DDS Touchstone Benchmark Suite

Version 1.1.3 beta

User Guide

PRISMTECH

Copyright Notice

IMPORTANT Note to Readers:

This is a beta version tutorial and is partially incomplete.

Tutorial Screen shots contained within this document where created with a pre-release version of DDS TouchStone. This version of Touchstone was project named “Pother”.

These screen shots will be corrected with the Touchstone named version for the GA release of the product.

Within Chapter 4 descriptions labelled ** have yet to be reviewed and completed.

Additional note this document is subject to change and review.

Table of Contents

1 Building DDS Touchstone

This chapter will explain how to install and integrate DDS Touchstone Benchmark Suite with OpenSplice DDS.

1.1 Prerequisites

In order to run DDS Touchstone with OpenSplice DDS an OpenSplice implementation must first be installed and configured on the target machine. The OpenSplice prerequisites are also required for DDS Touchstone, details of these can be found in the OpenSplice release notes.

For information on installing and configuring OpenSplice DDS see the Getting Started manual.

1.2 Installation for UNIX platforms

1. Unpack the DDS Touchstone distribution in a new directory on the target machine. For the rest of this section this directory will be referred to as <Touchstone_Home>

2. To build Touchstone first run the release.com script in the root OpenSplice directory, this will set the OpenSplice environment variables and allows Touchstone to build against the required DDS libraries.

3. Before building the Touchstone Tools the build scripts must be configured to the target machine platform. This can be done by using the provided configuration script (configure.sh) in the <Touchstone_HOME> directory.

4. Open a new shell in the <Touchstone_HOME> directory.

5. Use the following command to run the configuration script:

   ```
   . configure.sh
   ```

6. The script will then require the DDS vendor and Operating System abstraction implementation details. We recommend that for most Unix systems the 'OpenSpliceDDS' vendor and then the 'posix' OS abstraction be selected from the following lists:

   ```
   Available DDS vendors are:
   1 > OpenSpliceDDS
   Please select a DDS vendor number:

   Available OS abstraction implementations are:
   1 > OpenSpliceDDS
   2 > posix
   Please select an OS abstraction implementation number:
   ```

7. The next menu selects the DDS Touchstone tools to be installed.

   ```
   Available target components are:
   1 > touchstone_c
   2 > touchstone_cpp
   3 > touchstone_java
   4 > tools
   5 > <done>
   Select desired target component by number:
   ```

   In order to choose the desired tool set continue to pick tools from the menu list. Select <done> once this step is complete.

8. The Make files have now been configured for OpenSplice DDS for the Posix platform with the requested tools set. Finally the script will ask for a configuration name.

   ```
   Please enter a symbolic name for this configuration [g++.4.1.2]:
   ```

   Touchstone supports the use of multiple configurations. This name will be used to identify and to apply these environmental and internal Touchstone settings.

9. The Touchstone Tools can now be built by running the provided Makefile.

   ```
   gmake CONFIG_NAME=<chosen configuration name>
   ```

   Please note that the default configuration will be the first found by the makefile. This occurs when no <config_name> is specified.

10. Upon completion the Makefile will have created the following new executables in the <Touchstone_HOME>/bin/<config_name> directory:

    ```
    touchstone_c
    touchstone_cpp
    touchstone.jar
    watcher
    spotter
    errorLog
    recorder
    excelerator
    ```

    Also a new release.<config_name>.com will appear in the <Touchstone_HOME> directory this script sets the environment for the new Touchstone configuration. This script should be sourced before running any Touchstone application.

1.3 Installation for Windows platforms

1. In order to build DDS Touchstone on a Windows platform Visual Studio 7 or above must be installed and configured on the target machine. For further details on how to do this please refer to the Microsoft Visual Studio installation documentation.

2. Unpack the DDS Touchstone distribution in a new directory on the target machine. For the rest of this section this directory will be referred to as <DDSTouchstone_Home>

3. OpenSplice DDS must be correctly referenced in the environment variables. The environment variables should have been automatically configured as part of the OpenSplice windows installation.

4. To build the tools first open a command prompt in the <DDSTouchstone_HOME> directory.

5. Then run the following batch file:

   ```
   MakeWin.bat
   ```

6. Upon completion the MakeWin.bat batch file will have created the following Touchstone Tool executables in the bin directory:

   ```
   touchstone_c
   touchstone_cpp
   touchstone.jar
   watcher
   spotter
   errorLog
   recorder
   excelerator
   ```

2 Running Simple Touchstone Scenarios

DDS Touchstone is pre-built with examples scenarios that can produce benchmarking results for the following areas:

- Throughput
- Latency
- Latency with Threading Priorities

These scenarios are stored in Touchstone recording files, located in the <Touchstone_HOME>/examples directory.

To run an example scenario following the instructions given in the README files provided within each example directory.

The recordings file are in a plain test file format (with .dat extensions) and can be examined and edited create new test beds using a standard text editor.

2.1 Touchstone start scripts

In the <Touchstone_HOME>/bin there are scripts provided to help start and run Touchstone instances and the tools. These scripts invoke the Touchstone executables built in the previous chapter using a combination of default and user defined variables.

Before running any of the start scripts the release.<config_name>.com should be run in the appropriate shell. This will add <Touchstone_HOME>/bin to the PATH allowing the executables to be run from any location.

2.1.1 Touchstone C

The command to start a C instance of Touchstone is as follows:

```
touchstone_c <application_id> [<group_id>]
```

The application id and group id will be set to the values provided. When a group id is not provided then the group id will be set to the same value as the application id.

2.1.2 Touchstone C++

The command to start a C++ instance of Touchstone is as follows:

```
touchstone_cpp <application_id> [<group_id>]
```

The application id and group id will be set to the values provided. When a group id is not provided then the group id will be set to the same value as the application id.

2.1.3 Touchstone Java

The command to start a Java instance of Touchstone is as follows:

```
java touchstone.Main <application_id> [<group_id>]
```

The application id and group id will be set to the values provided. When a group id is not provided then the group id will be set to the same value as the application id.

2.1.4 start_touchstones.sh

This script is designed to allow simple uniform and batch creation of Touchstone instances. The script takes the following arguments:

```
start_touchstone.sh <language> <no#> <group id> <start id>
```

Where:

<language>	Selects the language for the touchstones. Choices are: c / cpp / java
<no#>	Defines number of touchstones to be created
<group id>	Group id for this collection of touchstones
<start id>	Starting id for touchstone instances

2.1.5 start_recording.sh

This is used to select and automatically run a Touchstone recording file. With this script recording files can only be read and are protected against being overwritten.

```
start_recording.sh <recording_file_name> [recorder_id]
```

Where:

<recording_file_name>	recording location
[recorder_id]	sets the recorder id, this is optional the default value is set to 99

This script uses the following recorder options:

--autoplay	Automatically begins to run the scenario.
-v	The verbose option provides screen output displaying the recorder actions.

3 Creating Touchstone Scenarios Tutorial

The easiest method of using DDS Touchstone is with the OpenSplice DDS Tuner tool. The Tuner provides the facilities for monitoring and controlling OpenSplice and the applications that use OpenSplice for the distribution of data. With the added functionality provided by the Touchstone Tools, the Tuner also becomes a flexible run-time interface for benchmark frameworks. For further details on using the OpenSplice DDS Tuner tool check the OpenSplice Tuner user manual, located in the OpenSplice DDS distribution.

This chapter will demonstrate how to create a simple DDS benchmark scenario using the DDS Touchstone along with the OpenSplice Tuner tool.

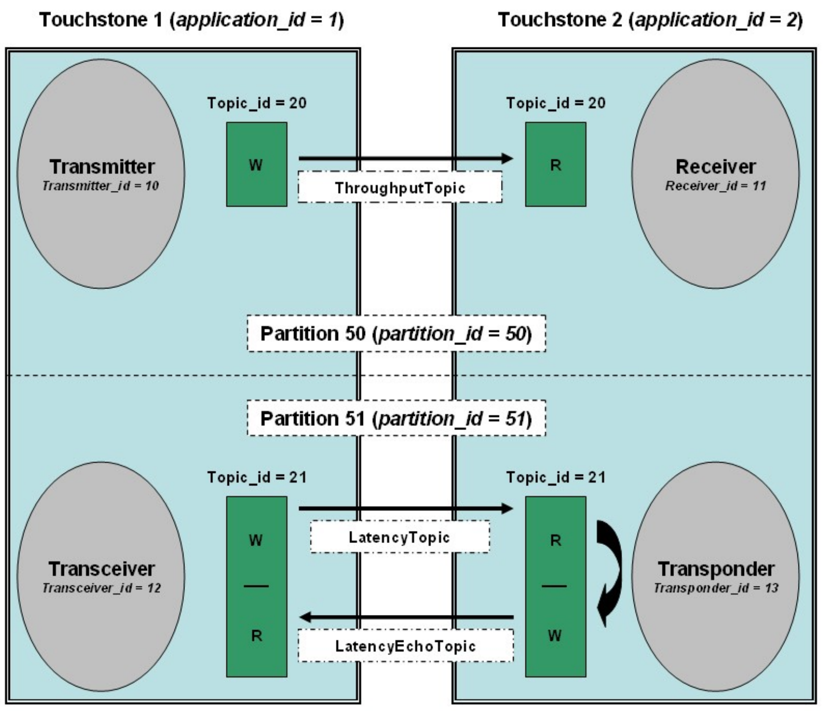


Figure 1: Tutorial Framework Abstract

This scenario example will include a Transmitter / Receiver pairing and a Transceiver / Transponder pairing.

Two Touchstone tool instances will be used, one Touchstone (application_id and group_id = 1) for the Transceiver and the Transmitter and a second Touchstone (application_id and group_id = 2) for the Transponder and Receiver.

There will also be two Partitions. The first will be for the Transmitter / Receiver pairing, the second for the Transceiver / Transponder pairing. In this framework partitions ids will start at 50.

The Transmitter / Receiver setup in Partition 50 will be used to perform Efficiency & Scalability benchmark testing using the Touchstone Throughput Topic types.

The Transceiver / Transponder configuration in Partition 51 will be used to perform Determinism & Dependability benchmark testing using the Touchstone Latency Topic types.

The resulting test reports will be collected and displayed using the Watcher tool

3.1 Starting OpenSplice DSS and the Tuner Tool

1. First set the OpenSplice environment variables. This can be done by running the following command in the root OpenSplice directory:

   ```
   . release.com
   ```

2. Start the OpenSplice daemon using:

   ```
   ospl start
   ```

3. The following command is used to start the OpenSplice Tuner tool:

   ```
   ospltun
   ```

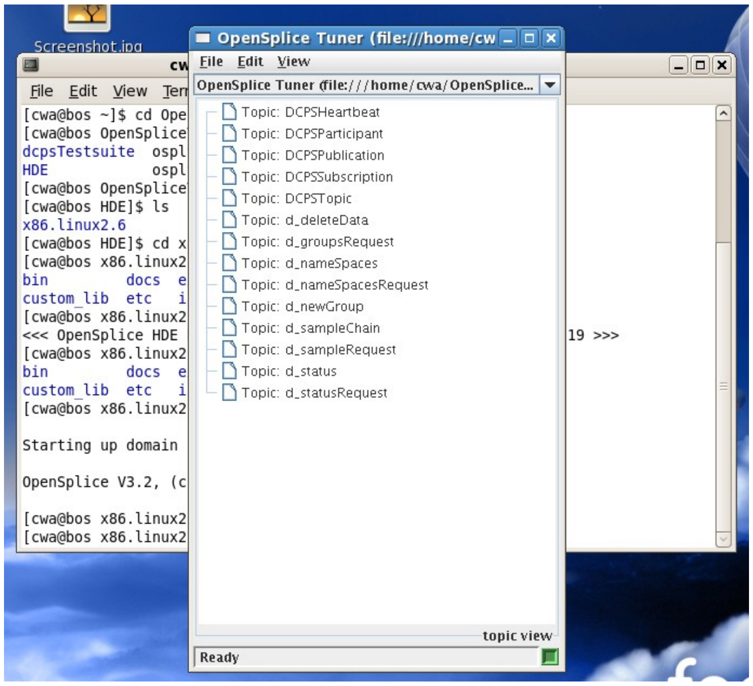
4. Once the Tuner program has been loaded, it will need to be connected to the OpenSplice instance started on the current machine.

   This is done by selecting from the Connect from the File menu.

5. Set the Domain URI in the Open Connection dialog box. This must be set to OpenSplice configuration file ospl.xml, which located in:

   <OSPL_HOME>/etc/config/ospl.xml
   (Where <OSPL_HOME> is the root OpenSplice DDS directory).

6. When this is set click OK to connect the Tuner


Screenshot.jpg
cw
File Edit View Ter
OpenSplice Tuner (file:///home/cw
File Edit View
OpenSplice Tuner (file:///home/cwa/OpenSplice...
topic view
Ready

```
[cwa@bos ~]$ cd Ope
[cwa@bos OpenSplice
dcpsTestsuite  ospl
HDE            ospl
[cwa@bos OpenSplice
[cwa@bos HDE]$ ls
x86.linux2.6
[cwa@bos HDE]$ cd x
[cwa@bos x86.linux2
bin         docs  e
custom_lib  etc   i
[cwa@bos x86.linux2
<<< OpenSplice HDE
[cwa@bos x86.linux2
bin         docs  e
custom_lib  etc   i
[cwa@bos x86.linux2

Starting up domain

OpenSplice V3.2, (c

[cwa@bos x86.linux2
[cwa@bos x86.linux2
```

```
19 >>>
```

- Topic: DCPSHeartbeat
- Topic: DCPSParticipant
- Topic: DCPSPublication
- Topic: DCPSSubscription
- Topic: DCPSTopic
- Topic: d_deleteData
- Topic: d_groupsRequest
- Topic: d_nameSpaces
- Topic: d_nameSpacesRequest
- Topic: d_newGroup
- Topic: d_sampleChain
- Topic: d_sampleRequest
- Topic: d_status
- Topic: d_statusRequest

Figure 2: OpenSplice Tuner

Figure 2 shows the OpenSplice Tuner connected to OpenSplice DDS before using the DDS Touchstone Tools.

3.2 Touchstone Tool

The Touchstone Tool is the core program of the DDS Touchstone suite. Touchstone provides the topics that will enable the control of the Transmitter, Receiver, Transceiver and Transponder components. With these components the user can create tests that will demonstrate determinism and efficiency benchmarking scenarios.

3.2.1 Application ids and Group ids

A Touchstone instance will have two references that a command can use to identify it. The Touchstone Application id is used to uniquely identify the Touchstone instance. Touchstone instances can also be grouped together using Touchstone Group id references. Touchstone instances that share a common group id are in the same grouping.

The advantage of using Touchstone groupings is that a single command can be issued simultaneously to a collection of Touchstone instances.

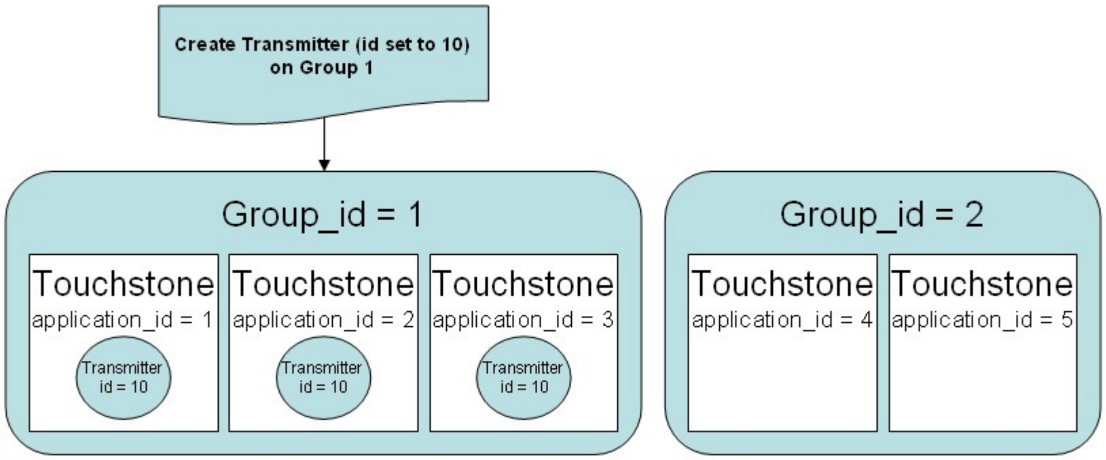


Figure 3 Touchstone Instance Grouping

The Figure 3 example shows when issuing a command to create a transmitter component on Group 1, all the Touchstone instances within that group will create a new transmitter. Touchstone 1, 2 and 3 now all have a new transmitter with their transmitter ids set to 10.

3.2.2 Starting Touchstone with the OpenSplice Tuner

1. To start the Touchstone Tool with the OpenSplice Tuner, the tuner first needs to be started and connected to an OpenSplice DDS daemon.

2. The Touchstone application links to some of the OpenSplice DDS libraries. The release.com script in the OpenSplice distribution must first be run to allow this:

   `cd <OSPL_HOME> -` where <OSPL_HOME> is the OpenSplice root directory
   `. release.com`

   Also in order to set the Touchstone run the release.<config_name>.com from the <Touchstone_HOME> directory.

   `cd <Touchstone_HOME> -` where <Touchstone_HOME> is the Touchstone root directory

   `. release.<CONFIG_NAME>.com`

3. The environment variables have now been set and Touchstone executables have been added to the $PATH. Create and move to a new work directory.

4. As multiple touchstone instances can be used in a single scenario, an Application ID and a Group ID must be provided so as to identify the Touchstone instance within the framework.

   The Touchstone Tool can be started with the following command:

   ```
   touchstone_c <application_id> [<group_id>]
   ```

   For the duration of this tutorial the C version of Touchstone will be referenced. However these instructions are generic and transferable to all other language implementations of Touchstone. (Use “`touchstone_cpp`” for C++ and “`java touchstone.Main`” for java)

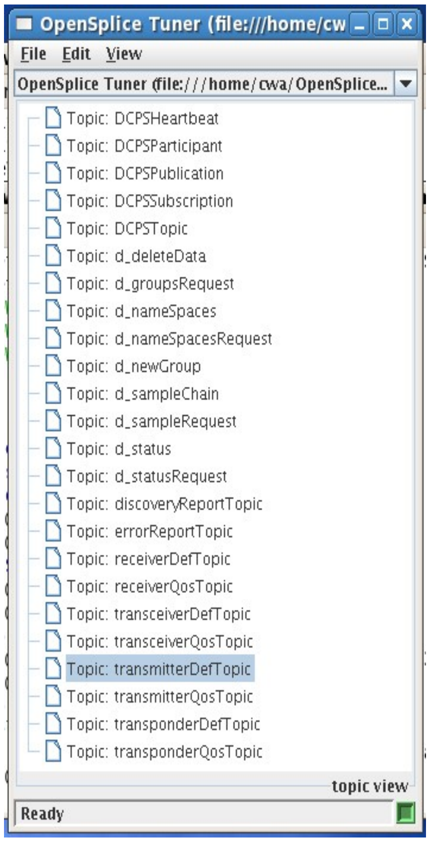
   The application id and group id will be set to the values provided. When a group id is not provided then the group id will be set to the same value as the application id.

   For this tutorial the first Touchstone instance (with an application_id and a group_id set to 1) can be created with the following command:

   ```
   touchstone_c 1
   ```

3.2.3 Touchstone Topics

Once started, the Touchstone Tool will inject the DDS Touchstone topic definitions into the OpenSplice middleware. To view these new topics select ‘Refresh Entity Tree’ in the Tuner Edit Menu.


OpenSplice Tuner (file:///home/cw
File Edit View
OpenSplice Tuner (file:///home/cwa/OpenSplice...
topic view
Ready

```
Topic: DCPSHeartbeat
Topic: DCPSParticipant
Topic: DCPSPublication
Topic: DCPSSubscription
Topic: DCPSTopic
Topic: d_deleteData
Topic: d_groupsRequest
Topic: d_nameSpaces
Topic: d_nameSpacesRequest
Topic: d_newGroup
Topic: d_sampleChain
Topic: d_sampleRequest
Topic: d_status
Topic: d_statusRequest
Topic: discoveryReportTopic
Topic: errorReportTopic
Topic: receiverDefTopic
Topic: receiverQosTopic
Topic: transceiverDefTopic
Topic: transceiverQosTopic
Topic: transmitterDefTopic
Topic: transmitterQosTopic
Topic: transponderDefTopic
Topic: transponderQosTopic
```

Figure 4: Tuner with the Test Suite Topics

As shown in Figure 4 there are now additional topics in topic view from the previously selection shown in Figure 2. These have been created and added by the Touchstone tool.

The Touchstone tool has now been successfully integrated with the OpenSplice Tuner

3.2.4 Touchstone Partitions

The Touchstone tool also adds new partitions into the OpenSplice middleware where the DDS Touchstone topics will reside. These can be examined in the partition view of the Tuner.

The DDSTouchStoneCommands partition contains topics that can be used to create, control and configure the framework infrastructure components such as Transmitters, Receivers, Transceivers and Transponders.

The DDSTouchStoneReports partition contains the report topic types that will allow components to publish information. These reportTopic types can then be subscribed to using the Watcher, Excelerator, Errorlog and Spotter tools.

3.3 Creating a Transmitter / Receiver Pairing

This section will demonstrate how to use the Definition Topics to create the Transmitter and Receiver components for this tutorial.

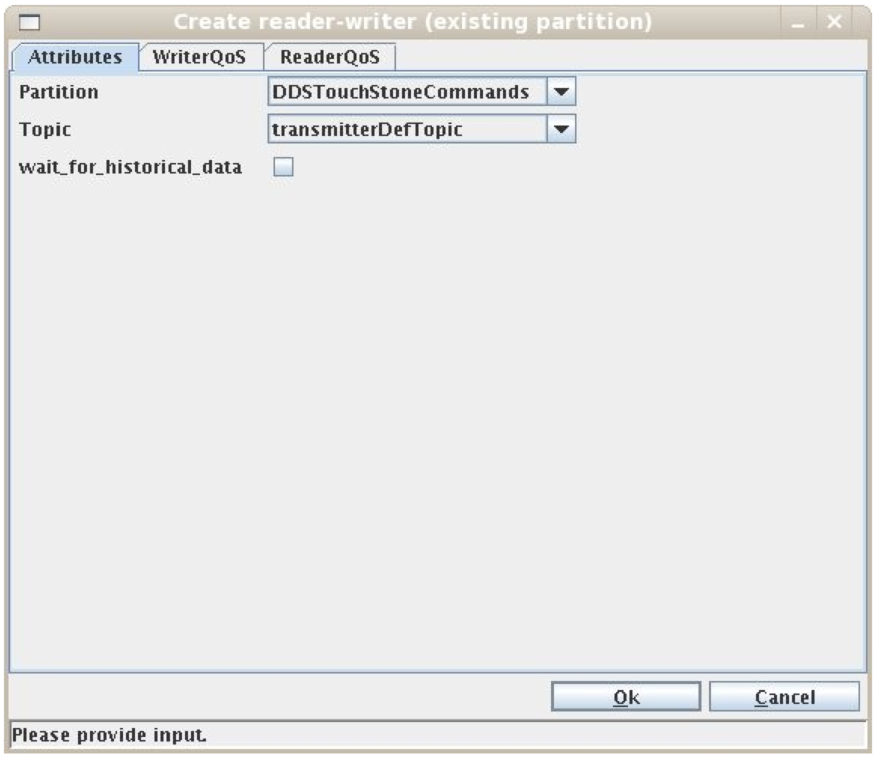
Each component has two topics associated with it in the DDSTouchStoneCommands partition, a DefTopic and a QosTopic. The DefTopic is used to create and control the component. The QosTopic can be used to configure the Qos settings for each component.

When used with the OpenSplice tuner the Touchstone DefTopic types provides a flexible interface to create and configure custom benchmark frameworks.

3.3.1 Creating a Transmitter

Steps 1-3 from this section can be used to create the other Touchstone components, by simply substituting DefTopic values where necessary.

1. To create a new Transmitter select Edit -> Create Reader-Writer -> Existing Partition from the Tuner menu

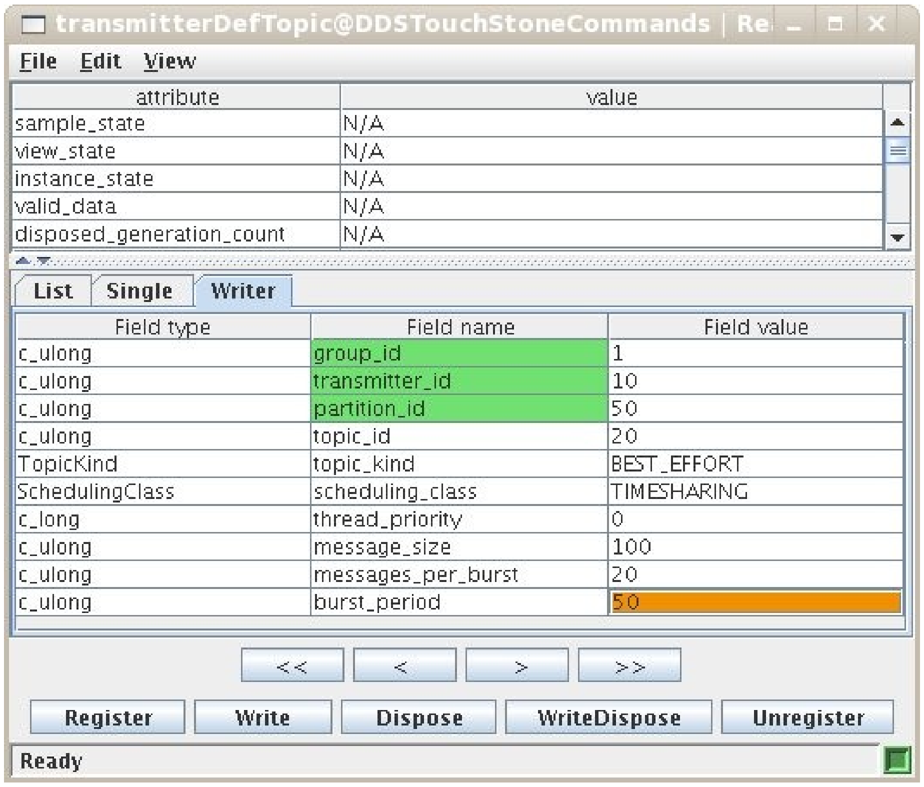


Partition DDSTouchStoneCommands
Topic transmitterDefTopic
wait_for_historical_data
Please provide input.

Figure 5: creating a Transmitter reader-writer

2. In the create reader-writer dialog box shown in Figure 5, set the topic to transmitterDefTopic. Then click OK.

3. A new dialog box will appear labelled transmitterDefTopic@DDSTouchStoneCommands. Select the writer tab. This will display a set of fields that can be used to configure the new transmitter.


transmitterDefTopic@DDSTouchStoneCommands | Re
File Edit View
List Single Writer
<< < > >>
Register Write Dispose WriteDispose Unregister
Ready

attribute	value
sample_state	N/A
view_state	N/A
instance_state	N/A
valid_data	N/A
disposed_generation_count	N/A

Field type	Field name	Field value
c_ulong	group_id	1
c_ulong	transmitter_id	10
c_ulong	partition_id	50
c_ulong	topic_id	20
TopicKind	topic_kind	BEST_EFFORT
SchedulingClass	scheduling_class	TIMESHARING
c_long	thread_priority	0
c_ulong	message_size	100
c_ulong	messages_per_burst	20
c_ulong	burst_period	50

Figure 6: transmitter field values

4. These writer fields need to be set and applied to the new transmitter. Figure 6 shows the example set of values to be used to create the tutorial transmitter.

Group_id	This must match the Touchstone group_id for the Touchstone instance that the component will reside on. When referencing a group containing multiple Touchstone instances, this command topic will be received and processed by all components in that grouping. This tutorial uses a Touchstone instance with the group_id set to 1.
transmitter_id (component_id)	This is used to uniquely identify the component to be created or edited. If this is the first time using this component id then a new component will be created with the given id. If a component with the same id value already exists then these field values with replace the values in the existing component attributes. For this example create a new Transmitter with a transmitter_id set to 10
Partition_id	This selects the partition that the transmitter will belong to. If the partition does not exist then one will be create with the given id. This value will be

	stored as a string. To create the Partition 50 for this example set this value to 50
topic_id	Sets the id for the Topic to be used. For this example set this value to 20 If this is the first time using this topic id then a new topic will be created with the given id. If a topic with the same id value already exists then these field values with replace the values in the existing topic attributes.
Topic_kind	Sets the Topic kind to either: Best_Effort, Reliable, Transient or Persistent Set to Best_Effort for this example
Scheduling_class	Sets the Operation System Scheduling Class to either: TimeSharing or Real-Time Set to TimeSharing for this example
Thread_priority	Sets the Operation System Thread Priority. Set to 0 for this example

5. The above fields are common to all the DDS Touchstone Components. Some components will have additional fields listed after these that are specific to that component type. For this transmitter these are the additional fields

Messages_per_burst	Sets the number of the Messages per write burst. Set the number of messages per burst to 20 for this example
Burst_period	Sets the interval between write bursts. These values are in milliseconds Set burst period to 1000ms, this will perform a single write burst every second

6. Once this form has been completed click “Write” to send the data to the OpenSplice middleware. The addressed touchstone instance (group_id = 1) will then read the data and create the new Transmitter

3.3.2 Creating a Receiver

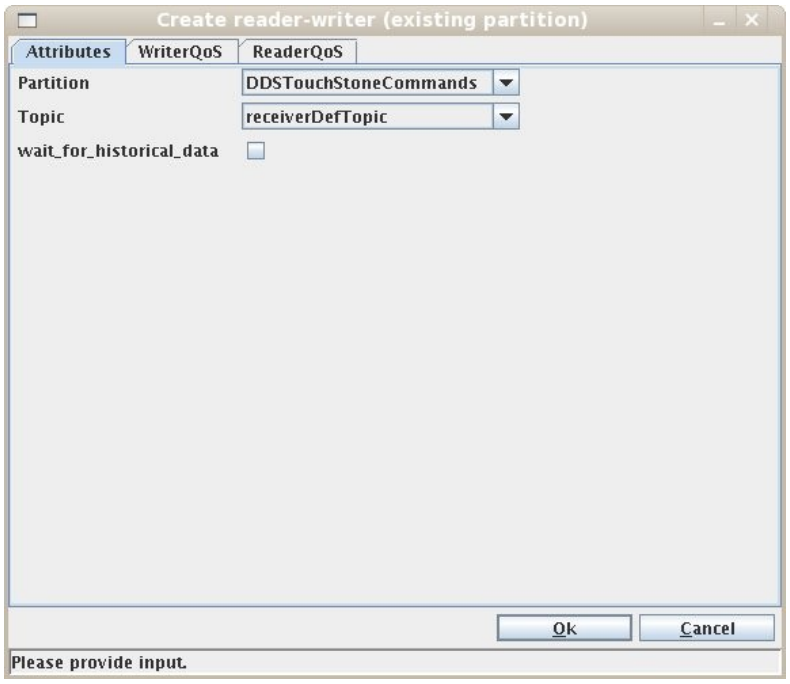
Once the transmitter has been created, it needs to be paired with a Receiver. This Receiver will be on the same Partition (Partition 50) as the Transmitter, but in this tutorial it will be created using a new separate Touchstone instance.

Note: Component pairings can also be created on the same Touchstone instance.

1. Start a Touchstone instance with the application_id and group_id set to 2

   ./touchstone_c 2

2. To create a new Receiver select Edit -> Create Reader-Writer -> Existing Partition from the Tuner menu



Partition: DDSTouchStoneCommands
Topic: receiverDefTopic
wait_for_historical_data: ☐
Please provide input.

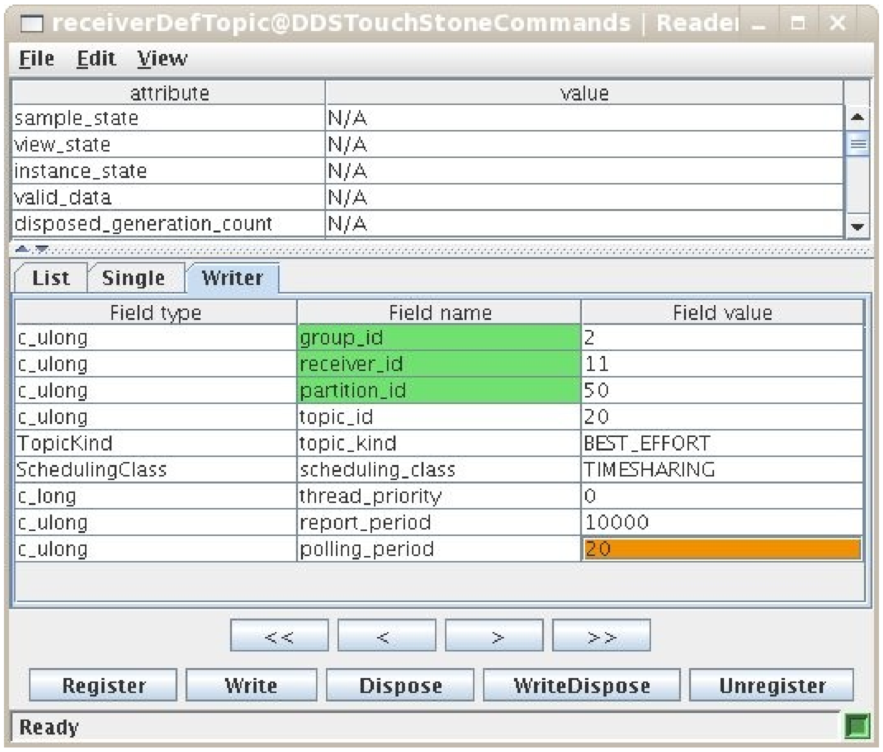
Figure 7: creating a Receiver reader-writer

3. In the create reader-writer dialog box shown in Figure 7, set the topic to receiverDefTopic. Then click OK.

4. A new dialog box will appear labelled receiverDefTopic@DDSTouchStoneCommands. Select the writer tab. This will display a set of fields that can be used to configure the new receiver.

5. To create the new receiver on the Touchstone 2 instance the group_id must be set to 2. The receiver must also have a matching Partition_id and Topic_id to its corresponding Transmitter, otherwise messages with not be received.

   Complete the receiver field values as shown in Figure 8 below.


receiverDefTopic@DDSTouchStoneCommands | Reader
File Edit View
List Single Writer
<< < > >>
Register Write Dispose WriteDispose Unregister
Ready

attribute	value
sample_state	N/A
view_state	N/A
instance_state	N/A
valid_data	N/A
disposed_generation_count	N/A

Field type	Field name	Field value
c_ulong	group_id	2
c_ulong	receiver_id	11
c_ulong	partition_id	50
c_ulong	topic_id	20
TopicKind	topic_kind	BEST_EFFORT
SchedulingClass	scheduling_class	TIMESHARING
c_long	thread_priority	0
c_ulong	report_period	10000
c_ulong	polling_period	20

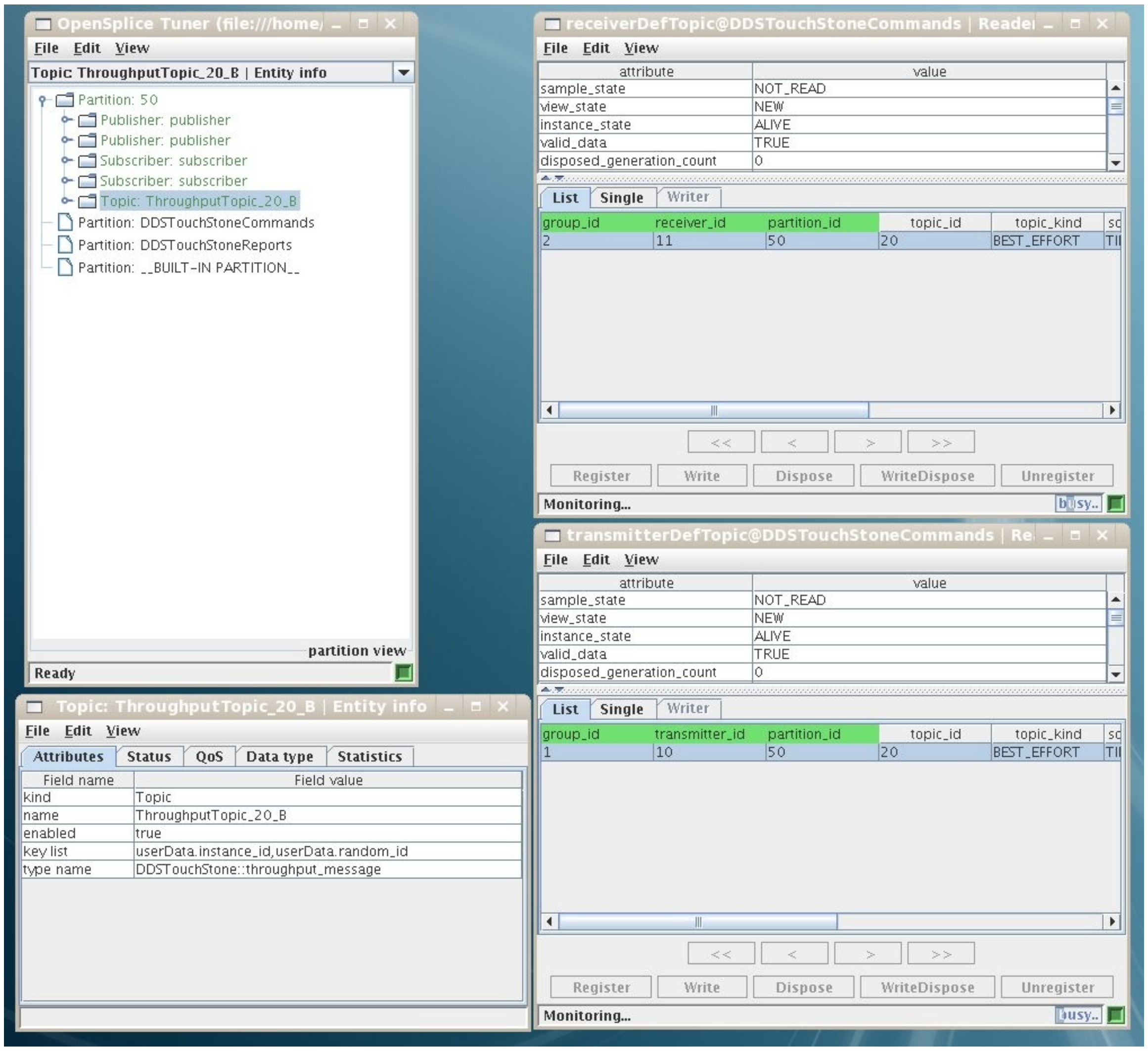
Figure 8: Receiver field values

6. The receiver has the two additional fields.

Report_period	Sets the interval between Report creations. This value is in milliseconds. For this example set the Report period to 10000ms. This will produce a report every 10 seconds.
Polling_period	Sets the interval between 'take' actions by the receiver. This value is in milliseconds. For this example set the Polling period to 20ms. The receiver will look for new data every 20 milliseconds.

7. Once this form has been completed click “Write” to write the data to the OpenSplice middleware, and create the receiver on Touchstone 2.

3.3.3 Throughput Topics


OpenSplice Tuner (file:///home/
File Edit View
Topic: ThroughputTopic_20_B | Entity info
partition view
Ready
receiverDefTopic@DDSTouchStoneCommands | Reader
transmitterDefTopic@DDSTouchStoneCommands | Re
List
Single
Writer
<< < > >>
Register
Write
Dispose
WriteDispose
Unregister
Monitoring...
busy..
Topic: ThroughputTopic_20_B | Entity info
Attributes
Status
QoS
Data type
Statistics

- Partition: 50
  - Publisher: publisher
  - Publisher: publisher
  - Subscriber: subscriber
  - Subscriber: subscriber
  - Topic: ThroughputTopic_20_B
- Partition: DDSTouchStoneCommands
- Partition: DDSTouchStoneReports
- Partition: __BUILT-IN PARTITION__

attribute	value
sample_state	NOT_READ
view_state	NEW
instance_state	ALIVE
valid_data	TRUE
disposed_generation_count	0

group_id	receiver_id	partition_id	topic_id	topic_kind	sc
2	11	50	20	BEST_EFFORT	TI

attribute	value
sample_state	NOT_READ
view_state	NEW
instance_state	ALIVE
valid_data	TRUE
disposed_generation_count	0

group_id	transmitter_id	partition_id	topic_id	topic_kind	sc
1	10	50	20	BEST_EFFORT	TI

Field name	Field value
kind	Topic
name	ThroughputTopic_20_B
enabled	true
key list	userData.instance_id,userData.random_id
type name	DDSTouchStone::throughput_message

Figure 9: Framework Summary

The first framework partition (Partition 50) has now been created and configured. Figure 9 shows the monitoring information for the new Transmitter and Receiver pairing. In addition when using the Tuner Partition View, Partition 50 is now viewable.

Inside Partition 50 is a new topic type ThroughputTopic_20_B.

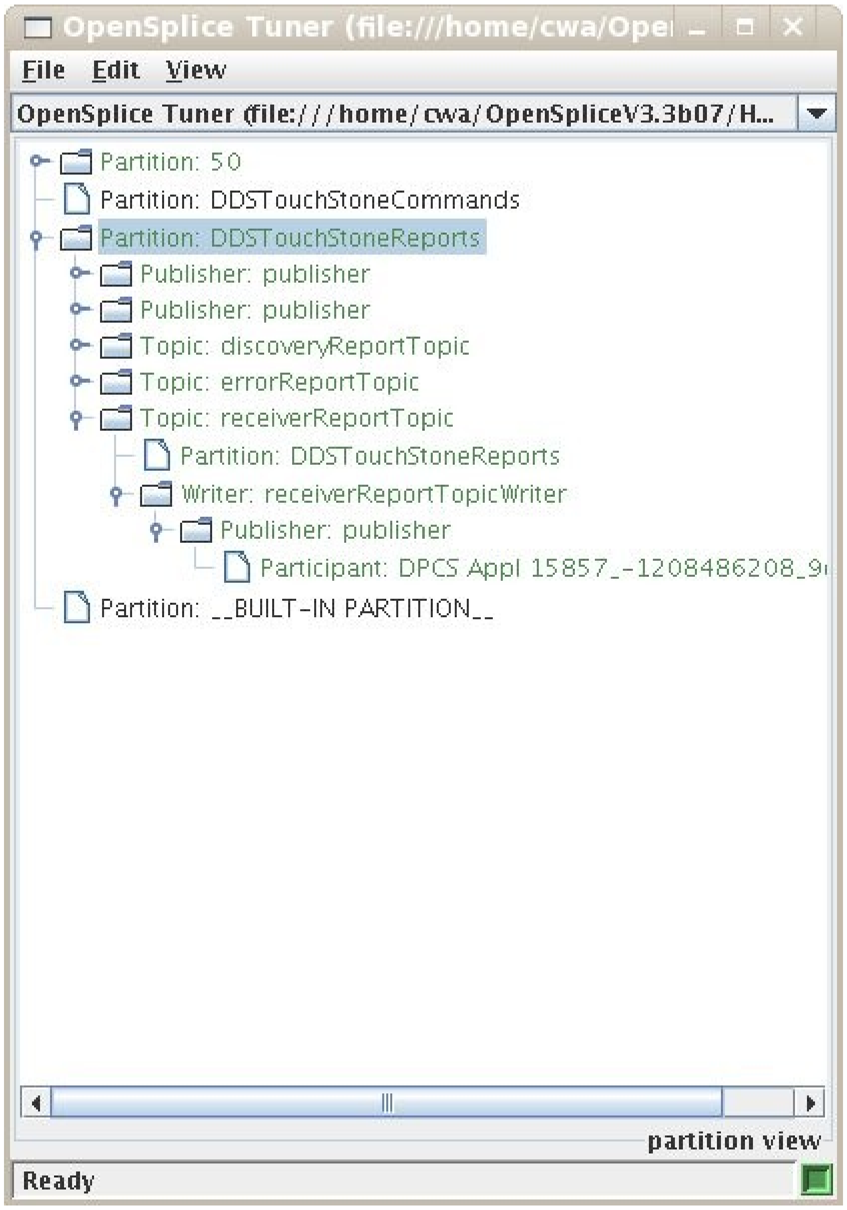
This Throughput Topic represents the new topic (topic_id = 20) used by the Transmitter and Receiver to communicate.

Throughput Topics are used in Touchstone frameworks to provide efficiency and scalability benchmarks, where data throughput and transport efficiency are to be measured.

3.4 Watcher Tool

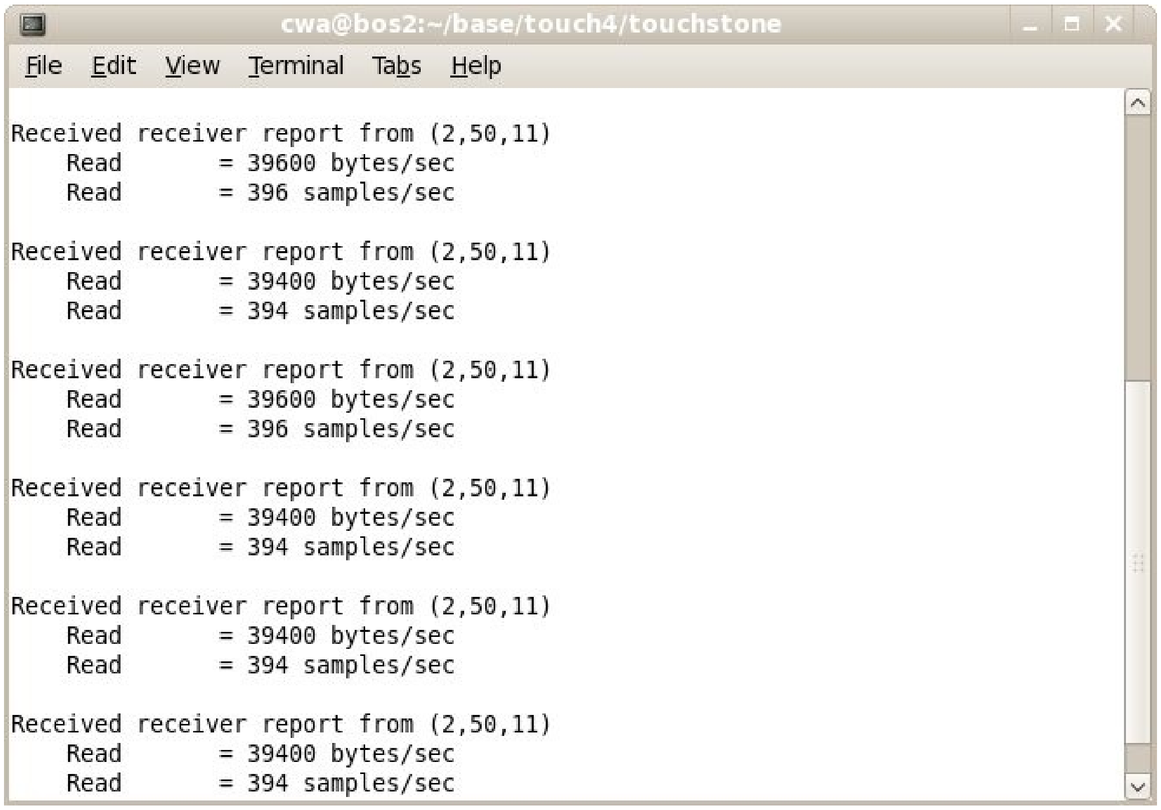
The Watcher tool can be used to view the report information created by the Touchstone report topics.

When using Touchstone components special report topics are created to gather information and results during testing. Components publish reports to these topics during run-time.


```
OpenSplice Tuner (file:///home/cwa/Ope
File  Edit  View
OpenSplice Tuner (file:///home/cwa/OpenSpliceV3.3b07/H...
Partition: 50
Partition: DDSTouchStoneCommands
Partition: DDSTouchStoneReports
    Publisher: publisher
    Publisher: publisher
    Topic: discoveryReportTopic
    Topic: errorReportTopic
    Topic: receiverReportTopic
        Partition: DDSTouchStoneReports
        Writer: receiverReportTopicWriter
            Publisher: publisher
                Participant: DPCS Appl 15857_-1208486208_9
Partition: __BUILT-IN PARTITION__
partition view
Ready
```
Figure 10: Report Topics in the Touchstone Reports Partition

In this framework a new Receiver Report Topic has been created, as shown in Figure 10.

Once started the Watcher tool will automatically subscribe to all component report topics on the OpenSplice middleware and display the information to the screen.


cwa@bos2:~/base/touch4/touchstone
File Edit View Terminal Tabs Help

```
Received receiver report from (2,50,11)
    Read        = 39600 bytes/sec
    Read        = 396 samples/sec

Received receiver report from (2,50,11)
    Read        = 39400 bytes/sec
    Read        = 394 samples/sec

Received receiver report from (2,50,11)
    Read        = 39600 bytes/sec
    Read        = 396 samples/sec

Received receiver report from (2,50,11)
    Read        = 39400 bytes/sec
    Read        = 394 samples/sec

Received receiver report from (2,50,11)
    Read        = 39400 bytes/sec
    Read        = 394 samples/sec

Received receiver report from (2,50,11)
    Read        = 39400 bytes/sec
    Read        = 394 samples/sec
```

Figure 11: Watcher Tool Output

Figure 11 shows the report information so far collected from the tutorial framework.

This information shows the new Transmitter / Receiver pairing current Throughput and Read rates.

The components are identifiable by the id address provided in the reports, for example:

(2, 50, 11) -> (application_id, partition_id, component_id)

In this scenario the component_id is the receiver_id for Receiver 11.

The Watcher can be started with the following command:

```
watcher
```

Note: It is recommended that the Watcher is run in a separate shell as the program output will print to the screen.

3.5 Creating a Transceiver / Transponder Pairing

The next step is to build on top of the Transmitter / Receiver pairing created in Section 3.3, by adding a Transceiver / Transponder pairing to provide some determinism & dependability benchmarks.

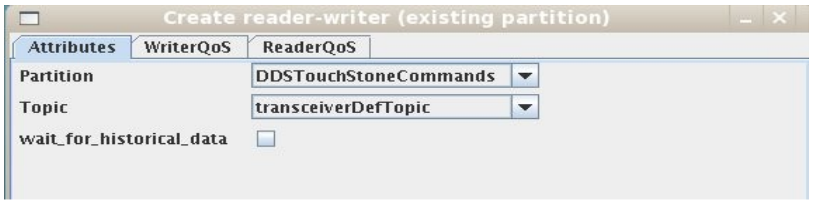
When using the OpenSplice Tuner tool, the benchmark frameworks can be expanded and altered dynamically on demand.

In this section a new Transceiver will be created on the Touchstone 1 group and a Transponder on the Touchstone 2 group. Both will be added to a new Partition with the partition_id set to 51.

3.5.1 Creating a Transceiver

1. To create a new Transceiver select Edit -> Create Reader-Writer -> Existing Partition from the Tuner menu

2. In the create reader-writer dialog box shown in Figure 7, set the topic to transceiverDefTopic. Then click OK.



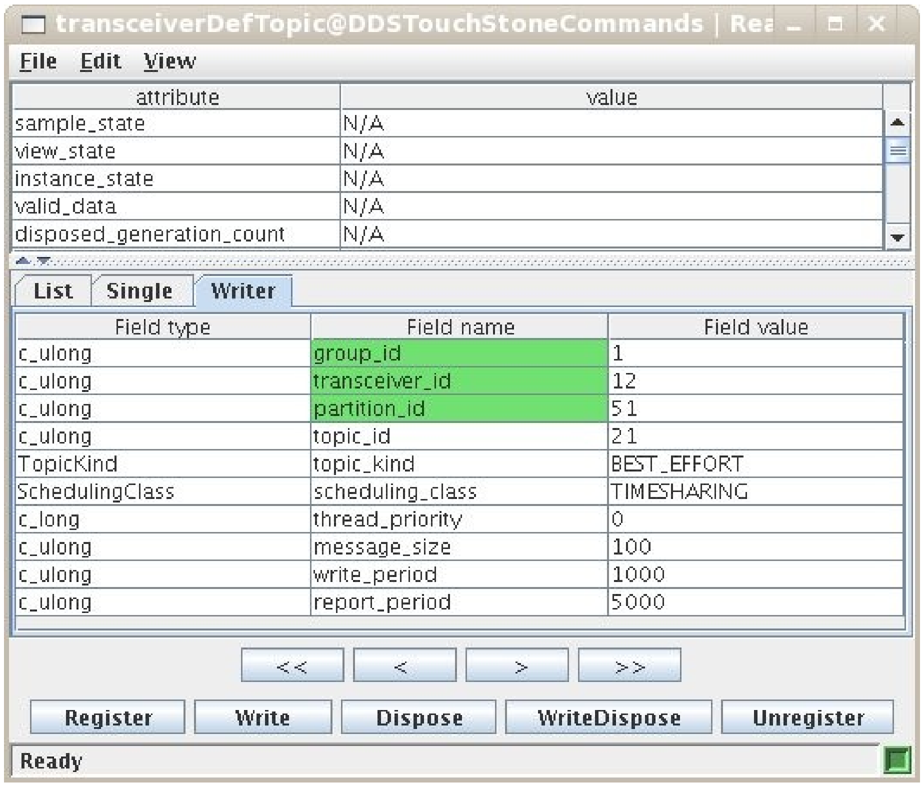
Partition	DDSTouchStoneCommands
Topic	transceiverDefTopic
wait_for_historical_data	☐

Figure 12: Creating a Transceiver

3. A new dialog box will appear labelled transceiverDefTopic@DDSTouchStoneCommands. Select the writer tab. This will display a set of fields that can be used to configure the new transceiver.

4. To create the transceiver on the Touchstone 1 group the group_id must be set to 1.

   Complete the transceiver field values as shown in Figure 13 below.


transceiverDefTopic@DDSTouchStoneCommands | Rea
File Edit View
List Single Writer
<< < > >>
Register Write Dispose WriteDispose Unregister
Ready

attribute	value
sample_state	N/A
view_state	N/A
instance_state	N/A
valid_data	N/A
disposed_generation_count	N/A

Field type	Field name	Field value
c_ulong	group_id	1
c_ulong	transceiver_id	12
c_ulong	partition_id	51
c_ulong	topic_id	21
TopicKind	topic_kind	BEST_EFFORT
SchedulingClass	scheduling_class	TIMESHARING
c_long	thread_priority	0
c_ulong	message_size	100
c_ulong	write_period	1000
c_ulong	report_period	5000

Figure 13: Transceiver field values

5. Using a partition_id of 51 will create a new partition for the Transceiver / Transponder pairing. The Transceiver has the following additional fields.

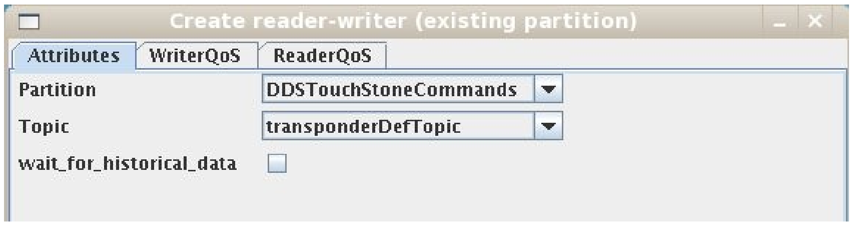
Message_size	Sets the size of the Messages. These values are in bytes Set message size to 100 bytes for this example
Write_period	Sets the interval between writes. These values are in milliseconds Set write period to 1000ms, this will perform a single write every second
Report_period	Sets the interval between Report creations. These values are in milliseconds. For this example set the Report period to 5000ms. This will produce a report every 5 seconds.

6. Once this form has been completed click “Write” to write the data to the OpenSplice middleware, and let Touchstone create the Transceiver.

3.5.2 Creating a Transponder

1. To create a new Transponder select Edit -> Create Reader-Writer -> Existing Partition from the Tuner menu

2. In the create reader-writer dialog box shown in Figure 14, set the topic to transponderDefTopic. Then click OK.



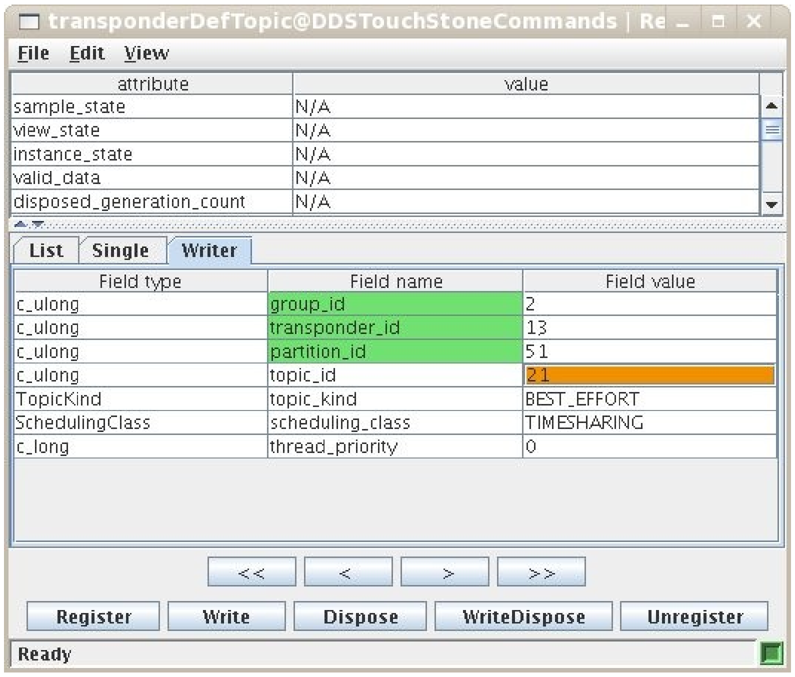
Partition	DDSTouchStoneCommands
Topic	transponderDefTopic
wait_for_historical_data	☐

Figure 14: Create a Transponder

3. In transponderDefTopic@DDSTouchStoneCommands dialog box, select the writer tab. This will display a set of fields that can be used to configure the new transponder.

4. To create the transponder on Touchstone 2 group the group_id must be set to 2. A Transponder must also have a matching Partition_id and Topic_id to its corresponding Transceiver, otherwise messages with not be received or responded to.

   Complete the transponder field values as shown in Figure 15 below.



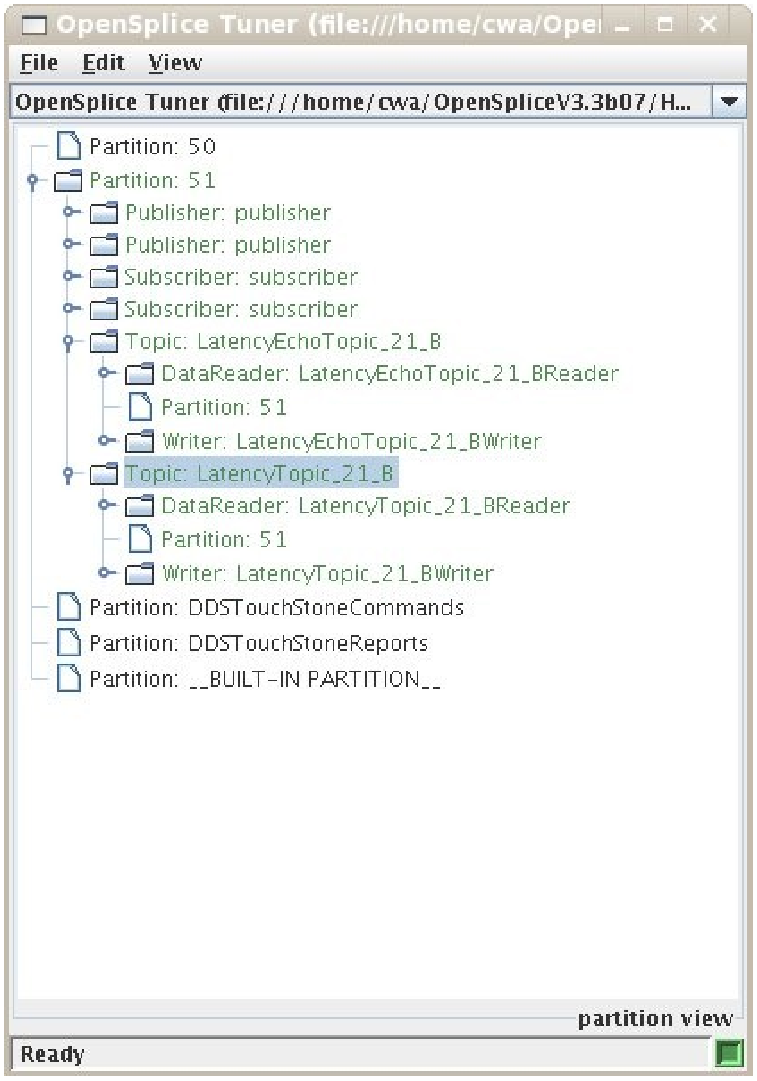
attribute	value
sample_state	N/A
view_state	N/A
instance_state	N/A
valid_data	N/A
disposed_generation_count	N/A

Field type	Field name	Field value
c_ulong	group_id	2
c_ulong	transponder_id	13
c_ulong	partition_id	51
c_ulong	topic_id	21
TopicKind	topic_kind	BEST_EFFORT
SchedulingClass	scheduling_class	TIMESHARING
c_long	thread_priority	0

Figure 15: Transponder Field values

5. Once this form has been completed click “Write” to write the data to the OpenSplice middleware and create the Transponder.

3.5.3 Latency Topics


OpenSplice Tuner (file:///home/cwa/Ope
File Edit View
OpenSplice Tuner (file:///home/cwa/OpenSpliceV3.3b07/H...
partition view
Ready

```
Partition: 50
Partition: 51
    Publisher: publisher
    Publisher: publisher
    Subscriber: subscriber
    Subscriber: subscriber
    Topic: LatencyEchoTopic_21_B
        DataReader: LatencyEchoTopic_21_BReader
        Partition: 51
        Writer: LatencyEchoTopic_21_BWriter
    Topic: LatencyTopic_21_B
        DataReader: LatencyTopic_21_BReader
        Partition: 51
        Writer: LatencyTopic_21_BWriter
Partition: DDSTouchStoneCommands
Partition: DDSTouchStoneReports
Partition: __BUILT-IN PARTITION__
```

Figure 16: Partition 51 View

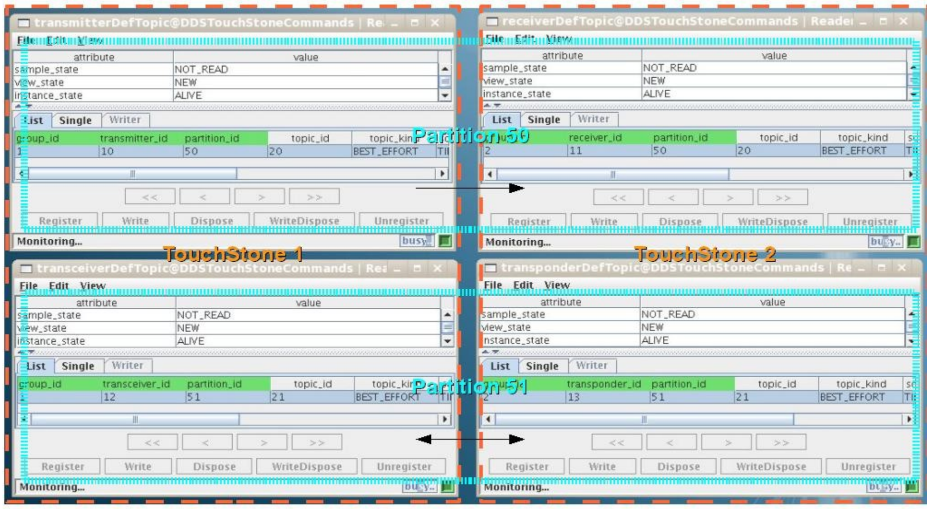
The second framework partition (Partition 51) has now been created. shows the Partition 51 view from the Tuner window. Inside Partition 51 are the new topic types:

LatencyEchoTopic_21_B
LatencyTopic_21_B

These Latency Topics represents the new topics (with topic_id = 21) used by the Transceiver and Transponder to communicate.

Latency Topics are used in frameworks designed to provide determinism and dependability benchmarks, where data latency, data urgency and data importance are to be measured.

3.6 Watcher Results


transmitterDefTopic@DDSTouchStoneCommands | Re
receiverDefTopic@DDSTouchStoneCommands | Reader
transceiverDefTopic@DDSTouchStoneCommands | Rea
transponderDefTopic@DDSTouchStoneCommands | Re
File Edit View
List Single Writer
Partition 50
Partition 51
TouchStone 1
TouchStone 2
<< < > >>
Register Write Dispose WriteDispose Unregister
Monitoring...
busy

attribute	value
sample_state	NOT_READ
view_state	NEW
instance_state	ALIVE

group_id	transmitter_id	partition_id	topic_id	topic_kind
1	10	50	20	BEST_EFFORT

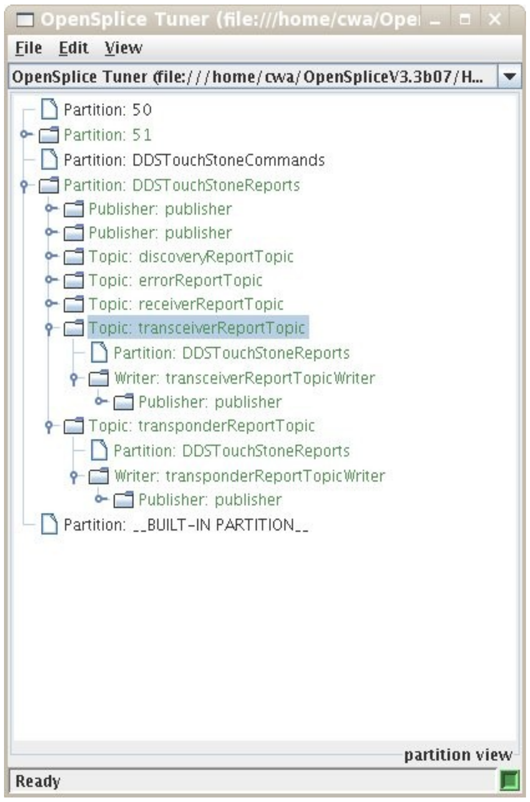
group_id	receiver_id	partition_id	topic_id	topic_kind
2	11	50	20	BEST_EFFORT

group_id	transceiver_id	partition_id	topic_id	topic_kind
1	12	51	21	BEST_EFFORT

group_id	transponder_id	partition_id	topic_id	topic_kind
2	13	51	21	BEST_EFFORT

Figure 17: Example Test Framework

Figure 17 shows the completed tutorial framework. Now that the Transceiver / Transponder pairing has been included into the framework. This will provide extras information to the Watcher output.


OpenSplice Tuner (file:///home/cwa/Ope
File Edit View
OpenSplice Tuner (file:///home/cwa/OpenSpliceV3.3b07/H...
partition view
Ready

```
Partition: 50
Partition: 51
Partition: DDSTouchStoneCommands
Partition: DDSTouchStoneReports
    Publisher: publisher
    Publisher: publisher
    Topic: discoveryReportTopic
    Topic: errorReportTopic
    Topic: receiverReportTopic
    Topic: transceiverReportTopic
        Partition: DDSTouchStoneReports
        Writer: transceiverReportTopicWriter
            Publisher: publisher
    Topic: transponderReportTopic
        Partition: DDSTouchStoneReports
        Writer: transponderReportTopicWriter
            Publisher: publisher
Partition: __BUILT-IN PARTITION__
```

Figure 18: Report Topics in Touchstone Report Partition

As shown in Figure 18, the addition of Partition 51 components has created new report topic publishers for the Transceiver and the Transponder. The information published to these report topics can be viewed in the watcher output. The following is the new output displayed by the Watcher for this tutorial:

```
Received receiver report from (2,50,11)
Throughput = 99 bytes/sec
Read       = 99 bytes/sec

Received Transceiver report from (1,51,12)
Send latency:
100.0% : cnt= 5, min= 159, avg= 162, max= 166, dev= 2.45
99.9% : cnt= 4, min= 159, avg= 161, max= 163, dev= 1.58
99.0% : cnt= 4, min= 159, avg= 161, max= 163, dev= 1.58
90.0% : cnt= 4, min= 159, avg= 161, max= 163, dev= 1.58
Send Source latency:
100.0% : cnt= 5, min= 4, avg= 4, max= 4, dev= 0.00
99.9% : cnt= 4, min= 4, avg= 4, max= 4, dev= 0.00
99.0% : cnt= 4, min= 4, avg= 4, max= 4, dev= 0.00
90.0% : cnt= 4, min= 4, avg= 4, max= 4, dev= 0.00
Send Delivery latency:
100.0% : cnt= 5, min= 46, avg= 48, max= 50, dev= 1.33
99.9% : cnt= 4, min= 46, avg= 48, max= 49, dev= 1.09
99.0% : cnt= 4, min= 46, avg= 48, max= 49, dev= 1.09
90.0% : cnt= 4, min= 46, avg= 48, max= 49, dev= 1.09
Send Arrival latency:
100.0% : cnt= 5, min= 108, avg= 110, max= 112, dev= 1.33
99.9% : cnt= 4, min= 108, avg= 109, max= 110, dev= 0.83
99.0% : cnt= 4, min= 108, avg= 109, max= 110, dev= 0.83
90.0% : cnt= 4, min= 108, avg= 109, max= 110, dev= 0.83
Echo latency:
100.0% : cnt= 5, min= 141, avg= 143, max= 145, dev= 1.41
99.9% : cnt= 4, min= 141, avg= 142, max= 144, dev= 1.12
99.0% : cnt= 4, min= 141, avg= 142, max= 144, dev= 1.12
90.0% : cnt= 4, min= 141, avg= 142, max= 144, dev= 1.12
Echo Source latency:
100.0% : cnt= 5, min= 2, avg= 3, max= 3, dev= 0.40
99.9% : cnt= 4, min= 2, avg= 3, max= 3, dev= 0.43
99.0% : cnt= 4, min= 2, avg= 3, max= 3, dev= 0.43
90.0% : cnt= 4, min= 2, avg= 3, max= 3, dev= 0.43
Echo Delivery latency:
100.0% : cnt= 5, min= 39, avg= 40, max= 41, dev= 0.75
99.9% : cnt= 4, min= 39, avg= 40, max= 41, dev= 0.71
99.0% : cnt= 4, min= 39, avg= 40, max= 41, dev= 0.71
90.0% : cnt= 4, min= 39, avg= 40, max= 41, dev= 0.71
Echo Arrival latency:
100.0% : cnt= 5, min= 99, avg= 100, max= 102, dev= 1.26
99.9% : cnt= 4, min= 99, avg= 100, max= 101, dev= 0.87
99.0% : cnt= 4, min= 99, avg= 100, max= 101, dev= 0.87
90.0% : cnt= 4, min= 99, avg= 100, max= 101, dev= 0.87
Trip latency:
100.0% : cnt= 5, min= 304, avg= 307, max= 310, dev= 2.48
99.9% : cnt= 4, min= 304, avg= 306, max= 310, dev= 2.29
99.0% : cnt= 4, min= 304, avg= 306, max= 310, dev= 2.29
90.0% : cnt= 4, min= 304, avg= 306, max= 310, dev= 2.29
Inter arrival time:
100.0% : cnt= 5, min= 1003931, avg= 1004062, max= 1004187, dev= 83.18
99.9% : cnt= 4, min= 1003931, avg= 1004030, max= 1004090, dev= 61.24
99.0% : cnt= 4, min= 1003931, avg= 1004030, max= 1004090, dev= 61.24
90.0% : cnt= 4, min= 1003931, avg= 1004030, max= 1004090, dev= 61.24
```

The Watcher now also receives reports from the Transceiver containing determinism & dependability benchmark information.

4 Advanced Touchstone Tools

In addition to the Touchstone and Watcher tools used in the Tutorial from the previous Chapter, DDS Touchstone provides extras tools to help manage and monitor benchmark frameworks.

4.1 Recorder Tool

The Recorder tool can be used to record and store benchmark framework configurations. The Recorder can also replay previously saved frameworks to re-demonstrate testing scenarios. The Recorder can be controlled and monitored via the OpenSplice Tuner.

4.1.1 Starting the Recorder with OpenSplice Tuner

1. To use the Recorder with the OpenSplice Tuner, first start the OpenSplice Daemon and connect the Tuner. Run the relevant release.<config_name>.com to set up the Touchstone environment.

2. To start the recorder tool run the following command:

   ```
   recorder [–v] <recorder_id> <filename>
   ```

   The –v option is used to turn on “Verbose” mode. This mode will display to screen the actions recorded by the Recorder

   <application_id> is used to identify and reference the recorder instance.

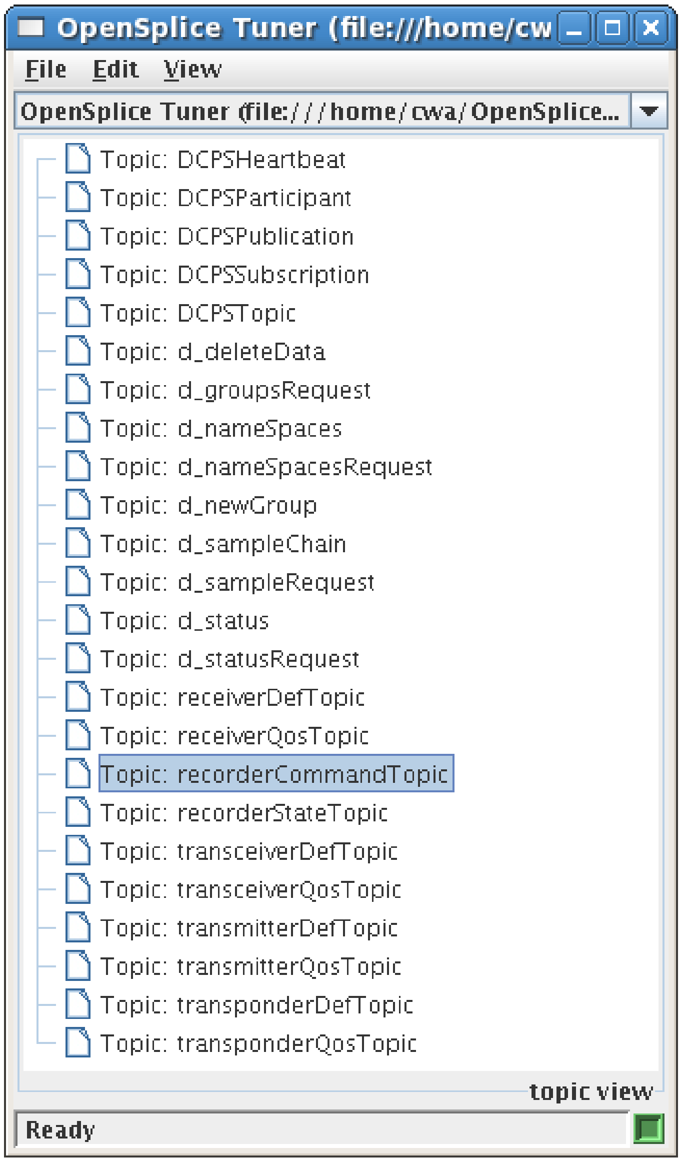
   <filename> provides a filename to a file for the recorder to store the framework actions to.

   For example:

   ```
   recorder –v 99 example1.txt
   ```

3. Once the Recorder has been started the following topics with be displayed in the Tuner:

   recorderCommandTopic
   recorderStateTopic


OpenSplice Tuner (file:///home/cw
File
Edit
View
OpenSplice Tuner (file:///home/cwa/OpenSplice...

Topic: DCPSHeartbeat
Topic: DCPSParticipant
Topic: DCPSPublication
Topic: DCPSSubscription
Topic: DCPSTopic
Topic: d_deleteData
Topic: d_groupsRequest
Topic: d_nameSpaces
Topic: d_nameSpacesRequest
Topic: d_newGroup
Topic: d_sampleChain
Topic: d_sampleRequest
Topic: d_status
Topic: d_statusRequest
Topic: receiverDefTopic
Topic: receiverQosTopic
Topic: recorderCommandTopic
Topic: recorderStateTopic
Topic: transceiverDefTopic
Topic: transceiverQosTopic
Topic: transmitterDefTopic
Topic: transmitterQosTopic
Topic: transponderDefTopic
Topic: transponderQosTopic

Figure 19: Recorder Topics

The recorderCommandTopic is used to send commands such as ‘Record’, ‘Play’ and ‘Stop ’to a recorder instance.

The recorderStateTopic can be used to monitor a recorders current state.

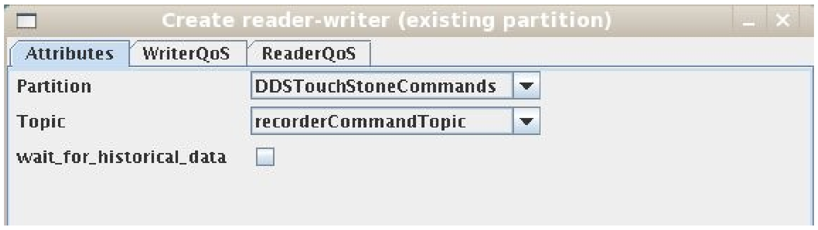
4.1.2 Recording a Framework

The Touchstone recorder has two recording modes: ‘RECORD’ and ‘COMPOSE’.

Recording mode records the time taken to issue each action. This means that a recording created with the ‘RECORD’ command will replay in the same time frame that it was created.

Composing mode disregards the time taken to issue each action, and simply records the commands in a sequential list. When a recording created using the ‘COMPOSE’ command is replayed the stored actions will be issued sequential without any time delay.

1. To set the Recorder to record, first create a new reader-writer for the recorderCommandTopic.


Create reader-writer (existing partition)

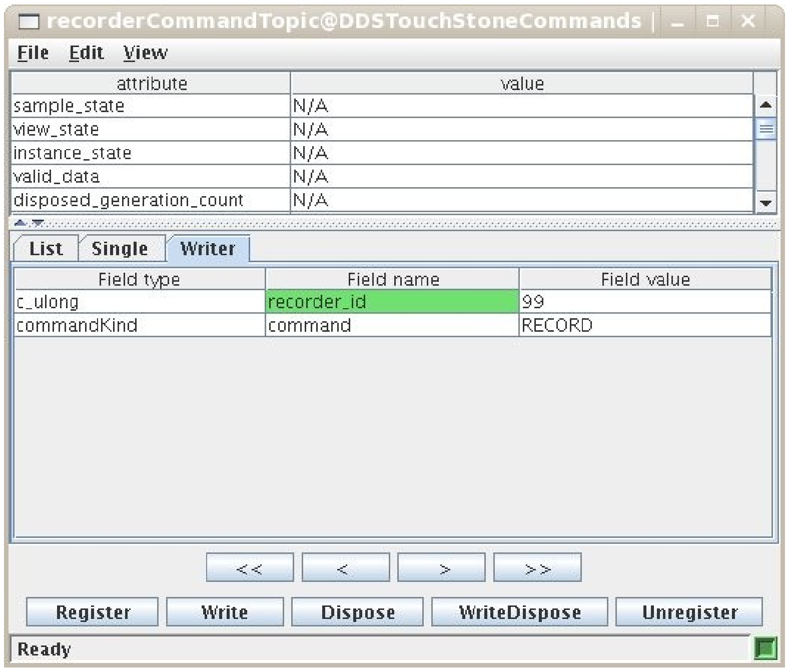
Attributes | WriterQoS | ReaderQoS

Partition	DDSTouchStoneCommands
Topic	recorderCommandTopic
wait_for_historical_data	☐

Figure 20: Create a recorder command reader-writer

   Set the partition value to ‘DDSTouchStoneCommands’ and the Topic value to ‘ recorderCommandTopic’. Then click OK to create the reader-writer

2. Once the reader-writer has been created select the writer tab, as shown in Figure 21.


recorderCommandTopic@DDSTouchStoneCommands

File Edit View

attribute	value
sample_state	N/A
view_state	N/A
instance_state	N/A
valid_data	N/A
disposed_generation_count	N/A

List | Single | Writer

Field type	Field name	Field value
c_ulong	recorder_id	99
commandKind	command	RECORD

Figure 21: Recorder topic fields

Recorder_id	`This must match the application_id for the Recorder instance that has been created.`
	
	`This scenario uses a Recorder instance with the application_id set to 99.`

Command	Selects the command to send to the Recorder. STOP — Stops Recording RECORD — Starts Recording, this command also the time taken to issue each action COMPOSE — Starts Recording, this command disregards the time taken to issue each action PLAY — Starts a Replay of a Recording REPEAT — Loops a Recording Playback QUIT — Shuts down the recorder For this example set the command to ‘RECORD’

For this scenario complete the fields as shown in and click ‘Write’ to send the command to the Recorder 99.

3. If the verbose mode is set to ‘ON’ the recorder will confirm it received the command in the terminal window.

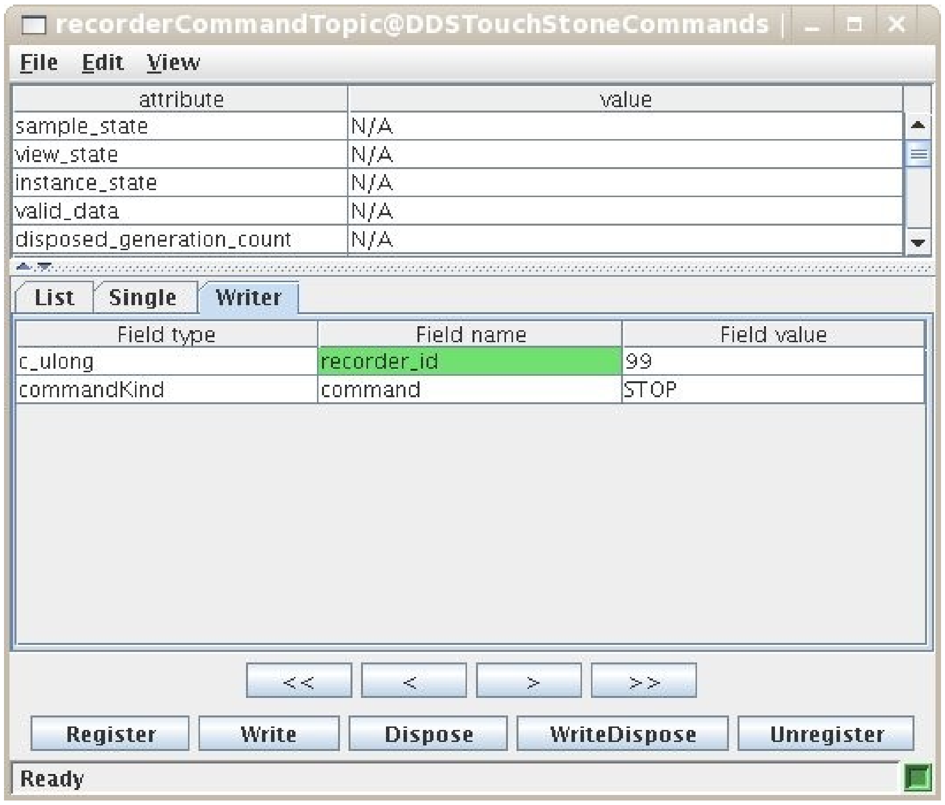
```
$ ./recorder -v 99 example1.rec
Recorder message: Switched on verbose mode
Recorder message: Using recorder_id 99
Recorder message: Using logfile "example1.rec"
Recorder message: Received recorder command RECORD
```

4. The recorder is now set to record. All future Touchstone component actions will be store in the example1.rec file.

When running the recorder with the Chapter 3 tutorial, the recorder output will be as follows:

```
$ ./recorder -v 99 example1.rec
Recorder message: Switched on verbose mode
Recorder message: Using recorder_id 99
Recorder message: Using logfile "example1.rec"
Recorder message: Received recorder command RECORD
Recorder message: Successfully recorded transmitterDef pother command to file
Recorder message: Successfully recorded transmitterQos pother command to file
Recorder message: Successfully recorded receiverDef pother command to file
Recorder message: Successfully recorded receiverQos pother command to file
Recorder message: Successfully recorded transceiverDef pother command to file
Recorder message: Successfully recorded transceiverQos pother command to file
Recorder message: Successfully recorded transponderDef pother command to file
Recorder message: Successfully recorded transponderQos pother command to file
```

To stop the recording, publish a new message to the Recorder using the reader-writer and issue the Stop command.


recorderCommandTopic@DDSTouchStoneCommands
File Edit View
List Single Writer
<< < > >>
Register Write Dispose WriteDispose Unregister
Ready

attribute	value
sample_state	N/A
view_state	N/A
instance_state	N/A
valid_data	N/A
disposed_generation_count	N/A

Field type	Field name	Field value
c_ulong	recorder_id	99
commandKind	command	STOP

Figure 22: Stopping the Recorder

To shutdown the recorder repeat the previous step with the Quit command. The recorder will now terminate.

```
Recorder message: Received recorder command STOP
Recorder message: Received recorder command QUIT
$
```

5. All the actions performed during the recording are now stored in the new recording file.

The DDS Touchstone distribution contains a completed tutorial recording. This is located in the following location:

<Touchstone_HOME>\examples\tutorial\tutorial_recording.dat

4.1.3 Replaying a Saved Framework

If a recording was created using the ‘RECORD’ command it will be replayed in the same time frame in which it was recorded. For example if a test framework took 5 minutes to create and record. Then the replay will also take 5 minutes to complete.

If a recording was created using the ‘COMPOSE’ command the recorder will replay all the recorded actions immediately.

1. The recorder will NOT start the Touchstone instances that where available when the recording was taken. Start all required Touchstone instances used by the recording, ensuring that their application ids and group ids are correct.

   From the tutorial recording the following touchstone instances are required:

   touchstone_c 1

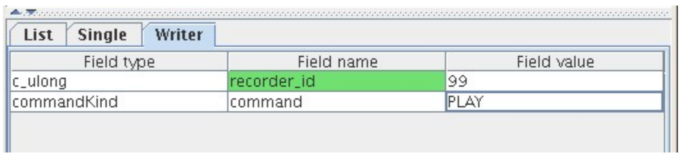
   touchstone_c 2

2. To replay a previous recording, start the Recorder tool providing the filename of the recording in the parameter list. For example:

   ```
   recorder -v 99 example1.rec
   ```

   Note: The “--`autoplay`” flag can also be used this will start the recording automatically without the need for the tuner. A script “start_recording.sh” is provide in the <Touchstone_HOME>/bin to allow automatic replaying of recordings. See section 2.1.5

3. Using the Tuner tool create a reader-writer for the recorderCommandTopic.



Field type	Field name	Field value
c_ulong	recorder_id	99
commandKind	command	PLAY

Figure 23: Sending a Play Command

   Set the recorder_id to match the Recorder instance and set the command to ‘PLAY’. Click Write to send the command.

4. The recorder will now replay the actions from the recorded file. The following is the recorder verbose information from the Tutorial recording:

```
$ ./recorder -v 99 example1.rec
Recorder message: Switched on verbose mode
Recorder message: Using recorder_id 99
Recorder message: Using logfile "example1.rec"
Recorder message: Received recorder command PLAY
Recorder message: Successfully read transmitterDef pother command from file
Recorder message: Successfully read transmitterQos pother command from file
Recorder message: Successfully read receiverDef pother command from file
Recorder message: Successfully read receiverQos pother command from file
Recorder message: Successfully read transceiverDef pother command from file
Recorder message: Successfully read transceiverQos pother command from file
Recorder message: Successfully read transponderDef pother command from file
Recorder message: Successfully read transponderQos pother command from file
Recorder message: Received recorder command STOP
```

5. By starting the Watcher tool the report information from tutorial recording can be viewed.

```
Received receiver report from (2,50,11)
    Throughput = 99 bytes/sec
    Read       = 99 bytes/sec

Received Transceiver report from (1,51,12)
    Send latency:
    100.0% : cnt= 5, min= 159, avg= 162, max= 166, dev= 2.45
    99.9% : cnt= 4, min= 159, avg= 161, max= 163, dev= 1.58
    99.0% : cnt= 4, min= 159, avg= 161, max= 163, dev= 1.58
    90.0% : cnt= 4, min= 159, avg= 161, max= 163, dev= 1.58
    Send Source latency:
    100.0% : cnt= 5, min= 4, avg= 4, max= 4, dev= 0.00
    99.9% : cnt= 4, min= 4, avg= 4, max= 4, dev= 0.00
    99.0% : cnt= 4, min= 4, avg= 4, max= 4, dev= 0.00
    90.0% : cnt= 4, min= 4, avg= 4, max= 4, dev= 0.00
    Send Delivery latency:
    100.0% : cnt= 5, min= 46, avg= 48, max= 50, dev= 1.33
    99.9% : cnt= 4, min= 46, avg= 48, max= 49, dev= 1.09
    99.0% : cnt= 4, min= 46, avg= 48, max= 49, dev= 1.09
    90.0% : cnt= 4, min= 46, avg= 48, max= 49, dev= 1.09
    Send Arrival latency:
    100.0% : cnt= 5, min= 108, avg= 110, max= 112, dev= 1.33
    99.9% : cnt= 4, min= 108, avg= 109, max= 110, dev= 0.83
    99.0% : cnt= 4, min= 108, avg= 109, max= 110, dev= 0.83
    90.0% : cnt= 4, min= 108, avg= 109, max= 110, dev= 0.83
    Echo latency:
    100.0% : cnt= 5, min= 141, avg= 143, max= 145, dev= 1.41
    99.9% : cnt= 4, min= 141, avg= 142, max= 144, dev= 1.12
    99.0% : cnt= 4, min= 141, avg= 142, max= 144, dev= 1.12
    90.0% : cnt= 4, min= 141, avg= 142, max= 144, dev= 1.12
    Echo Source latency:
    100.0% : cnt= 5, min= 2, avg= 3, max= 3, dev= 0.40
    99.9% : cnt= 4, min= 2, avg= 3, max= 3, dev= 0.43
    99.0% : cnt= 4, min= 2, avg= 3, max= 3, dev= 0.43
    90.0% : cnt= 4, min= 2, avg= 3, max= 3, dev= 0.43
    Echo Delivery latency:
    100.0% : cnt= 5, min= 39, avg= 40, max= 41, dev= 0.75
    99.9% : cnt= 4, min= 39, avg= 40, max= 41, dev= 0.71
    99.0% : cnt= 4, min= 39, avg= 40, max= 41, dev= 0.71
    90.0% : cnt= 4, min= 39, avg= 40, max= 41, dev= 0.71
    Echo Arrival latency:
    100.0% : cnt= 5, min= 99, avg= 100, max= 102, dev= 1.26
    99.9% : cnt= 4, min= 99, avg= 100, max= 101, dev= 0.87
    99.0% : cnt= 4, min= 99, avg= 100, max= 101, dev= 0.87
    90.0% : cnt= 4, min= 99, avg= 100, max= 101, dev= 0.87
    Trip latency:
    100.0% : cnt= 5, min= 304, avg= 307, max= 310, dev= 2.48
    99.9% : cnt= 4, min= 304, avg= 306, max= 310, dev= 2.29
    99.0% : cnt= 4, min= 304, avg= 306, max= 310, dev= 2.29
    90.0% : cnt= 4, min= 304, avg= 306, max= 310, dev= 2.29
    Inter arrival time:
    100.0% : cnt= 5, min= 1003931, avg= 1004062, max= 1004187, dev= 83.18
    99.9% : cnt= 4, min= 1003931, avg= 1004030, max= 1004090, dev= 61.24
    99.0% : cnt= 4, min= 1003931, avg= 1004030, max= 1004090, dev= 61.24
    90.0% : cnt= 4, min= 1003931, avg= 1004030, max= 1004090, dev= 61.24
```

6. The recording can be repeated by issuing the Repeat command.

4.1.4 Recorder State Topic

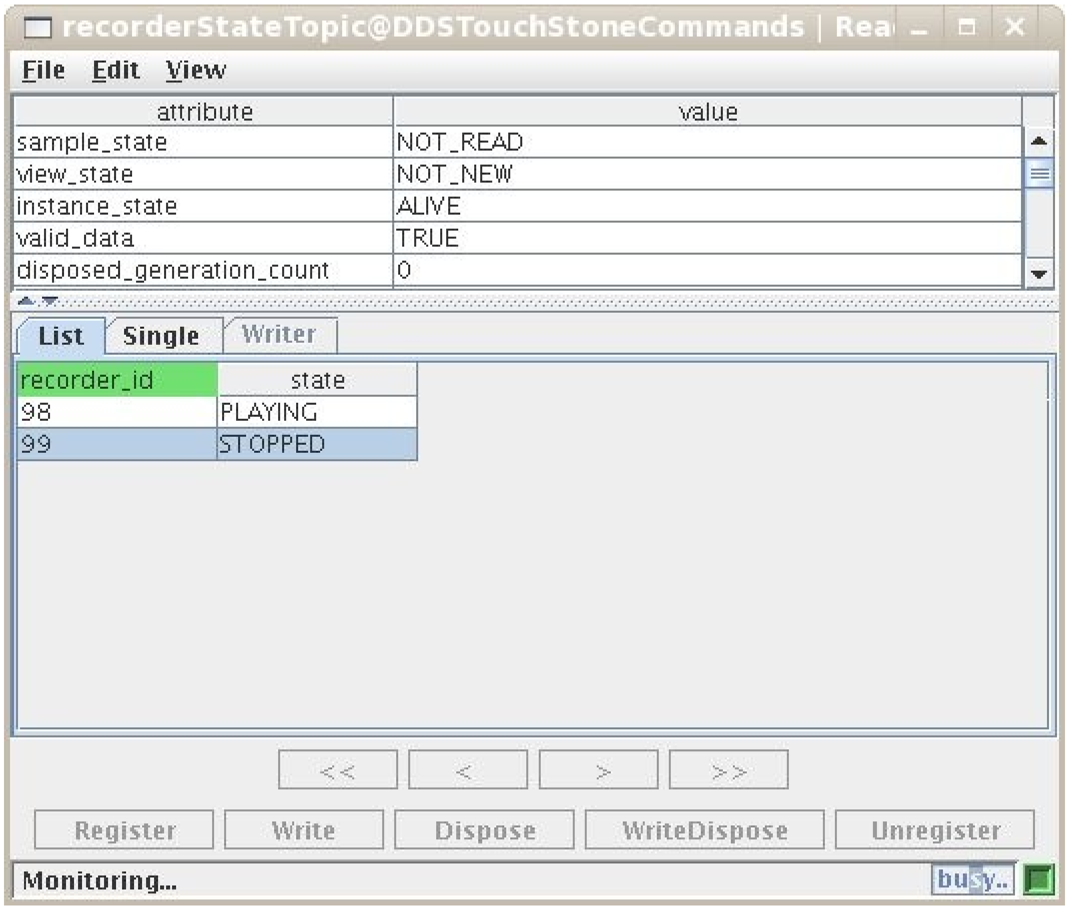
Touchstone also provides a recorderStateTopic. This can be used to check on the current state of any given recorder.

The state topic is best used with the OpenSplice Tuner monitoring feature.

To check the current state of the recorders used by the framework:

1. Create a recorderStateTopic ReaderWriter in the Tuner

2. Select ‘Starting Monitoring’ from the ‘Edit’ menu



attribute	value
sample_state	NOT_READ
view_state	NOT_NEW
instance_state	ALIVE
valid_data	TRUE
disposed_generation_count	0

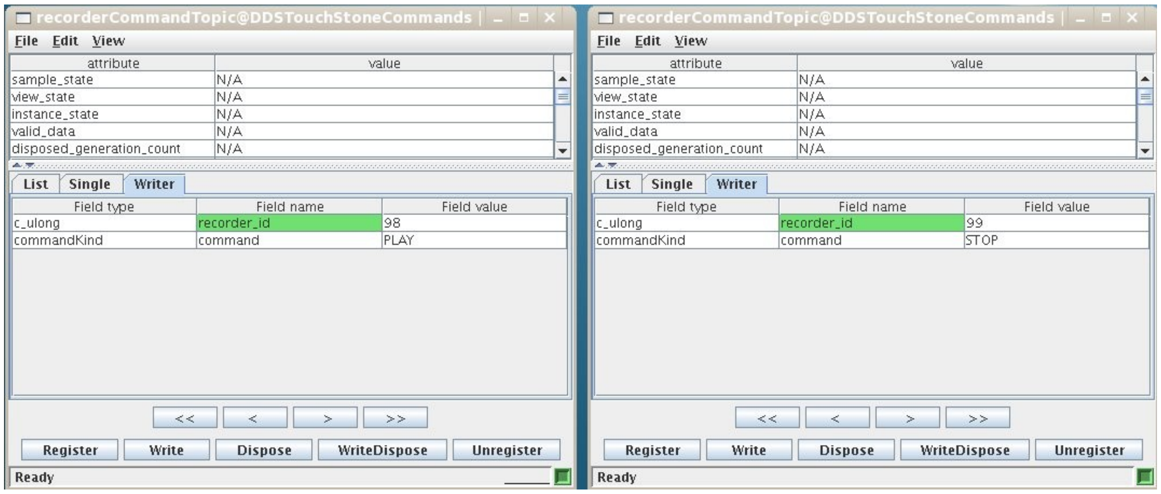
recorder_id	state
98	PLAYING
99	STOPPED

Monitoring...

Figure 24: recorder state monitoring

4.1.5 Using Multiple Recorders

Multiple recorder instances can be used to combine and dynamically expand test scenarios.


recorderCommandTopic@DDSTouchStoneCommands
File Edit View
List Single Writer
<< < > >>
Register Write Dispose WriteDispose Unregister
Ready

attribute	value
sample_state	N/A
view_state	N/A
instance_state	N/A
valid_data	N/A
disposed_generation_count	N/A

Field type	Field name	Field value
c_ulong	recorder_id	98
commandKind	command	PLAY

attribute	value
sample_state	N/A
view_state	N/A
instance_state	N/A
valid_data	N/A
disposed_generation_count	N/A

Field type	Field name	Field value
c_ulong	recorder_id	99
commandKind	command	STOP

Figure 25: using multiple recorders

Figure 25 shows two recorders being run simultaneously. Recorder 98 plays a new framework scenario containing a single Transmitter (id 14) / Receiver (id 15) pairing. Recorder 99 replays the Tutorial example from Chapter 3.

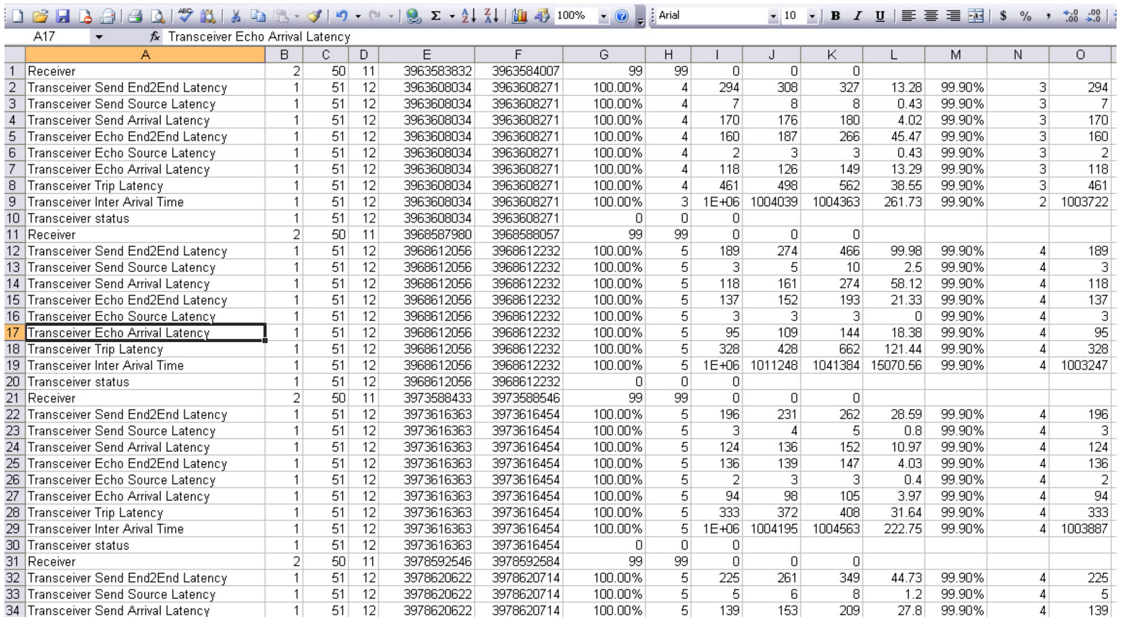
As shown in the Watcher output below, combining two recordings produces new test information.

```
Received receiver report from (2,50,15) /* From Recorder 98 */
    Throughput = 199 bytes/sec
    Read       = 199 bytes/sec

Received receiver report from (2,50,11) /* From Recorder 99 */
    Throughput = 99 bytes/sec
    Read       = 99 bytes/sec

Received Transceiver report from (1,51,12)
    Send latency:
    100.0% : cnt= 5, min= 230, avg= 275, max= 354, dev= 42.15
    99.9% : cnt= 4, min= 230, avg= 255, max= 275, dev= 16.14
    99.0% : cnt= 4, min= 230, avg= 255, max= 275, dev= 16.14
    90.0% : cnt= 4, min= 230, avg= 255, max= 275, dev= 16.14
    Send Source latency:
    100.0% : cnt= 5, min= 5, avg= 7, max= 10, dev= 1.72
    99.9% : cnt= 4, min= 5, avg= 6, max= 7, dev= 0.71
    99.0% : cnt= 4, min= 5, avg= 6, max= 7, dev= 0.71
    90.0% : cnt= 4, min= 5, avg= 6, max= 7, dev= 0.71
    Send Delivery latency:
    100.0% : cnt= 5, min= 85, avg= 112, max= 135, dev= 16.23
    99.9% : cnt= 4, min= 85, avg= 106, max= 119, dev= 12.69
    99.0% : cnt= 4, min= 85, avg= 106, max= 119, dev= 12.69
    90.0% : cnt= 4, min= 85, avg= 106, max= 119, dev= 12.69
    Send Arrival latency:
    100.0% : cnt= 5, min= 140, avg= 156, max= 209, dev= 26.59
    99.9% : cnt= 4, min= 140, avg= 143, max= 149, dev= 3.54
    99.0% : cnt= 4, min= 140, avg= 143, max= 149, dev= 3.54
    90.0% : cnt= 4, min= 140, avg= 143, max= 149, dev= 3.54
```

4.2 Excelerator Tool

The Excelerator tool produces the same results as the Watcher, but in a comma separated format.

This formatted output allows for the information to be machine processed more easily or to be loaded into a spreadsheet application, such as Microsoft Excel, as shown in Figure 26.



A17 Transceiver Echo Arrival Latency

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Receiver	2	50	11	3963583832	3963584007	99	99	0	0	0				
2	Transceiver Send End2End Latency	1	51	12	3963608034	3963608271	100.00%	4	294	308	327	13.28	99.90%	3	294
3	Transceiver Send Source Latency	1	51	12	3963608034	3963608271	100.00%	4	7	8	8	0.43	99.90%	3	7
4	Transceiver Send Arrival Latency	1	51	12	3963608034	3963608271	100.00%	4	170	176	180	4.02	99.90%	3	170
5	Transceiver Echo End2End Latency	1	51	12	3963608034	3963608271	100.00%	4	160	187	266	45.47	99.90%	3	160
6	Transceiver Echo Source Latency	1	51	12	3963608034	3963608271	100.00%	4	2	3	3	0.43	99.90%	3	2
7	Transceiver Echo Arrival Latency	1	51	12	3963608034	3963608271	100.00%	4	118	126	149	13.29	99.90%	3	118
8	Transceiver Trip Latency	1	51	12	3963608034	3963608271	100.00%	4	461	498	562	38.55	99.90%	3	461
9	Transceiver Inter Arival Time	1	51	12	3963608034	3963608271	100.00%	3	1E+06	1004039	1004363	261.73	99.90%	2	1003722
10	Transceiver status	1	51	12	3963608034	3963608271	0	0	0						
11	Receiver	2	50	11	3968587980	3968588057	99	99	0	0	0				
12	Transceiver Send End2End Latency	1	51	12	3968612056	3968612232	100.00%	5	189	274	466	99.98	99.90%	4	189
13	Transceiver Send Source Latency	1	51	12	3968612056	3968612232	100.00%	5	3	5	10	2.5	99.90%	4	3
14	Transceiver Send Arrival Latency	1	51	12	3968612056	3968612232	100.00%	5	118	161	274	58.12	99.90%	4	118
15	Transceiver Echo End2End Latency	1	51	12	3968612056	3968612232	100.00%	5	137	152	193	21.33	99.90%	4	137
16	Transceiver Echo Source Latency	1	51	12	3968612056	3968612232	100.00%	5	3	3	3	0	99.90%	4	3
17	Transceiver Echo Arrival Latency	1	51	12	3968612056	3968612232	100.00%	5	95	109	144	18.38	99.90%	4	95
18	Transceiver Trip Latency	1	51	12	3968612056	3968612232	100.00%	5	328	428	662	121.44	99.90%	4	328
19	Transceiver Inter Arival Time	1	51	12	3968612056	3968612232	100.00%	5	1E+06	1011248	1041384	15070.56	99.90%	4	1003247
20	Transceiver status	1	51	12	3968612056	3968612232	0	0	0						
21	Receiver	2	50	11	3973588433	3973588546	99	99	0	0	0				
22	Transceiver Send End2End Latency	1	51	12	3973616363	3973616454	100.00%	5	196	231	262	28.59	99.90%	4	196
23	Transceiver Send Source Latency	1	51	12	3973616363	3973616454	100.00%	5	3	4	5	0.8	99.90%	4	3
24	Transceiver Send Arrival Latency	1	51	12	3973616363	3973616454	100.00%	5	124	136	152	10.97	99.90%	4	124
25	Transceiver Echo End2End Latency	1	51	12	3973616363	3973616454	100.00%	5	136	139	147	4.03	99.90%	4	136
26	Transceiver Echo Source Latency	1	51	12	3973616363	3973616454	100.00%	5	2	3	3	0.4	99.90%	4	2
27	Transceiver Echo Arrival Latency	1	51	12	3973616363	3973616454	100.00%	5	94	98	105	3.97	99.90%	4	94
28	Transceiver Trip Latency	1	51	12	3973616363	3973616454	100.00%	5	333	372	408	31.64	99.90%	4	333
29	Transceiver Inter Arival Time	1	51	12	3973616363	3973616454	100.00%	5	1E+06	1004195	1004563	222.75	99.90%	4	1003887
30	Transceiver status	1	51	12	3973616363	3973616454	0	0	0						
31	Receiver	2	50	11	3978592546	3978592584	99	99	0	0	0				
32	Transceiver Send End2End Latency	1	51	12	3978620622	3978620714	100.00%	5	225	261	349	44.73	99.90%	4	225
33	Transceiver Send Source Latency	1	51	12	3978620622	3978620714	100.00%	5	5	6	8	1.2	99.90%	4	5
34	Transceiver Send Arrival Latency	1	51	12	3978620622	3978620714	100.00%	5	139	153	209	27.8	99.90%	4	139

Figure 26: Excelerator output displayed in Excel

The Excelerator can be started with the following command:

```
excelerator <filename>
```

<filename> provides a filename to a file for the Excelerator to output the information

4.3 Error Log Tool

Error Log is a tool that displays information published in the Error Report topic.

The errorReportTopic can be found in the DDSTouchStoneReports partition provided by DDS Touchstone. When an error occurs in one of the Touchstone components, details of the problem are published to this topic.

The Error Log tool subscribes to the error report topic and prints the information to the screen.

The following is an example Error Log output:

```
Received Error report from (1,50,14)
Message: ./pother line 1569:
    transmitter_del_topic: delete Topic failed => DDS_RETCODE_BAD_PARAMETER

Received Error report from (2,50,15)
Message: ./pother line 2127:
    receiver_del_topic: delete Topic failed => DDS_RETCODE_BAD_PARAMETER
```

The Error Log tool can be started with the following command:

```
errorlog
```

Note: It is recommended that the Error Log is run in a separate shell as the program output will print to the screen.

4.4 Spotter Tool

The Spotter program is similar to the Watcher and Error Log tools. The Spotter subscribes to the discoveryReportTopic in the DDSTouchStoneReports partition. Discovery information published to this topic from the Touchstone components. The Spotter retrieves all discoveryReportTopic information and prints it to the screen.

The following is an example Spotter.

```
Discovered DataReader (2,50,11):
   creation time: 7714.00 usec
   discovery time: 530790.00 usec
   So discovered 523076.00 usec after creation
```

From the output displayed above, the creation time is mapped to the creation_duration field in the discoveryReportTopic type. This is the time taken to create the component.

The discovery time is mapped to the discovery_time field. This is the total time taken for the component to be created and for it to receive its first message.

The discovered after creation time calculation is performed by the Spotter tool. This is not part of the discoveryTopicReport type.

The Spotter tool can be started with the following command:

```
spotter
```

Note: It is recommended that the Spotter is run in a separate shell as the program output will print to the screen.

5 Topic Reference Guide

The section lists and describes the Topics used in the DDS Touchstone suite.

5.1 Definition Topics

The Definition Topics are used to control and configure the DDS Touchstone component objects, such as Transmitters, Receivers, Transceivers and Transponders.

The following are the DDS Touchstone Definition Topics:

- transmitterDefTopic
- receiverDefTopic
- transceiverDefTopic
- transponderDefTopic

5.1.1 Common Fields

The fields common to all definition topics are listed below:

Field	Description	Value Type
Group_id	This must match the Touchstone group_id for the Touchstone instance that the component will reside on. When referencing a group containing multiple Touchstone instances, this command topic will be received and processed by all components in that grouping.	integer id
component_id transmitter_id receiver_id transceiver_id transponder_id	This id is used to uniquely identify the component to be created or edited. If this is the first time using this component id then a new component will be created with the given id. If a component with the same id value already exists then these field values with replace the values in the existing component attributes.	integer id
partition_id	Identifies the partition that the component will belong to. If the partition does not exist then one will be create with the given id.	integer id
topic_id	Sets the id for the Topic to be used for the component. If this is the first time using this topic id then a new topic will be created with the given id.	integer id

	field values with replace the values in the existing topic attributes. For the Transmitter or the Receiver components this will represent a throughput_topic For Transceiver or Transponder components this will represent a latency_topic	

topic_kind	Sets the topic communication type for the referenced topic:
	
	Best_Effort = Best Effort reliability with Volatile durability
	
	Reliable = Reliable with Volatile durability
	
	Transient = Reliable with Transient durability
	
	Persistent = Reliable with Persistent durability
		Best_Effort
		Reliable
		Transient
		Persistent

scheduling_class	Sets the operating system scheduling policy.	Timesharing RealTime
Thread_priority	Sets the thread priority	Priority Level: 0 (lowest) 100 (Highest)

5.1.2 transmitterDefTopic

In addition the common fields the transmitterDefTopic uses the following fields:

Message_size	Sets the size of the Messages to be sent	bytes
Messages_per_burst	Sets the number of the Messages per write burst.	
Burst_period	Sets the interval between write bursts. These values are in milliseconds	milliseconds

5.1.3 receiverDefTopic

In addition to the common fields the receiverDefTopic uses the following field:

Report_period	Sets the interval between receiverReportTopic publications.	milliseconds
Polling_period	Sets the interval between 'take' actions by the receiver. This value is in milliseconds.	milliseconds

5.1.4 transceiverDefTopic

In addition to the common fields the transceiverDefTopic uses the following field:

message_size	Sets the size of the Messages to be sent	bytes
write_period	Sets the interval between message publications.	milliseconds
Report_period	Sets the interval between receiverReportTopic publications.	milliseconds

5.1.5 transponderDefTopic

The transponderDefTopic does not use any additions fields other than the common definition topic fields.

5.2 Qos Topics

The Qos Topics are using to configure the Quality of Service attributes for a given DDS Touchstone component.

The following are the DDS Touchstone Qos Topics:

- transmitterQosTopic
- receiverQosTopic
- transceiverQosTopic
- transponderQosTopic

5.2.1 Common Fields

All Touchstone Qos topics contain the following common fields:

group_id
component_id (i.e. transmitter_id, receiver_id, transceiver_id or transponder_id)
partition_id

For a description of these fields see Section 5.1.1

5.2.2 transmitterQosTopic

qos.deadline.period.sec	The period in which a new a sample is written, measured in seconds	seconds
qos.deadline.period.nanosec	The period in which a new a sample is written, measured in nanoseconds	nanoseconds
qos.latency_budget.duration.sec	Used by the Data Distribution Service for optimization, measured in seconds	seconds
qos.latency_budget.duration.nanosec	Used by the Data Distribution Service for optimization, measured in nanoseconds	nanoseconds
qos.transport_priority.value	A priority hint for the underlying transport layer	

5.2.3 receiverQosTopic

qos.deadline.period.sec	The period in which a new a sample is written, measured in seconds	seconds
qos.deadline.period.nanosec	The period in which a new a sample is written, measured in seconds	nanoseconds
qos.latency_budget.duration.sec	Used by the Data Distribution Service for optimization, measured in seconds	seconds
qos.latency_budget.duration.nanosec	Used by the Data Distribution Service for optimization, measured in nanoseconds	nanoseconds

5.2.4 transceiverQosTopic

writer_qos.deadline.period.sec	The period in which a new a sample is written, measured in seconds	seconds
writer_qos.deadline.period.nanosec	The period in which a new a sample is written, measured in nanoseconds	nanoseconds
writer_qos.latency_budget.duration.sec	Used by the Data Distribution Service for optimization, measured in seconds	seconds
writer_qos.latency_budget.duration.nanosec	Used by the Data Distribution Service for optimization, measured in nanoseconds	nanoseconds
writer_qos.transport_priority.value	A priority hint for the underlying transport layer	
reader_qos.deadline.period.sec	The period within which a new sample is expected, measured in seconds	seconds

Reader_qos.deadline.period.nanosec	The period within which a new sample is expected, measured in nanoseconds	nanoseconds
Reader_qos.latency_budget.duration.sec	Used by the Data Distribution Service for optimization, measured in seconds	seconds
reader_qos.latency_budget.duration.nanosec	Used by the Data Distribution Service for optimization, measured in nanoseconds	nanoseconds

5.2.5 transponderQosTopic

qos.deadline.period.sec	The period in which a new a sample is written, measured in seconds	seconds
qos.deadline.period.nanosec	The period in which a new a sample is written, measured in nanoseconds	nanoseconds
qos.latency_budget.duration.sec	Used by the Data Distribution Service for optimization, measured in seconds	seconds
qos.latency_budget.duration.nanosec	Used by the Data Distribution Service for optimization, measured in nanoseconds	nanoseconds
qos.transport_priority.value	A priority hint for the underlying transport layer	

5.3 Report Topics

The DDS Touchstone report topics are used to allow components to publish benchmark information to the subscribing tools such as the Watcher, Excelerator, ErrorLog and Spotter.

The following are the DDS Touchstone report topics:

- transmitterReportTopic
- receiverReportTopic
- transceiverReportTopic
- transponderReportTopic
- errorReportTopic
- discoveryReportTopic

5.3.1 Common Fields

The Touchstone component report topics (i.e. transmitterReportTopic, recieverReportTopic, transceiverReportTopic and transponderReportTopic) contain the following common fields:

application_id	Identifies the Touchstone application instance that produced the report	integer id
component_id transmiiter_id receiver_id transceiver_id transponder_id	For a description of these fields see Section 5.1.1	integer id
partition_id	For a description of these fields see Section 5.1.1	integer id

The other Touchstone report topics (errorReportTopic and discoveryReportTopic) also share these fields with the exception of the component_id. This is because these report types are linked to a Touchstone instance and not a component instance

5.3.2 transmitterReportTopic

config_number	**	
write_status.deadlines_missed	Number of deadline	

** To be completed

5.3.3 receiverReportTopic

config_number	**	
throughput	**	
read	**	
reader_status.samples_lost	**	
reader_status.samples_rejected	**	
reader_status.deadlines_missed	Number of messages that have missed the period within which a new sample is expected	

** To be completed

5.3.4 transceiverReportTopic

config_number	**	
writer_status.deadlines_missed	Number of messages that have missed the period within which a new sample is sent	
reader_status.samples_lost	**	
reader_status.samples_rejected	**	
reader_status.deadlines_missed	Number of messages that have missed the period within which a new sample is expected	

** To be completed

The transceiverReportTopic also contains arrays of fields that contain the reports latency information. Each of the array elements represent a percentile of the latency measurements taken. Each percentile element has a field to describe the following statistical information:

- percentile
- sample_count
- minimum
- average
- maximum
- deviation

Each array contains four element sets of fields which represents the following fixed percentiles values of latency measurements taken:

- 100.0%
- 99.9%
- 99.0%
- 90.0%

The individual arrays represent the following latency measurement information.

inter_arrival_time	**
send_latency	**
echo_latency	**
trip_latency	**
send_source_latency	**
send_arrival_latency	**
send_trip_latency	**
echo_source_latency	**
echo_arrival_latency	**
echo_trip_latency	**

** To be completed

5.3.5 transponderReportTopic

config_number	**	
writer_status.deadlines_missed	Number of messages that have missed the period within which a new sample is sent	
reader_status.samples_lost	**	
reader_status.samples_rejected	**	
reader_status.deadlines_missed	Number of messages that have missed the period within which a new sample is expected	

** To be completed

5.3.6 errorReportTopic

entity_id	Identifies the component that submitted the report	integer id
message	Contains the error message	

5.3.7 discoveryReportTopic

entity_id	Identifies the component that submitted the report	integer id
report_kind	Specifies if the report is for a DataWriter or a DataReader	DataWriterDiscovery DataReaderDiscovery
samples_missed	**	
discovery_time	The discovery time is the total time taken for the component to be created, discovered and receive its first message. This time will include any possible writer action delays.	microseconds
creation_duration	Time taken to create the component	microseconds

5.4 Throughput Topic

application_id	Identifies the Touchstone application instance that contains the transmitter which created the topic	integer id
transmitter_id	Identifies the transmitter that publishes the throughput topic	integer id
random_id	**	integer id
sequence_number	Used to order the messages sent	
config_number	**	
creation_duration	Time taken to create the data writer in the Transmitter which sent the message	microseconds
creation_time	Timestamp from when the data writer in the Transmitter which sent the message was created	timestamp
write_timestamp	Timestamp from when the message was published	timestamp
payload_data	Message Payload	

** To be completed

5.5 Latency Topics

The following fields are in the LatencyTopic and the LatencyEchoTopic

application_id	Identifies the Touchstone application instance that contains the transceiver which created the topic	integer id
transceiver_id	Identifies the transceiver that publishes the latency topic	integer id
random_id	**	integer id
sequence_number	Used to order the messages sent	
config_number	**	
write_timestamp	Timestamp from when the message was published by the Transceiver	timestamp
echo_timestamp	Timestamp from when the message was republished by the Transponder	timestamp
source_latency	**	
arrival_latency	**	
send_latency	**	
payload_data	Message Payload	

** To be completed

5.6 Recorder Topics

5.6.1 recorderCommandTopic

recorder_id	Identifies the recorder to send the command to. This is done by using the recorder application_id as a reference	integer id
command	Selects the command to send to the Recorder. STOP Stops Recording RECORD Starts Recording, this command also the time taken to issue each action COMPOSE Starts Recording, this command disregards the time taken to issue each action PLAY Starts a Replay of a Recording REPEAT Repeats a Recording Playback QUIT Shuts down the recorder	STOP RECORD COMPOSE PLAY REPEAT QUIT

5.6.2 recorderStateTopic

recorder_id	Identifies which recorder to check the state of. This is done by using the recorder application_id as a reference	integer id
state	State values: STOPPED Recorder has Stopped RECORDING Currently Recording COMPOSING Currently Composing PLAYING Currently replaying a Recording TERMINATED Shuts down the recorder	STOPPED RECORDING COMPOSING PLAYING TERMINATED