The purpose of DDS logger is to non-invasively log events occurring in DDS domains. The events include registration of new type, discovery of new topics, addition on new participants, subscription to topics, and messages between publishers and subscribers.

Version 1.0:

1. The logger is supported only on Windows 7 platform and with RTI Connext DDS.
2. Every data type supported by RTI Connext DDS (except WSTRING) is captured and logged by the logger.
3. Only events occurring in DDS domain 0 are captured and logged by the logger.
1 Install RTI Connext ................................................................. 2
2 Install Java ............................................................................... 3
3 Install Eclipse ........................................................................... 4
4 Install OpenICE ......................................................................... 4
5 Install DDS Logger ...................................................................... 5
6 Using the Logger with OpenICE .................................................. 6
7 Understanding the Log File ......................................................... 6
7.1 Log Format .............................................................................. 7

1 Install RTI Connext

1. Navigate to www.rti.com/downloads/connext-files.html in a web browser. This page should list various bundles of RTI Connext 5.1.0 software.

2. Depending on the processor architecture of your Windows box – 32-bit (x86) v/s 64-bit (x64), click the appropriate download link for Windows.

3. Download and run the installer or run directly from Oracle. The software should get installed into C:\Program Files\RTI. We refer to the above folder as RTI directory.

   Note: Depending on the processor architectures of RTI Connext your Windows box, the software might get installed into c:\Program Files (x86)\RTI folder. Please check on your system.

4. The bundle will place rti_license.dat file in the RTI directory. If this is an evaluation license, it will have limited validity. Please contact RTI for a license with longer validity (and place it in RTI directory.)

5. To verify the installation of RTI Connext, do the following.
   a. Launch RTI through the installer or double click on the RTI Launcher 5.1.0 shortcut on the desktop (or execute <RTI directory>\RTI_Launcher_5.1.0\bin\i386Win32VC2005.exe). This will launch RTI Launcher application and bring up a window with four tabs.
   b. Click on the Utilities tab.
   c. Click on Shapes Demo. If you are requested to authorize network access for the application (via Windows Security Alert), click Allow Access. This will open a demo window.
   d. In the demo window, click on Square under Publish and click Ok in the next dialog. You should see a blue square moving in the demo window.
2 Install Java

1. Download Java JDK compatible (depending on the processor architecture of the Windows box – 32-bit (x86) v/s 64-bit (x64)) from http://www.oracle.com/technetwork/java/javase/downloads/.

2. Run the downloaded installer. The path for Java should be `C:/Program Files/Java/jdk[version #]`. For example, if you installed JDK version 8 update 20, the path to Java will be `C:\Program Files\Java\jdk1.8.0_20`.

Note: Navigate to the folder to find the exact version number. Also, depending on the processor architectures of JDK and your Windows box, the software might get installed into `c:\Program Files (x86)\Java` folder. Please check on your system.

3. Open Control Panel, go to System and Security, click System, click Advanced System Settings, click Advanced tab, and click Environment Variables button.

4. Add two User variables:
   - Variable: HOME
     Value: %USERPROFILE%
   - Variable: JAVA_HOME
     Value: `C:\Program Files\Java\jdk[version #]`

5. To verify the installation and variable configuration, open the Command Prompt and type "`%JAVA_HOME%\bin\java.exe -version`" into the command prompt (with the quotes). If the executable is found and it displays the version of Java, then the set up was successful.

e. Next, click on Square under Subscribe and click Ok in the next dialog. You should see a blue square with red trails. If so, the installation was successful!!
3 Install Eclipse


2. Install Eclipse by unzipping the bundle at c:\. This will create the folder c:\eclipse.

3. Start Eclipse by clicking on eclipse.exe in c:\eclipse. Go to Help menu and click Install New Software. Select Luna from the Work with: drop down list. Type Marketplace in the search box. After the list populates, install Marketplace Client by selecting Marketplace Client and clicking on Next, Accepting the license and Finish. If you are prompted to restart eclipse, click yes.

4. When Eclipse restarts, click on Eclipse Marketplace in Help menu. In the Eclipse Marketplace dialog, type Gradle Integration in the Find box and click Go, or press Enter.

5. Click Install button for the Gradle Integration for Eclipse (4.4) 3.6.2 Release. Click Confirm/Finish buttons in the next dialogs. This will install Gradle Integration plugin. Ignore warnings such as: “The following solutions are not available: Gradle Integration for Eclipse (4.4) 3.6.2 RELEASE. Proceed?”.

4 Install OpenICE


2. Unzip the downloaded file in c:\. This will create a folder with naming similar to mdpnpc-code-*. Rename this folder to mdpnpc-code.

3. Ensure that you are connected to the Internet; preferably, via a fast connection.

   Open a Command Prompt and navigate to c:\mdpnpc-code. Execute .\gradlew.bat. This will setup Gradle required to build OpenICE.

   This step can take a while as various required software and libraries will be downloaded from the web.

4. Open Eclipse.

5. Click Import in File menu.

6. Click Gradle followed by Gradle Project and Next.

7. Browse to c:\mdpnpc-code as the root folder.

8. Click Build Model.

9. Select the root project mdpnpc-code.

10. Click Finish to import the projects. Again, this step can take a while as various software and libraries are downloaded from the Internet.
11. To verify OpenICE is functioning correctly
   a. Open Demo Apps project.
   c. Right click on Main.java and click on Run As->Java Application.
   d. In the displayed dialog, select ICE_Device_Interface as the Application and Simulated Pulse Oximeter as the Device Type, and click Start Simulated Pulse Oximeter.
   e. If you see a dialog similar to the one below, then you have a functioning installation of OpenICE!!

5  Install DDS Logger

2. Unzip the bundle in c:\. It will create the folder c:\dds-logger with the following structure. Make sure there isn’t an extra folder such as dds-logger-v#.
   a. libs folder contains required libraries.
   b. output folder will contains generated logs.
   c. Licenses folder will contain library licenses.
   d. startLogger.bat.
3. To start the logger, double click on the c:\dds-logger\startLogger.bat file. This will start the logger in a command interpreter (prompt) window.
4. To stop the logger, press enter in the command interpreter window.

5. To verify the installation, start and stop the logger and check for the presence of a log file in c:.dds-logger\output folder.

6 Using the Logger with OpenICE

1. Start the logger, double click on the c:.dds-logger\startLogger.bat file. This will start the logger in a command interpreter (prompt) window.

2. Open Eclipse and run the OpenICE demo:
   a. Open Demo Apps project.
   c. Right click on Main.java and click on Run As->Java Application.
   d. In the displayed dialog, select ICE_Device_Interface as the Application and Simulated Pulse Oximeter as the Device Type, and click Start Simulated Pulse Oximeter.

3. Exit the OpenICE demo by closing the window and exit the logger by pressing Enter in the command interpreter (prompt) window.

4. To verify the OpenICE log file, check for the presence of a log file in c:.dds-logger\output folder that matches the name of the log file displayed in the logger command interpreter (prompt) window.

7 Understanding the Log File

The log file is written in JSON format. It is recommended to use a JSON formatter (e.g., http://www.jsoneditoronline.org) for reformatting the JSON output.

In every log, the first entry is always the log creation date. The entry is an array of events of the following kinds: Participant Connection, Type Registration, Topic Subscription, and Messages.

Here is an example Event:

```json
{
    "Event_Type": "Message",
    "Member_Count": 6,
    "Type_Name": "ice::Numeric",
    "Milliseconds": 1414427826241,
    "Members": [
      {
        "Data": "CuCMmSNP1NB8egcuwHMbgl9gWvh2kDYk0kp6",
        "Name": "unique_device_identifier",
        "Kind_Name": "string"
      },
      {
        "Data": "MDC_PULS_OXIM_SAT_O2",
        "Name": "metric_id",
        "Kind_Name": "string"
      },
      {"Data": 0,
```
Each message contains a timestamp (Milliseconds) of when the logger received the message.

7.1 Log Format

The log has 2 base objects: Log Start is start time of the log and Events is an array of logged events.

There are 5 types of event objects that can appear in the Events array:

1. Participant Connected event occurs when a new participant connects to DDS. It contains the domain ID, participant key, participant name, and the millisecond timestamp of the event.
2. Registered Type event occurs when a new data type is registered in the DDS. It contains the type name, kind of the type, number of members, members, and the millisecond timestamp of the event. The members can be more types.
3. Discovered Topic event occurs when a topic starts publishing data. It contains the topic name, type name, participant key, and the millisecond timestamp of the message.
4. Topic Subscription event occurs when a participant subscribes to a topic. It contains the topic name, type name, participant key and the millisecond timestamp of the message.
5. *Message* event occurs for every message that gets sent on a topic. It contains the type name, kind of type, number of members, and the millisecond timestamp of the message. In addition, it also contains an array of members. Each member object contains the member name, data/value, and data kind/type. The data field changes according to the data type.

Among the following data types supported by DDS, logger supports the data types in black. Other data types will be supported in a future release.

- `TK_ALIAS`
- `TK_STRING`
- `TK_CHAR`
- `TK_LONG`
- `TK_LONGLONG`
- `TK_BOOLEAN`
- `TK_WCHAR`
- `TK_DOUBLE`
- `TK_LONGDOUBLE`
- `TK_FLOAT`
- `TK_SHORT`
- `TK_USHORT`
- `TK_OCTET`
- `TK_STRUCT`
- `TK_ENUM`
- `TK_ARRAY`
- `TK_SEQUENCE`
- `TK_UNION`
- `TK_VALUE`
- `TK_WSTRING`

For unsupported data types, only its type (no data) is reported in the log.

*Log*: Object

- **Log Start**: Integer. This provides the logging start time as the number of milliseconds elapsed since Jan 1, 1970.
- **Events**: Array. Every event has a field named *Event_Type*. The domain of values for this field along with the corresponding sub-fields in the event is given below.

*Participant Connected*: Object

- **Domain_ID**: Integer
- **Milliseconds**: Integer. This provides the time when the event was logged (as the number of milliseconds elapsed since Jan 1, 1970).
- **Participant_Key**: Array of Integers
- **Participant_Name**: String

*Registered Type*: Object

- **Kind_Name**: String
- **Milliseconds**: Object
- **Member_Count**: Integer
- **Members**: Array of Types
- **Name**: String

*Discovered Topic*: Object

- **Milliseconds**: Integer
- **Participant_Key**: Array of Integers
- **Topic_Name**: String
- **Type_Name**: String

*Topic Subscription*: Object

- **Milliseconds**: Integer
- **Participant_Key**: Array of Integers
- **Topic_Name**: String
- **Type_Name**: String

*Message*: Object

- **Kind_Name**: String
- **Member_Count**: Integer
- **Members**: Array. The element objects contain the following fields.
  - **Data**: The type of the value depends on the kind of the member.
  - **Kind_Name**: String
  - **Name**: String
Milliseconds: Integer
Type_Name: String