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

4.2 RTI Distributed Logger Version Module . . . 7

4.2.1 Detailed Description . . . 8

4.2.2 Typedef Documentation . . . 8

4.2.2.1 RTI_DL_DistLogger_LibraryVersion . . . 8

4.2.3 Function Documentation . . . 8

4.2.3.1 RTI_DL_DistLogger_get_version() . . . 8

4.2.3.2 RTI_DL_DistLogger_get_api_build_version() . . . 9

4.2.3.3 RTI_DL_DistLogger_get_api_version_string() . . . 9

4.2.3.4 RTI_DLDistLogger_get_version() . . . 9

4.2.3.5 RTI_DLDistLogger_get_api_build_version_string() . . . 9

4.2.3.6 RTI_DLDistLogger_get_api_version_string() . . . 9

5 Class Documentation 11

5.1 RTI_DLDistLogger Class Reference . . . 11

5.1.1 Detailed Description . . . 12

5.1.2 Member Function Documentation . . . 12

5.1.2.1 getInstance() . . . 13

5.1.2.2 setFilterLevel() . . . 13

5.1.2.3 setOptions() . . . 14

5.1.2.4 setRTILoggerPrintFormat() . . . 15

5.1.2.5 setRTILoggerVerbosityByCategory() . . . 15

5.1.2.6 logMessageWithLevelCategory() . . . 15

5.1.2.7 logMessageWithParams() . . . 16

5.1.2.8 fatal() . . . 16

5.1.2.9 severe() . . . 16

5.1.2.10 error() . . . 17

5.1.2.11 warning() . . . 17

5.1.2.12 notice() . . . 17

5.1.2.13 info() . . . 18
5.1.2.14 debug() . . . 18
5.1.2.15 trace() . . . 18
5.1.2.16 log() . . . 19
5.1.2.17 finalizeInstance() . . . 19
5.1.2.18 getFatalLogLevel() . . . 19
5.1.2.19 getErrorLogLevel() . . . 20
5.1.2.20 getWarningLogLevel() . . . 20
5.1.2.21 getNoticeLogLevel() . . . 20
5.1.2.22 getInfoLogLevel() . . . 20
5.1.2.23 getDebugLogLevel() . . . 21
5.2 RTI_DLOptions Class Reference . . . 21
5.2.1 Detailed Description . . . 22
5.2.2 Constructor & Destructor Documentation . . . 22
5.2.2.1 RTI_DLOptions() . . . 22
5.2.2.2 ∼RTI_DLOptions() . . . 23
5.2.3 Member Function Documentation . . . 23
5.2.3.1 getCDomainParticipantI() . . . 23
5.2.3.2 setDomainParticipant() . . . 23
5.2.3.3 isRemoteAdministrationEnabled() . . . 24
5.2.3.4 setRemoteAdministrationEnabled() . . . 24
5.2.3.5 getFilterLevel() . . . 24
5.2.3.6 setFilterLevel() . . . 25
5.2.3.7 getDomainId() . . . 25
5.2.3.8 setDomainId() . . . 26
5.2.3.9 getQosLibrary() . . . 26
5.2.3.10 setQosLibrary() . . . 26
5.2.3.11 getQosProfile() . . . 27
5.2.3.12 setQosProfile() . . . 27
5.2.3.13 getQueueSize() . . . 28
5.2.3.14 setQueueSize() . . . 28
5.2.3.15 getThreadSettings() . . . 28
5.2.3.16 setThreadSettings() . . . 29
5.2.3.17 getApplicationKind() . . . 29
5.2.3.18 setApplicationKind() . . . 29
5.2.3.19 getEchoToStdout() . . . 30
5.2.3.20 setEchoToStdout() . . . 30
5.2.3.21 getLogInfrastructureMessages() . . . 30
5.2.3.22 setLogInfrastructureMessages() . . . 31

5.3 RTI_DLUtility Class Reference 31
5.3.1 Detailed Description 31
5.3.2 Member Function Documentation 31
5.3.2.1 sleep() 31

Index **33**

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_DLOptions** (p. 21). 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_DLDistLogger::setOptions** (p. 14).
3. Acquire an **RTI_DLDistLogger** (p. 11) instance using **RTI_DLDistLogger::getInstance** (p. 12). 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_DLDistLogger::error** (p. 17), **RTI_DLDistLogger::fatal** (p. 16),... 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_DLDist↩ Logger::setFilterLevel** (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
RTI Distributed Logger Version Module 7

Chapter 3

Class Index

3.1 Class List

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

RTI_DLDistLogger
This class contains RTI Distributed Logger singleton and the methods to configure and use it 11
RTI_DLOptions
This class contains RTI Distributed Logger configurable options 21
RTI_DLUtility
Commodity functions for using RTI Distributed Logger 31

Chapter 4

Module Documentation

4.1 RTI Distributed Logger API

Main API to use RTI Distributed Logger singleton.

Modules

- **RTI Distributed Logger Version Module**

Classes

- class **RTI_DLOptions**

  This class contains RTI Distributed Logger configurable options.
- class **RTI_DLUtility**

  Commodity functions for using RTI Distributed Logger.
- class **RTI_DLDistLogger**

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

4.1.1 Detailed Description

Main API to use RTI Distributed Logger singleton.

4.2 RTI Distributed Logger Version Module

Typedefs

- typedef struct DDS_ProductVersion_t **RTI_DL_DistLogger_LibraryVersion**

  The RTI Distributed Logger library version.

Functions

- const **RTI_DL_DistLogger_LibraryVersion** * **RTI_DL_DistLogger_get_version** (void)

  Get the RTI Distributed Logger C library version.
- const char * **RTI_DL_DistLogger_get_api_build_version** (void)

  Get the RTI Distributed Logger C build version.
- const char * **RTI_DL_DistLogger_get_api_version_string** (void)

  Get the RTI Distributed Logger C api version as a string.
- const **RTI_DL_DistLogger_LibraryVersion** * **RTI_DLDistLogger_get_version** ()

  Get the Distributed Logger C++ library version.
- std::string **RTI_DLDistLogger_get_api_build_version_string** ()

  Get the Distributed Logger C++ build version.
- std::string **RTI_DLDistLogger_get_api_version_string** ()

  Get the Distributed Logger C++ build version as a string.

4.2.1 Detailed Description

4.2.2 Typedef Documentation

4.2.2.1 RTI_DL_DistLogger_LibraryVersion

```
typedef struct DDS_ProductVersion_t  RTI_DL_DistLogger_LibraryVersion
```

The RTI Distributed Logger library version.

4.2.3 Function Documentation

4.2.3.1 RTI_DL_DistLogger_get_version()

```
const  RTI_DL_DistLogger_LibraryVersion * RTI_DL_DistLogger_get_version (
            void  )
```

Get the RTI Distributed Logger C library version.

4.2.3.2 RTI_DL_DistLogger_get_api_build_version()

```
const char * RTI_DL_DistLogger_get_api_build_version (
            void  )
```

Get the RTI Distributed Logger C build version.

4.2.3.3 RTI_DL_DistLogger_get_api_version_string()

```
const char * RTI_DL_DistLogger_get_api_version_string (
            void  )
```

Get the RTI Distributed Logger C api version as a string.

4.2.3.4 RTI_DLDistLogger_get_version()

```
const  RTI_DL_DistLogger_LibraryVersion * RTI_DLDistLogger_get_version ( )
```

Get the Distributed Logger C++ library version.

4.2.3.5 RTI_DLDistLogger_get_api_build_version_string()

```
std::string RTI_DLDistLogger_get_api_build_version_string ( )
```

Get the Distributed Logger C++ build version.

4.2.3.6 RTI_DLDistLogger_get_api_version_string()

```
std::string RTI_DLDistLogger_get_api_version_string ( )
```

Get the Distributed Logger C++ build version as a string.

Chapter 5

Class Documentation

5.1 RTI_DLDistLogger Class Reference

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

`#include <rti_dl_cpp.h>`

Public Member Functions

- DDS_ReturnCode_t **setFilterLevel** (DDS_Long newLevel)

  Sets the filter level for the singleton.
- bool **setRTILoggerPrintFormat** (NDDS_Config_LogPrintFormat logPrintFormat)

  Sets PrintFormat of RTI Logger to the provided value.
- void **setRTILoggerVerbosityByCategory** (NDDS_Config_LogCategory category, NDDS_Config_LogVerbosity verbosity)

  Updates the LogCategory and the LogVerbosity in RTI Logger to the provided value.
- void **logMessageWithLevelCategory** (int logLevel, const char ∗message, const char ∗category)

  Logs a message at the specified level and with the specified category.
- void **logMessageWithParams** (const struct RTI_DL_DistLogger_MessageParams &params)

  *Performs the same function as **logMessageWithLevelCategory*** (p. 15) *except that it also allows specification of the message timestamp.*
- void **fatal** (const char ∗message)

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

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

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

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

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

Logs a info message.

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

  Logs a debug message.

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

  Logs a trace message.

- void **log** (int logLevel, const char *message)

  Logs a message.

- void **finalizeInstance** ()

  Finalizes and delete the RTI_DL_DistLogger singleton.

- DDS_Long **getFatalLogLevel** ()

  Return Fatal log level.

- DDS_Long **getErrorLogLevel** ()

  Return Error log level.

- DDS_Long **getWarningLogLevel** ()

  Return Warning log level.

- DDS_Long **getNoticeLogLevel** ()

  Return Notice log level.

- DDS_Long **getInfoLogLevel** ()

  Return Info log level.

- DDS_Long **getDebugLogLevel** ()

  Return Debug log level.

Static Public Member Functions

- static **RTI_DLDistLogger** * **getInstance** ()

  Provides access to the RTI_DL_DistLogger singleton.

- static bool **setOptions** (**RTI_DLOptions** &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 **RTI_DLOptions** (p. 21) class to customize the singleton **BEFORE** creating it.

5.1.2 Member Function Documentation

5.1.2.1 getInstance()

```
static RTI_DLDistLogger * RTI_DLDistLogger::getInstance ( ) [static]
```

Provides access to the RTI_DL_DistLogger singleton.

This method provides access to the singleton instance of the RTI_DL_DistLogger. 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 RTI_DL_DistLogger.

5.1.2.2 setFilterLevel()

```
DDS_ReturnCode_t RTI_DLDistLogger::setFilterLevel (
            DDS_Long newLevel )
```

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

in	*newLevel*	The new filter level.

Returns

Error code. DDS_RETCODE_OK if everything goes as planned.

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 5.2 *Filter Levels*

Connext DDS API constant	Distributed Logger IDL constant	Distributed Logger API getter
NDDS_CONFIG_LOG_FATAL_ERROR	RTI_DL_FATAL_LEVEL	**getFatalLogLevel** (p. 19)
NDDS_CONFIG_LOG_LEVEL_ERROR	RTI_DL_ERROR_LEVEL	**getErrorLogLevel** (p. 19)
NDDS_CONFIG_LOG_LEVEL_↩ WARNING	RTI_DL_WARNING_LEVEL	**getWarningLogLevel** (p. 20)
NDDS_CONFIG_LOG_LEVEL_↩ STATUS_LOCAL	RTI_DL_NOTICE_LEVEL	**getNoticeLogLevel** (p. 20)
NDDS_CONFIG_LOG_LEVEL_↩ STATUS_REMOTE	RTI_DL_INFO_LEVEL	**getInfoLogLevel** (p. 20)
NDDS_CONFIG_LOG_LEVEL_DEBUG	RTI_DL_DEBUG_LEVEL	**getDebugLogLevel** (p. 20)

5.1.2.3 setOptions()

```
static bool RTI_DLDistLogger::setOptions (
              RTI_DLOptions & options )  [static]
```

Assigns configurable options to RTI Distributed Logger before its creation.

Assigns the values in the `options` to the **RTI_DLDistLogger** (p. 11) singleton **before** its creation. If the singleton was already created, this function has no effect.

Parameters

in	*options*	The new options to be assigned to the **RTI_DLDistLogger** (p. 11) singleton.

Precondition

The **singleton** must be NULL.

Returns

RTI_TRUE if everything went as planned. RTI_FALSE if RTI Distributed Logger already existed.

5.1.2.4 setRTILoggerPrintFormat()

```
bool RTI_DLDistLogger::setRTILoggerPrintFormat (
            NDDS_Config_LogPrintFormat logPrintFormat )
```

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

in	*logPrintFormat*	The new PrintFormat.

Returns

RTI_TRUE if everything went as planned.

5.1.2.5 setRTILoggerVerbosityByCategory()

```
void RTI_DLDistLogger::setRTILoggerVerbosityByCategory (
            NDDS_Config_LogCategory category,
            NDDS_Config_LogVerbosity verbosity )
```

Updates the LogCategory and the LogVerbosity 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

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

5.1.2.6 logMessageWithLevelCategory()

```
void RTI_DLDistLogger::logMessageWithLevelCategory (
            int logLevel,
            const char * message,
            const char * category )
```

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

Parameters

in	*logLevel*	The level of the log message.
in	*message*	The message to log.
in	*category*	The category of the log message.

5.1.2.7 logMessageWithParams()

```
void RTI_DLDistLogger::logMessageWithParams (
            const struct RTI_DL_DistLogger_MessageParams & params )
```

Performs the same function as **logMessageWithLevelCategory** (p. 15) except that it also allows specification of the message timestamp.

Parameters

in	*params*	Parameters.

5.1.2.8 fatal()

```
void RTI_DLDistLogger::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

in	*message*	The message to log.

5.1.2.9 severe()

```
void RTI_DLDistLogger::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

in	*message*	The message to log.

5.1.2.10 error()

```
void RTI_DLDistLogger::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

in	*message*	The message to log.

5.1.2.11 warning()

```
void RTI_DLDistLogger::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

in	*message*	The message to log.

5.1.2.12 notice()

```
void RTI_DLDistLogger::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

in	*message*	The message to log.

5.1.2.13 info()

```
void RTI_DLDistLogger::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

in	*message*	The message to log.

5.1.2.14 debug()

```
void RTI_DLDistLogger::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

in	*message*	The message to log.

5.1.2.15 trace()

```
void RTI_DLDistLogger::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

in	*message*	The message to log.

5.1.2.16 log()

```
void RTI_DLDistLogger::log (
            int logLevel,
            const char * message )
```

Logs a message.

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

Parameters

in	*logLevel*	The level of the log message.
in	*message*	The message to log.

5.1.2.17 finalizeInstance()

```
void RTI_DLDistLogger::finalizeInstance ( )
```

Finalizes and delete the RTI_DL_DistLogger singleton.

Unset the connections with DDS, frees memory and set singleton to NULL.

5.1.2.18 getFatalLogLevel()

```
DDS_Long RTI_DLDistLogger::getFatalLogLevel ( )
```

Return Fatal log level.

Returns

Fatal log level.

5.1.2.19 getErrorLogLevel()

```
DDS_Long RTI_DLDistLogger::getErrorLogLevel ( )
```

Return Error log level.

Returns

Error log level.

5.1.2.20 getWarningLogLevel()

```
DDS_Long RTI_DLDistLogger::getWarningLogLevel ( )
```

Return Warning log level.

Returns

Warning log level.

5.1.2.21 getNoticeLogLevel()

```
DDS_Long RTI_DLDistLogger::getNoticeLogLevel ( )
```

Return Notice log level.

Returns

Notice log level.

5.1.2.22 getInfoLogLevel()

```
DDS_Long RTI_DLDistLogger::getInfoLogLevel ( )
```

Return Info log level.

Returns

Info log level.

5.1.2.23 getDebugLogLevel()

```
DDS_Long RTI_DLDistLogger::getDebugLogLevel ( )
```

Return Debug log level.

Returns

Debug log level.

5.2 RTI_DLOptions Class Reference

This class contains RTI Distributed Logger configurable options.

```
#include <rti_dl_cpp.h>
```

Public Member Functions

- **RTI_DLOptions** ()
  Default constructor.
- ~**RTI_DLOptions** ()
  Default destructor.
- DDS_DomainParticipant * **getCDomainParticipantI** ()
  Gets current C DomainParticipant.
- DDS_ReturnCode_t **setDomainParticipant** (DDSDomainParticipant *domainParticipant)
  Sets current DomainParticipant.
- bool **isRemoteAdministrationEnabled** ()
  Sets current DomainParticipant.
- DDS_ReturnCode_t **setRemoteAdministrationEnabled** (bool enabled)
  Enables or disables remote administration.
- int **getFilterLevel** ()
  Gets the log filter level.
- DDS_ReturnCode_t **setFilterLevel** (int filterLevel)
  Sets the log filter level.
- int **getDomainId** ()
  Gets the current Domain ID.
- DDS_ReturnCode_t **setDomainId** (int domainId)
  Sets the current Domain ID.
- char * **getQosLibrary** ()
  Gets the Qos Library that will be used to create to RTI Distributed Logger.
- DDS_ReturnCode_t **setQosLibrary** (const char *qosLibrary)
  Sets the Qos Library that will be used to create to RTI Distributed Logger.
- char * **getQosProfile** ()
  Gets the Qos Profile that will be used to create to RTI Distributed Logger.
- DDS_ReturnCode_t **setQosProfile** (const char *qosProfile)

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

- int **getQueueSize** ()

  Gets the maximum queue size that will be assigned to RTI Distributed Logger.
- DDS_ReturnCode_t **setQueueSize** (int queueSize)

  Sets the maximum queue size that will be assigned to RTI Distributed Logger.
- struct DDS_ThreadSettings_t * **getThreadSettings** ()

  Get the thread settings of the publishing threadthat will be created by RTI Distributed Logger.
- DDS_ReturnCode_t **setThreadSettings** (const struct DDS_ThreadSettings_t *threadSettings)

  Sets the settings of the publishing thread that will be created by RTI Distributed Logger.
- char * **getApplicationKind** ()

  Gets the application kind that will be associated with RTI Distributed Logger.
- DDS_ReturnCode_t **setApplicationKind** (const char *applicationKind)

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

  Gets the current value of echoToStdout.
- DDS_ReturnCode_t **setEchoToStdout** (bool echoToStdout)

  Sets the value of echoToStdout.
- bool **getLogInfrastructureMessages** ()

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

  Enables or disables automatic logging of infrastructure messages.

Friends

- class **RTI_DLDistLogger**

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 **RTI_DLOptions** (p. 21) object to RTI Distributed Logger singleton before calling to **RTI_DLDistLogger::getInstance** (p. 12).

5.2.2 Constructor & Destructor Documentation

5.2.2.1 RTI_DLOptions()

```
RTI_DLOptions::RTI_DLOptions ( )
```

Default constructor.

Allocates memory for a **RTI_DLOptions** (p. 21) class and sets the default values needed to create the DL.

See also

~**RTI_DLOptions()** (p. 22)

5.2.2.2 ~RTI_DLOptions()

```
RTI_DLOptions::~RTI_DLOptions ( )
```

Default destructor.

Frees **RTI_DLOptions** (p. 21) class. Checks whether the pointer exists or not. If `domainParticipant` was assigned, its memory is not released.

5.2.3 Member Function Documentation

5.2.3.1 getCDomainParticipantI()

```
DDS_DomainParticipant * RTI_DLOptions::getCDomainParticipantI ( )
```

Gets current C DomainParticipant.

If a customized DomainParticipant have been set, this function returns a pointer to the internal C DomainParticipant that will be used to create DDS entities (Topics, Publishers, Subscribers, DataWriters, and DataReaders) needed by RTI Distributed Logger. The default value is NULL.

Returns

Pointer to the DDS_DomainParticipant contained in the class. Otherwise, NULL.

5.2.3.2 setDomainParticipant()

```
DDS_ReturnCode_t RTI_DLOptions::setDomainParticipant (
            DDSDomainParticipant * domainParticipant )
```

Sets current DomainParticipant.

There are several use cases in which you may want to use your own DomainParticipant to create RTI Distributed Logger internal DDS entities. By calling this function, you will have control of the DomainParticipant used by RTI Distributed Logger. Once assigned, **`domainParticipant`** `will` be used to create the DDS entities (Topics, Publishers, Subscribers, DataWriters, & DataReaders) needed by RTI Distributed Logger.

When the DomainParticipant is set to a non-NULL value it is the user's responsibility to handle the memory associated with it.

Parameters

in	*domainParticipant*	Pointer to a valid DDS_DomainParticipant

Postcondition

Memory must be released by the user **after** finalizing RTI Distributed Logger using **RTI_DLDistLogger::finalize↩ Instance** (p. 19).

5.2.3.3 isRemoteAdministrationEnabled()

```
bool RTI_DLOptions::isRemoteAdministrationEnabled ( )
```

Sets current DomainParticipant.

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 the Remote Administration is enabled. Otherwise false.

5.2.3.4 setRemoteAdministrationEnabled()

```
DDS_ReturnCode_t RTI_DLOptions::setRemoteAdministrationEnabled (
            bool enabled )
```

Enables or disables remote administration.

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

Parameters

in	*enabled*	Boolean indicating new remote administration value.

5.2.3.5 getFilterLevel()

```
int RTI_DLOptions::getFilterLevel ( )
```

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. See **setFilterLevel()** (p. 25) for more information on filter levels.

Returns

integer containing the log filter level of the options.

See also

RTI_DLDistLogger::setFilterLevel (p. 13)

5.2.3.6 setFilterLevel()

```
DDS_ReturnCode_t RTI_DLOptions::setFilterLevel (
            int filterLevel )
```

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

in	*filterLevel*	The log filter level of the options (see above).

Returns

Error code. DDS_RETCODE_OK if everything goes as planned.

See also

RTI_DLDistLogger::setFilterLevel (p. 13)

5.2.3.7 getDomainId()

```
int RTI_DLOptions::getDomainId ( )
```

Gets the current Domain ID.

Returns the Domain ID to be used by RTI Distributed Logger. This value is only used if **setDomainParticipant** (p. 23) has NOT been called. Otherwise, the Domain ID used is the one of the DomainParticipant assigned.

The default value is 0.

Returns

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

5.2.3.8 setDomainId()

```
DDS_ReturnCode_t RTI_DLOptions::setDomainId (
            int domainId )
```

Sets the current Domain ID.

Updated the Domain ID to be used by RTI Distributed Logger. This value is only used if **setDomainParticipant** (p. 23) has NOT been called. Otherwise, the Domain ID used is the one of the DomainParticipant assigned.

The default value is 0.

Parameters

in	*domain↩ Id*	Int representing a DDS Domain ID. Max value: 250

Returns

DDS_RETCODE_OK if the operation was a success.

5.2.3.9 getQosLibrary()

```
char * RTI_DLOptions::getQosLibrary ( )
```

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 RTI Distributed Logger to use. The default value is "".

Returns

String containing the current Qos Library. Do not modify the content of this pointer directly nor delete it.

5.2.3.10 setQosLibrary()

```
DDS_ReturnCode_t RTI_DLOptions::setQosLibrary (
            const char * qosLibrary )
```

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

in	*qosLibrary*	String containing the new Qos Library. A copy of the string is made.

Returns

DDS_RETCODE_OK if the operation was a success.

5.2.3.11 getQosProfile()

```
char * RTI_DLOptions::getQosProfile ( )
```

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. The default value is "".

Returns

String containing the current Qos Profile. Do not modify the content of this pointer directly nor delete it.

5.2.3.12 setQosProfile()

```
DDS_ReturnCode_t RTI_DLOptions::setQosProfile (
            const char * qosProfile )
```

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

in	*qosProfile*	String containing the new Qos Profile. A copy of the string is made.

Returns

DDS_RETCODE_OK if the operation was a success.

5.2.3.13 getQueueSize()

```
int RTI_DLOptions::getQueueSize ( )
```

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 setQueueSize()

```
DDS_ReturnCode_t RTI_DLOptions::setQueueSize (
            int queueSize )
```

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

in	*queueSize*	Integer containing the new maximum queue size.

Returns

DDS_RETCODE_OK if the operation was a success.

5.2.3.15 getThreadSettings()

```
struct DDS_ThreadSettings_t * RTI_DLOptions::getThreadSettings ( )
```

Get the thread settings of the publishing threadthat 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

threadPriority Integer containing the current thread priority.

5.2.3.16 setThreadSettings()

```
DDS_ReturnCode_t RTI_DLOptions::setThreadSettings (
            const struct DDS_ThreadSettings_t * threadSettings )
```

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

in	*threadSettings*	A valid DDS_ThreadSettings_t struct with the settings for Distributed Logger's write thread.

Returns

DDS_RETCODE_OK if the operation was a success.

5.2.3.17 getApplicationKind()

```
char * RTI_DLOptions::getApplicationKind ( )
```

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. The default value is "".

Returns

String containing the current application kind.

5.2.3.18 setApplicationKind()

```
DDS_ReturnCode_t RTI_DLOptions::setApplicationKind (
            const char * applicationKind )
```

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. The default value is "".

Parameters

in	*applicationKind*	String containing the new application kind. The maximum length of the string is 255.

5.2.3.19 getEchoToStdout()

```
bool RTI_DLOptions::getEchoToStdout ( )
```

Gets the current value of echoToStdout.

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

Returns

Boolean to the current echoToStdout value

5.2.3.20 setEchoToStdout()

```
DDS_ReturnCode_t RTI_DLOptions::setEchoToStdout (
            bool echoToStdout )
```

Sets the value of echoToStdout.

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

Parameters

in	*echoToStdout*	New value of set self.echoToStdout.

5.2.3.21 getLogInfrastructureMessages()

```
bool RTI_DLOptions::getLogInfrastructureMessages ( )
```

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

Boolean to the current LogInfrastructureMessages value.

5.2.3.22 setLogInfrastructureMessages()

```
DDS_ReturnCode_t RTI_DLOptions::setLogInfrastructureMessages (
            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 NDDSConfigLogger's set_output_device() API will be unregistered. Similarly Distributed Logger will stop logging infrastructure messages if the set_output_device() API is used to register a new device after the creation of the Distributed Logger instance.

Parameters

in	*log*	Boolean indicating whether to enable this feature.

Returns

DDS_RETCODE_OK if the operation was a success.

5.3 RTI_DLUtility Class Reference

Commodity functions for using RTI Distributed Logger.

`#include <rti_dl_cpp.h>`

Static Public Member Functions

- static void **sleep** (int msec)
  Blocks the calling thread for the specified duration.

5.3.1 Detailed Description

Commodity functions for using RTI Distributed Logger.

5.3.2 Member Function Documentation

5.3.2.1 sleep()

```
static void RTI_DLUtility::sleep (
            int msec )  [static]
```

Blocks the calling thread for the specified duration.

Note that the achievable resolution of sleep is OS-dependent. That is, do not assume that you can sleep for 1 ms just because you can specify a 1-ms sleep duration via the API. The sleep resolution on most operating systems is usually 10 ms or greater.

Parameters

in	*msec*	Sleep duration in milliseconds.

Index

~RTI_DLOptions
    RTI_DLOptions, 22

debug
    RTI_DLDistLogger, 18

error
    RTI_DLDistLogger, 17

fatal
    RTI_DLDistLogger, 16
finalizeInstance
    RTI_DLDistLogger, 19

getApplicationKind
    RTI_DLOptions, 29
getCDomainParticipantI
    RTI_DLOptions, 23
getDebugLogLevel
    RTI_DLDistLogger, 20
getDomainId
    RTI_DLOptions, 25
getEchoToStdout
    RTI_DLOptions, 30
getErrorLogLevel
    RTI_DLDistLogger, 19
getFatalLogLevel
    RTI_DLDistLogger, 19
getFilterLevel
    RTI_DLOptions, 24
getInfoLogLevel
    RTI_DLDistLogger, 20
getInstance
    RTI_DLDistLogger, 12
getLogInfrastructureMessages
    RTI_DLOptions, 30
getNoticeLogLevel
    RTI_DLDistLogger, 20
getQosLibrary
    RTI_DLOptions, 26
getQosProfile
    RTI_DLOptions, 27
getQueueSize
    RTI_DLOptions, 27
getThreadSettings
    RTI_DLOptions, 28
getWarningLogLevel
    RTI_DLDistLogger, 20

info
    RTI_DLDistLogger, 18
isRemoteAdministrationEnabled
    RTI_DLOptions, 24

log
    RTI_DLDistLogger, 19
logMessageWithLevelCategory
    RTI_DLDistLogger, 15
logMessageWithParams
    RTI_DLDistLogger, 16

notice
    RTI_DLDistLogger, 17

RTI Distributed Logger API, 7
RTI Distributed Logger Version Module, 7
    RTI_DL_DistLogger_get_api_build_version, 8
    RTI_DL_DistLogger_get_api_version_string, 9
    RTI_DL_DistLogger_get_version, 8
    RTI_DL_DistLogger_LibraryVersion, 8
    RTI_DLDistLogger_get_api_build_version_string, 9
    RTI_DLDistLogger_get_api_version_string, 9
    RTI_DLDistLogger_get_version, 9
RTI_DL_DistLogger_get_api_build_version
    RTI Distributed Logger Version Module, 8
RTI_DL_DistLogger_get_api_version_string
    RTI Distributed Logger Version Module, 9
RTI_DL_DistLogger_get_version
    RTI Distributed Logger Version Module, 8
RTI_DL_DistLogger_LibraryVersion
    RTI Distributed Logger Version Module, 8
RTI_DLDistLogger, 11
    debug, 18
    error, 17
    fatal, 16
    finalizeInstance, 19
    getDebugLogLevel, 20
    getErrorLogLevel, 19
    getFatalLogLevel, 19
    getInfoLogLevel, 20
    getInstance, 12
    getNoticeLogLevel, 20

getWarningLogLevel, 20
info, 18
log, 19
logMessageWithLevelCategory, 15
logMessageWithParams, 16
notice, 17
setFilterLevel, 13
setOptions, 14
setRTILoggerPrintFormat, 14
setRTILoggerVerbosityByCategory, 15
severe, 16
trace, 18
warning, 17
RTI_DLDistLogger_get_api_build_version_string
RTI Distributed Logger Version Module, 9
RTI_DLDistLogger_get_api_version_string
RTI Distributed Logger Version Module, 9
RTI_DLDistLogger_get_version
RTI Distributed Logger Version Module, 9
RTI_DLOptions, 21
~RTI_DLOptions, 22
getApplicationKind, 29
getCDomainParticipantI, 23
getDomainId, 25
getEchoToStdout, 30
getFilterLevel, 24
getLogInfrastructureMessages, 30
getQosLibrary, 26
getQosProfile, 27
getQueueSize, 27
getThreadSettings, 28
isRemoteAdministrationEnabled, 24
RTI_DLOptions, 22
setApplicationKind, 29
setDomainId, 25
setDomainParticipant, 23
setEchoToStdout, 30
setFilterLevel, 25
setLogInfrastructureMessages, 30
setQosLibrary, 26
setQosProfile, 27
setQueueSize, 28
setRemoteAdministrationEnabled, 24
setThreadSettings, 28
RTI_DLUtility, 31
sleep, 31

setApplicationKind
RTI_DLOptions, 29
setDomainId
RTI_DLOptions, 25
setDomainParticipant
RTI_DLOptions, 23
setEchoToStdout
RTI_DLOptions, 30
setFilterLevel
RTI_DLDistLogger, 13
RTI_DLOptions, 25
setLogInfrastructureMessages
RTI_DLOptions, 30
setOptions
RTI_DLDistLogger, 14
setQosLibrary
RTI_DLOptions, 26
setQosProfile
RTI_DLOptions, 27
setQueueSize
RTI_DLOptions, 28
setRemoteAdministrationEnabled
RTI_DLOptions, 24
setRTILoggerPrintFormat
RTI_DLDistLogger, 14
setRTILoggerVerbosityByCategory
RTI_DLDistLogger, 15
setThreadSettings
RTI_DLOptions, 28
severe
RTI_DLDistLogger, 16
sleep
RTI_DLUtility, 31

trace
RTI_DLDistLogger, 18

warning
RTI_DLDistLogger, 17