`, `discard`	`block`
⚙ `quarkus.log.handler.file."file-handlers".rotation.max-file-size` The maximum file size of the log file after which a rotation is executed.	Memory Size ❓	
⚙ `quarkus.log.handler.file."file-handlers".rotation.max-backup-index` The maximum number of backups to keep.	int	`1`

	Type	Default
⚙ `quarkus.log.handler.file."file-handlers".rotation.file-suffix` File handler rotation file suffix. Example fileSuffix: .yyyy-MM-dd	string	
⚙ `quarkus.log.handler.file."file-handlers".rotation.rotate-on-boot` Indicates whether to rotate log files on server initialization.	boolean	`true`
Syslog handlers	**Type**	**Default**
⚙ `quarkus.log.handler.syslog."syslog-handlers".enable` If syslog logging should be enabled	boolean	`false`
⚙ `quarkus.log.handler.syslog."syslog-handlers".endpoint` The IP address and port of the syslog server	host:port	`localhost:514`
⚙ `quarkus.log.handler.syslog."syslog-handlers".app-name` The app name used when formatting the message in RFC5424 format	string	
⚙ `quarkus.log.handler.syslog."syslog-handlers".hostname` The name of the host the messages are being sent from	string	

| `quarkus.log.handler.syslog."syslog-handlers".facility`

Sets the facility used when calculating the priority of the message as defined by RFC-5424 and RFC-3164 | `kernel, user-level, mail-system, system-daemons, security, syslogd, line-printer, network-news, uucp, clock-daemon, security2, ftp-daemon, ntp, log-audit, log-alert, clock-daemon2, local-use-0, local-use-1, local-use-2, local-use-3, local-use-4, local-use-5, local-use-6, local-use-7` | `user-level` |

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

	Type	Default
⚙ `quarkus.log.handler.syslog."syslog-handlers".async.queue-length` The queue length to use before flushing writing	int	512
⚙ `quarkus.log.handler.syslog."syslog-handlers".async.overflow` Determine whether to block the publisher (rather than drop the message) when the queue is full	`block`, `discard`	`block`
Console logging	**Type**	**Default**
⚙ `quarkus.log.console.enable` If console logging should be enabled	boolean	`true`
⚙ `quarkus.log.console.format` 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	`%d{yyyy-MM-dd HH:mm:ss,SSS} %-5p [%c{3.}] (%t) %s%e%n`
⚙ `quarkus.log.console.level` The console log level.	`Level`	`ALL`
⚙ `quarkus.log.console.color` 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	
⚙ `quarkus.log.console.darken` 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`
⚙ `quarkus.log.console.async` Indicates whether to log asynchronously	boolean	`false`

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

	Type	Default
⚙️ `quarkus.log.file.rotation.max-file-size` The maximum file size of the log file after which a rotation is executed.	Memory Size ❓	
⚙️ `quarkus.log.file.rotation.max-backup-index` The maximum number of backups to keep.	int	1
⚙️ `quarkus.log.file.rotation.file-suffix` File handler rotation file suffix. Example fileSuffix: .yyyy-MM-dd	string	
⚙️ `quarkus.log.file.rotation.rotate-on-boot` Indicates whether to rotate log files on server initialization.	boolean	`true`
Syslog logging	**Type**	**Default**
⚙️ `quarkus.log.syslog.enable` If syslog logging should be enabled	boolean	`false`
⚙️ `quarkus.log.syslog.endpoint` The IP address and port of the syslog server	host:port	`localhost:514`
⚙️ `quarkus.log.syslog.app-name` The app name used when formatting the message in RFC5424 format	string	
⚙️ `quarkus.log.syslog.hostname` The name of the host the messages are being sent from	string	

⚙ `quarkus.log.syslog.facility` Sets the facility used when calculating the priority of the message as defined by RFC-5424 and RFC-3164	`kernel, user-level, mail-system, system-daemons, security, syslogd, line-printer, network-news, uucp, clock-daemon, security2, ftp-daemon, ntp, log-audit, log-alert, clock-daemon2, local-use-0, local-use-1, local-use-2, local-use-3, local-use-4, local-use-5, local-use-6, local-use-7`	`user-level`

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

⚙ `quarkus.log.syslog.async.queue-length` The queue length to use before flushing writing	int	512
⚙ `quarkus.log.syslog.async.overflow` Determine whether to block the publisher (rather than drop the message) when the queue is full	`block`, `discard`	`block`
Log cleanup filters - internal use	**Type**	**Default**
⚙ `quarkus.log.filter."filters".if-starts-with` The message starts to match	list of string	`inherit`

ℹ *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.

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
`quarkus.log.category."<category-name>".level`	`INFO` [1]	The level to use to configure the category named `<category-name>`. The quotes are necessary.
`quarkus.log.category."<category-name>".useParentHandlers`	`true`	Specify whether or not this logger should send its output to its parent logger.
`quarkus.log.category."<category-name>".handlers=[<handler>]`	`empty` [2]	The names of the handlers that you want to attach to a specific category.

ℹ 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
`quarkus.log.level`	`INFO`	The default minimum log level for every log category.

Format String

The logging format string supports the following symbols:

Symbol	Summary	Description
`%%`	%	Renders a simple % character.
`%c`	Category	Renders the category name.
`%C`	Source class	Renders the source class name. [3]
`%d{xxx}`	Date	Renders a date with the given date format string, which uses the syntax defined by `java.text.SimpleDateFormat`.
`%e`	Exception	Renders the thrown exception, if any.
`%F`	Source file	Renders the source file name.[3]
`%h`	Host name	Renders the system simple host name.
`%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.[3]
`%L`	Source line	Renders the source line number.[3]
`%m`	Full Message	Renders the log message plus exception (if any).
`%M`	Source method	Renders the source method name.[3]
`%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.

Symbol	Summary	Description
%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

```xml
<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 - ⚙ Configuration property overridable at runtime

Configuration property	Type	Default
⚙ `quarkus.log.console.json` Determine whether to enable the JSON console formatting extension, which disables "normal" console formatting.	boolean	`true`
⚙ `quarkus.log.console.json.pretty-print` Enable "pretty printing" of the JSON record. Note that some JSON parsers will fail to read pretty printed output.	boolean	`false`
⚙ `quarkus.log.console.json.date-format` The date format to use. The special string "default" indicates that the default format should be used.	string	`default`
⚙ `quarkus.log.console.json.record-delimiter` The special end-of-record delimiter to be used. By default, no delimiter is used.	string	
⚙ `quarkus.log.console.json.zone-id` The zone ID to use. The special string "default" indicates that the default zone should be used.	string	`default`

⚙ `quarkus.log.console.json.exception-output-type` The exception output type to specify.	`detailed`, `formatted`, `detailed-and-formatted`	`detailed`
⚙ `quarkus.log.console.json.print-details` 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.

> ℹ 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
```

> ℹ 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%e%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%e%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.

[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