RTI Distributed Logger API

Generated by Doxygen 1.9.3

1 RTI Distributed Logger API 1

1.1 General Usage. . . . 1

1.2 Feedback and Support for this Release. . . . 2

2 Module Index 3

2.1 Modules . . . 3

3 Class Index 5

3.1 Class List . . . 5

4 Module Documentation 7

4.1 RTI Distributed Logger API . . . 7

4.1.1 Detailed Description . . . 7

5 Class Documentation 9

5.1 rti::dist_logger::DistLogger Class Reference . . . 9

5.1.1 Detailed Description . . . 11

5.1.2 Member Function Documentation . . . 11

5.1.2.1 get_instance() . . . 11

5.1.2.2 finalize() . . . 11

5.1.2.3 set_options() . . . 13

5.1.2.4 set_filter_level() . . . 13

5.1.2.5 set_print_format() . . . 14

5.1.2.6 set_verbosity() . . . 15

5.1.2.7 log_message() [1/4] . . . 15

5.1.2.8 log_message() [2/4] . . . 15

5.1.2.9 log_message() [3/4] . . . 16

5.1.2.10 log_message() [4/4] . . . 16

5.1.2.11 fatal() [1/2] . . . 17

5.1.2.12 fatal() [2/2] . . . 17

5.1.2.13 severe() [1/2] . . . 17

5.1.2.14 severe() [2/2] . . . 18

5.1.2.15 error() [1/2] . . . 18

5.1.2.16 error() [2/2] . . . 18

5.1.2.17 warning() [1/2] . . . 19

5.1.2.18 warning() [2/2] . . . 19

5.1.2.19 notice() [1/2] . . . 19

5.1.2.20 notice() [2/2] . . . 20

5.1.2.21 info() [1/2] . . . 20

5.1.2.22 info() [2/2] . . . 20

5.1.2.23 debug() [1/2] . . . 21

5.1.2.24 debug() [2/2] . . . 21
5.1.2.25 trace() [1/2] . . . 21
5.1.2.26 trace() [2/2] . . . 22
5.1.2.27 log() [1/2] . . . 22
5.1.2.28 log() [2/2] . . . 22
5.1.2.29 get_fatal_log_level() . . . 24
5.1.2.30 get_error_log_level() . . . 24
5.1.2.31 get_warning_log_level() . . . 25
5.1.2.32 get_notice_log_level() . . . 25
5.1.2.33 get_info_log_level() . . . 25
5.1.2.34 get_debug_log_level() . . . 26
5.2 rti::dist_logger::DistLoggerOptions Class Reference . . . 26
5.2.1 Detailed Description . . . 27
5.2.2 Constructor & Destructor Documentation . . . 27
5.2.2.1 DistLoggerOptions() . . . 28
5.2.3 Member Function Documentation . . . 28
5.2.3.1 domain_participant() [1/2] . . . 28
5.2.3.2 domain_participant() [2/2] . . . 28
5.2.3.3 remote_administration_enabled() [1/2] . . . 28
5.2.3.4 remote_administration_enabled() [2/2] . . . 29
5.2.3.5 filter_level() [1/2] . . . 29
5.2.3.6 filter_level() [2/2] . . . 29
5.2.3.7 domain_id() [1/2] . . . 30
5.2.3.8 domain_id() [2/2] . . . 30
5.2.3.9 qos_library() [1/2] . . . 30
5.2.3.10 qos_library() [2/2] . . . 31
5.2.3.11 qos_profile() [1/2] . . . 31
5.2.3.12 qos_profile() [2/2] . . . 31
5.2.3.13 queue_size() [1/2] . . . 32
5.2.3.14 queue_size() [2/2] . . . 32
5.2.3.15 thread_settings() [1/2] . . . 32
5.2.3.16 thread_settings() [2/2] . . . 33
5.2.3.17 application_kind() [1/2] . . . 33
5.2.3.18 application_kind() [2/2] . . . 33
5.2.3.19 echo_to_stdout() [1/2] . . . 34
5.2.3.20 echo_to_stdout() [2/2] . . . 34
5.2.3.21 log_infrastructure_messages() [1/2] . . . 34
5.2.3.22 log_infrastructure_messages() [2/2] . . . 35
5.2.3.23 operator==() . . . 35

5.2.3.24 operator"!=() 35
5.2.4 Friends And Related Function Documentation 36
5.2.4.1 swap 36

Index **37**

Chapter 1

RTI Distributed Logger API

Real-Time Innovations, Inc.

RTI Distributed Logger is an API that allows publishing log messages to a DDS Topic. This will allow your distributed application to send log messages that can be monitored using RTI Monitor or RTI Admin Console.

Generally, this API is used by an external logging infrastructure, such as
RTI Logger, to distribute the logging messages using RTI Connext DDS. It is also possible to use the API directly, creating your own logging infrastructure.

1.1 General Usage.

We strongly recommend that you use one of our shipped examples as a guideline to configure and use the RTI Distributed Logger.

In general, the steps to customize and use the RTI Distributed Logger are the followings:

1. Customize your RTI Distributed Logger using **rti::dist_logger::DistLoggerOptions** (p. 26). These options must be assigned to the RTI Distributed Logger singleton **before** creating it for the first time.
2. Assign the newly created options to the RTI Distributed Logger using **rti::dist_logger::DistLogger::set_options** (p. 13).
3. Acquire an **rti::dist_logger::DistLogger** (p. 9) instance using **rti::dist_logger::DistLogger::get_instance** (p. 11). Make sure the instance was not created before or the option setup will have no effect.
4. Finally, send your own log messages using RTI Distributed Logger logging functions: **rti::dist_logger::Dist↩ Logger::error** (p. 18), **rti::dist_logger::DistLogger::fatal** (p. 17), ... The internal RTI Logger's log messages will be sent automatically once the RTI Distributed Logger is created (no additional configuration required).

For details on the logging and filter levels used to by the RTI Distributed Logger, refer to the function **rti::dist_logger↩ ::DistLogger::set_filter_level** (p. 13) and the main IDL file: <RTI Connext Installation Folder>/resource/idl/distlog.idl.

You can learn how to use RTI Distributed Logger through
RTI Distributed Logger, in the Core Libraries User's Manual, or our shipped examples.

1.2 Feedback and Support for this Release.

For more information, visit our knowledge base, accessible from `http://community.rti.com/` to see sample code, general information on RTI Connext, performance information, troubleshooting tips, and technical details.

By its very nature, the knowledge base is continuously evolving and improving. We hope that you will find it helpful. If there are questions that you would like to see addressed or comments you would like to share, please send email to `support@rti.com`. We can only guarantee a response for customers with a current maintenance contract or subscription. To purchase a maintenance contract or subscription, contact your local RTI representative (see `http://www.rti.com/company/contact.html`), send an email request to `sales@rti.com`, or call +1 (408) 990-7400.

Please do not hesitate to contact RTI with questions or comments about this release. We welcome any input on how to improve RTI Distributed Logger and RTI Connext to suit your needs.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

RTI Distributed Logger API 7

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

rti::dist_logger::DistLogger
This class contains RTI Distributed Logger singleton and the methods to configure and use it 9
rti::dist_logger::DistLoggerOptions
This class contains RTI Distributed Logger configurable options 26

Chapter 4

Module Documentation

4.1 RTI Distributed Logger API

Main APIs to use the Modern C++ RTI Distributed Logger.

Classes

- class **rti::dist_logger::DistLogger**

  This class contains RTI Distributed Logger singleton and the methods to configure and use it.
- class **rti::dist_logger::DistLoggerOptions**

  This class contains RTI Distributed Logger configurable options.

4.1.1 Detailed Description

Main APIs to use the Modern C++ RTI Distributed Logger.

Chapter 5

Class Documentation

5.1 rti::dist_logger::DistLogger Class Reference

This class contains RTI Distributed Logger singleton and the methods to configure and use it.

`#include <DistLogger.hpp>`

Inherits dds::core::Reference< DistLoggerImpl >.

Public Member Functions

- void **set_filter_level** (int new_level)

  Sets the filter level for the singleton.
- void **set_print_format** (const rti::config::PrintFormat &log_format)

  Sets PrintFormat of RTI Logger to the provided value.
- void **set_verbosity** (const rti::config::LogCategory &category, const rti::config::Verbosity &verbosity)

  Updates the rti::config::LogCategory and the rti::config::Verbosity in RTI Logger to the provided value.
- void **log_message** (int log_level, const char *message, const char *category)

  Logs a message at the specified level and with the specified category.
- void **log_message** (int log_level, const std::string &message, const std::string &category)

  Logs a message at the specified level and with the specified category.
- void **log_message** (int log_level, const char *message, const char *category, const dds::core::Time &timestamp)

  *Performs the same function as **log_message(int, const std::string&, const std::string&)** (p. 15) except that it also allows specification of the message timestamp.*
- void **log_message** (int log_level, const std::string &message, const std::string &category, const dds::core::Time &timestamp)

  *Performs the same function as **log_message(int, const std::string&, const std::string&)** (p. 15) except that it also allows specification of the message timestamp.*
- void **fatal** (const char *message)

  Logs a fatal message.
- void **fatal** (const std::string &message)

  Logs a fatal message.

- void **severe** (const char *message)

  Logs a severe message.
- void **severe** (const std::string &message)

  Logs a severe message.
- void **error** (const char *message)

  Logs a error message.
- void **error** (const std::string &message)

  Logs a error message.
- void **warning** (const char *message)

  Logs a warning message.
- void **warning** (const std::string &message)

  Logs a warning message.
- void **notice** (const char *message)

  Logs a notice message.
- void **notice** (const std::string &message)

  Logs a notice message.
- void **info** (const char *message)

  Logs a info message.
- void **info** (const std::string &message)

  Logs a info message.
- void **debug** (const char *message)

  Logs a debug message.
- void **debug** (const std::string &message)

  Logs a debug message.
- void **trace** (const char *message)

  Logs a trace message.
- void **trace** (const std::string &message)

  Logs a trace message.
- void **log** (int log_level, const char *message)

  Logs a message.
- void **log** (int log_level, const std::string &message)

  Logs a message.
- int **get_fatal_log_level** () const

  Gets Fatal log level.
- int **get_error_log_level** () const

  Gets Error log level.
- int **get_warning_log_level** () const

  Gets Warning log level.
- int **get_notice_log_level** () const

  Gets Notice log level.
- int **get_info_log_level** () const

  Gets Info log level.
- int **get_debug_log_level** () const

  Gets Debug log level.

Static Public Member Functions

- static **DistLogger** & **get_instance** ()

  *Provides access to the **DistLogger** (p. 9) singleton.*
- static void **finalize** ()

  *Finalizes the **DistLogger** (p. 9) singleton.*
- static void **set_options** (const **rti::dist_logger::DistLoggerOptions** &options)

  Assigns configurable options to RTI Distributed Logger before its creation.

5.1.1 Detailed Description

This class contains RTI Distributed Logger singleton and the methods to configure and use it.

This class is a **singleton**, so only one copy will be created. Use the **DistLoggerOptions** (p. 26) class to customize the singleton **BEFORE** creating it.

5.1.2 Member Function Documentation

5.1.2.1 get_instance()

```
static DistLogger & rti::dist_logger::DistLogger::get_instance ( ) [static]
```

Provides access to the **DistLogger** (p. 9) singleton.

This method provides access to the singleton instance of the **DistLogger** (p. 9). If the instance does not exist, it will be created. Note that no change in options is allowed after the singleton is created.

Returns

The singleton instance of the **DistLogger** (p. 9).

Exceptions

dds::core::Error	on failure.

5.1.2.2 finalize()

```
static void rti::dist_logger::DistLogger::finalize ( ) [static]
```

Finalizes the **DistLogger** (p. 9) singleton.

This method must be called before the application ends.

Exceptions

dds::core::Exception	on failure.

5.1.2.3 set_options()

```
static void rti::dist_logger::DistLogger::set_options (
            const  rti::dist_logger::DistLoggerOptions & options )  [static]
```

Assigns configurable options to RTI Distributed Logger before its creation.

Assigns the values in the `options` to the **DistLogger** (p. 9) singleton **before** its creation. If the singleton was already created, this method has no effect. If a Domain Participant is provided ih the `options`, the Distributed Logger instance must be finalized before finalizing the Domain Participant Factory.

Parameters

options	The new options to be assigned to the **DistLogger** (p. 9) singleton.

Exceptions

dds::core::IllegalOperationError	when the **DistLogger** (p. 9) singleton has already been created or dds::core::Exception on failure setting the options.

5.1.2.4 set_filter_level()

```
void rti::dist_logger::DistLogger::set_filter_level (
            int new_level )
```

Sets the filter level for the singleton.

Filtering controls what type of log messages are sent. This enables filtering on the *sending* side (in the application using Distributed Logger). You can also enable filtering on the *receiving* side (such as in a tool that receives log messages, like RTI Admin Console or RTI Monitor).

Supported Filter Levels: See the table below. The default value is RTI_DL_INFO_LEVEL. The default value is RTI_↩ DL_INFO_LEVEL.

Or see the IDL file provided with Distributed Logger (file: <RTI Connext Installation Folder>/resource/idl/distlog.idl).In the IDL file, there are language-specific bindings. Prepend the appropriate binding to the filter level shown in the IDL. For example, one of the filter levels in the IDL is INFO_LEVEL. To refer to this filter level in C++, use RTI_DL_INFO_LEVEL.

Log messages up to and including the specified filter level will pass through the filter and be sent on the network. For example, the WARNING level allows log messages up to and including the WARNING level (FATAL, SEVERE, ERROR, and WARNING) to be sent; any other log messages with numerically higher filter levels (INFO, DEBUG, and TRACE) will be filtered out and not sent.

Parameters

new_level	The new filter level.

Exceptions

dds::core::Exception	on failure.

The following table shows how RTI Connext DDS log levels map to RTI Distributed Logger log filter levels.

Not all RTI Distributed Logger levels appear in this table because there is not a one-to-one mapping between RTI Connext DDS and RTI Distributed Logger levels (there are more levels in RTI Distributed Logger). You can obtain the filter level using a constant defined in the Distributed Logger IDL file or a getter function in the Distributed Logger API.

Table 5.7 *Filter Levels*

Connext DDS API constant	Distributed Logger IDL constant	Distributed Logger API getter
rti::config::LogLevel::FATAL_ERROR	RTI_DL_FATAL_LEVEL	**get_fatal_log_level** (p. 24)
rti::config::LogLevel::EXCEPTION	RTI_DL_ERROR_LEVEL	**get_error_log_level** (p. 24)
rti::config::LogLevel::WARNING	RTI_DL_WARNING_LEVEL	**get_warning_log_level** (p. 24)
rti::config::LogLevel::STATUS_LOCAL	RTI_DL_NOTICE_LEVEL	**get_notice_log_level** (p. 25)
rti::config::LogLevel::STATUS_REMOTE	RTI_DL_INFO_LEVEL	**get_info_log_level** (p. 25)
rti::config::LogLevel::STATUS_ALL	RTI_DL_DEBUG_LEVEL	**get_debug_log_level** (p. 25)

5.1.2.5 set_print_format()

```
void rti::dist_logger::DistLogger::set_print_format (
             const rti::config::PrintFormat & log_format )
```

Sets PrintFormat of RTI Logger to the provided value.

This method is mostly a pass-through to the appropriate call in RTI DDS library, but it also has the benefit of keeping the state of RTI Logger and RTI Distributed Logger in sync.

Parameters

log_format	The new PrintFormat.

Exceptions

dds::core::Error	on failure.

5.1.2.6 set_verbosity()

```
void rti::dist_logger::DistLogger::set_verbosity (
            const rti::config::LogCategory & category,
            const rti::config::Verbosity & verbosity )
```

Updates the rti::config::LogCategory and the rti::config::Verbosity in RTI Logger to the provided value.

This method is mostly a pass-through to the appropriate call in RTI DDS library, but it also has the benefit of keeping the state of RTI Logger and RTI Distributed Logger in sync.

Parameters

category	The category for which to set the verbosity.
verbosity	The new verbosity level for the category.

Exceptions

dds::core::Error	on failure.

5.1.2.7 log_message() [1/4]

```
void rti::dist_logger::DistLogger::log_message (
            int log_level,
            const char * message,
            const char * category )
```

Logs a message at the specified level and with the specified category.

Parameters

log_level	The level of the log message.
message	The message to log.
category	The category of the log message.

5.1.2.8 log_message() [2/4]

```
void rti::dist_logger::DistLogger::log_message (
            int log_level,
```

```
            const std::string & message,
            const std::string & category )
```

Logs a message at the specified level and with the specified category.

Parameters

log_level	The level of the log message.
message	The message to log.
category	The category of the log message.

5.1.2.9 log_message() [3/4]

```
void rti::dist_logger::DistLogger::log_message (
            int log_level,
            const char * message,
            const char * category,
            const dds::core::Time & timestamp )
```

Performs the same function as **log_message(int, const std::string&, const std::string&)** (p. 15) except that it also allows specification of the message timestamp.

Parameters

log_level	The level of the log message.
message	The message to log.
category	The category of the log message.
timestamp	The timestamp of the log message.

5.1.2.10 log_message() [4/4]

```
void rti::dist_logger::DistLogger::log_message (
            int log_level,
            const std::string & message,
            const std::string & category,
            const dds::core::Time & timestamp )
```

Performs the same function as **log_message(int, const std::string&, const std::string&)** (p. 15) except that it also allows specification of the message timestamp.

Parameters

log_level	The level of the log message.

Parameters

message	The message to log.
category	The category of the log message.
timestamp	The timestamp of the log message.

5.1.2.11 fatal() [1/2]

```
void rti::dist_logger::DistLogger::fatal (
            const char * message )
```

Logs a fatal message.

The message received is logged using the default category and the RTI_DL_FATAL_LEVEL log level.

Parameters

message	The message to log.

5.1.2.12 fatal() [2/2]

```
void rti::dist_logger::DistLogger::fatal (
            const std::string & message )
```

Logs a fatal message.

The message received is logged using the default category and the RTI_DL_FATAL_LEVEL log level.

Parameters

message	The message to log.

5.1.2.13 severe() [1/2]

```
void rti::dist_logger::DistLogger::severe (
            const char * message )
```

Logs a severe message.

The message received is logged using the default category and the RTI_DL_SEVERE_LEVEL log level.

Parameters

message	The message to log.

5.1.2.14 severe() [2/2]

```
void rti::dist_logger::DistLogger::severe (
            const std::string & message )
```

Logs a severe message.

The message received is logged using the default category and the RTI_DL_SEVERE_LEVEL log level.

Parameters

message	The message to log.

5.1.2.15 error() [1/2]

```
void rti::dist_logger::DistLogger::error (
            const char * message )
```

Logs a error message.

The message received is logged using the default category and the RTI_DL_ERROR_LEVEL log level.

Parameters

message	The message to log.

5.1.2.16 error() [2/2]

```
void rti::dist_logger::DistLogger::error (
            const std::string & message )
```

Logs a error message.

The message received is logged using the default category and the RTI_DL_ERROR_LEVEL log level.

Parameters

message	The message to log.

5.1.2.17 warning() [1/2]

```
void rti::dist_logger::DistLogger::warning (
            const char * message )
```

Logs a warning message.

The message received is logged using the default category and the RTI_DL_WARNING_LEVEL log level.

Parameters

message	The message to log.

5.1.2.18 warning() [2/2]

```
void rti::dist_logger::DistLogger::warning (
            const std::string & message )
```

Logs a warning message.

The message received is logged using the default category and the RTI_DL_WARNING_LEVEL log level.

Parameters

message	The message to log.

5.1.2.19 notice() [1/2]

```
void rti::dist_logger::DistLogger::notice (
            const char * message )
```

Logs a notice message.

The message received is logged using the default category and the RTI_DL_NOTICE_LEVEL log level.

Parameters

message	The message to log.

5.1.2.20 notice() [2/2]

```
void rti::dist_logger::DistLogger::notice (
            const std::string & message )
```

Logs a notice message.

The message received is logged using the default category and the RTI_DL_NOTICE_LEVEL log level.

Parameters

message	The message to log.

5.1.2.21 info() [1/2]

```
void rti::dist_logger::DistLogger::info (
            const char * message )
```

Logs a info message.

The message received is logged using the default category and the RTI_DL_INFO_LEVEL log level.

Parameters

message	The message to log.

5.1.2.22 info() [2/2]

```
void rti::dist_logger::DistLogger::info (
            const std::string & message )
```

Logs a info message.

The message received is logged using the default category and the RTI_DL_INFO_LEVEL log level.

Parameters

message	The message to log.

5.1.2.23 debug() [1/2]

```
void rti::dist_logger::DistLogger::debug (
            const char * message )
```

Logs a debug message.

The message received is logged using the default category and the RTI_DL_DEBUG_LEVEL log level.

Parameters

message	The message to log.

5.1.2.24 debug() [2/2]

```
void rti::dist_logger::DistLogger::debug (
            const std::string & message )
```

Logs a debug message.

The message received is logged using the default category and the RTI_DL_DEBUG_LEVEL log level.

Parameters

message	The message to log.

5.1.2.25 trace() [1/2]

```
void rti::dist_logger::DistLogger::trace (
            const char * message )
```

Logs a trace message.

The message received is logged using the default category and the RTI_DL_TRACE_LEVEL log level.

Parameters

message	The message to log.

5.1.2.26 trace() [2/2]

```
void rti::dist_logger::DistLogger::trace (
            const std::string & message )
```

Logs a trace message.

The message received is logged using the default category and the RTI_DL_TRACE_LEVEL log level.

Parameters

message	The message to log.

5.1.2.27 log() [1/2]

```
void rti::dist_logger::DistLogger::log (
            int log_level,
            const char * message )
```

Logs a message.

The message received is logged using the default category and the given log level.

Parameters

log_level	The level of the log message.
message	The message to log.

5.1.2.28 log() [2/2]

```
void rti::dist_logger::DistLogger::log (
            int log_level,
            const std::string & message )
```

Logs a message.

The message received is logged using the default category and the given log level.

Parameters

log_level	The level of the log message.
message	The message to log.

5.1.2.29 get_fatal_log_level()

```
int rti::dist_logger::DistLogger::get_fatal_log_level ( ) const
```

Gets Fatal log level.

This method returns the currently defined value for RTI_DL_FATAL_LEVEL defined in <RTI Connext Installation Folder>/resource/idl/distlog.idl

See also

set_filter_level (p. 13) and **log_message(int, const std::string&, const std::string&)** (p. 15);

Returns

Fatal log level.

5.1.2.30 get_error_log_level()

```
int rti::dist_logger::DistLogger::get_error_log_level ( ) const
```

Gets Error log level.

This method returns the currently defined value for RTI_DL_ERROR_LEVEL defined in <RTI Connext Installation Folder>/resource/idl/distlog.idl

See also

set_filter_level (p. 13) and **log_message(int, const std::string&, const std::string&)** (p. 15);

Returns

Error log level.

5.1.2.31 get_warning_log_level()

```
int rti::dist_logger::DistLogger::get_warning_log_level ( ) const
```

Gets Warning log level.

This method returns the currently defined value for RTI_DL_WARNING_LEVEL defined in <RTI Connext Installation Folder>/resource/idl/distlog.idl

See also

set_filter_level (p. 13) and **log_message(int, const std::string&, const std::string&)** (p. 15);

Returns

Warning log level.

5.1.2.32 get_notice_log_level()

```
int rti::dist_logger::DistLogger::get_notice_log_level ( ) const
```

Gets Notice log level.

This method returns the currently defined value for RTI_DL_NOTICE_LEVEL defined in <RTI Connext Installation Folder>/resource/idl/distlog.idl

See also

set_filter_level (p. 13) and **log_message(int, const std::string&, const std::string&)** (p. 15);

Returns

Notice log level.

5.1.2.33 get_info_log_level()

```
int rti::dist_logger::DistLogger::get_info_log_level ( ) const
```

Gets Info log level.

This method returns the currently defined value for RTI_DL_INFO_LEVEL defined in <RTI Connext Installation Folder>/resource/idl/distlog.idl

See also

set_filter_level (p. 13) and **log_message(int, const std::string&, const std::string&)** (p. 15);

Returns

Info log level.

5.1.2.34 get_debug_log_level()

```
int rti::dist_logger::DistLogger::get_debug_log_level ( ) const
```

Gets Debug log level.

This method returns the currently defined value for RTI_DL_DEBUG_LEVEL defined in <RTI Connext Installation Folder>/resource/idl/distlog.idl

See also

set_filter_level (p. 13) and **log_message(int, const std::string&, const std::string&)** (p. 15);

Returns

Debug log level.

5.2 rti::dist_logger::DistLoggerOptions Class Reference

This class contains RTI Distributed Logger configurable options.

```
#include <rti/distlogger/DistLoggerOptions.hpp>
```

Public Member Functions

- **DistLoggerOptions** ()
  Creates a new object.
- dds::domain::DomainParticipant **domain_participant** () const
  Gets currently set Domain Participant.
- void **domain_participant** (const dds::domain::DomainParticipant &domain_participant)
  Sets current Domain Participant.
- bool **remote_administration_enabled** () const
  Indicates whether or not remote administration is enabled.
- void **remote_administration_enabled** (bool enabled)
  Enables or disables remote administration.
- int **filter_level** () const
  Gets the log filter level.
- void **filter_level** (int filter_level)
  Sets the log filter level.
- int **domain_id** () const
  Gets the current Domain ID.
- void **domain_id** (int domain_id)
  Sets the current Domain ID.
- rti::core::optional_value< std::string > **qos_library** () const
  Gets the Qos Library that will be used to create to RTI Distributed Logger.
- void **qos_library** (const rti::core::optional_value< std::string > &qos_library)

Sets the Qos Library that will be used to create to RTI Distributed Logger.

- rti::core::optional_value< std::string > **qos_profile** () const

  Gets the Qos Profile that will be used to create to RTI Distributed Logger.
- void **qos_profile** (const rti::core::optional_value< std::string > &qos_profile)

  Sets the Qos Profile that will be used to create to RTI Distributed Logger.
- int **queue_size** () const

  Gets the maximum queue size that will be assigned to RTI Distributed Logger.
- void **queue_size** (int queue_size)

  Sets the maximum queue size that will be assigned to RTI Distributed Logger.
- rti::core::ThreadSettings **thread_settings** () const

  Get the thread settings of the publishing thread that will be created by RTI Distributed Logger.
- void **thread_settings** (const rti::core::ThreadSettings &thread_settings)

  Sets the settings of the publishing thread that will be created by RTI Distributed Logger.
- rti::core::optional_value< std::string > **application_kind** () const

  Gets the application kind that will be associated with RTI Distributed Logger.
- void **application_kind** (const rti::core::optional_value< std::string > &application_kind)

  Sets the application kind that will be associated with RTI Distributed Logger.
- bool **echo_to_stdout** () const

  Indicates whether this feature is enabled.
- void **echo_to_stdout** (bool echo_to_stdout)

  Enables or disables this feature.
- bool **log_infrastructure_messages** () const

  Indicates whether Distributed Logger should also log all messages produced by infrastructure code automatically.
- void **log_infrastructure_messages** (bool log)

  Enables or disables automatic logging of infrastructure messages.
- bool **operator==** (const **DistLoggerOptions** &other) const

  Compares two objects for equality.
- bool **operator!=** (const **DistLoggerOptions** &other) const

  Compares two objects for inequality.

Friends

- void **swap** (**DistLoggerOptions** &left, **DistLoggerOptions** &right) noexcept

  Swaps two objects.

5.2.1 Detailed Description

This class contains RTI Distributed Logger configurable options.

Use this API to customize RTI Distributed Logger to adapt it to your needs. Assign the newly created **DistLogger↩ Options** (p. 26) object to the RTI Distributed Logger singleton before calling to **DistLogger::get_instance()** (p. 11).

5.2.2 Constructor & Destructor Documentation

5.2.2.1 DistLoggerOptions()

```
rti::dist_logger::DistLoggerOptions::DistLoggerOptions ( )
```

Creates a new object.

5.2.3 Member Function Documentation

5.2.3.1 domain_participant() [1/2]

```
dds::domain::DomainParticipant rti::dist_logger::DistLoggerOptions::domain_participant ( ) const
```

Gets currently set Domain Participant.

Returns

The current Domain Participant if set. Otherwise, dds::core::null.

5.2.3.2 domain_participant() [2/2]

```
void rti::dist_logger::DistLoggerOptions::domain_participant (
            const dds::domain::DomainParticipant & domain_participant )
```

Sets current Domain Participant.

There are several use cases in which you may want to use your own Domain Participant to create RTI Distributed Logger internal DDS entities. By calling this function, you will have control of the Domain Participant used by RTI Distributed Logger. Once assigned, `domain_participant` will be used to create the DDS entities (Topics, Publishers, Subscribers, DataWriters, & DataReaders) needed by RTI Distributed Logger. If `domain_participant` is set, the associated Distributed Logger instance must be finalized before finalizing the Domain Participant Factory.

Parameters

domain_participant	The dds::domain::DomainParticipant to use.

5.2.3.3 remote_administration_enabled() [1/2]

```
bool rti::dist_logger::DistLoggerOptions::remote_administration_enabled ( ) const
```

Indicates whether or not remote administration is enabled.

If true, remote administration Topics will be created and used. If false, they will not be created. The default value is true.

Returns

true if remote administration is enabled. Otherwise false.

5.2.3.4 remote_administration_enabled() [2/2]

```
void rti::dist_logger::DistLoggerOptions::remote_administration_enabled (
            bool enabled )
```

Enables or disables remote administration.

Sets remote administration of the Topics to the value passed as parameter.

Parameters

enabled	A bool indicating whether or not to enable remote administration.

5.2.3.5 filter_level() [1/2]

```
int rti::dist_logger::DistLoggerOptions::filter_level ( ) const
```

Gets the log filter level.

Supported Filter Levels: See the IDL file provided with RTI Distributed Logger (<RTI Connext Installation Folder>/resource/idl/distlog.idl). The default value is RTI_DL_INFO_LEVEL.

Returns

integer containing the log filter level of the options.

5.2.3.6 filter_level() [2/2]

```
void rti::dist_logger::DistLoggerOptions::filter_level (
            int filter_level )
```

Sets the log filter level.

Filtering controls what type of log messages are sent. This enables filtering on the *sending* side (in the application using RTI Distributed Logger). You can also enable filtering on the *receiving* side (such as in a tool that receives log messages, like RTI Admin Console or RTI Monitor).

Supported Filter Levels: See the IDL file provided with RTI Distributed Logger (<RTI Connext Installation Folder>/resource/idl/distlog.idl). The default value is RTI_DL_INFO_LEVEL.

Parameters

filter_level	The log filter level of the options (see above).

5.2.3.7 domain_id() [1/2]

```
int rti::dist_logger::DistLoggerOptions::domain_id ( ) const
```

Gets the current Domain ID.

Returns the Domain ID to be used by RTI Distributed Logger. This value is only used if a Domain Participant has not been set. Otherwise, the Domain ID used is the one of the set Domain Participant

The default value is 0.

Returns

Integer containing the DDS Domain ID where RTI Distributed Logger instance will be created.

5.2.3.8 domain_id() [2/2]

```
void rti::dist_logger::DistLoggerOptions::domain_id (
            int domain_id )
```

Sets the current Domain ID.

Update the Domain ID to be used by RTI Distributed Logger. This value is only used if a Domain Participant has not been set. Otherwise, the Domain ID used is the one of the set Domain Participant.

The default value is 0.

Parameters

domain_id	Int representing a DDS Domain ID. Max value: 250

5.2.3.9 qos_library() [1/2]

```
rti::core::optional_value< std::string > rti::dist_logger::DistLoggerOptions::qos_library ( )
const
```

Gets the Qos Library that will be used to create to RTI Distributed Logger.

The QoS library name to use when creating DDS entities. It is up to the application to ensure that this library is available at run time for the Distributed Logger to use.

Returns

rti::core::optional_value<std::string> containing the current Qos Library.

5.2.3.10 qos_library() [2/2]

```
void rti::dist_logger::DistLoggerOptions::qos_library (
            const rti::core::optional_value< std::string > & qos_library )
```

Sets the Qos Library that will be used to create to RTI Distributed Logger.

The QoS Library name to use when creating DDS entities. It is up to the application to ensure that this library is available at run time for the Distributed Logger to use.

Parameters

qos_library	rti::core::optional_value<std::string> containing the Qos Library to set.

5.2.3.11 qos_profile() [1/2]

```
rti::core::optional_value< std::string > rti::dist_logger::DistLoggerOptions::qos_profile ( )
const
```

Gets the Qos Profile that will be used to create to RTI Distributed Logger.

The QoS Profile name to use when creating DDS entities. It is up to the application to ensure that this profile is available at run time for the Distributed Logger to use.

Returns

rti::core::optional_value<std::string> containing the current Qos Profile.

5.2.3.12 qos_profile() [2/2]

```
void rti::dist_logger::DistLoggerOptions::qos_profile (
            const rti::core::optional_value< std::string > & qos_profile )
```

Sets the Qos Profile that will be used to create to RTI Distributed Logger.

The QoS Profile name to use when creating DDS entities. It is up to the application to ensure that this profile is available at run time for the Distributed Logger to use.

Parameters

qos_profile	rti::core::optional_value<std::string> containing the Qos Profile to set.

5.2.3.13 queue_size() [1/2]

```
int rti::dist_logger::DistLoggerOptions::queue_size ( ) const
```

Gets the maximum queue size that will be assigned to RTI Distributed Logger.

This field controls the size of the queue used to temporarily store log messages from the application until they can be written out to DDS. The default value is RTI_DL_DEFAULT_QUEUE_SIZE.

Returns

Integer containing the current maximum queue size.

5.2.3.14 queue_size() [2/2]

```
void rti::dist_logger::DistLoggerOptions::queue_size (
            int queue_size )
```

Sets the maximum queue size that will be assigned to RTI Distributed Logger.

This field controls the size of the queue used to temporarily store log messages from the application until they can be written out to DDS.

Parameters

queue_size	Integer containing the new maximum queue size.

5.2.3.15 thread_settings() [1/2]

```
rti::core::ThreadSettings rti::dist_logger::DistLoggerOptions::thread_settings ( ) const
```

Get the thread settings of the publishing thread that will be created by RTI Distributed Logger.

This field contains general settings of the publishing thread in Distributed Logger, like the thread priority, stack size, and other settings.

Returns

The currently set rti::core::ThreadSettings

5.2.3.16 thread_settings() [2/2]

```
void rti::dist_logger::DistLoggerOptions::thread_settings (
            const rti::core::ThreadSettings & thread_settings )
```

Sets the settings of the publishing thread that will be created by RTI Distributed Logger.

This parameter is used to control thread parameters such as priority, stack size, and other settings. These settings will only affect the publishing thread created by RTI Distributed Logger.

Parameters

thread_settings	A valid rti::core::ThreadSettings object with the settings for Distributed Logger's write thread.

5.2.3.17 application_kind() [1/2]

```
rti::core::optional_value< std::string > rti::dist_logger::DistLoggerOptions::application_kind (
) const
```

Gets the application kind that will be associated with RTI Distributed Logger.

This char pointer is sent along on the state Topic and can help identify different applications within a system. You can use this field however you think it will help you most.

Returns

rti::core::optional_value<std::string> containing the current application kind.

5.2.3.18 application_kind() [2/2]

```
void rti::dist_logger::DistLoggerOptions::application_kind (
            const rti::core::optional_value< std::string > & application_kind )
```

Sets the application kind that will be associated with RTI Distributed Logger.

This string is sent along on the state Topic and can help identify different applications within a system. You can use this field however you think it will help you most.

Parameters

application_kind	rti::core::optional_value<std::string> containing the new application kind. The maximum length of the string is 255.

5.2.3.19 echo_to_stdout() [1/2]

```
bool rti::dist_logger::DistLoggerOptions::echo_to_stdout ( ) const
```

Indicates whether this feature is enabled.

If true, Distributed Logger prints the log messages received to the stdout. The default value is true.

Returns

true if echo_to_stdout is enabled. Otherwise false.

5.2.3.20 echo_to_stdout() [2/2]

```
void rti::dist_logger::DistLoggerOptions::echo_to_stdout (
            bool echo_to_stdout )
```

Enables or disables this feature.

If true, Distributed Logger prints the log messages received to the stdout. The default value is true.

Parameters

echo_to_stdout	A bool indicating whether or not to enable this feature.

5.2.3.21 log_infrastructure_messages() [1/2]

```
bool rti::dist_logger::DistLoggerOptions::log_infrastructure_messages ( ) const
```

Indicates whether Distributed Logger should also log all messages produced by infrastructure code automatically.

If true, messages produced by either the middleware or the tools will be forwarded to Distributed Logger.

Returns

true if log_infrastructure_messages is enabled. Otherwise false.

5.2.3.22 log_infrastructure_messages() [2/2]

```
void rti::dist_logger::DistLoggerOptions::log_infrastructure_messages (
            bool log )
```

Enables or disables automatic logging of infrastructure messages.

If true, messages produced by either the middleware or the tools will be forwarded to Distributed Logger. In this mode Distributed Logger uses NDDS_Config_Logger_set_output_device() API to receive messages from the middleware. Hence any device previously registered with the rti::config::Logger::set_device() API will be unregistered. Similarly Distributed Logger will stop logging infrastructure messages if the set_device() API is used to register a new device after the creation of the Distributed Logger instance.

Parameters

log	bool indicating whether or not to enable this feature.

5.2.3.23 operator==()

```
bool rti::dist_logger::DistLoggerOptions::operator== (
            const DistLoggerOptions & other ) const
```

Compares two objects for equality.

Parameters

other	The other object to compare.

5.2.3.24 operator"!=()

```
bool rti::dist_logger::DistLoggerOptions::operator!= (
            const DistLoggerOptions & other ) const
```

Compares two objects for inequality.

Parameters

other	The other object to compare.

5.2.4 Friends And Related Function Documentation

5.2.4.1 swap

```
void swap (
            DistLoggerOptions & left,
            DistLoggerOptions & right )  [friend]
```

Swaps two objects.

Parameters

left	One of the objects to swap.
right	The other object to swap.

Index

application_kind
    rti::dist_logger::DistLoggerOptions, 33

debug
    rti::dist_logger::DistLogger, 21
DistLoggerOptions
    rti::dist_logger::DistLoggerOptions, 27
domain_id
    rti::dist_logger::DistLoggerOptions, 30
domain_participant
    rti::dist_logger::DistLoggerOptions, 28

echo_to_stdout
    rti::dist_logger::DistLoggerOptions, 34
error
    rti::dist_logger::DistLogger, 18

fatal
    rti::dist_logger::DistLogger, 17
filter_level
    rti::dist_logger::DistLoggerOptions, 29
finalize
    rti::dist_logger::DistLogger, 11

get_debug_log_level
    rti::dist_logger::DistLogger, 25
get_error_log_level
    rti::dist_logger::DistLogger, 24
get_fatal_log_level
    rti::dist_logger::DistLogger, 24
get_info_log_level
    rti::dist_logger::DistLogger, 25
get_instance
    rti::dist_logger::DistLogger, 11
get_notice_log_level
    rti::dist_logger::DistLogger, 25
get_warning_log_level
    rti::dist_logger::DistLogger, 24

info
    rti::dist_logger::DistLogger, 20

log
    rti::dist_logger::DistLogger, 22
log_infrastructure_messages
    rti::dist_logger::DistLoggerOptions, 34
log_message
    rti::dist_logger::DistLogger, 15, 16

notice
    rti::dist_logger::DistLogger, 19, 20

operator!=
    rti::dist_logger::DistLoggerOptions, 35
operator==
    rti::dist_logger::DistLoggerOptions, 35

qos_library
    rti::dist_logger::DistLoggerOptions, 30, 31
qos_profile
    rti::dist_logger::DistLoggerOptions, 31
queue_size
    rti::dist_logger::DistLoggerOptions, 32

remote_administration_enabled
    rti::dist_logger::DistLoggerOptions, 28, 29
RTI Distributed Logger API, 7
rti::dist_logger::DistLogger, 9
    debug, 21
    error, 18
    fatal, 17
    finalize, 11
    get_debug_log_level, 25
    get_error_log_level, 24
    get_fatal_log_level, 24
    get_info_log_level, 25
    get_instance, 11
    get_notice_log_level, 25
    get_warning_log_level, 24
    info, 20
    log, 22
    log_message, 15, 16
    notice, 19, 20
    set_filter_level, 13
    set_options, 13
    set_print_format, 14
    set_verbosity, 15
    severe, 17, 18
    trace, 21, 22
    warning, 19
rti::dist_logger::DistLoggerOptions, 26
    application_kind, 33
    DistLoggerOptions, 27

domain_id, 30
domain_participant, 28
echo_to_stdout, 34
filter_level, 29
log_infrastructure_messages, 34
operator!=, 35
operator==, 35
qos_library, 30, 31
qos_profile, 31
queue_size, 32
remote_administration_enabled, 28, 29
swap, 36
thread_settings, 32, 33

set_filter_level
rti::dist_logger::DistLogger, 13
set_options
rti::dist_logger::DistLogger, 13
set_print_format
rti::dist_logger::DistLogger, 14
set_verbosity
rti::dist_logger::DistLogger, 15
severe
rti::dist_logger::DistLogger, 17, 18
swap
rti::dist_logger::DistLoggerOptions, 36

thread_settings
rti::dist_logger::DistLoggerOptions, 32, 33
trace
rti::dist_logger::DistLogger, 21, 22

warning
rti::dist_logger::DistLogger, 19