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 Class Index **3**

2.1 Class List . . . 3

3 Class Documentation **5**

3.1 DistLogger Class Reference . . . 5

3.1.1 Detailed Description . . . 5

3.1.2 Member Function Documentation . . . 6

3.1.2.1 setOptions() . . . 6

3.1.2.2 getInstance() . . . 6

3.1.2.3 fatal() . . . 6

3.1.2.4 severe() . . . 7

3.1.2.5 error() . . . 7

3.1.2.6 warning() . . . 7

3.1.2.7 notice() . . . 8

3.1.2.8 info() . . . 8

3.1.2.9 debug() . . . 8

3.1.2.10 trace() . . . 9

3.1.2.11 log() [1/3] . . . 9

3.1.2.12 log() [2/3] . . . 9

3.1.2.13 log() [3/3] . . . 10

3.1.2.14 setRTILoggerVerbosityByCategory() . . . 10

3.1.2.15 setRTILoggerPrintFormat() . . . 11

3.1.2.16 getFilterLevel() . . . 11

3.1.2.17 setFilterLevel() . . . 11

3.1.2.18 isEnabled() . . . 12

3.1.2.19 getRtpsHostId() . . . 12

3.1.2.20 getRtpsAppId() . . . 13

3.1.2.21 delete() . . . 13

3.2 LogMessageData Class Reference . . . 13

3.2.1 Detailed Description . . . 13

3.2.2 Constructor & Destructor Documentation . . . 13

3.2.2.1 LogMessageData() [1/3] . . . 13

3.2.2.2 LogMessageData() [2/3] . . . 14

3.2.2.3 LogMessageData() [3/3] . . . 14

3.2.3 Member Data Documentation . . . 15

3.2.3.1 level . . . 15

3.2.3.2 message . . . 15
3.2.3.3 category . . . 15
3.2.3.4 writeTime . . . 15
3.3 Options Class Reference . . . 16
3.3.1 Detailed Description . . . 16
3.3.2 Constructor & Destructor Documentation . . . 16
3.3.2.1 Options() . . . 17
3.3.3 Member Function Documentation . . . 17
3.3.3.1 isEchoToStdout() . . . 17
3.3.3.2 setEchoToStdout() . . . 17
3.3.3.3 getLogInfrastructureMessages() . . . 17
3.3.3.4 setLogInfrastructureMessages() . . . 18
3.3.3.5 getDomainParticipant() . . . 18
3.3.3.6 setDomainParticipant() . . . 18
3.3.3.7 isRemoteAdministrationEnabled() . . . 18
3.3.3.8 setRemoteAdministrationEnabled() . . . 19
3.3.3.9 getFilterLevel() . . . 19
3.3.3.10 setFilterLevel() . . . 19
3.3.3.11 getDomainId() . . . 20
3.3.3.12 setDomainId() . . . 20
3.3.3.13 getQosLibrary() . . . 21
3.3.3.14 setQosLibrary() . . . 21
3.3.3.15 getQosProfile() . . . 21
3.3.3.16 setQosProfile() . . . 21
3.3.3.17 getThreadPriority() . . . 22
3.3.3.18 setThreadPriority() . . . 22
3.3.3.19 getQueueSize() . . . 22
3.3.3.20 setQueueSize() . . . 22
3.3.3.21 getApplicationKind() . . . 23
3.3.3.22 setApplicationKind() . . . 23
3.3.4 Member Data Documentation . . . 23
3.3.4.1 DEFAULT_QUEUE_SIZE . . . 23
3.4 Version Class Reference . . . 23
3.4.1 Detailed Description . . . 24
3.4.2 Member Function Documentation . . . 24
3.4.2.1 getInstance() . . . 24
3.4.2.2 toString() . . . 24

Index **25**

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 **com.rti.dl.Options** (p. 16). 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 **com.rti.dl.DistLogger.setOptions()** (p. 6).
3. Acquire an **com.rti.dl.DistLogger** (p. 5) instance using **com.rti.dl.DistLogger.getInstance()** (p. 6). 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: **com.rti.dl.DistLogger.error()** (p. 7), **com.rti.dl.DistLogger.fatal()** (p. 6),... 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 **com.rti.dl.DistLogger.setFilterLevel()** (p. 11) 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

Class Index

2.1 Class List

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

DistLogger 5
LogMessageData 13
Options 16
Version 23

Chapter 3

Class Documentation

3.1 DistLogger Class Reference

Public Member Functions

- void **fatal** (String message)
- void **severe** (String message)
- void **error** (String message)
- void **warning** (String message)
- void **notice** (String message)
- void **info** (String message)
- void **debug** (String message)
- void **trace** (String message)
- void **log** (int level, String message)
- void **log** (int level, String message, String category)
- void **log** (**LogMessageData** messageData)
- void **setRTILoggerVerbosityByCategory** (LogCategory category, LogVerbosity verbosity)
- void **setRTILoggerPrintFormat** (LogPrintFormat logPrintFormat)
- int **getFilterLevel** ()
- void **setFilterLevel** (int filterLevel)
- boolean **isEnabled** ()
- int **getRtpsHostId** ()
- int **getRtpsAppId** ()

Static Public Member Functions

- static boolean **setOptions** (**Options** options)
- static synchronized **DistLogger** **getInstance** ()
- static void **delete** ()

3.1.1 Detailed Description

This class provides an API to publish log messages to a DDS Topic. Generally, this class is used by an application logging infrastructure to integrate it into an application although the API can certainly be used directly as well.

3.1.2 Member Function Documentation

3.1.2.1 setOptions()

```
static boolean setOptions (
            Options options )  [static]
```

Sets the options used during creation of the Distributed Logger singleton. It is important to note that this method is only successful if the Distributed Logger singleton does not exist.

Parameters

options	The options to use during creation of the singleton.

Returns

true if successful, false otherwise.

3.1.2.2 getInstance()

```
static synchronized  DistLogger getInstance ( )  [static]
```

This method provides access to the singleton instance of this class. 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 this class.

3.1.2.3 fatal()

```
void fatal (
            String message )
```

Log a message at the fatal level.

Parameters

message	The log message.

References **DistLogger.log()**.

3.1.2.4 severe()

```
void severe (
            String message )
```

Log a message at the severe level.

Parameters

message	The log message.

References **DistLogger.log()**.

3.1.2.5 error()

```
void error (
            String message )
```

Log a message at the error level.

Parameters

message	The log message.

References **DistLogger.log()**.

Referenced by **DistLogger.delete()**.

3.1.2.6 warning()

```
void warning (
            String message )
```

Log a message at the warning level.

Parameters

message	The log message.

References **DistLogger.log()**.

3.1.2.7 notice()

```
void notice (
             String message )
```

Log a message at the notice level.

Parameters

message	The log message.

References **DistLogger.log()**.

3.1.2.8 info()

```
void info (
             String message )
```

Log a message at the info level.

Parameters

message	The log message.

References **DistLogger.log()**.

3.1.2.9 debug()

```
void debug (
             String message )
```

Log a message at the debug level.

Parameters

message	The log message.

References **DistLogger.log()**.

3.1.2.10 trace()

```
void trace (
            String message )
```

Log a message at the trace level.

Parameters

message	The log message.

References **DistLogger.log()**.

3.1.2.11 log() [1/3]

```
void log (
            int level,
            String message )
```

Log a message at the specified level.

Parameters

level	The level of the log message.
message	The log message.

References **DistLogger.log()**.

Referenced by **DistLogger.debug()**, **DistLogger.error()**, **DistLogger.fatal()**, **DistLogger.info()**, **DistLogger.log()**, **DistLogger.notice()**, **DistLogger.severe()**, **DistLogger.trace()**, and **DistLogger.warning()**.

3.1.2.12 log() [2/3]

```
void log (
            int level,
            String message,
            String category )
```

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

Parameters

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

References **DistLogger.log()**.

3.1.2.13 log() [3/3]

```
void log (
            LogMessageData messageData )
```

Log a message with the specified message data

Parameters

messageData	

References **Options.isEchoToStdout()**, **DistLogger.isEnabled()**, **LogMessageData.level**, and **LogMessage↩ Data.message**.

3.1.2.14 setRTILoggerVerbosityByCategory()

```
void setRTILoggerVerbosityByCategory (
            LogCategory category,
            LogVerbosity verbosity )
```

This method is mostly a pass-through to the appropriate call in the RTI DDS library but it also has the benefit of keeping the state of the logger in synch with this setting in the library (there is no other way to do so).

Parameters

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

References **DistLogger.isEnabled()**.

3.1.2.15 setRTILoggerPrintFormat()

```
void setRTILoggerPrintFormat (
            LogPrintFormat logPrintFormat )
```

This method is mostly a pass-through to the appropriate call in the RTI DDS library but it also has the benefit of keeping the state of the logger in synch with this setting in the library (there is no other way to do so).

Parameters

logPrintFormat	The new print format.

References **DistLogger.isEnabled()**.

3.1.2.16 getFilterLevel()

```
int getFilterLevel ( )
```

Gets the filter level.

Supported Filter Levels: See the IDL file provided with Distributed Logger (ndds.<version>/resource/rtidistlogger/idl/rtidistlog.idl)

The default value is INFO_LEVEL. See **setFilterLevel()** (p. 11) for more information on filter levels.

Returns

The current filter level.

3.1.2.17 setFilterLevel()

```
void setFilterLevel (
            int filterLevel )
```

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 INFO_LEVEL. Or see the IDL file provided with Distributed Logger (file: ndds.<version>/resource/rtidistlogger/idl/rtidistlog.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 Java, use com.rti.dl.INFO_LEVEL.VALUE.

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

filterLevel	The new log level.

The following table shows how Connext log levels map to Distributed Logger log filter levels. Not all Distributed Logger levels appear in this table because
there is not a one-to-one mapping between Connext and Distributed Logger levels (there are more levels in Distributed Logger).

Table 3.16 *Filter Levels*

Connext DDS API constant	Distributed Logger IDL constant
NDDS_CONFIG_LOG_FATAL_ERROR	FATAL_LEVEL
NDDS_CONFIG_LOG_LEVEL_ERROR	ERROR_LEVEL
NDDS_CONFIG_LOG_LEVEL_WARNING	WARNING_LEVEL
NDDS_CONFIG_LOG_LEVEL_STATUS_LOCAL	NOTICE_LEVEL
NDDS_CONFIG_LOG_LEVEL_STATUS_REMOTE	INFO_LEVEL
NDDS_CONFIG_LOG_LEVEL_DEBUG	DEBUG_LEVEL

3.1.2.18 isEnabled()

```
boolean isEnabled ( )
```

Returns

true if this logger is enabled, false otherwise.

Referenced by **DistLogger.log()**, **DistLogger.setRTILoggerPrintFormat()**, and **DistLogger.setRTILogger↩ VerbosityByCategory()**.

3.1.2.19 getRtpsHostId()

```
int getRtpsHostId ( )
```

Returns

The RTPS host id. This value comes directly from the `DomainParticipant` associated with this instance.

3.1.2.20 getRtpsAppId()

```
int getRtpsAppId ( )
```

Returns

The RTPS application id. This value comes directly from the `DomainParticipant` associated with this instance.

3.1.2.21 delete()

```
static void delete ( )  [static]
```

Delete DDS entities and all resources associated with this object. It is not possible to restart this object once it has been deleted.

References **DistLogger.error()**, and **Options.getLogInfrastructureMessages()**.

3.2 LogMessageData Class Reference

Public Member Functions

- **LogMessageData** (int **level**, String **message**, String **category**, Time_t **writeTime**)
- **LogMessageData** (int **level**, String **message**, String **category**)
- **LogMessageData** (int **level**, String **message**)

Public Attributes

- int **level**
- String **message**
- String **category**
- Time_t **writeTime**

3.2.1 Detailed Description

This class encapsulates all parameters that define a log message, including the message itself, log level, category and timestamp.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 LogMessageData() [1/3]

```
LogMessageData (
            int level,
            String message,
            String category,
            Time_t writeTime )
```

Convenience constructor to set all **LogMessageData** (p. 13) fields

Parameters

level	The level of the log message.
message	The message.
category	The category of the log message.
writeTime	The time the message was written

References **LogMessageData.category**, **LogMessageData.level**, **LogMessageData.message**, and **LogMessageData.writeTime**.

3.2.2.2 LogMessageData() [2/3]

```
LogMessageData (
            int level,
            String message,
            String category )
```

LogMessageData (p. 13) constructor, writeTime is set to current time.

Parameters

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

References **LogMessageData.category**, **LogMessageData.level**, and **LogMessageData.message**.

3.2.2.3 LogMessageData() [3/3]

```
LogMessageData (
            int level,
            String message )
```

LogMessageData (p. 13) constructor with default category.

Parameters

level	The level of the log message.
message	The message.

References **LogMessageData.level**, and **LogMessageData.message**.

3.2.3 Member Data Documentation

3.2.3.1 level

```
int level
```

The level of the log message.

Referenced by **DistLogger.log()**, and **LogMessageData.LogMessageData()**.

3.2.3.2 message

```
String message
```

The message to log. It must be non-NULL.

Referenced by **DistLogger.log()**, and **LogMessageData.LogMessageData()**.

3.2.3.3 category

```
String category
```

The category of the log message.

Referenced by **LogMessageData.LogMessageData()**.

3.2.3.4 writeTime

```
Time_t writeTime
```

The timestamp with which the message will be logged.

Referenced by **LogMessageData.LogMessageData()**.

3.3 Options Class Reference

Public Member Functions

- boolean **isEchoToStdout** ()
- void **setEchoToStdout** (boolean echoToStdout)
- boolean **getLogInfrastructureMessages** ()
- void **setLogInfrastructureMessages** (boolean log)
- **Options** ()
- DomainParticipant **getDomainParticipant** ()
- void **setDomainParticipant** (DomainParticipant domainParticipant)
- boolean **isRemoteAdministrationEnabled** ()
- void **setRemoteAdministrationEnabled** (boolean enableRemoteAdministration)
- int **getFilterLevel** ()
- void **setFilterLevel** (int filterLevel)
- int **getDomainId** ()
- void **setDomainId** (int domainId)
- String **getQosLibrary** ()
- void **setQosLibrary** (String qosLibrary)
- String **getQosProfile** ()
- void **setQosProfile** (String qosProfile)
- int **getThreadPriority** ()
- void **setThreadPriority** (int threadPriority)
- int **getQueueSize** ()
- void **setQueueSize** (int queueSize)
- String **getApplicationKind** ()
- void **setApplicationKind** (String applicationKind)

Static Public Attributes

- static final int **DEFAULT_QUEUE_SIZE** = 128

3.3.1 Detailed Description

This class contains the runtime options for the Distributed Logger. It is important to note that these options can only be set *before* the Distributed Logger has been instantiated.

Author

RTI

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Options()

```
Options ( )
```

The constructor to create a new instance of the **Options** (p. 16) class.

References **Options.DEFAULT_QUEUE_SIZE**.

3.3.3 Member Function Documentation

3.3.3.1 isEchoToStdout()

```
boolean isEchoToStdout ( )
```

Returns

true if echoing messages to stdout, false otherwise.

Referenced by **DistLogger.log()**.

3.3.3.2 setEchoToStdout()

```
void setEchoToStdout (
            boolean echoToStdout )
```

Parameters

echoToStdout	Set to true to echo messages to stdout, false otherwise.

3.3.3.3 getLogInfrastructureMessages()

```
boolean getLogInfrastructureMessages ( )
```

Returns

true if Distributed Logger should log messages produced by other RTI products (tools, middleware, etc)

Referenced by **DistLogger.delete()**.

3.3.3.4 setLogInfrastructureMessages()

```
void setLogInfrastructureMessages (
            boolean log )
```

Parameters

log	true if Distributed Logger should log messages produced by other RTI products (tools, middleware, etc)

3.3.3.5 getDomainParticipant()

```
DomainParticipant getDomainParticipant ( )
```

Returns

The `DomainParticipant` to be used for creating DDS entities. Note that this can be null.

3.3.3.6 setDomainParticipant()

```
void setDomainParticipant (
            DomainParticipant domainParticipant )
```

Set the `DomainParticipant` to use when creating DDS entities to support the distribution of log messages.

Parameters

domainParticipant	The `DomainParticipant` cannot be null.

3.3.3.7 isRemoteAdministrationEnabled()

```
boolean isRemoteAdministrationEnabled ( )
```

Returns

true if remote administration is enabled, false otherwise.

3.3.3.8 setRemoteAdministrationEnabled()

```
void setRemoteAdministrationEnabled (
             boolean enableRemoteAdministration )
```

Set whether or not remote administration will be used.

Parameters

enableRemoteAdministration	

3.3.3.9 getFilterLevel()

```
int getFilterLevel ( )
```

Get the filter level.

Supported Filter Levels: See the IDL file provided with Distributed Logger (ndds.<version>/resource/rtidistlogger/idl/rtidistlog.idl)

The default value is INFO_LEVEL. See **setFilterLevel()** (p. 19) for more information on filter levels.

Returns

The filter level. Messages with a less-verbose level will be filtered out and not sent.

3.3.3.10 setFilterLevel()

```
void setFilterLevel (
             int filterLevel )
```

Sets the filter level.

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 IDL file provided with Distributed Logger (file: ndds.<version>/resource/rtidistlogger/idl/rtidistlog.idl). The default value is INFO_LEVEL. 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 Java, use com.rti.dl.INFO_LEVEL.VALUE.

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

filterLevel	The new filter level. This value must be >= `SILENT_LEVEL.VALUE`.

The following table shows how Connext log levels map to Distributed Logger log filter levels. Not all Distributed Logger levels appear in this table because
there is not a one-to-one mapping between Connext and Distributed Logger levels (there are more levels in Distributed Logger).

Table 3.25 *Filter Levels*

Connext	Distributed Logger
NDDS_CONFIG_LOG_LEVEL_ERROR	ERROR_LEVEL
NDDS_CONFIG_LOG_LEVEL_WARNING	WARNING_LEVEL
NDDS_CONFIG_LOG_LEVEL_STATUS_LOCAL	NOTICE_LEVEL
NDDS_CONFIG_LOG_LEVEL_STATUS_REMOTE	INFO_LEVEL
NDDS_CONFIG_LOG_LEVEL_DEBUG	DEBUG_LEVEL

3.3.3.11 getDomainId()

```
int getDomainId ( )
```

Returns

The domain id which will be associated with all of the Distributed Logger topics.

3.3.3.12 setDomainId()

```
void setDomainId (
            int domainId )
```

Set the domain id to use for connecting to DDS.

Parameters

domain↩ Id	The new domain id must be >= 0.

3.3.3.13 getQosLibrary()

```
String getQosLibrary ( )
```

Returns

The QoS library name used when creating DDS entities.

3.3.3.14 setQosLibrary()

```
void setQosLibrary (
            String qosLibrary )
```

Set the QoS library name used when creating DDS entities.

Parameters

qosLibrary	The QoS library name.

3.3.3.15 getQosProfile()

```
String getQosProfile ( )
```

Returns

The QoS profile name used when creating DDS entities.

3.3.3.16 setQosProfile()

```
void setQosProfile (
            String qosProfile )
```

Set the QoS profile name used when creating DDS entities.

Parameters

qosProfile	The QoS profile name.

3.3.3.17 getThreadPriority()

```
int getThreadPriority ( )
```

Returns

The thread priority of the thread that publishes the log messages to DDS.

3.3.3.18 setThreadPriority()

```
void setThreadPriority (
            int threadPriority )
```

Set the thread priority of the publishing thread.

Parameters

threadPriority	Must be >= `Thread.MIN_PRIORITY` and <= `Thread.MAX_PRIORITY`.

3.3.3.19 getQueueSize()

```
int getQueueSize ( )
```

Returns

The queue size used to buffer log messages until they are written to DDS.

3.3.3.20 setQueueSize()

```
void setQueueSize (
            int queueSize )
```

Sets the queue size.

Parameters

queueSize	Must be > 0.

3.3.3.21 getApplicationKind()

```
String getApplicationKind ( )
```

Returns

The application kind string.

3.3.3.22 setApplicationKind()

```
void setApplicationKind (
            String applicationKind )
```

Set the application kind string. This string can be used for any purpose that is useful or interesting.

Parameters

applicationKind	A string to be associated with the Distributed Logger. The maximum length of the string is 255.

3.3.4 Member Data Documentation

3.3.4.1 DEFAULT_QUEUE_SIZE

```
final int DEFAULT_QUEUE_SIZE = 128  [static]
```

The default size for the queue that stores log messages waiting to be written to DDS.

Referenced by **Options.Options()**.

3.4 Version Class Reference

Public Member Functions

- String **toString** ()

Static Public Member Functions

- static **Version** **getInstance** ()

3.4.1 Detailed Description

This class contains the version number of Distributed Logger.

Author

RTI

3.4.2 Member Function Documentation

3.4.2.1 getInstance()

```
static Version getInstance ( ) [static]
```

Get the **Version** (p. 23) instance

3.4.2.2 toString()

```
String toString ( )
```

Return the Distributed Logger version number as a String.

Index

category
    LogMessageData, 15

debug
    DistLogger, 8
DEFAULT_QUEUE_SIZE
    Options, 23
delete
    DistLogger, 13
DistLogger, 5
    debug, 8
    delete, 13
    error, 7
    fatal, 6
    getFilterLevel, 11
    getInstance, 6
    getRtpsAppId, 12
    getRtpsHostId, 12
    info, 8
    isEnabled, 12
    log, 9, 10
    notice, 8
    setFilterLevel, 11
    setOptions, 6
    setRTILoggerPrintFormat, 10
    setRTILoggerVerbosityByCategory, 10
    severe, 7
    trace, 9
    warning, 7

error
    DistLogger, 7

fatal
    DistLogger, 6

getApplicationKind
    Options, 23
getDomainId
    Options, 20
getDomainParticipant
    Options, 18
getFilterLevel
    DistLogger, 11
    Options, 19
getInstance
    DistLogger, 6
    Version, 24
getLogInfrastructureMessages
    Options, 17
getQosLibrary
    Options, 21
getQosProfile
    Options, 21
getQueueSize
    Options, 22
getRtpsAppId
    DistLogger, 12
getRtpsHostId
    DistLogger, 12
getThreadPriority
    Options, 22

info
    DistLogger, 8
isEchoToStdout
    Options, 17
isEnabled
    DistLogger, 12
isRemoteAdministrationEnabled
    Options, 18

level
    LogMessageData, 15
log
    DistLogger, 9, 10
LogMessageData, 13
    category, 15
    level, 15
    LogMessageData, 13, 14
    message, 15
    writeTime, 15

message
    LogMessageData, 15

notice
    DistLogger, 8

Options, 16
    DEFAULT_QUEUE_SIZE, 23
    getApplicationKind, 23

getDomainId, 20
getDomainParticipant, 18
getFilterLevel, 19
getLogInfrastructureMessages, 17
getQosLibrary, 21
getQosProfile, 21
getQueueSize, 22
getThreadPriority, 22
isEchoToStdout, 17
isRemoteAdministrationEnabled, 18
Options, 16
setApplicationKind, 23
setDomainId, 20
setDomainParticipant, 18
setEchoToStdout, 17
setFilterLevel, 19
setLogInfrastructureMessages, 17
setQosLibrary, 21
setQosProfile, 21
setQueueSize, 22
setRemoteAdministrationEnabled, 18
setThreadPriority, 22

setApplicationKind
Options, 23
setDomainId
Options, 20
setDomainParticipant
Options, 18
setEchoToStdout
Options, 17
setFilterLevel
DistLogger, 11
Options, 19
setLogInfrastructureMessages
Options, 17
setOptions
DistLogger, 6
setQosLibrary
Options, 21
setQosProfile
Options, 21
setQueueSize
Options, 22
setRemoteAdministrationEnabled
Options, 18
setRTILoggerPrintFormat
DistLogger, 10
setRTILoggerVerbosityByCategory
DistLogger, 10
setThreadPriority
Options, 22
severe
DistLogger, 7
toString
Version, 24
trace
DistLogger, 9

Version, 23
getInstance, 24
toString, 24

warning
DistLogger, 7
writeTime
LogMessageData, 15