OpenSplice Automated Testing and Debugging Tool

Version 6.x

User Guide

PrismTech

OpenSplice Automated Testing and Debugging Tool

User Guide

PrismTech

Part Number: OS-TTUG

Doc Issue 07, 15 Jan 14

CONTENTS

Table of Contents

Preface

About the User Guide

The OpenSplice Automated Testing and Debugging Tool *User Guide* is intended to provide a complete reference on how to configure the tool and use it to test applications generated with the OpenSplice DDS software.

This *User Guide* is intended to be used after the OpenSplice DDS software has been installed and configured according to the instructions in the OpenSplice *Getting Started Guide*.

Intended Audience

This OpenSplice Automated Testing and Debugging Tool *User Guide* is for everyone using the tool (which is usually referred to as the Tester) to assist in developing and debugging their DDS applications with OpenSplice DDS software.

Organisation

Chapter 1, *Introduction*, provides general information about the Automated Testing and Debugging Tool.

Chapter 2, *Getting Started*, gives an introduction to the use of the Tester, with descriptions of the main features.

Chapter 3, *Familiarization Exercises*, shows how to perform some typical tasks with step-by-step instructions.

Chapter 4, *Command Reference* has a complete list of all the commands available.

Chapter 5, *Scripting* describes how to automate repetitive testing procedures with scripts and macros, provides a list of all of the built-in script instructions, and shows how different scripting languages can be installed and used with the Tester.

Chapter 6, *Message Interfaces* has information about testing applications with non-DDS interfaces.

Appendix A, *Scripting BNF*, contains the complete Scripting BNF listing for reference.

Conventions

The conventions listed below are used to guide and assist the reader in understanding this *User Guide*.

⚠ Item of special significance or where caution needs to be taken.

i Item contains helpful hint or special information.

WIN Information applies to Windows (*e.g.* XP, 2003, Windows 7) only.

UNIX Information applies to Unix-based systems (*e.g.* Solaris) only.

Hypertext links are shown as *blue italic underlined*.

On-Line (PDF) versions of this document: Cross-references such as ‘see *Contacts* on page 11’ act as hypertext links: click on the reference to jump to the item.

```
%  Commands or input which the user enters on the
   command line of a computer terminal
```

`Courier` fonts indicate programming code, commands, file names, and values stored in variables and fields.

Extended code fragments and log file contents are shown in shaded boxes:

```
NameComponent newName[] = new NameComponent[1];

// set id field to “example” and kind field to an empty string
newName[0] = new NameComponent (“example”, ““);
```

Italics and ***Italic Bold*** are used to indicate new terms, or emphasise an item.

Sans-serif and **Sans-serif Bold** are used to indicate components of a Graphical User Interface (GUI) or an Integrated Development Environment (IDE), such as a Cancel button, and sequences of actions, such as selecting **File > Save** from a menu.

The names of keyboard keys are shown in sans-serif small caps, *e.g.* Return. (Combinations of keys to be pressed simultaneously have their names joined with a ‘plus’ sign: Ctrl+C and Ctrl+Alt+Delete.) Names of navigation keys and keys on the numeric pad are spelled out (*e.g.* Left, Down, Plus, Minus).

Angle brackets **< >** enclosing code, command arguments, and similar types of text strings, are used to indicate ‘placeholders’ to be replaced by user-supplied values.

Step 1: One of several steps required to complete a task.

Contacts

PrismTech can be reached at the following contact points for information and technical support.

USA Corporate Headquarters

PrismTech Corporation
400 TradeCenter
Suite 5900
Woburn, MA
01801
USA

Tel: +1 781 569 5819

European Head Office

PrismTech Limited
PrismTech House
5th Avenue Business Park
Gateshead
NE11 0NG
UK

Tel: +44 (0)191 497 9900
Fax: +44 (0)191 497 9901

Web:	*http://www.prismtech.com*	
Technical questions:	*crc@prismtech.com*	(Customer Response Center)
Sales enquiries:	*sales@prismtech.com*	

Using Automated Testing and Debugging Tool

CHAPTER 1 Introduction

This chapter provides a brief introduction to the Tester.

1.1 Features

The OpenSplice Automated Testing and Debugging Tool provides an easy way of displaying messages produced in OpenSplice and also provides means to publish messages manually or with a script.

(The OpenSplice Automated Testing and Debugging Tool is usually referred to as Tester; the name `ospltest` is used when referring to the executable program.)

This tool is made with the software tester, and the way he performs his job, in mind. A pre-defined list of topics of interest can be provided. For all topics a reader is created in the correct partition. Once started, the tool receives all instances of the topics of interest and will display them in the sample list in the order they were produced (using the source time stamp). This makes it very easy to see when topics are produced and in what order. It also provides feedback about unexpected updates.

Other features of Tester include the ability to:

- dump a selection of topic(s) to a file
- dump all logged topic instances to a file
- filter the sample list based on key
- filter the sample list based on key and topic name
- filter the sample list based on key
- create a script with a selection of previously sent or received topics
- compare topic samples
- edit topic samples and then write them or dispose the topic instance
- create new topic samples and write them or dispose the topic instances

1.2 Location of Tester in the OpenSplice architecture

Tester is complementary to OpenSplice Tuner (`ospltun`). Tuner supports ‘white box’ application monitoring and tuning, and Tester supports ‘black box’ system testing, debugging, data capture, analysis, and visualization.

1.3 Things to Know

⚠ **NOTE:** Tester uses the internal Control & Monitoring (C&M) API for access to OpenSplice. At this time Tester only supports OpenSplice DDS systems.

Tester can be used both locally (*via* shared memory or single process) and/or remotely (*via* a SOAP connection to the SOAP service).

Tester uses a periodic poll to read data (the default poll period is 250 ms). The normal restrictions for storage scope apply (only keys defined with the topic separate topics for reading, if topics with the same key are produced within a polling period, then only the last topic is read).

Tester uses the default QoS for writing (as provided by the first application which registers the topic) and the weakest QoS for reading. However when specifying the topic in `add topic(s)` or in the topic file the QoS can be given, this QoS must be compatible with the topic QoS as defined when the topic was registered.

⚠ **NOTE:** In order for the Tester system browser to correctly show the complete system, OpenSplice Durability services have to be properly configured so that the transient ‘built-in-topics’ are properly aligned when new nodes join the system. Monitoring the built-in topic sample set on different nodes will quickly reveal any failure in correct lining-up of transient data (in which case there will be different sets visible on different nodes). Monitoring the DCPSHeartbeat built-in topic will reveal fundamental connectivity issues in your system (you should see as many unique heartbeats as there are interconnected nodes in the system).

1.4 Prerequisites

Tester is included in the standard OpenSplice installation.

Tester’s minimum system requirements are the same as for OpenSplice itself; please refer to the Release Notes for both Tester and OpenSplice for details. The OpenSplice DDS *Getting Started Guide* contains additional information about installation and configuration on various systems.

Note that to compile plugins you will also need to have `ant` and JDK1.6 installed (see Section 6.2, *Getting Started with a Message Interface*, on page 85).

Tester has been implemented in the Java Language, and it uses the OpenSplice Command and Management (C&M) API.

Although Tester uses the C&M API, it doesn’t depend on a locally installed or running instantiation of OpenSplice. It can operate either ‘co-located’ with a running DDS target system, or it can operate in ‘remote-connection’ mode (like the Tuner).

When Tester is run on the same platform as OpenSplice, it uses the `OSPL_HOME` environment variable to find the necessary OpenSplice library files. It also uses `OSPL_URI` as its default OpenSplice configuration.

When Tester connects to a remote 'target' platform using SOAP it doesn't use any local environment variables, it just needs to be installed on the machine where you run it.

(Note that the OpenSplice Tuner can be started with a `-uri` command-line parameter (see Section 2.1.2, *Starting - Remote Connection*). This is a new feature that is actually used by the Tester in the system browser, where you can spawn a Tuner (see Section 3.7.6, *To Spawn a Tuner from the System Browser*, on page 41) that then connects to the node/application that the browser is pointing to.)

CHAPTER 2 Getting Started

This chapter describes how to use the Tester's main features.

2.1 Starting and Stopping Tester

Tester may be started by running the OpenSplice Tester application or from a command prompt and `oslptest` with the following command line arguments:

`-?` *or* `-help`	Display the command line options
`-ns`	No splash screen
`-uri <uri>`	URI to connect
`-nac`	No auto-connect upon application start
`-s <path to script>`	Script to run
`-b <path to batch file>`	Batch script to run
`-noplugins`	Do not process plugins
`-plugindir <dir>`	Extra plugin directory to search
`-headless`	Run script or batch script without the GUI
`-rc <port>`	Enable remote control (*via* `<port>`)
`-dsl <language>`	Default script language
`-l <path to topic file>`	Load topics from file

(These preferences can also be set *via* **File > Preferences**.)

2.1.1 Starting - Local Connection

WIN On Windows, use either of the shortcuts created by the installer (on the desktop and in **Start > Programs**) to start Tester.

UNIX On Linux go to the installation directory and execute the command:

```
%  ospltest
```

This will start Tester with separate windows.

2.1.2 Starting - Remote Connection

To connect to a remote platform, execute the command:

```
%  ospltest -uri http://perf1.perfnet.ptnl:50000
```

(Port number 50000 is the default port in a standard DDS shared-memory deployment.)

2.1.3 Stopping

Stop Tester either by using the menu option **File > Exit** or by clicking on the main window ‘close’ button ☒.

2.1.4 Remotely Controlling Tester

Starting Tester in remote control mode *e.g.* "`ospltest -rc <port> -headless`" allows Tester to be controlled from another application, shell script, *etc.*

Use cases for remote control include:

- Using Tester in combination with a commercial or proprietary test system;
- Within a continuous build and test environment this would provide more options to control DDS testing in combination with other application-specific testing;
- In an integrated development environment like Eclipse using Junit for testing.

A Tester instance is controlled *via* a TCP/IP connection. Text-based commands are sent over this connection.

The remote control application can be used by executing the command:

```
    ospltestrc [-p <port>] [-h <host>] <command>
```

where

`<host>` is the host name of the machine that the Tester you wish to control is running on

`<port>` is the port that Tester is listening on (specified by the `-rc` option when Tester was started)

`<command>` is the command to send to Tester.

The remote control commands are:

`stop`	terminate the Tester instance
`batch <batch name>`	execute the batch with `<batch name>`
`script <script name>`	execute the script with `<script name>`

`scenario <scenario>`	execute a scenario which is provided in full text on a single line (new lines "\n" are replaced by "&nbsp")
`connect <optional uri>`	connect to a specified or the default Domain uri
`disconnect`	

When a command is completed the following is reported on a single line:

```
Done
```

When a batch is executed, for each scenario two lines are returned to the test controller:

```
Scenario: <index> of <count> execute: <scenario name>
Scenario: <index> result: <result>
```

2.2 Trying out Tester

Once you have started Tester, you can get a feeling for how to use it with a few simple exercises:

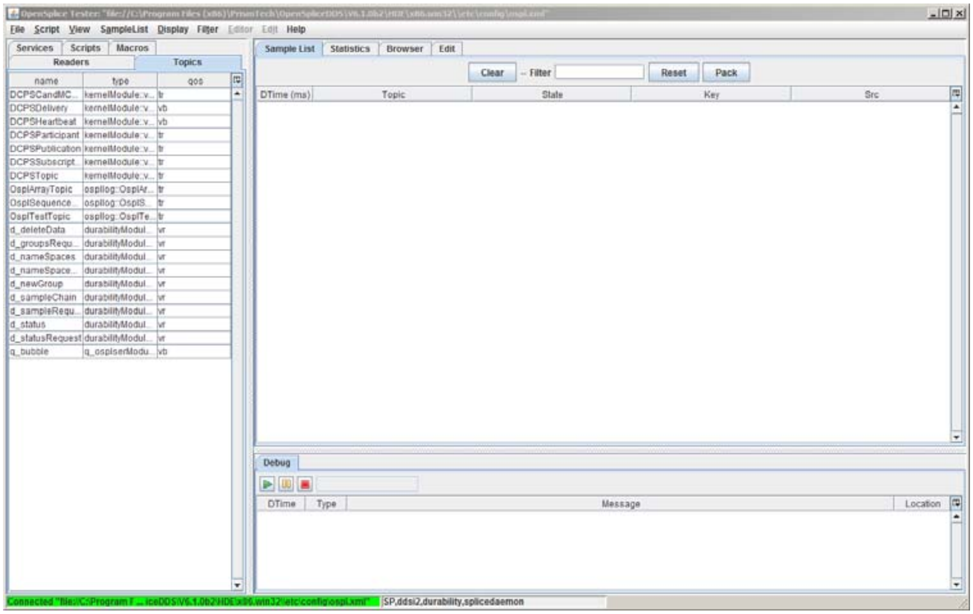
- Create a default reader for some of the registered topics.
- Double-click one of the samples and see all the fields of the topic.
- Browse through the list of samples using the arrow keys.
- Select a topic in the sample list and press F9, then select a field for display in the sample list.
- Select another topic (it need not be of the same type as the one displayed in the topic instance window) and press F2 for a comparison between the two topics.
- Select a topic in the sample list or in the topic list and press F4, then in the Write topic window set the fields to the desired value and write or dispose the topic.
- Choose **File > Dump** on the sample list and save the information of the topic samples in the sample list to a file.
- Have a try with the scripting, it can make your life a lot easier (especially with recurring tasks).

The rest of this chapter describes the features that you will use when you try these exercises.

2.3 Tester Windows

2.3.1 Main Window

Once started Tester presents the user with the following main window.



name	type	qos
DCPSCandMC...	kernelModule::v...	tr
DCPSDelivery	kernelModule::v...	vb
DCPSHeartbeat	kernelModule::v...	vb
DCPSParticipant	kernelModule::v...	tr
DCPSPublication	kernelModule::v...	tr
DCPSSubscript...	kernelModule::v...	tr
DCPSTopic	kernelModule::v...	tr
OsplArrayTopic	ospllog::OsplAr...	tr
OsplSequence...	ospllog::OsplS...	tr
OsplTestTopic	ospllog::OsplTe...	tr
d_deleteData	durabilityModul...	vr
d_groupsRequ...	durabilityModul...	vr
d_nameSpaces	durabilityModul...	vr
d_nameSpace...	durabilityModul...	vr
d_newGroup	durabilityModul...	vr
d_sampleChain	durabilityModul...	vr
d_sampleRequ...	durabilityModul...	vr
d_status	durabilityModul...	vr
d_statusRequest	durabilityModul...	vr
q_bubble	q_osplserModu...	vb

Connected "file://C:\Program F ... iceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl.xml"   SP,ddsi2,durability,splicedaemon

Figure 1 Tester main window

The Command Menu (below) provides direct access to most of the Tester capabilities.

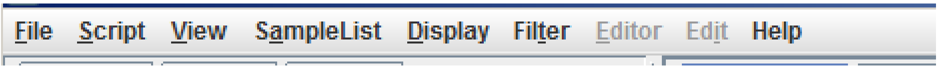


Figure 2 Tester command menu

The Tester main window has three sub-frames:

1. Main tabbed frame for selecting items from a list, such as topics, scenarios, and readers or writers.
2. Working area frame where you will do most of your work such as editing scenarios, investigate samples, and capturing statistics
3. Debug frame used to debug scripts and macros.

2.3.2 Overview Windows

The user can select the type of resource to work with by selecting tabs. These can be the Services and Topics in the system, the Scripts and Macros they have installed, or the Readers for the current Tester timeline.

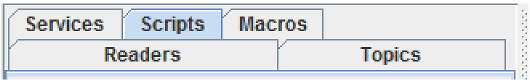
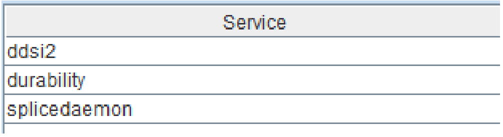


Figure 3 Tester resource tabs

2.3.2.1 Services

Lists the installed services. This is a read-only list.



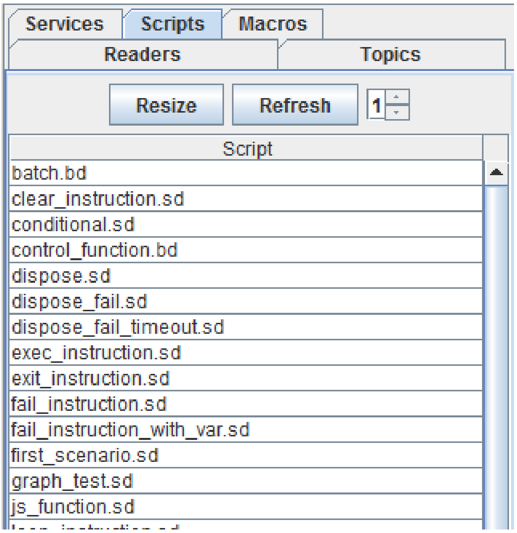
Service
ddsi2
durability
splicedaemon

Figure 4 Tester Services list

2.3.2.2 Scripts

The script list provides a convenient way of selecting an existing script for editing or execution. The list is filled at startup or when clicking the Refresh button. All files in the specified script directory are added to the list. The script directory (or directories) are specified in the preference page.

A script can be selected in the script editor by single-clicking the entry in the table. When the entry is double-clicked the script is loaded in the script editor and executed.

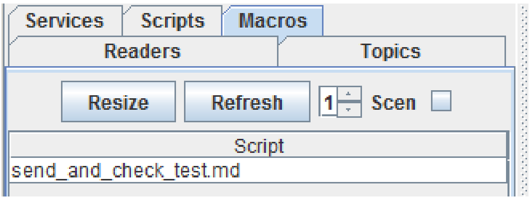


Script
batch.bd
clear_instruction.sd
conditional.sd
control_function.bd
dispose.sd
dispose_fail.sd
dispose_fail_timeout.sd
exec_instruction.sd
exit_instruction.sd
fail_instruction.sd
fail_instruction_with_var.sd
first_scenario.sd
graph_test.sd
js_function.sd

Figure 5 Tester scripts tab

2.3.2.3 Macros

The Macros List is similar to the Scripts List.

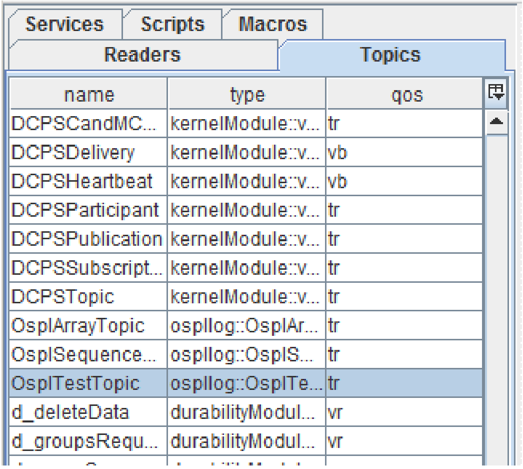


Script
send_and_check_test.md

Figure 6 Tester macros tab

2.3.2.4 Topics

The topics list displays the list of registered topics.

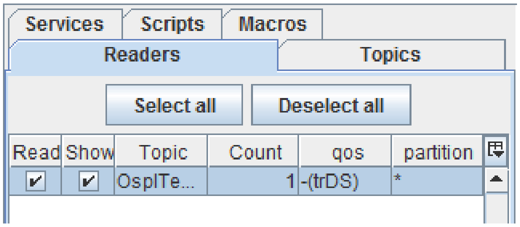


name	type	qos
DCPSCandMC...	kernelModule::v...	tr
DCPSDelivery	kernelModule::v...	vb
DCPSHeartbeat	kernelModule::v...	vb
DCPSParticipant	kernelModule::v...	tr
DCPSPublication	kernelModule::v...	tr
DCPSSubscript...	kernelModule::v...	tr
DCPSTopic	kernelModule::v...	tr
OsplArrayTopic	ospllog::OsplAr...	tr
OsplSequence...	ospllog::OsplS...	tr
OsplTestTopic	ospllog::OsplTe...	tr
d_deleteData	durabilityModul...	vr
d_groupsRequ...	durabilityModul...	vr

Figure 7 Tester topics tab

2.3.2.5 Readers

The readers list displays the readers (and implicit topic writers) for the current Tester timeline. The default name for a reader is the same as the name of the topic it is subscribed to. For each reader the count of received samples (as available in the sample list) is displayed. A check box is provided for changing the read state or the show state. When Read is unchecked the reader stops reading topics. When Show is unchecked the topic of that topic will not be displayed in the sample list.



Read	Show	Topic	Count	qos	partition
☑	☑	OsplTe...	1	-(trDS)	*

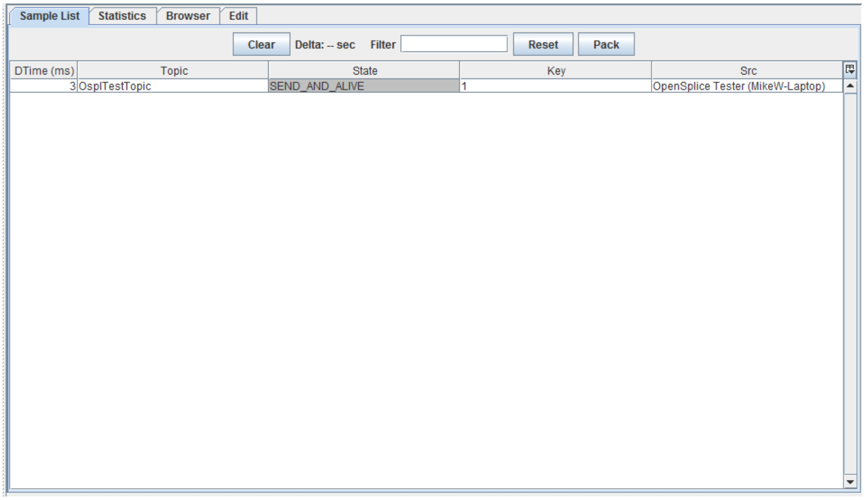
Figure 8 Tester readers tab

2.3.3 Working Windows

These windows support testing activities.

2.3.3.1 Sample List Window

Used to view and generate samples for the current timeline (Readers).

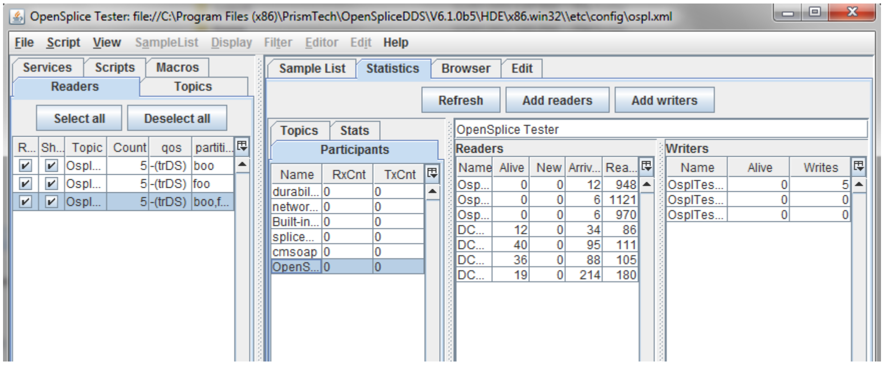


DTime (ms)	Topic	State	Key	Src
3	OsplTestTopic	SEND_AND_ALIVE	1	OpenSplice Tester (MikeW-Laptop)

Figure 9 Sample list window

2.3.3.2 Statistics Window

The statistics window provides statistics for the topics in use, like write count, number of alive topics, *etc.*. Statistics are gathered from the local copy of OpenSplice. To gather statistics from remote nodes, use OpenSplice Tuner.



OpenSplice Tester: file://C:\Program Files (x86)\PrismTech\OpenSpliceDDS\V6.1.0b5\HDE\x86.win32\\etc\config\ospl.xml

R...	Sh...	Topic	Count	qos	partiti...
☑	☑	Ospl...	5	-(trDS)	boo
☑	☑	Ospl...	5	-(trDS)	foo
☑	☑	Ospl...	5	-(trDS)	boo,f...

Name	RxCnt	TxCnt
durabil...	0	0
networ...	0	0
Built-in...	0	0
splice...	0	0
cmsoap	0	0
OpenS...	0	0

OpenSplice Tester

Readers

Name	Alive	New	Arriv...	Rea...
Osp...	0	0	12	948
Osp...	0	0	6	1121
Osp...	0	0	6	970
DC...	12	0	34	86
DC...	40	0	95	111
DC...	36	0	88	105
DC...	19	0	214	180

Writers

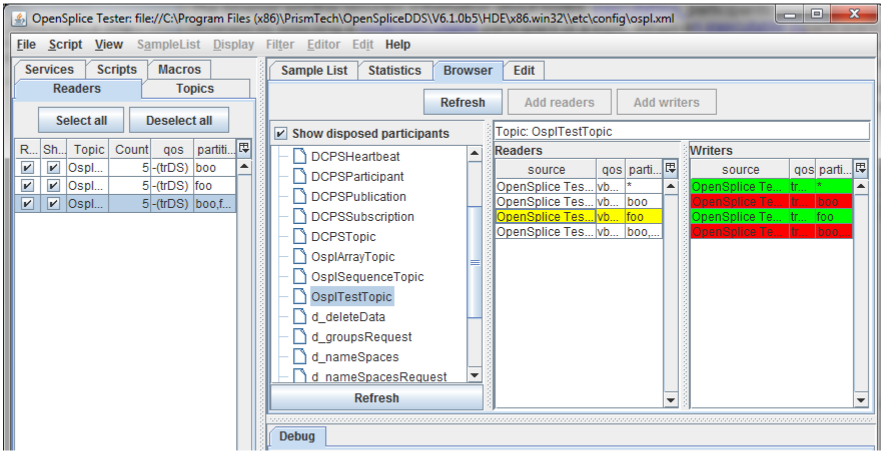
Name	Alive	Writes
OsplTes...	0	5
OsplTes...	0	0
OsplTes...	0	0

Figure 10 Statistics window

2.3.3.3 Browser Window

The browser window provides information about nodes, executables, participants (applications), readers, writers and topics. Information can be browsed by selecting a node/executable participant or a topic. When an executable or participant is selected the reader and writer lists (subscribed and published topics) for that executable/participant are shown. Together with the topic name concise information about the QoS and partition is shown. When the mouse cursor is hovered over the QoS value the hint will show detailed information about the QoS.

When a topic is selected the list of participant readers (subscribe) and writers (publish) are shown, together with concise information about the QoS and partition. By selecting a row in either the reader of writer list the compatible readers/writers will be shown in green and non compatible (by QoS/partition) readers/writers will be shown in red.



OpenSplice Tester: file://C:\Program Files (x86)\PrismTech\OpenSpliceDDS\V6.1.0b5\HDE\x86.win32\\etc\config\ospl.xml

R...	Sh...	Topic	Count	qos	partiti...
☑	☑	Ospl...	5	-(trDS)	boo
☑	☑	Ospl...	5	-(trDS)	foo
☑	☑	Ospl...	5	-(trDS)	boo,f...

☑ Show disposed participants

- DCPSHeartbeat
- DCPSParticipant
- DCPSPublication
- DCPSSubscription
- DCPSTopic
- OsplArrayTopic
- OsplSequenceTopic
- OsplTestTopic
- d_deleteData
- d_groupsRequest
- d_nameSpaces
- d_nameSpacesRequest

Topic: OsplTestTopic

Readers

source	qos	parti...
OpenSplice Tes...	vb...	*
OpenSplice Tes...	vb...	boo
OpenSplice Tes...	vb...	foo
OpenSplice Tes...	vb...	boo,...

Writers

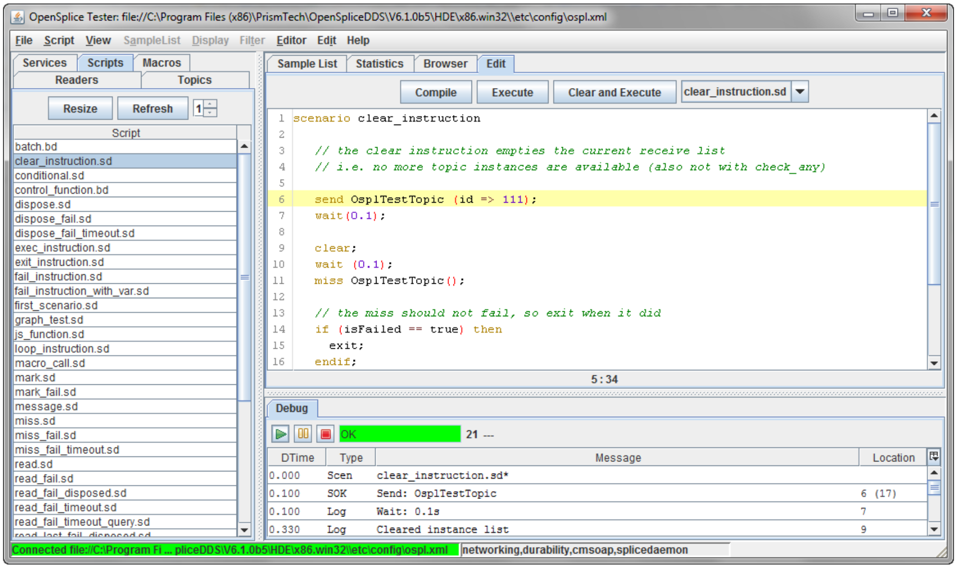
source	qos	parti...
🟩 OpenSplice Te...	tr...	*
🟥 OpenSplice Te...	tr...	boo
🟩 OpenSplice Te...	tr...	foo
🟥 OpenSplice Te...	tr...	boo,...

Figure 11 Tester browser window

2.3.4 Scripting Windows

2.3.4.1 Edit Window

The script window is used for editing scripts. The editor supports syntax highlighting, auto-completion, and more.



Script
batch.bd
clear_instruction.sd
conditional.sd
control_function.bd
dispose.sd
dispose_fail.sd
dispose_fail_timeout.sd
exec_instruction.sd
exit_instruction.sd
fail_instruction.sd
fail_instruction_with_var.sd
first_scenario.sd
graph_test.sd
js_function.sd
loop_instruction.sd
macro_call.sd
mark.sd
mark_fail.sd
message.sd
miss.sd
miss_fail.sd
miss_fail_timeout.sd
read.sd
read_fail.sd
read_fail_disposed.sd
read_fail_timeout.sd
read_fail_timeout_query.sd

```
scenario clear_instruction

  // the clear instruction empties the current receive list
  // i.e. no more topic instances are available (also not with check_any)

  send OsplTestTopic (id => 111);
  wait(0.1);

  clear;
  wait (0.1);
  miss OsplTestTopic();

  // the miss should not fail, so exit when it did
  if (isFailed == true) then
    exit;
  endif;
```

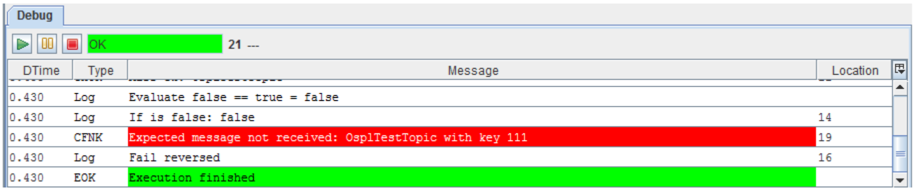
DTime	Type	Message	Location
0.000	Scen	clear_instruction.sd*	
0.100	SOK	Send: OsplTestTopic	6 (17)
0.100	Log	Wait: 0.1s	7
0.330	Log	Cleared instance list	9

Connected file://C:\Program Fi ... pliceDDS\V6.1.0b5\HDE\x86.win32\\etc\config\ospl.xml networking,durability,cmsoap,splicedaemon

Figure 12 Script editing window

2.3.4.2 Debug Window

The debug window displays compile and execution results. Details can be filtered. Positive results are highlighted with green, negative results are highlighted with red.



DTime	Type	Message	Location
0.430	Log	Evaluate false == true = false	
0.430	Log	If is false: false	14
0.430	CFNK	Expected message not received: OsplTestTopic with key 111	19
0.430	Log	Fail reversed	16
0.430	EOK	Execution finished	

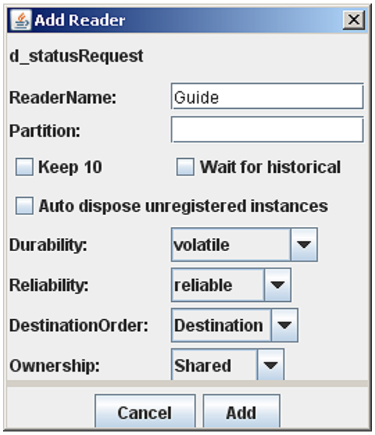
Figure 13 Debug window

2.3.5 Other Windows

The following dialog windows will be used.

2.3.5.1 Add Reader Window

Used to create/define a new Reader.


Add Reader

d_statusRequest

ReaderName: Guide

Partition:

□ Keep 10 □ Wait for historical

□ Auto dispose unregistered instances

Durability: volatile

Reliability: reliable

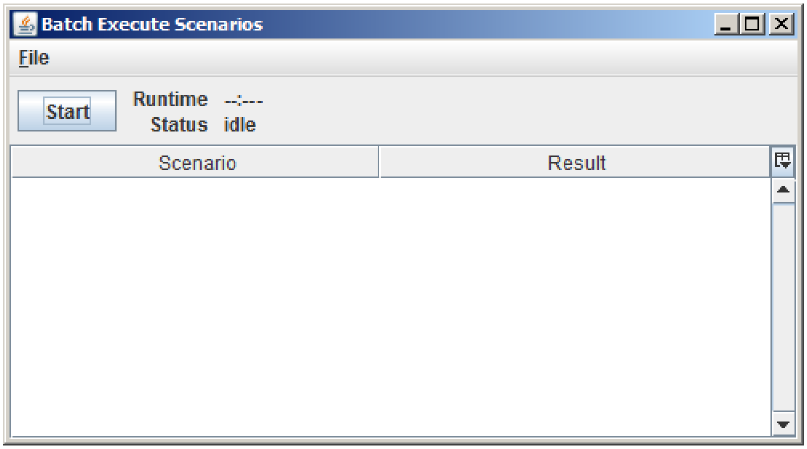
DestinationOrder: Destination

Ownership: Shared

Figure 14 Add Reader dialog

2.3.5.2 Batch Window

Used to Start a batch scenario and display the test results.


Batch Execute Scenarios

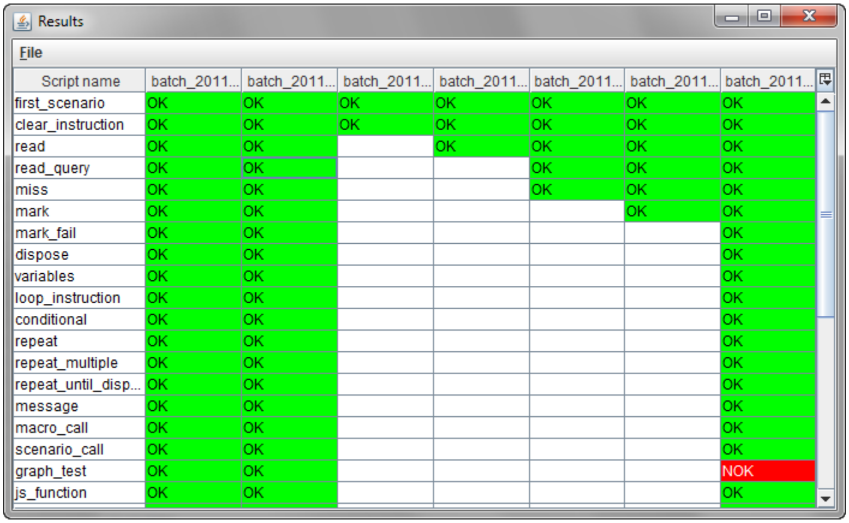
Runtime --:---
Status idle

Scenario	Result

Figure 15 Batch Execute Scenarios window

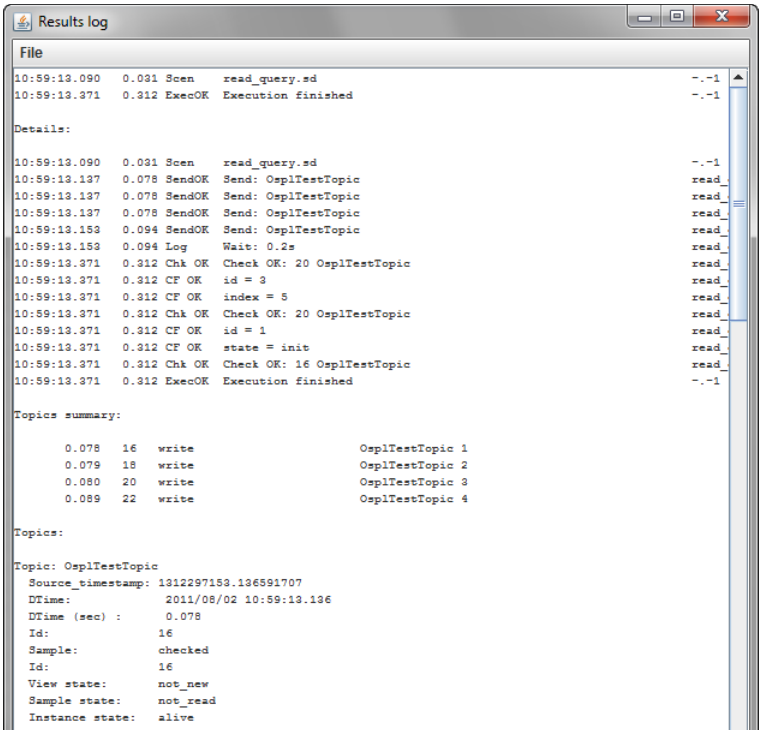
2.3.5.3 Batch Results Window

Displays the detailed results of a batch of scripts. Detailed individual test result can be viewed by double clicking on a test result



Script name	batch_2011...	batch_2011...	batch_2011...	batch_2011...	batch_2011...	batch_2011...	batch_2011...
first_scenario	OK	OK	OK	OK	OK	OK	OK
clear_instruction	OK	OK	OK	OK	OK	OK	OK
read	OK	OK		OK	OK	OK	OK
read_query	OK	OK			OK	OK	OK
miss	OK	OK			OK	OK	OK
mark	OK	OK				OK	OK
mark_fail	OK	OK					OK
dispose	OK	OK					OK
variables	OK	OK					OK
loop_instruction	OK	OK					OK
conditional	OK	OK					OK
repeat	OK	OK					OK
repeat_multiple	OK	OK					OK
repeat_until_disp...	OK	OK					OK
message	OK	OK					OK
macro_call	OK	OK					OK
scenario_call	OK	OK					OK
graph_test	OK	OK					NOK
js_function	OK	OK					OK

Figure 16 Batch results window



```
10:59:13.090   0.031 Scen     read_query.sd                                          -.-1
10:59:13.371   0.312 ExecOK   Execution finished                                     -.-1

Details:

10:59:13.090   0.031 Scen     read_query.sd                                          -.-1
10:59:13.137   0.078 SendOK   Send: OsplTestTopic                                    read_
10:59:13.137   0.078 SendOK   Send: OsplTestTopic                                    read_
10:59:13.137   0.078 SendOK   Send: OsplTestTopic                                    read_
10:59:13.153   0.094 SendOK   Send: OsplTestTopic                                    read_
10:59:13.153   0.094 Log      Wait: 0.2s                                             read_
10:59:13.371   0.312 Chk OK   Check OK: 20 OsplTestTopic                             read_
10:59:13.371   0.312 CF OK    id = 3                                                 read_
10:59:13.371   0.312 CF OK    index = 5                                              read_
10:59:13.371   0.312 Chk OK   Check OK: 20 OsplTestTopic                             read_
10:59:13.371   0.312 CF OK    id = 1                                                 read_
10:59:13.371   0.312 CF OK    state = init                                           read_
10:59:13.371   0.312 Chk OK   Check OK: 16 OsplTestTopic                             read_
10:59:13.371   0.312 ExecOK   Execution finished                                     -.-1

Topics summary:

      0.078   16   write                      OsplTestTopic 1
      0.079   18   write                      OsplTestTopic 2
      0.080   20   write                      OsplTestTopic 3
      0.089   22   write                      OsplTestTopic 4

Topics:

Topic: OsplTestTopic
  Source_timestamp: 1312297153.136591707
  DTime:             2011/08/02 10:59:13.136
  DTime (sec) :      0.078
  Id:               16
  Sample:           checked
  Id:               16
  View state:       not_new
  Sample state:     not_read
  Instance state:   alive
```

Figure 17 Detailed Batch results log

2.3.5.4 Chart Window

Used to plot topic field values.

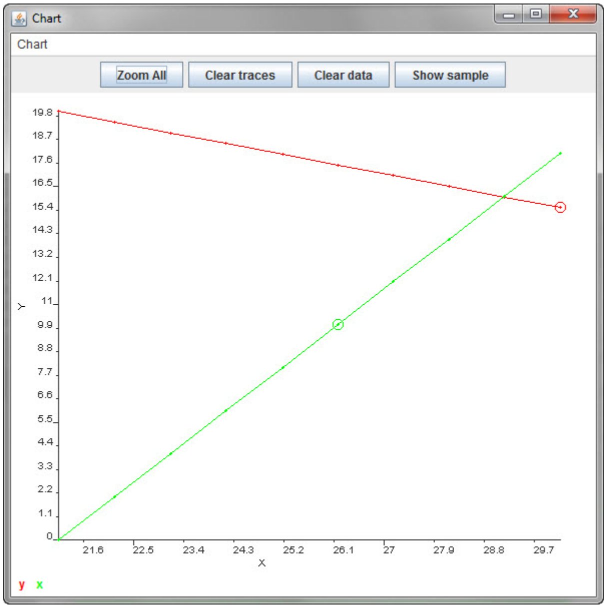
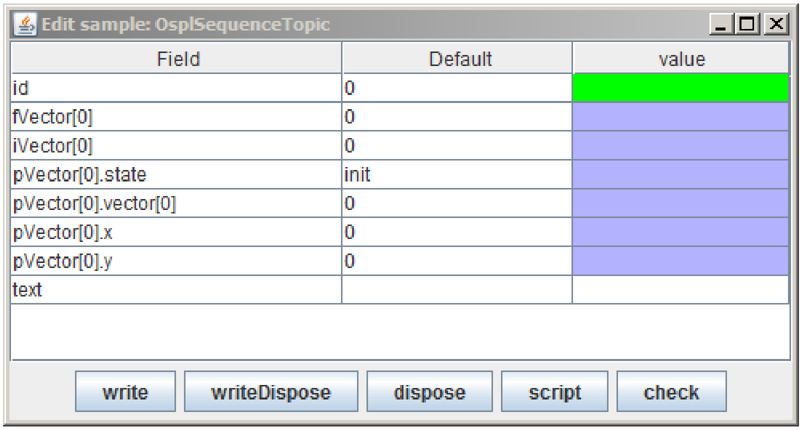


Figure 18 Topic field values graph

2.3.5.5 Edit Sample Window

Used to create samples for a selected topic.


Edit sample: OsplSequenceTopic

Field	Default	value
id	0	🟩
fVector[0]	0	🟦
iVector[0]	0	🟦
pVector[0].state	init	🟦
pVector[0].vector[0]	0	🟦
pVector[0].x	0	🟦
pVector[0].y	0	🟦
text		

Figure 19 Edit sample window

2.3.5.6 Topic Instance Window

The topic sample window is used for displaying field values of a topic. It can be opened by double-clicking a sample in the sample list or by pressing F3 (additional) or F2 (additional with compare) in the sample list while a sample is selected. Special fields are highlighted with colors:

Key field	*(Green)*
Foreign key	*(Yellow)*
Different (compare only)	*(Red)*
Not existing (compare only)	*(Orange)*

When a field is selected, Ctrl+H will toggle between normal and hexadecimal representation, and Ctrl+D will toggle between normal and degrees/radians representation.

CHAPTER 3 Familiarization Exercises

This chapter gives step-by-step instructions for using the Tester to perform many typical tasks to help you bcome familiar with the way it operates.

The exercises in this chapter assume that OpenSplice and the Tester have been succesfully installed. These illustrations make use of the example data supplied with the product.

3.1 Starting the Tester

OpenSplice must already be running before you start the Tester.

Step 1: Start OpenSplice DDS

Step 2: Start the Tester :

UNIX
- On Linux, run `ospltest`.

WIN
- On Windows, choose **OpenSplice DDS Tester** from the Start menu (see *Figure 20* below) *or* run `ospltest` from the OpenSplice DDS command prompt.

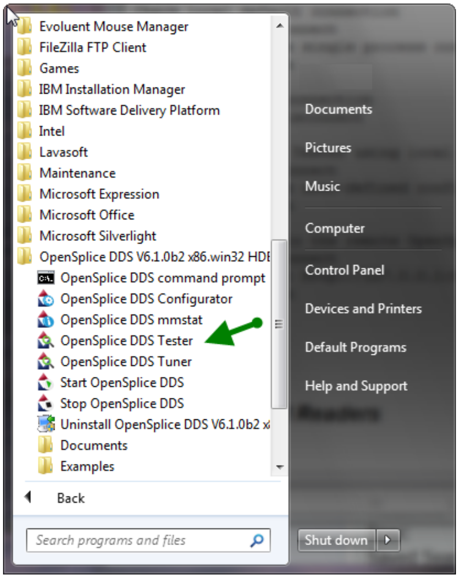


Figure 20 Starting Tester

3.2 Connection management

When it starts the Tester will automatically try to establish a connection to a running instance of OpenSplice using the default URI. You can also make or break connections from the main window by following the steps given below.

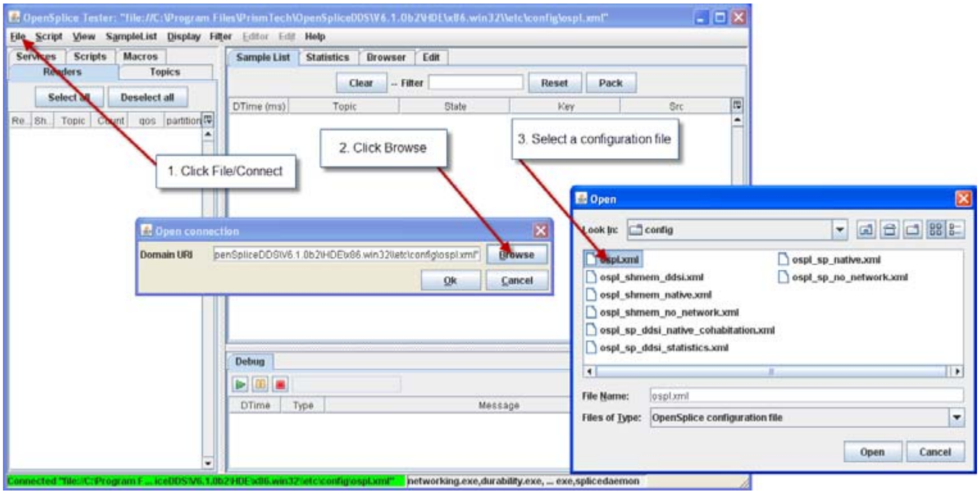
The command line option `-nac` stops tester from making a connection at startup, and with the `-uri` command line option a connection to an alternative URI can be made at startup.

3.2.1 To Connect to a local OpenSplice instance

Step 1: Choose **File > Connect**.

Step 2: Set the path *or* Browse to the configuration file (*e.g.*, `file://<OpenSplice install dir>/etc/config/ospl.xml`).

Step 3: Click the OK button.



OpenSplice Tester: "file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl.xml"

1. Click File/Connect

2. Click Browse

3. Select a configuration file

Open connection

Domain URI: penSpliceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl.xml"

Open

Look In: config

File Name: ospl.xml

Files of Type: OpenSplice configuration file

Connected "file://C:\Program F... iceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl.xml" networking.exe,durability.exe, ... exe,splicedaemon

Figure 21 Connecting to a local OpenSplice instance

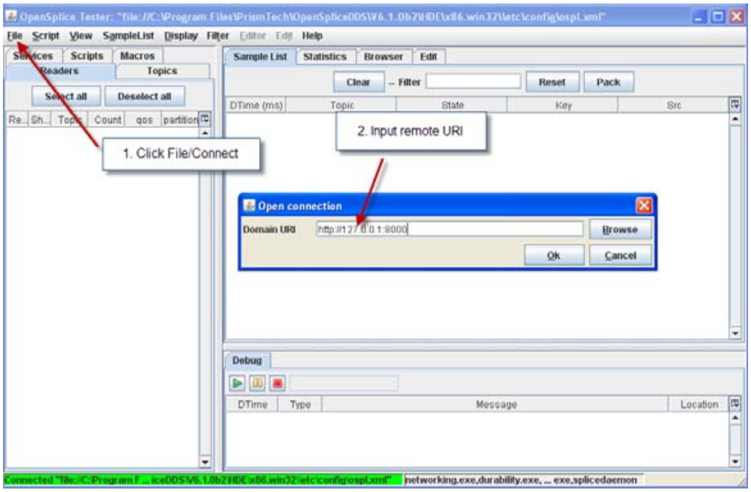
3.2.2 To Connect to a remote OpenSplice instance

Step 1: Choose **File > Connect**.

Step 2: Enter the URI for the remote OpenSplice system (*e.g.*, `http://127.0.0.1:8000`).

> **NOTE:** The port number must be set to the port number as configured for the SOAP service of the remote OpenSplice instance.

Step 3: Click the OK button.



OpenSplice Tester: "file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\\etc\config\ospl.xml"

1. Click File/Connect

2. Input remote URI

Open connection

Domain URI: http://127.0.0.1:8000

Connected "file://C:\Program F ... iceDDS\V6.1.0b2\HDE\x86.win32\\etc\config\ospl.xml" networking.exe,durability.exe, ... exe,splicedaemon

Figure 22 Connecting to a remote OpenSplice instance

3.2.3 To Disconnect

Step 1: Choose **File > Disconnect**.

3.2.4 To Exit Tester

Step 1: Choose **File > Exit** *or* click the Close button on the Tester main window.

3.3 Topics and Readers

Tester can subscribe to multiple topics. These Readers will comprise a timeline for testing. Samples of those topics are automatically read and displayed in the Sample List. Tester readers can also be used to write or edit samples.

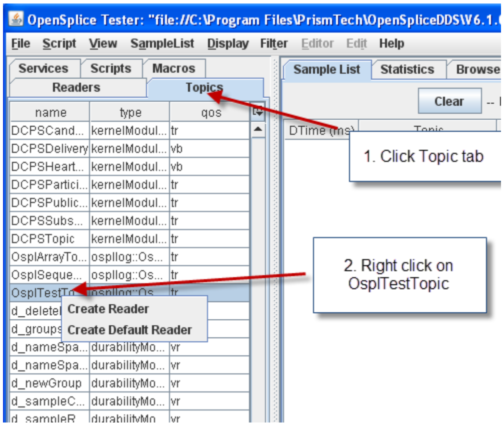
3.3.1 The Topic list

Check the Topic list. Make sure that the Tester is connected to the default URI.

3.3.2 To Add a Reader from the Topic list

Step 1: Select the Topics tab.

Step 2: Right-click OsplTestTopic.


OpenSplice Tester: "file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Sample List Statistics Browse
Clear
DTime (ms) Topic
Create Reader
Create Default Reader

name	type	qos
DCPSCand...	kernelModul...	tr
DCPSDelivery	kernelModul...	vb
DCPSHeart...	kernelModul...	vb
DCPSPartici...	kernelModul...	tr
DCPSPublic...	kernelModul...	tr
DCPSSubs...	kernelModul...	tr
DCPSTopic	kernelModul...	tr
OsplArrayTo...	ospllog::Os...	tr
OsplSeque...	ospllog::Os...	tr
OsplTestTo	ospllog::Os	tr
d_deletel		
d_groups		
d_nameSpa...	durabilityMo...	vr
d_nameSpa...	durabilityMo...	vr
d_newGroup	durabilityMo...	vr
d_sampleC...	durabilityMo...	vr
d_sampleR	durabilityMo	vr

1. Click Topic tab

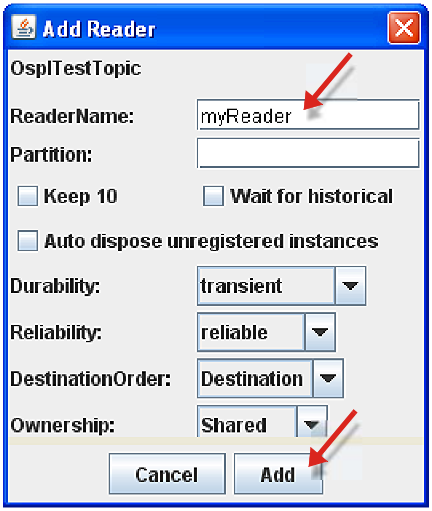
2. Right click on OsplTestTopic

Figure 23 Create Readers from the Topics list

Step 3: Choose **Create Default Reader** from the pop-up menu. The reader will automatically be named the same as the topic.

Step 4: Choose **Create Reader** and modify the (writer) QoS or reader name if desired.

Step 5: Click Add.



OsplTestTopic

ReaderName: myReader

Partition:

☐ Keep 10 ☐ Wait for historical

☐ Auto dispose unregistered instances

Durability: transient

Reliability: reliable

DestinationOrder: Destination

Ownership: Shared

Figure 24 Create myReader

Step 6: Open the Readers tab and you will see the readers you just created.

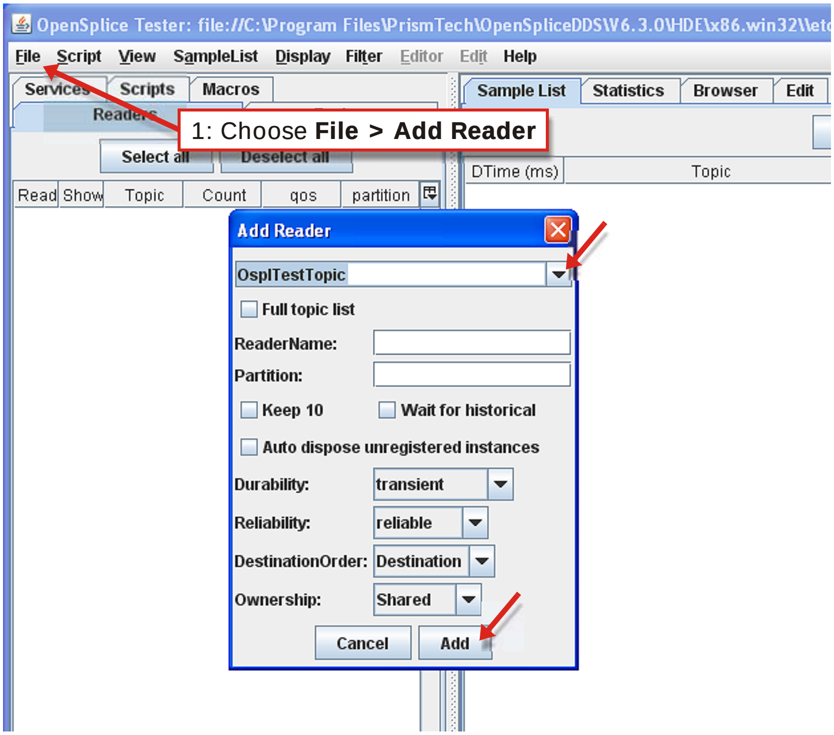
3.3.3 To Add a Reader from the File menu

Step 1: Select the Readers tab.

Step 2: Choose **File > Add Reader**.

Step 3: Select OsplTestTopic from the drop-down list.

Step 4: Click Add.



OpenSplice Tester: file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.3.0\HDE\x86.win32\etc
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers
Select all Deselect all
Read Show Topic Count qos partition
Sample List Statistics Browser Edit
DTime (ms) Topic
Add Reader
Cancel Add

1: Choose **File > Add Reader**

OsplTestTopic

□ Full topic list

ReaderName:

Partition:

□ Keep 10 □ Wait for historical

□ Auto dispose unregistered instances

Durability: transient

Reliability: reliable

DestinationOrder: Destination

Ownership: Shared

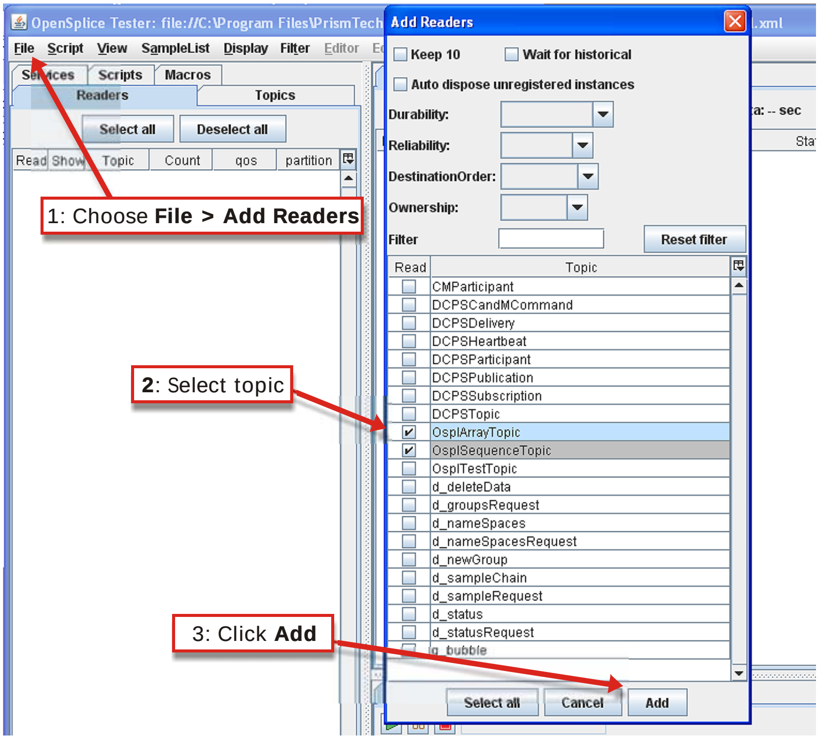
Figure 25 Adding a Reader from the File menu

3.3.4 To Add multiple Readers to the Tester timeline

Step 1: Choose **File > Add Readers**.

Step 2: Type `ospl` in the filter field to limit the list of topics. Select OsplArrayTopic and OsplSequenceTopic.

Step 3: Click Add.



1: Choose **File > Add Readers**

2: Select topic

3: Click **Add**

Add Readers

□ Keep 10 □ Wait for historical

□ Auto dispose unregistered instances

Read	Topic
□	CMParticipant
□	DCPSCandMCommand
□	DCPSDelivery
□	DCPSHeartbeat
□	DCPSParticipant
□	DCPSPublication
□	DCPSSubscription
□	DCPSTopic
☑	OsplArrayTopic
☑	OsplSequenceTopic
□	OsplTestTopic
□	d_deleteData
□	d_groupsRequest
□	d_nameSpaces
□	d_nameSpacesRequest
□	d_newGroup
□	d_sampleChain
□	d_sampleRequest
□	d_status
□	d_statusRequest
□	g_bubble

Figure 26 Adding multiple Readers

3.3.5 To Save the current Readers to a file

If you need to preserve the Readers for a timeline, you can save the current Readers list.

Step 1: Choose **File > Save Readers List**.

Step 2: Enter a name for the new file.

Step 3: Click Save.

3.3.6 To Remove all Readers

Step 1: Choose **File > Remove all Readers**.

3.3.7 To Load Readers from a saved file

Step 1: Choose **File > Load Readers List**.

Step 2: Select the name of the saved file.

Step 3: Click Load.

3.3.8 To Delete a Reader

Step 1: Select OsplTestTopic reader from the list.

Step 2: Press the DELETE key *or* right-click on OsplTestTopic and choose **Delete Reader** from the pop-up menu.

3.4 Samples

3.4.1 Writing and Editing Samples

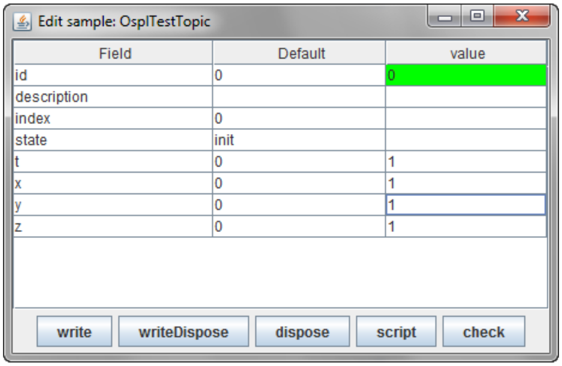
3.4.1.1 To Write Sample Topic data

Step 1: Select OsplTestTopic reader from the list.

Step 2: Press F4 *or* choose **Edit Sample** from the pop-up menu.

Step 3: Enter following values for the fields in the list:

id: `0`
t: `1`, x: `1`, y: `1`, z: `1`



Field	Default	value
id	0	0
description		
index	0	
state	init	
t	0	1
x	0	1
y	0	1
z	0	1

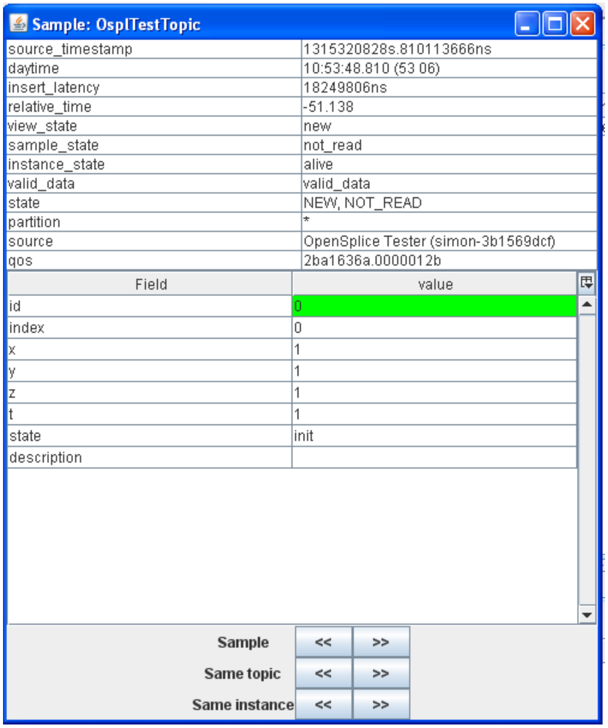
Figure 27 Entering sample topic data

Step 4: Click the write button.

Step 5: Close the Edit Sample window.

3.4.1.2 To display detailed information on sample data

Step 1: Double-click on the first OsplTestTopic sample in the Sample List window.



source_timestamp	1315320828s.810113666ns
daytime	10:53:48.810 (53 06)
insert_latency	18249806ns
relative_time	-51.138
view_state	new
sample_state	not_read
instance_state	alive
valid_data	valid_data
state	NEW, NOT_READ
partition	*
source	OpenSplice Tester (simon-3b1569dcf)
qos	2ba1636a.0000012b

Field	value
id	0
index	0
x	1
y	1
z	1
t	1
state	init
description	

Figure 28 Display detailed sample data information

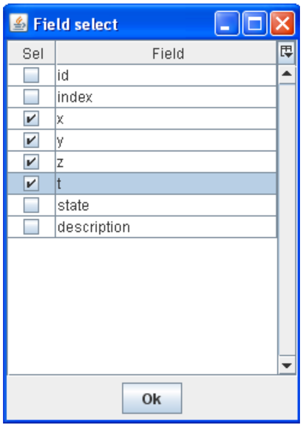
3.4.1.3 To Display extra fields

By default the Sample List displays topic-independent fields. You can add topic-specific fields as follows:

Step 1: Select any sample.

Step 2: Press F9 *or* right-click and choose **Select Extra Fields** from the pop-up menu.

Step 3: Click to select (‘check’) x, y, z and t.

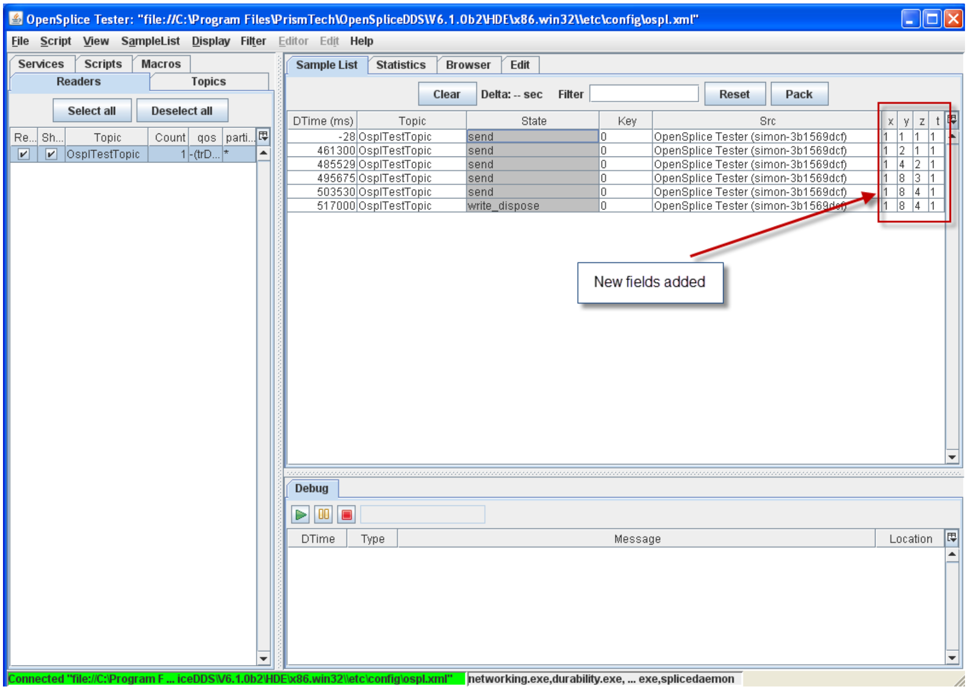


Sel	Field
☐	id
☐	index
☑	x
☑	y
☑	z
☑	t
☐	state
☐	description

Figure 29 Selecting extra fields to display

Step 4: Click OK.

The selected fields will be added to the Sample List.



OpenSplice Tester: "file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\\etc\config\ospl.xml"
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Select all Deselect all
Sample List Statistics Browser Edit
Clear Reset Pack
Debug
DTime Type Message Location

Re...	Sh...	Topic	Count	qos	parti...
☑	☑	OsplTestTopic	1	-(trD...	*

Delta: -- sec Filter

DTime (ms)	Topic	State	Key	Src	x	y	z	t
-28	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	1	1	1
461300	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	2	1	1
485529	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	4	2	1
495675	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	8	3	1
503530	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1
517000	OsplTestTopic	write_dispose	0	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1

New fields added

Connected "file://C:\Program F ... iceDDS\V6.1.0b2\HDE\x86.win32\\etc\config\ospl.xml" networking.exe,durability.exe, ... exe,splicedaemon

Figure 30 New fields added

3.4.1.4 To Edit a sample

Step 1: Select the first sample.

Step 2: Press F4 *or* choose **Edit Sample** from the pop-up menu.

Step 3: Enter following values in the fields:

id: `0`, x: `1`, y: `2`, z: `1`, t: `1`

Step 4: Click Write.

Step 5: Enter following values in the fields:

id: `0`, x: `1`, y: `4`, z: `2`, t: `1`

Step 6: Click Write.

Step 7: Enter following values in the fields:

id: `0`, x: `1`, y: `8`, z: `3`, t: `1`

Step 8: Click Write.

Step 9: Enter following values in the fields:

id: `1`, x: `1`, y: `8`, z: `4`, t: `1`

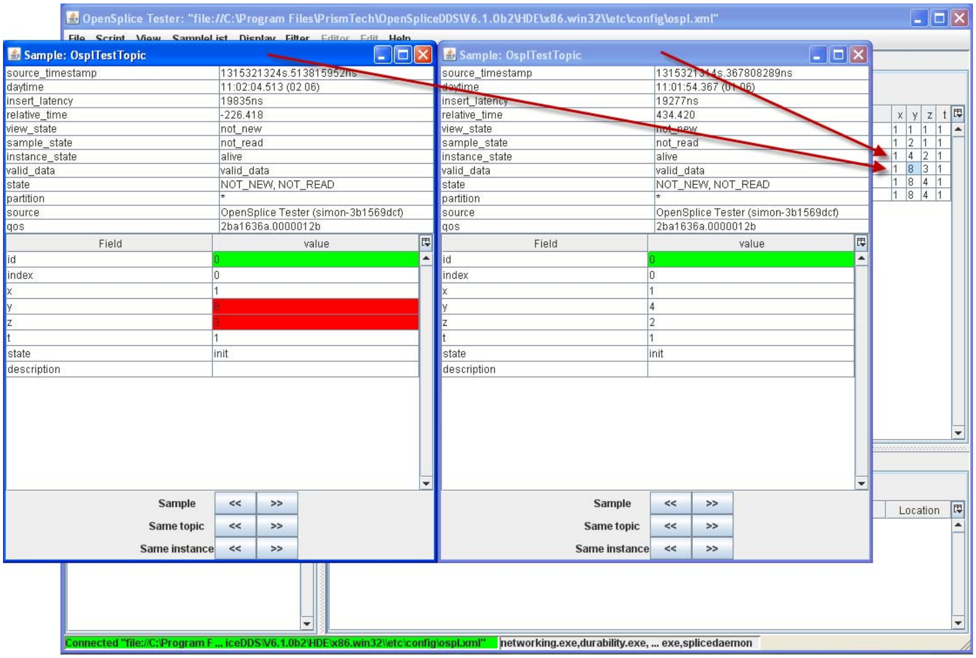
Step 10:Click WriteDispose.

3.4.1.5 To Compare two samples

Step 1: Double-click the sample with the values id: `0`, x: `1`, y: `4`, z: `2`, t: `1`.

Step 2: Select the sample with the values id: `0`, x: `1`, y: `8`, z: `3`, t: `1`.

Step 3: Press F2 *or* choose **Compare Samples** from the pop-up menu.


OpenSplice Tester: "file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\\etc\config\ospl.xml"
File Script View SampleList Display Filter Editor Edit Help
Sample: OsplTestTopic
Sample << >>
Same topic << >>
Same instance << >>
Location

Sample: OsplTestTopic

source_timestamp	1315321324s.513815952ns
daytime	11:02:04.513 (02 06)
insert_latency	19835ns
relative_time	-226.418
view_state	not_new
sample_state	not_read
instance_state	alive
valid_data	valid_data
state	NOT_NEW, NOT_READ
partition	*
source	OpenSplice Tester (simon-3b1569dcf)
qos	2ba1636a.0000012b

Field	value
id	0
index	0
x	1
y	8
z	3
t	1
state	init
description	

Sample: OsplTestTopic

source_timestamp	1315321314s.367808289ns
daytime	11:01:54.367 (01 06)
insert_latency	19277ns
relative_time	434.420
view_state	not_new
sample_state	not_read
instance_state	alive
valid_data	valid_data
state	NOT_NEW, NOT_READ
partition	*
source	OpenSplice Tester (simon-3b1569dcf)
qos	2ba1636a.0000012b

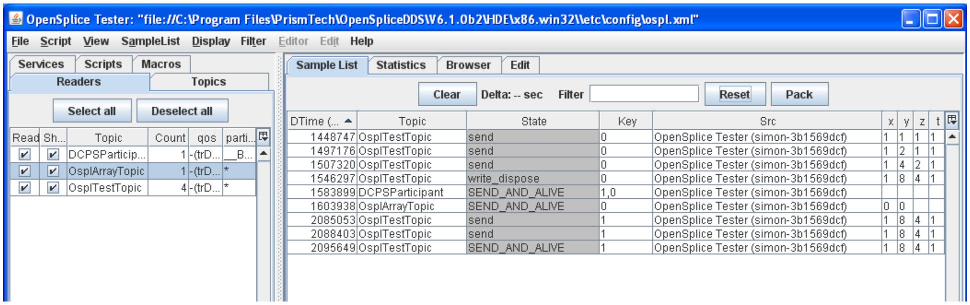
Field	value
id	0
index	0
x	1
y	4
z	2
t	1
state	init
description	

x	y	z	t
1	1	1	1
1	2	1	1
1	4	2	1
1	8	3	1
1	8	4	1
1	8	4	1

Connected "file://C:/Program F ... iceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl.xml" networking.exe,durability.exe, ... exe,splicedaemon

Figure 31 Comparing samples

3.5 Filtering


OpenSplice Tester: "file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\\etc\config\ospl.xml"
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Select all Deselect all
Sample List Statistics Browser Edit
Clear Delta: -- sec Filter Reset Pack

Read	Sh...	Topic	Count	qos	parti...
☑	☑	DCPSParticip...	1	-(trD...	__B...
☑	☑	OsplArrayTopic	1	-(trD...	*
☑	☑	OsplTestTopic	4	-(trD...	*

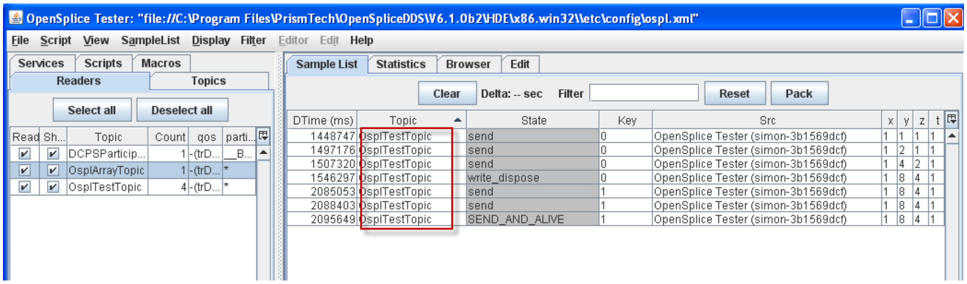
DTime (...	Topic	State	Key	Src	x	y	z	t
1448747	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	1	1	1
1497176	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	2	1	1
1507320	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	4	2	1
1546297	OsplTestTopic	write_dispose	0	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1
1583899	DCPSParticipant	SEND_AND_ALIVE	1,0	OpenSplice Tester (simon-3b1569dcf)				
1603938	OsplArrayTopic	SEND_AND_ALIVE	0	OpenSplice Tester (simon-3b1569dcf)	0	0		
2085053	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1
2088403	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1
2095649	OsplTestTopic	SEND_AND_ALIVE	1	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1

Figure 32 Filtering: un-filtered Topic list

3.5.1 To Filter the Sample List on a Topic

Step 1: Select the OsplTestTopic sample.

Step 2: Press F5 *or* choose **Filter on Topic** from the pop-up menu.


OpenSplice Tester: "file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\\etc\config\ospl.xml"
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Select all Deselect all
Sample List Statistics Browser Edit
Clear Delta: -- sec Filter Reset Pack

Read	Sh...	Topic	Count	qos	parti...
☑	☑	DCPSParticip...	1	-(trD...	__B...
☑	☑	OsplArrayTopic	1	-(trD...	*
☑	☑	OsplTestTopic	4	-(trD...	*

DTime (ms)	Topic	State	Key	Src	x	y	z	t
1448747	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	1	1	1
1497176	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	2	1	1
1507320	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	4	2	1
1546297	OsplTestTopic	write_dispose	0	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1
2085053	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1
2088403	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1
2095649	OsplTestTopic	SEND_AND_ALIVE	1	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1

Figure 33 Sample List filtered by Topic

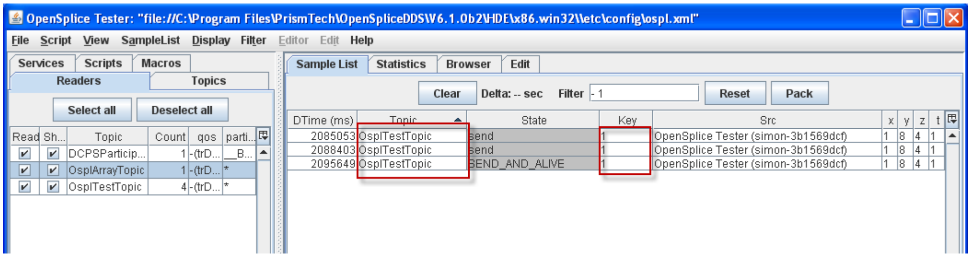
3.5.2 To Reset Filters and display all samples

Step 1: Press F7 *or* choose **Reset filter** from the pop-up menu *or* click the Reset button on the Sample List window.

3.5.3 To Filter on both Topic and Key

Step 1: Select OsplTestTopic with id(key): `1`.

Step 2: Press F5 *or* choose **Filter on topic and key** from the pop-up menu.


OpenSplice Tester: "file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\\etc\config\ospl.xml"
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Select all Deselect all
Sample List Statistics Browser Edit
Clear Delta: -- sec Filter - 1 Reset Pack

Read	Sh...	Topic	Count	qos	parti...
☑	☑	DCPSParticip...	1	-(trD...	__B...
☑	☑	OsplArrayTopic	1	-(trD...	*
☑	☑	OsplTestTopic	4	-(trD...	*

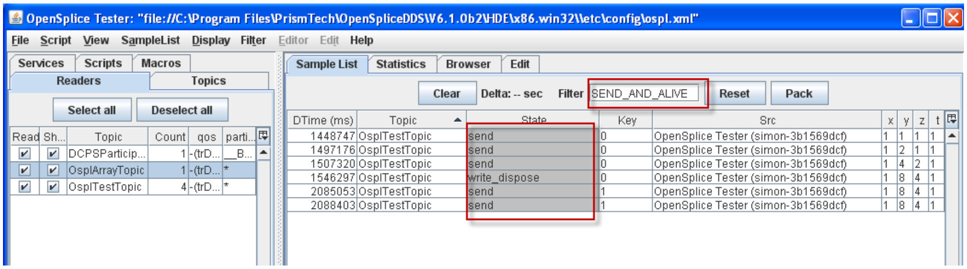
DTime (ms)	Topic	State	Key	Src	x	y	z	t
2085053	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1
2088403	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1
2095649	OsplTestTopic	SEND_AND_ALIVE	1	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1

Figure 34 Sample List filtered by Topic and Key

3.5.4 Filter samples on State

Step 1: Select a sample with a State of `SEND AND ALIVE`.

Step 2: Choose **Filter on State** from the pop-up menu.


OpenSplice Tester: "file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\\etc\config\ospl.xml"
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Select all Deselect all
Sample List Statistics Browser Edit
Clear Delta: -- sec Filter SEND_AND_ALIVE Reset Pack

Read	Sh...	Topic	Count	qos	parti...
☑	☑	DCPSParticip...	1	-(trD...	__B...
☑	☑	OsplArrayTopic	1	-(trD...	*
☑	☑	OsplTestTopic	4	-(trD...	*

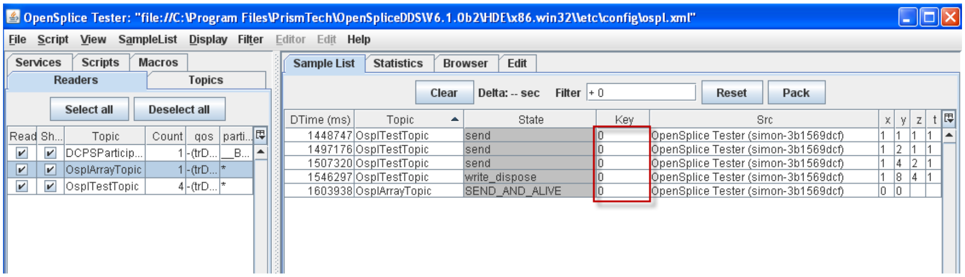
DTime (ms)	Topic	State	Key	Src	x	y	z	t
1448747	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	1	1	1
1497176	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	2	1	1
1507320	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	4	2	1
1546297	OsplTestTopic	write_dispose	0	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1
2085053	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1
2088403	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1

Figure 35 Sample List filtered by State

3.5.5 To Filter Samples on Key value

Step 1: Select OsplTestTopic with id(key): `0`.

Step 2: Choose **Filter on key** from the pop-up menu.


OpenSplice Tester: "file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\\etc\config\ospl.xml"
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Select all Deselect all
Sample List Statistics Browser Edit
Clear Delta: -- sec Filter Reset Pack

Filter: + 0

Read	Sh...	Topic	Count	qos	parti...
☑	☑	DCPSParticip...	1	-(trD...	__B...
☑	☑	OsplArrayTopic	1	-(trD...	*
☑	☑	OsplTestTopic	4	-(trD...	*

DTime (ms)	Topic	State	Key	Src	x	y	z	t
1448747	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	1	1	1
1497176	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	2	1	1
1507320	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	4	2	1
1546297	OsplTestTopic	write_dispose	0	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1
1603938	OsplArrayTopic	SEND_AND_ALIVE	0	OpenSplice Tester (simon-3b1569dcf)	0	0		

Figure 36 Sample List filtered by Key value

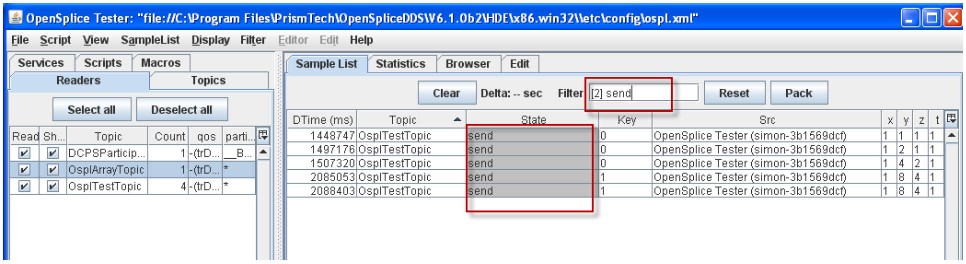
3.5.6 Filter on column text

Step 1: Select the State column of any sample.

Step 2: Choose **Filter on column text** from the pop-up menu.

Step 3: Type in ‘`send`’.

Step 4: Press the ENTER key.


OpenSplice Tester: "file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\\etc\config\ospl.xml"
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Select all Deselect all
Sample List Statistics Browser Edit
Clear Delta: -- sec Filter Reset Pack

Filter: [2] send

Read	Sh...	Topic	Count	qos	parti...
☑	☑	DCPSParticip...	1	-(trD...	__B...
☑	☑	OsplArrayTopic	1	-(trD...	*
☑	☑	OsplTestTopic	4	-(trD...	*

DTime (ms)	Topic	State	Key	Src	x	y	z	t
1448747	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	1	1	1
1497176	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	2	1	1
1507320	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	4	2	1
2085053	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1
2088403	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	1	8	4	1

Figure 37 Sample List filtered by column text

3.5.7 Find specific text

Step 1: Press CTRL+F to open the Find dialog.

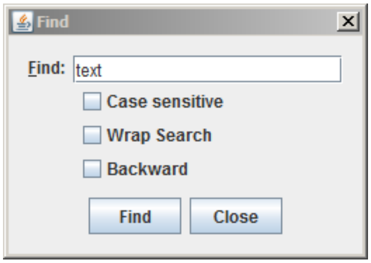


Figure 38 The Find dialog

Step 2: Type in the text to search for, and select any of the options if required.

Step 3: Click Find. The first occurrence of the search text is highlighted.

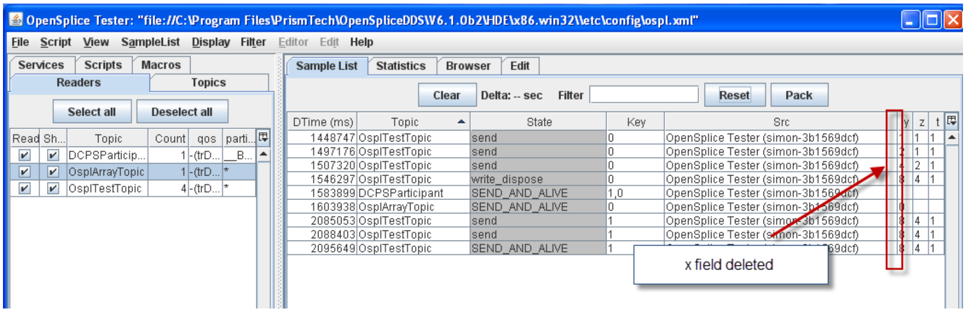
Step 4: Click Find again to find the next occurrence of the search text.

3.6 Working with Samples

3.6.1 To Delete a column from the Sample List table

Step 1: Select the x column of any sample.

Step 2: Press the DELETE key.


OpenSplice Tester: "file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\\etc\config\ospl.xml"
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Select all Deselect all
Sample List Statistics Browser Edit
Clear Delta: -- sec Filter Reset Pack
x field deleted

Read	Sh...	Topic	Count	qos	parti...
☑	☑	DCPSParticip...	1	-(trD...	__B...
☑	☑	OsplArrayTopic	1	-(trD...	*
☑	☑	OsplTestTopic	4	-(trD...	*

DTime (ms)	Topic	State	Key	Src	y	z	t
1448747	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	1	1
1497176	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	2	1	1
1507320	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	4	2	1
1546297	OsplTestTopic	write_dispose	0	OpenSplice Tester (simon-3b1569dcf)	8	4	1
1583899	DCPSParticipant	SEND_AND_ALIVE	1,0	OpenSplice Tester (simon-3b1569dcf)			
1603938	OsplArrayTopic	SEND_AND_ALIVE	0	OpenSplice Tester (simon-3b1569dcf)	0		
2085053	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	8	4	1
2088403	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	8	4	1
2095649	OsplTestTopic	SEND_AND_ALIVE	1	69dcf)	8	4	1

Figure 39 Column deleted from Sample List display

3.6.2 To Chart Sample Data

Using any list of samples:

Step 1: Select the z column of any sample and press the X key.

Step 2: Select the y column of any sample and press the Y key.

Step 3: Choose **SampleList > Show Chart** or press ALT+SHIFT+C to display the chart.

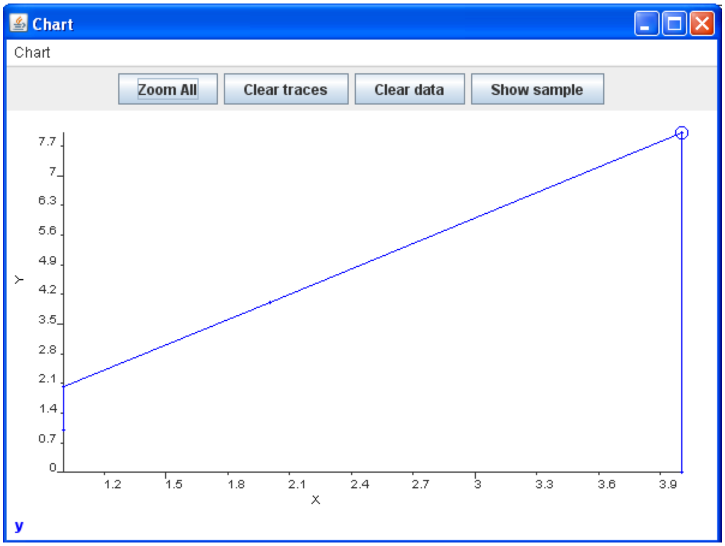


Figure 40 Chart of Sample data

3.6.3 To Dump a sample list to a file

Step 1: Choose **SampleList > Dump**.

Step 2: Enter a name for the file to save.

Step 3: Click Save.

3.6.4 To Dump selected Samples only

Step 1: Select OsplTestTopic with key: `1`.

Step 2: Choose **SampleList > Dump Selection**.

Step 3: Enter a name for the file to save.

Step 4: Click Save.

3.6.5 To Dump to a CSV format file

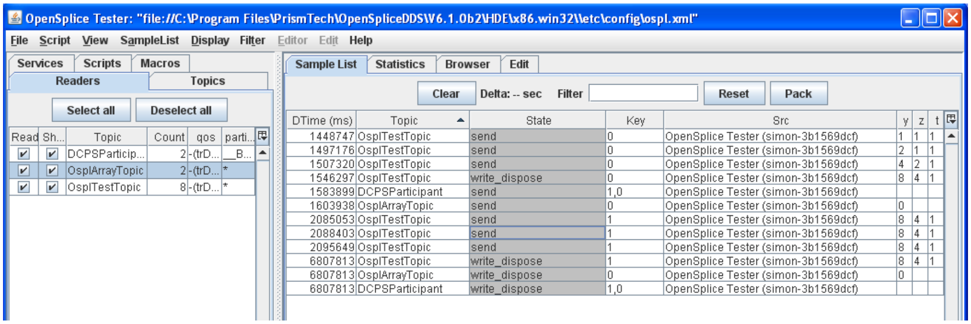
Step 1: Choose **SampleList > Dump to CSV**.

Step 2: Enter a name for the file to save.

Step 3: Click Save.

3.6.6 To Dispose data with Alive state

Step 1: Choose **SampleList > Dispose Alive**.


OpenSplice Tester: "file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\\etc\config\ospl.xml"
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Select all Deselect all
Sample List Statistics Browser Edit
Clear Delta: -- sec Filter Reset Pack

Read	Sh...	Topic	Count	qos	parti...
☑	☑	DCPSParticip...	2	-(trD...	__B...
☑	☑	OsplArrayTopic	2	-(trD...	*
☑	☑	OsplTestTopic	8	-(trD...	*

DTime (ms)	Topic	State	Key	Src	y	z	t
1448747	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	1	1	1
1497176	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	2	1	1
1507320	OsplTestTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	4	2	1
1546297	OsplTestTopic	write_dispose	0	OpenSplice Tester (simon-3b1569dcf)	8	4	1
1583899	DCPSParticipant	send	1,0	OpenSplice Tester (simon-3b1569dcf)			
1603938	OsplArrayTopic	send	0	OpenSplice Tester (simon-3b1569dcf)	0		
2085053	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	8	4	1
2088403	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	8	4	1
2095649	OsplTestTopic	send	1	OpenSplice Tester (simon-3b1569dcf)	8	4	1
6807813	OsplTestTopic	write_dispose	1	OpenSplice Tester (simon-3b1569dcf)	8	4	1
6807813	OsplArrayTopic	write_dispose	0	OpenSplice Tester (simon-3b1569dcf)	0		
6807813	DCPSParticipant	write_dispose	1,0	OpenSplice Tester (simon-3b1569dcf)			

Figure 41 Disposing data with ‘Alive’ state

3.6.7 To Translate Sample data to test script

Step 1: Choose **SampleList > Diff Script**.

The Scripting commands to replicate all of the sample data will be inserted into the current scenario in the Edit window.

3.6.8 Translate selected sample to test script

Step 1: Select a set of samples.

Step 2: Choose **SampleList > DiffScript Selection**.

The Scripting commands to replicate this subset of the sample data will be inserted into the current scenario in the Edit window.

3.7 System Browser (Browser window)

3.7.1 Browse tree

The System Browser is used to examine the Nodes, Participants, and Topics in your system using a tree paradigm.

Step 1: Choose **View > Browser** *or* click the Browser tab of the main window.

Step 2: Expand the all tree.

Step 3: Select Tester participant from the Browser tree. Note that your own Tester is highlighted in yellow in the tree.

```
All participants
- OpenSplice Tester
```

Step 4: Select Built-in participant from

```
Nodes
  + <your machine name>
     + java.exe
        - Built-in participant
```

Step 5: Select Build-in participant from

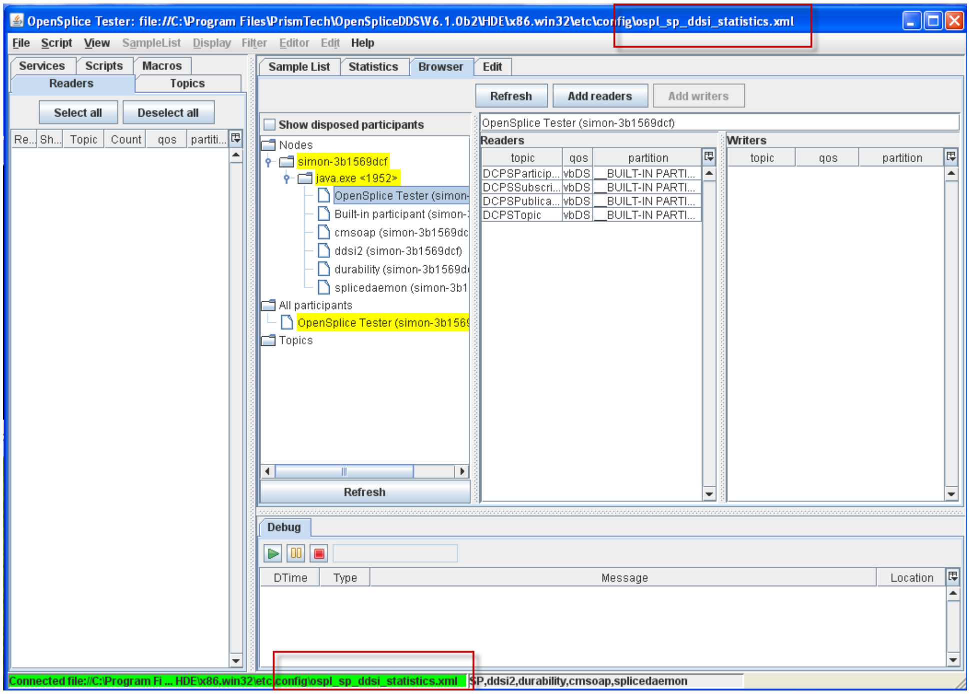
```
Nodes
  + <your machine name>
     + java.exe
        - ddsi2
```

Step 6: View readers and writers of durability service. Select Build-in participant from

```
Nodes
  + <your machine name>
     + java.exe
        - durability
```

Step 7: View readers and writers of splicedaemon. Select Build-in participant from

```
Nodes
  + <your machine name>
     + java.exe
        - splicedaemon
```



OpenSplice Tester: file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml

Re...	Sh...	Topic	Count	qos	partiti...

☐ Show disposed participants

```
Nodes
  simon-3b1569dcf
    java.exe <1952>
      OpenSplice Tester (simon-
      Built-in participant (simon-
      cmsoap (simon-3b1569dc
      ddsi2 (simon-3b1569dcf)
      durability (simon-3b1569d
      splicedaemon (simon-3b1
All participants
  OpenSplice Tester (simon-3b1569
Topics
```

OpenSplice Tester (simon-3b1569dcf)

Readers

topic	qos	partition
DCPSParticip...	vbDS	__BUILT-IN PARTI...
DCPSSubscri...	vbDS	__BUILT-IN PARTI...
DCPSPublica...	vbDS	__BUILT-IN PARTI...
DCPSTopic	vbDS	__BUILT-IN PARTI...

Writers

topic	qos	partition

DTime	Type	Message	Location

Connected file://C:\Program Fi ... HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml SP,ddsi2,durability,cmsoap,splicedaemon

Figure 42 Browser window

(The red boxes in the illustration indicate the current Open Connection.)

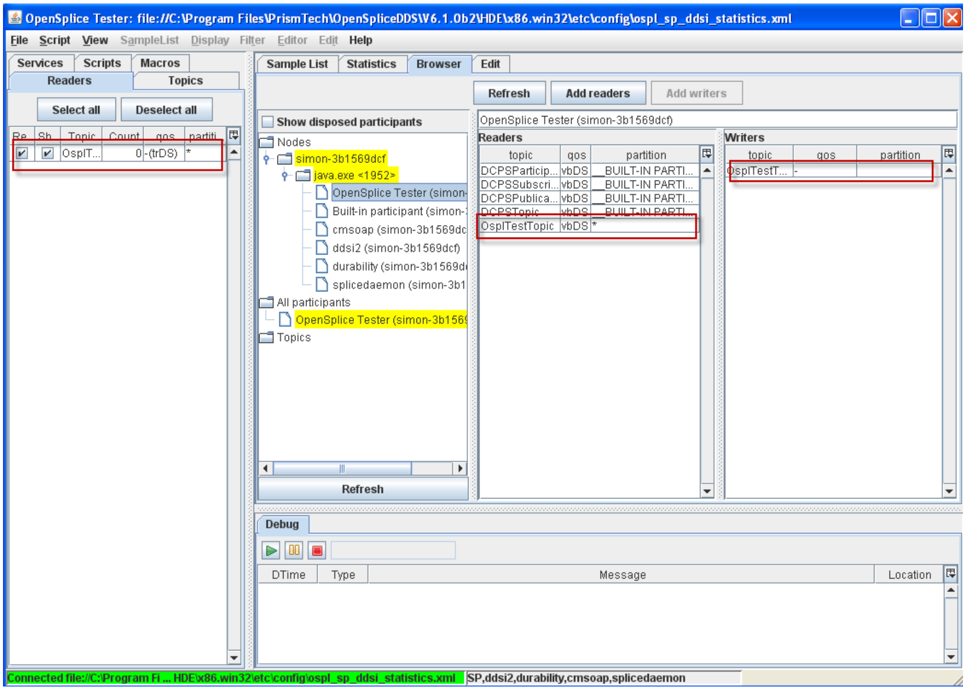
3.7.2 Readers and Writers tables are updated when a new Reader is created

Step 1: Open the Browser window.

Step 2: Select OpenSplice Tester participant from the All participants tree.

Step 3: Create a new OsplTestTopic reader (see section 3.3.2, *To Add a Reader from the Topic list*, on page 23, for instructions).

Step 4: The Readers and Writers table will be updated.



OpenSplice Tester: file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Select all Deselect all
Sample List Statistics Browser Edit
Refresh Add readers Add writers
□ Show disposed participants
Nodes
simon-3b1569dcf
java.exe <1952>
OpenSplice Tester (simon-
Built-in participant (simon-
cmsoap (simon-3b1569dc
ddsi2 (simon-3b1569dcf)
durability (simon-3b1569d
splicedaemon (simon-3b1
All participants
OpenSplice Tester (simon-3b156
Topics
Refresh
Debug
DTime Type Message Location

Re	Sh	Topic	Count	qos	partiti
☑	☑	OsplT...	0	-(trDS)	*

OpenSplice Tester (simon-3b1569dcf)

Readers

topic	qos	partition
DCPSParticip...	vbDS	__BUILT-IN PARTI...
DCPSSubscri...	vbDS	__BUILT-IN PARTI...
DCPSPublica...	vbDS	__BUILT-IN PARTI...
DCPSTopic	vbDS	__BUILT-IN PARTI...
OsplTestTopic	vbDS	*

Writers

topic	qos	partition
OsplTestT...	-	

Connected file://C:\Program Fi ... HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml SP,ddsi2,durability,cmsoap,splicedaemon

Figure 43 Readers and Writers table updated

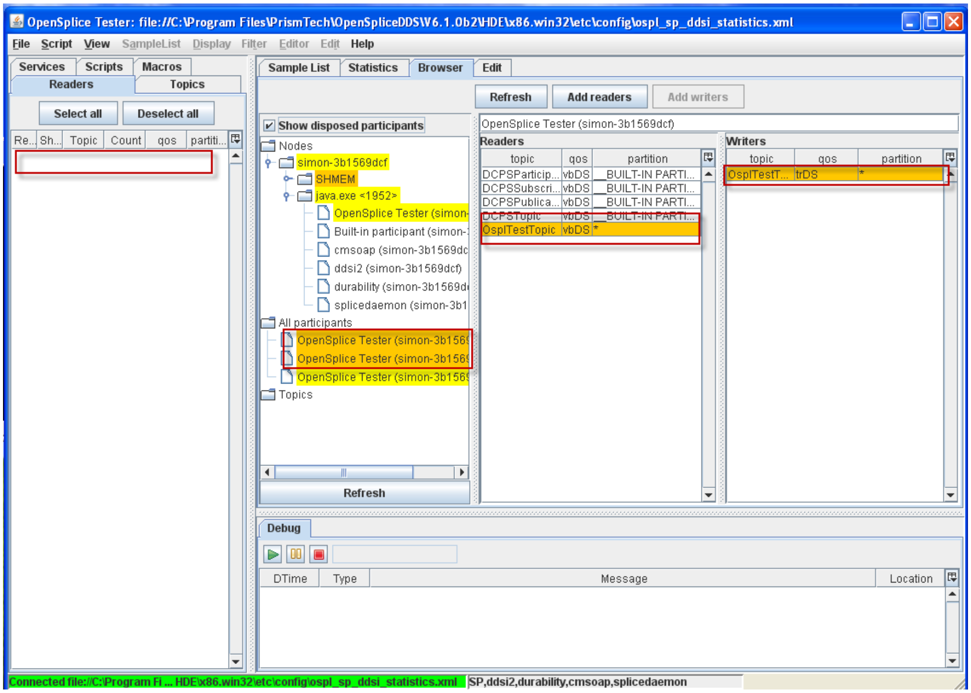
3.7.3 Readers and Writers tables are updated when a new Reader is deleted

Step 1: Open the Browser window.

Step 2: Select the OpenSplice Tester participant from the All participants tree.

Step 3: Delete the existing OsplTestTopic reader.

Step 4: The deleted reader will be highlighted with orange to indicate that the reader is disposed.


OpenSplice Tester: file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Select all Deselect all
Re.. Sh... Topic Count qos partiti...
Sample List Statistics Browser Edit
Refresh Add readers Add writers
Show disposed participants
Nodes
simon-3b1569dcf
SHMEM
java.exe <1952>
OpenSplice Tester (simon-
Built-in participant (simon-
cmsoap (simon-3b1569dc
ddsi2 (simon-3b1569dcf)
durability (simon-3b1569d
splicedaemon (simon-3b1
All participants
OpenSplice Tester (simon-3b156
OpenSplice Tester (simon-3b156
OpenSplice Tester (simon-3b156
Topics
Refresh
Debug
DTime Type Message Location

OpenSplice Tester (simon-3b1569dcf)

Readers

topic	qos	partition
DCPSParticip...	vbDS	__BUILT-IN PARTI...
DCPSSubscri...	vbDS	__BUILT-IN PARTI...
DCPSPublica...	vbDS	__BUILT-IN PARTI...
DCPSTopic	vbDS	__BUILT-IN PARTI...
OsplTestTopic	vbDS	*

Writers

topic	qos	partition
OsplTestT...	trDS	*

Connected file://C:/Program Fi ... HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml SP,ddsi2,durability,cmsoap,splicedaemon

Figure 44 Reader deleted

3.7.4 To Check Reader and Writer compatibility

Step 1: Choose **Create Reader** from the pop-up menu from OsplTestTopic.

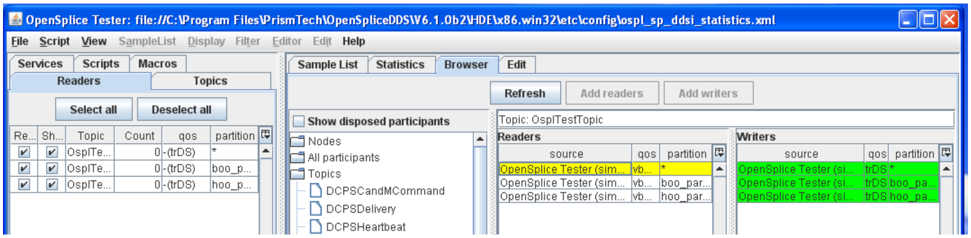
Step 2: Enter `boo` for the name and `boo_partition` for the partition.

Step 3: Create another reader with `hoo` for the name and `hoo_partition` for the partition.

Step 4: Choose **Create Default Reader** to create a default reader.

Step 5: Open the Browser window and select Topics/OsplTestTopic from the browser tree.

Step 6: Select a Reader with `*` partition from the Readers table.


OpenSplice Tester: file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Select all Deselect all
Sample List Statistics Browser Edit
Refresh Add readers Add writers
Show disposed participants
Nodes
All participants
Topics
DCPSCandMCommand
DCPSDelivery
DCPSHeartbeat

Re...	Sh...	Topic	Count	qos	partition
☑	☑	OsplTe...	0	-(trDS)	*
☑	☑	OsplTe...	0	-(trDS)	boo_p...
☑	☑	OsplTe...	0	-(trDS)	hoo_p...

Topic: OsplTestTopic

Readers

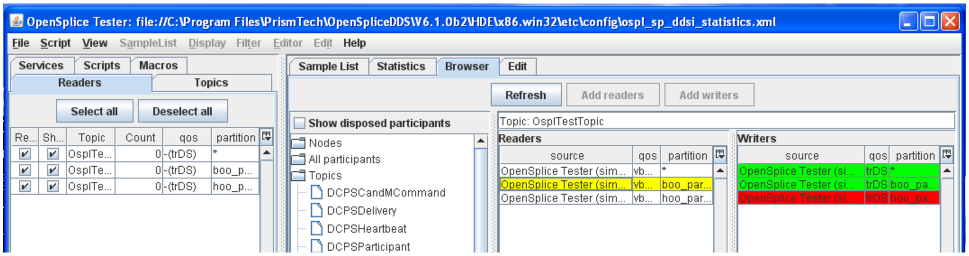
source	qos	partition
OpenSplice Tester (sim...	vb...	*
OpenSplice Tester (sim...	vb...	boo_par...
OpenSplice Tester (sim...	vb...	hoo_par...

Writers

source	qos	partition
OpenSplice Tester (si...	trDS	*
OpenSplice Tester (si...	trDS	boo_pa...
OpenSplice Tester (si...	trDS	hoo_pa...

Figure 45 Reader with ‘*’ partition selected

Step 7: Select a Reader with `boo` partition from the Readers table.


OpenSplice Tester: file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Select all Deselect all
Sample List Statistics Browser Edit
Refresh Add readers Add writers
Nodes
All participants
Topics
DCPSCandMCommand
DCPSDelivery
DCPSHeartbeat
DCPSParticipant

Re...	Sh...	Topic	Count	qos	partition
☑	☑	OsplTe...	0	-(trDS)	*
☑	☑	OsplTe...	0	-(trDS)	boo_p...
☑	☑	OsplTe...	0	-(trDS)	hoo_p...

□ Show disposed participants

Topic: OsplTestTopic

Readers

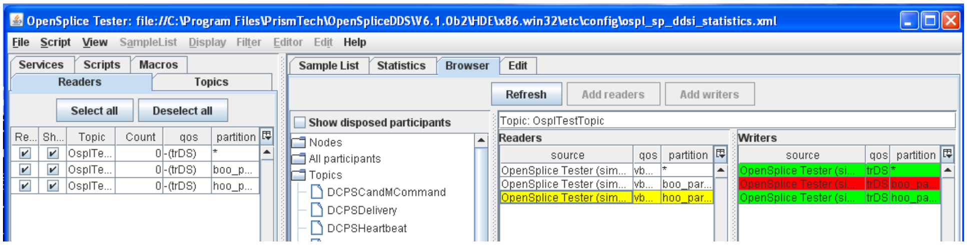
source	qos	partition
OpenSplice Tester (sim...	vb...	*
OpenSplice Tester (sim...	vb...	boo_par...
OpenSplice Tester (sim...	vb...	hoo_par...

Writers

source	qos	partition
OpenSplice Tester (si...	trDS	*
OpenSplice Tester (si...	trDS	boo_pa...
OpenSplice Tester (si...	trDS	hoo_pa...

Figure 46 Reader with ‘hoo’ partition selected

Step 8: Select a Reader with hoo partition from the Readers table.


OpenSplice Tester: file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml
File Script View SampleList Display Filter Editor Edit Help
Services Scripts Macros
Readers Topics
Select all Deselect all
Sample List Statistics Browser Edit
Refresh Add readers Add writers
Nodes
All participants
Topics
DCPSCandMCommand
DCPSDelivery
DCPSHeartbeat

Re...	Sh...	Topic	Count	qos	partition
☑	☑	OsplTe...	0	-(trDS)	*
☑	☑	OsplTe...	0	-(trDS)	boo_p...
☑	☑	OsplTe...	0	-(trDS)	hoo_p...

□ Show disposed participants

Topic: OsplTestTopic

Readers

source	qos	partition
OpenSplice Tester (sim...	vb...	*
OpenSplice Tester (sim...	vb...	boo_par...
OpenSplice Tester (sim...	vb...	hoo_par...

Writers

source	qos	partition
OpenSplice Tester (si...	trDS	*
OpenSplice Tester (si...	trDS	boo_pa...
OpenSplice Tester (si...	trDS	hoo_pa...

Figure 47 Reader with ‘boo’ partition selected

i In the Browser window, Readers/Writers are highlighted with red to indicate incompatibility with the selected Writer/Reader (yellow).

3.7.5 To Show Disposed Participants from the Browser tree

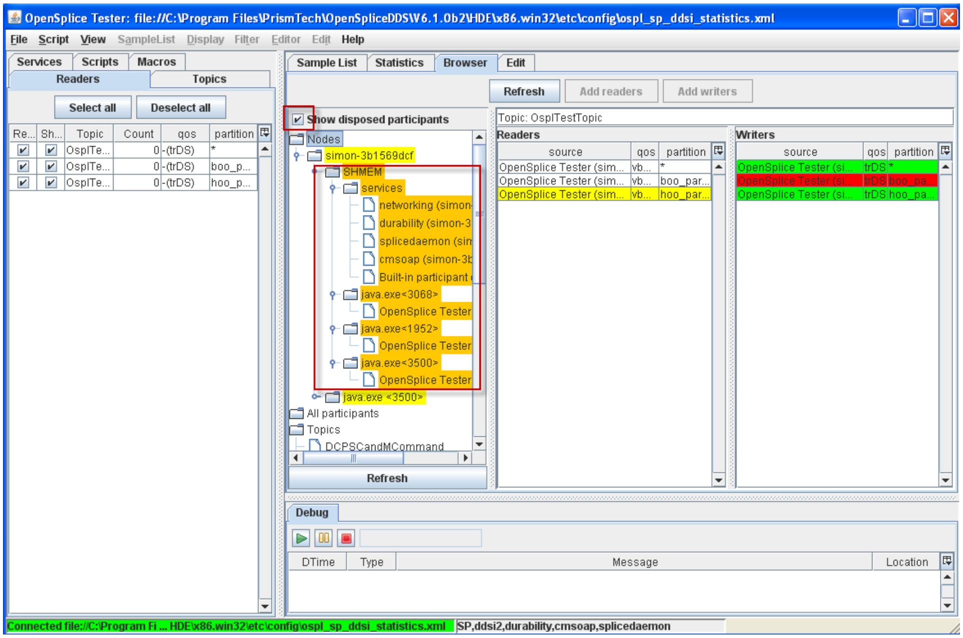
Step 1: Open the Browser window.

Step 2: Select (check) Show disposed participants.

Step 3: Expand the Nodes tree.

Step 4: Expand the All participants tree.

Step 5: De-select (un-check) Show disposed participants.



OpenSplice Tester: file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml

Re...	Sh...	Topic	Count	qos	partition
☑	☑	OsplTe...	0	-(trDS)	*
☑	☑	OsplTe...	0	-(trDS)	boo_p...
☑	☑	OsplTe...	0	-(trDS)	hoo_p...

☑ Show disposed participants

```
Nodes
  simon-3b1569dcf
    SHMEM
      services
        networking (simon-
        durability (simon-3
        splicedaemon (sim
        cmsoap (simon-3b
        Built-in participant
      java.exe<3068>
        OpenSplice Tester
      java.exe<1952>
        OpenSplice Tester
      java.exe<3500>
        OpenSplice Tester
    java.exe <3500>
All participants
Topics
  DCPSCandMCommand
```

Topic: OsplTestTopic

Readers

source	qos	partition
OpenSplice Tester (sim...	vb...	*
OpenSplice Tester (sim...	vb...	boo_par...
OpenSplice Tester (sim...	vb...	hoo_par...

Writers

source	qos	partition
OpenSplice Tester (si...	trDS	*
OpenSplice Tester (si...	trDS	boo_pa...
OpenSplice Tester (si...	trDS	hoo_pa...

DTime	Type	Message	Location

Connected file://C:\Program Fi ... HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml SP,ddsi2,durability,cmsoap,splicedaemon

Figure 48 Disposed participants

3.7.6 To Spawn a Tuner from the System Browser

Any domain participant that is part of a configuration that includes a SOAP service should have the **Start Tuner** pop-up menu.

Step 1: Connect Tester using the `ospl_sp_ddsi_statistics.xml` configuration file in the `etc/config` directory. (See section 3.2.1, *To Connect to a local OpenSplice instance*, on page 22.)

Step 2: Open the Browser window.

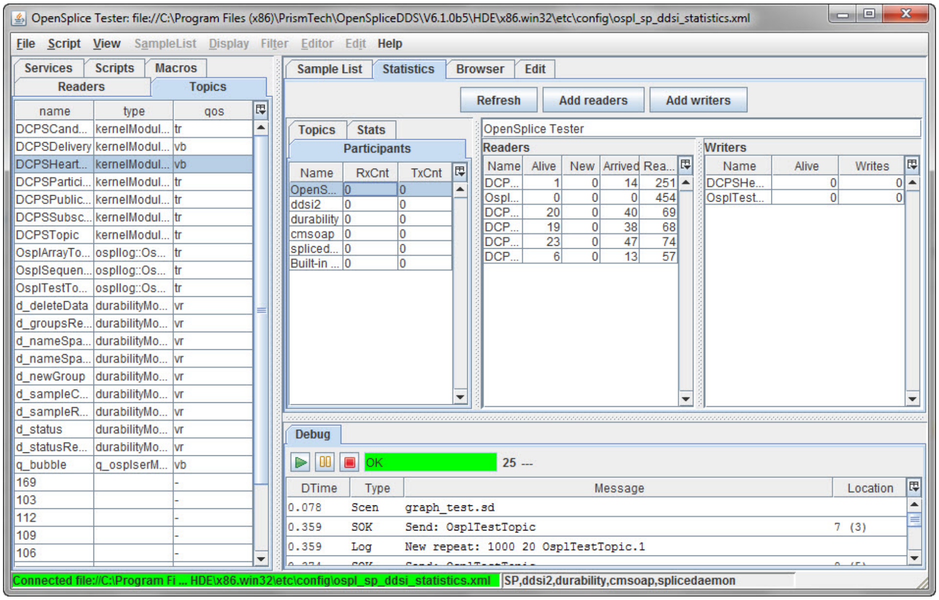
Step 3: Expand the Nodes tree.

Step 4: Right-click on the OpenSplice Tester participant.

Step 5: Choose **Start Tuner** from the pop-up menu.

3.7.7 Statistics

First, connect Tester using the `ospl_sp_ddsi_statistics.xml` configuration file in the `etc/config` directory. (See section 3.2.1, *To Connect to a local OpenSplice instance*, on page 22.) (Note that statistics can only be gathered from the Tester process.)



OpenSplice Tester: file://C:\Program Files (x86)\PrismTech\OpenSpliceDDS\V6.1.0b5\HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml

name	type	qos
DCPSCand...	kernelModul...	tr
DCPSDelivery	kernelModul...	vb
DCPSHeart...	kernelModul...	vb
DCPSPartici...	kernelModul...	tr
DCPSPublic...	kernelModul...	tr
DCPSSubsc...	kernelModul...	tr
DCPSTopic	kernelModul...	tr
OsplArrayTo...	ospllog::Os...	tr
OsplSequen...	ospllog::Os...	tr
OsplTestTo...	ospllog::Os...	tr
d_deleteData	durabilityMo...	vr
d_groupsRe...	durabilityMo...	vr
d_nameSpa...	durabilityMo...	vr
d_nameSpa...	durabilityMo...	vr
d_newGroup	durabilityMo...	vr
d_sampleC...	durabilityMo...	vr
d_sampleR...	durabilityMo...	vr
d_status	durabilityMo...	vr
d_statusRe...	durabilityMo...	vr
q_bubble	q_osplserM...	vb
169		-
103		-
112		-
109		-
106		-

Name	RxCnt	TxCnt
OpenS...	0	0
ddsi2	0	0
durability	0	0
cmsoap	0	0
spliced...	0	0
Built-in ...	0	0

OpenSplice Tester

Readers

Name	Alive	New	Arrived	Rea...
DCP...	1	0	14	251
Ospl...	0	0	0	454
DCP...	20	0	40	69
DCP...	19	0	38	68
DCP...	23	0	47	74
DCP...	6	0	13	57

Writers

Name	Alive	Writes
DCPSHe...	0	0
OsplTest...	0	0

DTime	Type	Message	Location
0.078	Scen	graph_test.sd	
0.359	SOK	Send: OsplTestTopic	7 (3)
0.359	Log	New repeat: 1000 20 OsplTestTopic.1	

Connected file://C:\Program Fi ... HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml SP,ddsi2,durability,cmsoap,splicedaemon

Figure 49 The Statistics window

3.7.7.1 Statistics - participants

3.7.7.1.1 Write sample topics and check statistics window content

Step 1: Create a default reader for OsplTestTopic.

Step 2: Write four samples.

Step 3: Open the Participant tab of the Statistics window.

Step 4: Select the OpenSplice Tester participant from the list.

3.7.7.2 Statistics - topics

3.7.7.2.1 Write sample topics and check statistics window content

Step 1: Create a default reader for OsplTestTopic.

Step 2: Write four samples.

Step 3: Open the Topics tab of the Statistics window.

Step 4: Select OsplTestTopic from the list.

3.8 Scripting

3.8.1 To Create a New Scenario

Note that you can only have one scenario open at a time. To avoid losing changes in the current scenario you must save it (see section *3.8.4* below) before creating a new scenario or selecting a different one from the drop-down list of recently-used scenarios (next to the Clear and Execute button).

Step 1: Choose **Editor > New Scenario** to create a new scenario and open it in the editor, *or* if the Editor window is already open, press CTRL+N to create and open a new scenario. A warning is displayed if there are unsaved changes in the current scenario.

Step 2: In the File Save dialog that appears, specify the location of the new scenario and give it a name.

3.8.2 To Create a New Macro

Step 1: Choose **Editor > New Macro** to create a new macro and open it in the editor, *or* if the Editor window is already open, press CTRL+M to create and open a new macro. You can have multiple macros open at the same time. Use the drop-down list next to the Clear and Execute button to see or select them.

Step 2: In the File Save dialog that appears, specify the location of the new macro and give it a name.

3.8.3 To Edit an Existing Scenario or Macro

Step 1: Choose **Editor > Open** from the top menu.

Step 2: In the dialog that appears, type in or browse to the location of the macro or scenario you wish to open, then click Open.

3.8.4 To Save an open Scenario or Macro

Save the current scenario or macro.

Step 1: Choose **File > Save** from the top menu *or* press CTRL+S.

Step 2: If the scenario or macro has been saved before, then it is immediately saved, over-writing the previous version.

Step 3: If the scenario or macro has *not* been saved before, a Save As... dialog appears; type in or browse to an appropriate location and enter a name for the scenario or macro, the click Save.

3.8.5 To Complete and Compile a Scenario

This function ‘wraps’ the current text in the Edit window with ‘scenario’ and ‘end scenario’.

Complete is only used when a new scenario is created without a template, from **DiffScript** or the Write button in a sample editor. **Compile** is only needed when you do not want to execute, but just check the syntax.

Step 1: Choose **Edit > Complete** from the top menu.

Step 2: Click the Compile button.

Step 3: Click the Execute button.

Step 4: Click the Clear and Execute button.

3.8.6 Script selection

Step 1: 1. Expand the Script Selection drop-down list of recently-used scripts near the Clear and Execute button.

3.8.7 Code completion

The Tester has a 'code completion' function which reduces the amount of typing that you have to do reduces the chances of errors. For example, you can press the CTRL+SPACE keys after you have typed the first few characters of a reader name and the Tester will display a list of the names of the readers which start with the same characters and you can choose the one you want.

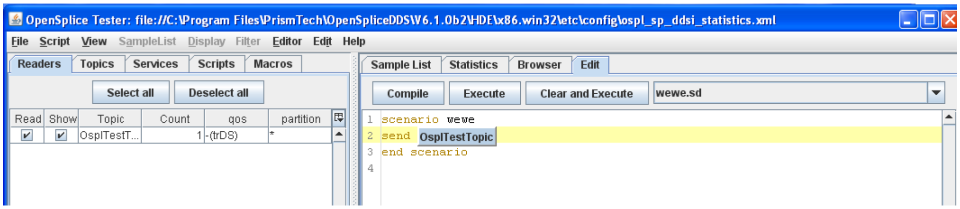
Assuming that the `OsplTestTopic` reader already exists, and that a new script is open in the Edit tab:

Step 1: Complete the current scenario by choosing **Editor > Complete** from the top menu. (Note that it is generally preferable to start from a template.)

Step 2: Type '`send Ospl`' then press CTRL+SPACE.

Step 3: 'OsplTestTopic' appears; press ENTER to accept it, and the instruction is completed.

This also pops up the sample editor, enabling you to set the arguments. The sample editor can also be activated by CTRL+SPACE when the cursor is in the instruction, or CTRL+LEFT-CLICK on the instruction.



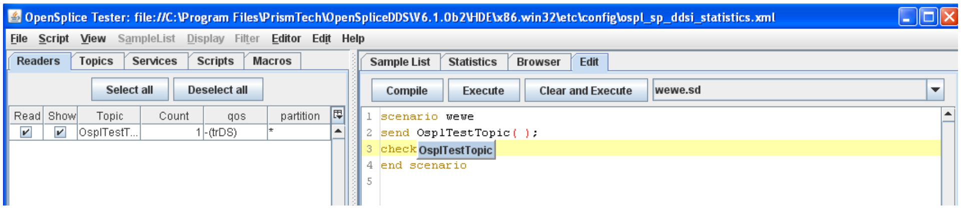
Read	Show	Topic	Count	qos	partition
☑	☑	OsplTestT...	1	-(trDS)	*

```
scenario wewe
send OsplTestTopic
end scenario
```

Figure 50 Code completion (send)

Step 4: Type '`check Ospl`' then press CTRL+SPACE.

Step 5: 'OsplTestTopic' appears; press ENTER to accept it, and the instruction is completed.



Read	Show	Topic	Count	qos	partition
☑	☑	OsplTestT...	1	-(trDS)	*

```
scenario wewe
send OsplTestTopic( );
check OsplTestTopic
end scenario
```

Figure 51 Code completion (check)

3.9 Execute and Debug

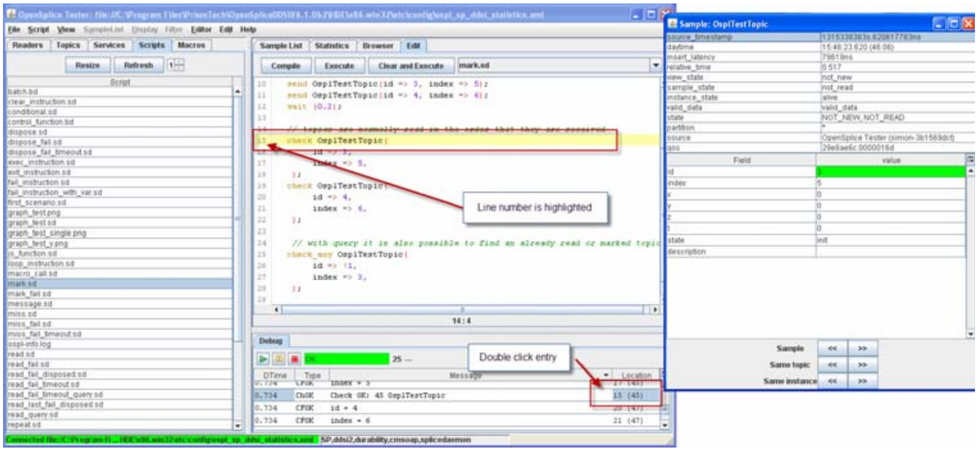
3.9.1 To Run the Current Script

Step 1: Click the Execute (‘Play’ ▶) button in the Debug window to run the current script.

Step 2: While the script is still executing, click the ‘Pause’ ⏸ button in the Debug window.

Step 3: While the script is still executing, click the ‘Stop’ ■ button in the Debug window.

Step 4: In the Debug window, double-click the entry where the column Location has a value of 6. Double-clicking on an entry in the Debug window highlights the relevant line in the Editor window.


OpenSplice Tester: file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml
File
Script
View
SampleList
Display
Filter
Editor
Edit
Help
Readers
Topics
Services
Scripts
Macros
Resize
Refresh
Sample List
Statistics
Browser
Edit
Compile
Execute
Clear and Execute
mark.sd
14 : 4
Line number is highlighted
Debug
OK
25 ---
Double click entry
Connected file://C:\Program Fi....HDE\x86.win32\etc\config\ospl_sp_ddsi_statistics.xml
SP,ddsi2,durability,cmsoap,splicedaemon
Sample: OsplTestTopic
Sample
Same topic
Same instance
<<
>>

Script
batch.bd
clear_instruction.sd
conditional.sd
control_function.bd
dispose.sd
dispose_fail.sd
dispose_fail_timeout.sd
exec_instruction.sd
exit_instruction.sd
fail_instruction.sd
fail_instruction_with_var.sd
first_scenario.sd
graph_test.png
graph_test.sd
graph_test_single.png
graph_test_y.png
js_function.sd
loop_instruction.sd
macro_call.sd
mark.sd
mark_fail.sd
message.sd
miss.sd
miss_fail.sd
miss_fail_timeout.sd
ospl-info.log
read.sd
read_fail.sd
read_fail_disposed.sd
read_fail_timeout.sd
read_fail_timeout_query.sd
read_last_fail_disposed.sd
read_query.sd
repeat.sd

```
  send OsplTestTopic(id => 3, index => 5);
  send OsplTestTopic(id => 4, index => 6);
  wait (0.2);

  // topics are normally read in the order that they are received
  check OsplTestTopic(
      id => 3,
      index => 5,
   );
  check OsplTestTopic(
      id => 4,
      index => 6,
   );

   // with query it is also possible to find an already read or marked topic
  check_any OsplTestTopic(
      id => !1,
      index => 3,
   );
```

DTime	Type	Message	Location
0.734	CFOK	index = 5	17 (45)
0.734	ChOK	Check OK: 45 OsplTestTopic	15 (45)
0.734	CFOK	id = 4	20 (47)
0.734	CFOK	index = 6	21 (47)

source_timestamp	1315338383s.820817783ns
daytime	15:46:23.620 (46.06)
insert_latency	79619ns
relative_time	0.517
view_state	not_new
sample_state	not_read
instance_state	alive
valid_data	valid_data
state	NOT_NEW_NOT_READ
partition	*
source	OpenSplice Tester (simon-3b1569dcf)
qos	29e8aefc.0000016d

Field	value
id	3
index	5
x	0
y	0
z	0
t	0
state	init
description	

Figure 52 Debugging a script

3.9.2 Batch execution (Batch window)

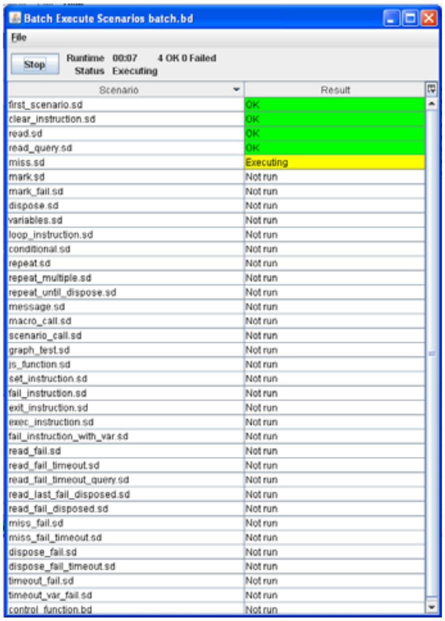
Load and run batch scenario.

Step 1: Choose **Script > Batch** from the top menu.

Step 2: In the Batch window, choose **File > Load batch**.

Step 3: Select `batch.bd` in the example script directory.

Step 4: Click the Start button.



Runtime 00:07 4 OK 0 Failed
Status Executing

Scenario	Result
first_scenario.sd	OK
clear_instruction.sd	OK
read.sd	OK
read_query.sd	OK
miss.sd	Executing
mark.sd	Not run
mark_fail.sd	Not run
dispose.sd	Not run
variables.sd	Not run
loop_instruction.sd	Not run
conditional.sd	Not run
repeat.sd	Not run
repeat_multiple.sd	Not run
repeat_until_dispose.sd	Not run
message.sd	Not run
macro_call.sd	Not run
scenario_call.sd	Not run
graph_test.sd	Not run
js_function.sd	Not run
set_instruction.sd	Not run
fail_instruction.sd	Not run
exit_instruction.sd	Not run
exec_instruction.sd	Not run
fail_instruction_with_var.sd	Not run
read_fail.sd	Not run
read_fail_timeout.sd	Not run
read_fail_timeout_query.sd	Not run
read_last_fail_disposed.sd	Not run
read_fail_disposed.sd	Not run
miss_fail.sd	Not run
miss_fail_timeout.sd	Not run
dispose_fail.sd	Not run
dispose_fail_timeout.sd	Not run
timeout_fail.sd	Not run
timeout_var_fail.sd	Not run
control_function.bd	Not run

Figure 53 Batch execution

3.9.3 To Run a Batch Script from the Command Line

Step 1: Change directory to the example scripts directory where the `batch.bd` is found (`<OSPL_HOME>/examples/` ...).

Step 2: Run `ospltest -e -b batch.bd`.

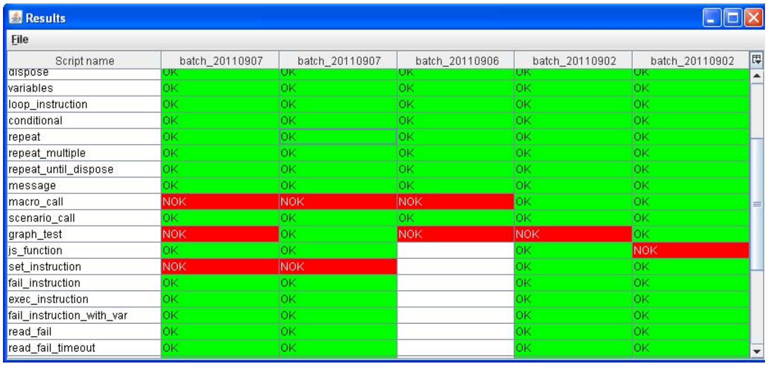
3.9.4 Batch results

3.9.4.1 Load batch result

Step 1: Choose **Scripts > Batch results** from the top menu.

Step 2: With the Batch results window open, choose **File > Load result** from the top menu.

Step 3: Select the batch result file from the batch run.



Script name	batch_20110907	batch_20110907	batch_20110906	batch_20110902	batch_20110902
dispose	OK	OK	OK	OK	OK
variables	OK	OK	OK	OK	OK
loop_instruction	OK	OK	OK	OK	OK
conditional	OK	OK	OK	OK	OK
repeat	OK	OK	OK	OK	OK
repeat_multiple	OK	OK	OK	OK	OK
repeat_until_dispose	OK	OK	OK	OK	OK
message	OK	OK	OK	OK	OK
macro_call	NOK	NOK	NOK	OK	OK
scenario_call	OK	OK	OK	OK	OK
graph_test	NOK	OK	NOK	NOK	OK
js_function	OK	OK		OK	NOK
set_instruction	NOK	NOK		OK	OK
fail_instruction	OK	OK		OK	OK
exec_instruction	OK	OK		OK	OK
fail_instruction_with_var	OK	OK		OK	OK
read_fail	OK	OK		OK	OK
read_fail_timeout	OK	OK		OK	OK

Figure 54 Batch results

3.9.4.2 Scan regression folder for batch results

Step 1: Choose **File > Scan Regression** from the top menu.

Step 2: Double-click the test result column of any test.

The results displayed will appear similar to the example in *Figure 54*.

3.9.4.3 Scan regression for specified directory

Step 1: Choose **File > Scan Regression dir** from the top menu.

Step 2: Select the directory (folder) that contains batch results.

The results displayed will appear similar to the example in *Figure 54*.

3.10 Adding virtual fields

Virtual fields are fields with calculated values. For example, a translation from radians to degrees, or from cartesian to polar coordinates. The virtual field can be provided in Java (inside a plugin, see section 3.11, *Plugins*, on page 48) or a script language (see Chapter 5, *Scripting*, on page 71, and the following example).

3.10.1 Add virtual fields to the topic

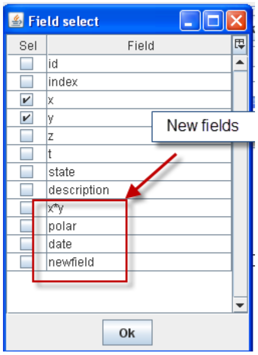
Step 1: Choose **File > Add fields** from the top menu.

Step 2: Browse to the example directory and select `fields.txt`.

Step 3: Open the SampleList window.

Step 4: Select the OsplTestTopic sample.

Step 5: Add extra fields from the pop-up menu.



Field select

Sel	Field
□	id
□	index
☑	x
☑	y
□	z
□	t
□	state
□	description
□	x*y
□	polar
□	date
□	newfield

Figure 55 Adding extra fields to a sample

3.11 Plugins

Plugins can extend the functionality of Tester by providing virtual fields (see section 3.10, *Adding virtual fields*, on page 47), or additional interfaces. Plugins are automatically loaded upon startup from the specified plugin directory. Two sample plugins are provided with Tester: `SimplePlugin` adds virtual fields, and `TestInterface` adds a UDP/IP message interface (see Chapter 6, *Message Interfaces*, on page 85).

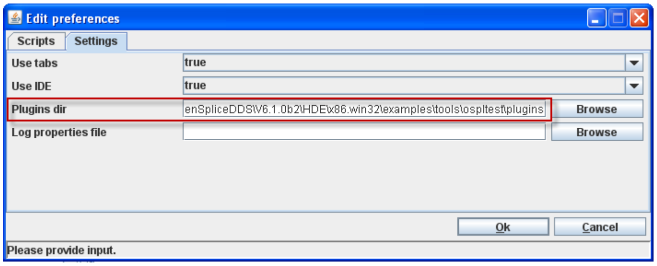
3.11.1 Install / Uninstall plugins

Step 1: Go to the `examples/tools/ospltest/SimplePlugin` directory.

Step 2: Run `ant` from the command console to build the `SimplePlugin` example.

Step 3: Run Tester and choose **File > Preference** from the top menu.

Step 4: In the Settings tab set the correct value for Plugins dir and click OK.



Edit preferences

Settings

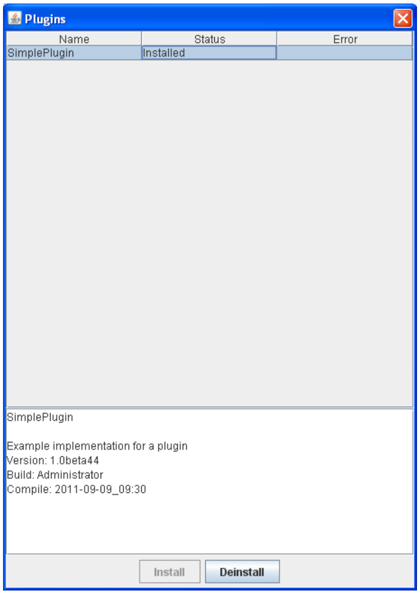
Use tabs	true
Use IDE	true
Plugins dir	enSpliceDDS\V6.1.0b2\HDE\x86.win32\examples\tools\ospltest\plugins
Log properties file	

Please provide input.

Figure 56 Setting the path to the Plugins directory

Step 5: Choose **File > Plugins** from the top menu.

Step 6: Click SimplePlugin to select it.



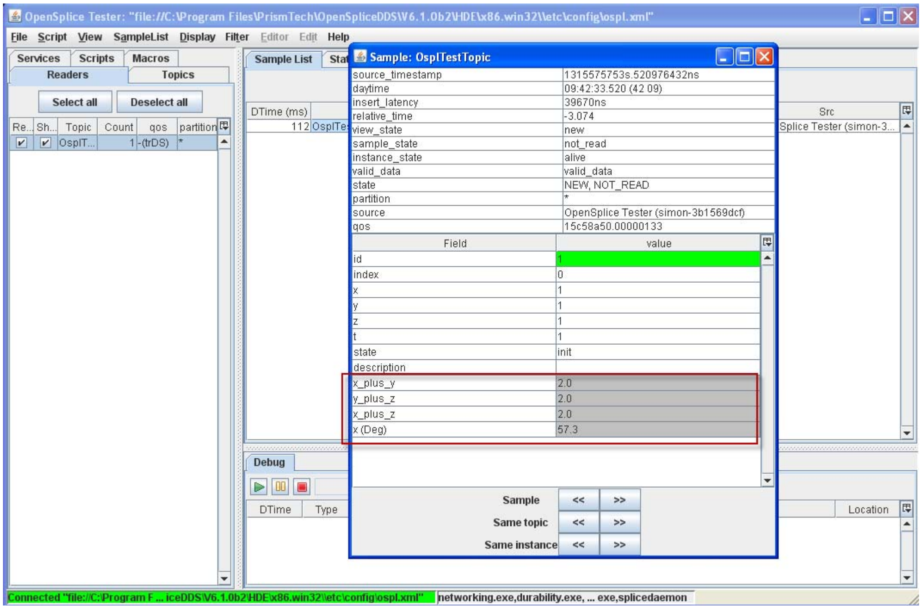
Name	Status	Error
SimplePlugin	Installed	

SimplePlugin

Example implementation for a plugin
Version: 1.0beta44
Build: Administrator
Compile: 2011-09-09_09:30

Figure 57 The SimplePlugin example

Step 7: Double-click any OsplTestTopic data in the SampleList window. New fields are added.



Re...	Sh...	Topic	Count	qos	partition
☑	☑	OsplT...	1	-(trDS)	*

DTime (ms)		Src
112	OsplTes	Splice Tester (simon-3...

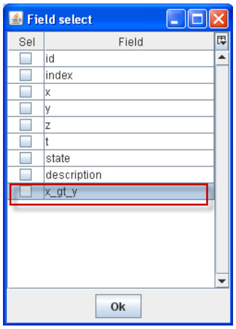
source_timestamp	1315575753s.520976432ns
daytime	09:42:33.520 (42 09)
insert_latency	39670ns
relative_time	-3.074
view_state	new
sample_state	not_read
instance_state	alive
valid_data	valid_data
state	NEW, NOT_READ
partition	*
source	OpenSplice Tester (simon-3b1569dcf)
qos	15c58a50.00000133

Field	value
id	1
index	0
x	1
y	1
z	1
t	1
state	init
description	
x_plus_y	2.0
y_plus_z	2.0
x_plus_z	2.0
x (Deg)	57.3

Connected "file://C:\Program F ... iceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl.xml" networking.exe,durability.exe, ... exe,splicedaemon

Figure 58 Fields added to OsplTestTopic sample

Step 8: In the Select extra field dialog (F9), one more field is added.



Field select

Sel	Field
☐	id
☐	index
☐	x
☐	y
☐	z
☐	t
☐	state
☐	description
☐	x_gt_y

Figure 59 Extra field added

3.12 More on Virtual fields

Additional virtual fields can be provided *via* a plugin or *via* a script.

3.12.1 Adding Virtual Fields *via* plugin

Override the class:

```
ExtraTopicField
```

Compile this class in a plugin and in the ‘install’ function register the extra fields with:

```
connection.registerExtraField(<instance of extra topic
field class>);
```

An example of a plugin with an extra field is provided in `examples`:

```
<OSPL_HOME>/examples/tools/ospltest/plugins/SimplePlugin
```

3.12.2 Adding Virtual Fields *via* script

A script file can be loaded using the top menu: **File > Add Fields**.

The script file has the following syntax:

```
[#!<language>]
<name of the field>
<name of the applicable topic>
<script which returns a value and can have multiple lines>
next_field
<name of the field>
<name of the applicable topic>
<script which returns a value and can have multiple lines>
```

The language description is optional. 1 to *n* fields can be described in a single file.

The data of the sample is available in an object variable which is pushed to the script engine before the execution of the script. The object sample provides the following functions:

```
String getDayTime();
long getTime();
long getId();
String getMsgName();
String getKey();
String getInstanceState();
boolean isALive();
String getSource();
String getFieldValue(String fieldname);
```

These functions can be used to retrieve data from the current sample and determine the value for the extra field. An example of a script file is provided in `examples/tools/ospltest/fields.txt`.

CHAPTER

4 Command Reference

This chapter lists all of the Tester's commands and describes their operation.

4.1 Introduction

The commands are described below in the order in which they appear in the menus (starting at the top left).

Where a menu option also has a keyboard shortcut, it is given in Small Capitals. Some menu options can also be invoked by Buttons in appropriate tabs or windows.

4.2 Menus

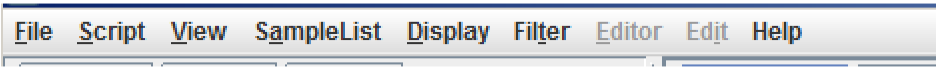


Figure 60 Tester main menu

4.2.1 File

File > Connect, Ctrl+Shift+C

Open a connection to a Domain.

File > Disconnect, Ctrl+Shift+D

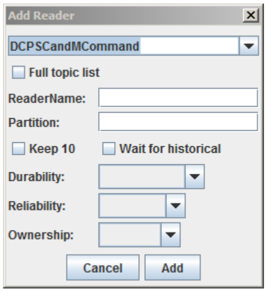
Disconnect from a Domain.

File > Remove All Readers

Remove all previously-added Readers.

File > Add Reader

Add a single topic Reader.



DCPSCandMCommand

□ Full topic list

ReaderName:

Partition:

□ Keep 10 □ Wait for historical

Durability:

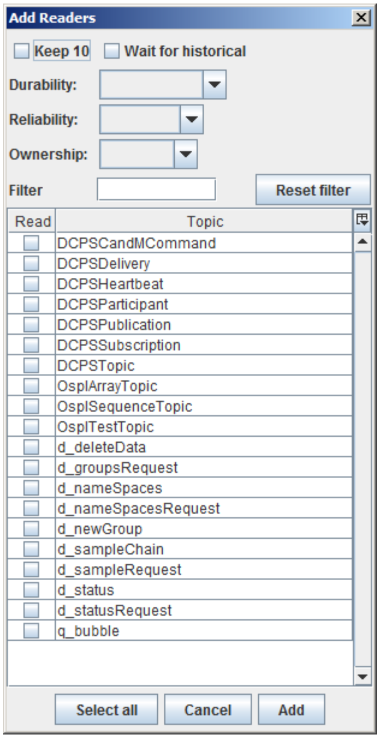
Reliability:

Ownership:

Figure 61 Add Reader dialog

File > Add Readers

Add multiple Readers by selecting from the Topic List.



□ Keep 10 □ Wait for historical

Durability:

Reliability:

Ownership:

Filter

Read	Topic
□	DCPSCandMCommand
□	DCPSDelivery
□	DCPSHeartbeat
□	DCPSParticipant
□	DCPSPublication
□	DCPSSubscription
□	DCPSTopic
□	OsplArrayTopic
□	OsplSequenceTopic
□	OsplTestTopic
□	d_deleteData
□	d_groupsRequest
□	d_nameSpaces
□	d_nameSpacesRequest
□	d_newGroup
□	d_sampleChain
□	d_sampleRequest
□	d_status
□	d_statusRequest
□	q_bubble

Figure 62 Add Readers from Topic list

File > Save Readers List

Save the current list of topics to a file. The keys, QoS, wait for historical info will be preserved.

The format of the readers list file (and the add reader specification) is:

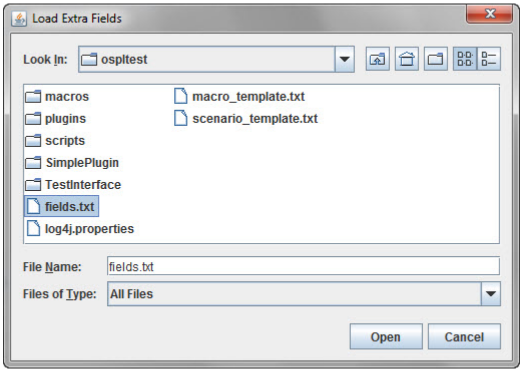
```
<!>[#QOS#]topic_name[|readername][\[partitionname\]]
<optional_key> <optional_foreign_key1> <optional_foreign_key2>
<optional_foreign_key3>
```

File > Load Readers List

Load a topics file. Topics already in the list will not be recreated.

File > Add Fields

Load new fields. Example `field.txt` is located in the `example` directory.



Look In: ospltest

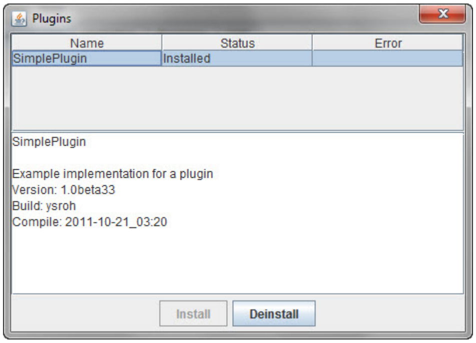
File Name: fields.txt

Files of Type: All Files

Figure 63 Load Extra Fields dialog

File > Plugins

Install/Uninstall Plugins. The example `SimplePlugin` plugin is located in the `example` directory. It must be compiled and put in to the `plugins` directory specified in Preference page.



Name	Status	Error
SimplePlugin	Installed	

SimplePlugin

Example implementation for a plugin
Version: 1.0beta33
Build: ysroh
Compile: 2011-10-21_03:20

Figure 64 Plugins dialog

File > Save Layout

Save the current layout of the windows in a file, this can later be used to organize the windows in the same way. Save Layout is only applicable to non-IDE mode.

File > Load Layout

Load a specific layout of the windows as previously saved (select by file on the disk) with Save Layout. Load Layout is only applicable to non-IDE mode.

File > Preferences

Can be used to change the locations of the `macros` and `scripts` directories. (See also section 2.1, *Starting and Stopping Tester*, on page 7).

File > Exit

Quit the application.

4.2.2 Script

Script > Script Editor, ALT+SHIFT+S

Open the script Edit window.

Script > Debug Window

Open the script Debug window.

Script > Scripts

Open the scripts window which allows for quick access to scripts found on the script path (as defined in the `ospltest.properties`). (See also *File > Preferences Can be used to change the locations of the `macros` and `scripts` directories. (See also section 2.1, Starting and Stopping Tester, on page 7).* above and section 2.1, *Starting and Stopping Tester*, on page 7).

Script > Macros

Open the macros window which allows for quick access to the macros found in the macro path (as defined in the `ospltest.properties`). (See also *File > Preferences Can be used to change the locations of the `macros` and `scripts` directories. (See also section 2.1, Starting and Stopping Tester, on page 7).* above and section 2.1, *Starting and Stopping Tester*, on page 7).

Script > Batch, ALT+SHIFT+B

Open the Batch Execute window for the batch execution of several scripts

Script > Batch Results

Display the results of the batch run.

4.2.3 View

View > Samples, ALT+1

Open the Sample List window.

View > Statistics, ALT+2

Open the Statistics window.

View > Browser, ALT+3

Open the Browser window.

4.2.4 SampleList

The Sample List displays the current list of read samples. The list is sorted on source time (timestamp) of the topic samples. Topics Samples are only displayed when the Show checkbox in the Reader list is checked (note that un-checking Show does not delete the topics Samples). A double-click in the list results in the topic being displayed in the Sample window.

The state displayed with the topic is the Sample state of the sample. When the state of the topic is `alive` then if this is the last Sample with that key it is displayed as `ALIVE_AND_KICKING` for received samples and `ALIVE_AND_SEND` for samples sent by Tester. This makes it very easy to spot topics which are not disposed.

When exactly two topics are selected the difference between the source timestamps is displayed.

The following menus are only active when the Sample List tab is selected showing samples. (If you are in the Browser tab, for example, then the menus will not be active (they will be 'greyed out')).

SampleList > Clear, Clear button
Clear the Sample List.

SampleList > Dump
Dump the contents of the current (filtered) Sample List to a file.

SampleList > Dump Selection, P (*also* CTRL+P *and* ALT+P)
Write the current selection content to a file.

SampleList > Dump to CSV
Write the contents in CSV format.

SampleList > Dispose Alive
Dispose all topics in the Sample list with a state `alive and kicking` (*i.e.* all last Samples of a topic with a given key which are still alive), this function can be used to clean up (dispose left alive samples) a list after a test.

SampleList > Diff Script
Create a list of instructions in the current scenario which reproduces the list of samples in the Sample list. The diff means that only fields which do not have the default value or are a key/switch field are used in the script.

SampleList > Diff Script Selection
Create a diff script for the current selection of samples.

SampleList > Show Chart, ALT+SHIFT+C
Display the chart window. To fill the chart with data select a column with numeric values and press Y. This will add a trace with the values of the column, using the time received on the X axis. Multiple traces can be added. Select a filter to limit to the appropriate values. To display a scatter plot, clear the traces

and select the column to use on the x-axis, then press X. After this select the column with values for the Y axis and press Y. It is also possible to automatically create multiple traces based on a key value. First select the column to be used as key and press K before the Y column is selected.

F2

Compare two topic Samples. Select the first topic Sample in the Sample window (by double-clicking), then select the second topic Sample and press F2. The samples will be displayed side by side with the differences marked in the window of the second topic (normally the left window). A field marked in red is different, a field marked in orange was not found in the first topic Sample. If not different then (foreign) key fields will be marked in green and yellow. (See also section 2.3.5.6, *Topic Instance Window*, on page 19.)

F3

Display a topic Sample in a separate Sample window.

F4

Open the topic edit window with the values of the selected topic.

F9

Fields of the current selected topic sample can be added for display in the Sample list. Fields are displayed based on name. Any topic Sample with a field of that name will provide the value of the field. A field column can be deleted by selecting a cell in the column and then pressing delete.

4.2.5 Display

When the Sample List is open these commands allow the user to adjust the window display attributes to their needs.

Display > Font Smaller, CTRL+MINUS

Decreases the font size of the Sample List window.

Display > Font Larger, CTRL+PLUS

Increases the font size of the Sample List window.

Display > Day Time

Toggles the Dtime column format between number of milliseconds (ms) and time-of-day (hh:mm:ss.ms).

Display > Colors

Toggles the display of colors (on or off).

Display > Refresh

Refreshes the Sample List window.

Display > Only Show Alive

Filters the samples to display samples in the ‘alive’ state.

4.2.6 Filter

When the Sample List is open these commands enable you to filter the displayed samples based on the Topic and Key attributes of the current sample.

The filter can also be applied by typing the key directly in the filter window. Add a + (plus) sign in front of the key value to filter including foreign key relations (it is not possible to filter on key and topic name when entering the key manually). The filter can also be reset by clicking the Reset button.

Filter > Topic, CTRL+F5
: Filter on topic name.

Filter > Topic and Key, F5
: Filter on key and topic name.

Filter > Key, F6
: Filter on key only (so all topics with the same value for key are displayed).

Filter > Resets, F7
: Clears the filter.

F8
: Filter on the key value and also allow forward foreign key relations (*i.e.* find topics which have a key which matches a foreign key of an already displayed topic.

F12
: Filter all messages with the same sample state.

F
: Filter based on text in a column, the column is listed in the filter box (*i.e.* `[<column>]`) add the text on which to filter and then press ENTER.

4.2.7 Editor

When the Edit window is open these commands allow the user to create and manage Scenarios and Macros.

Editor > New Scenario, CTRL+N
: Create a new scenario. A File Save dialog will be displayed to provide the filename of the scenario. The initial scenario will be created using the template `scenario_template.txt` which is found in the installation directory.

Editor > New Macro, CTRL+M
: Create a new macro. A File Save dialog will be displayed to provide the filename of the macro. The initial macro will be created using the template `macro_template.txt` which is found in the installation directory.

Editor > Open, CTRL+O

Opens the File Open dialog, the selected Script or Macro file will be loaded in the editor.

Editor > Save, CTRL+S

Save the current script to disk (to the same file as it was loaded/created).

Editor > Save As, CTRL+SHIFT+S

Opens the Save dialog for entering a filename to which the current script will be saved.

Editor > Complete, CTRL+SHIFT+C, CTRL+T

Completes the Scenario by inserting '`start scenario`' and '`end scenario`' text at the beginning and end of the current file.

4.2.8 Edit

When the Edit window is open these commands provide basic text editing capabilities.

Edit > Cut, **Edit > Copy**, **Edit > Paste**, **Edit > Find/Replace**

Traditional text editing commands. The standard key combinations (such as CTRL+X and CTRL+C) are also recognized.

Edit > Format, CTRL+SHIFT+F, CTRL+I

Automatically formats the text in the current edit window. Formatting removes extra blank lines and normalizes the indentation.

4.2.8.1 Keyboard-only commands

Some functions are not accessible from the menu bar; these are mostly common editing commands that are invoked with standard ('traditional') key combinations ('shortcuts').

CRL+A

Select all text in the current field or editor window.

CTRL+E

Execute the current scenario.

CTRL+SPACE

Complete the scenario at the current location. If the cursor is on an empty line, the list of possible commands is shown; on a complete command, the appropriate editor for that command is opened (if available).

CTRL+Z

Undo the last command.

4.2.8.2 Macro Recorder

The Tester has a simple macro recorder, intended for *ad hoc* use, controlled by keyboard commands only. It can record and store a single un-named macro which is only retained for the current session (until the Tester is closed).

CTRL+SHIFT+R
Start recording a new macro. Any previously-recorded macro is deleted.

CTRL+SHIFT+S
Stop recording.

CTRL+SHIFT+M
Play the recorded macro.

4.3 Lists

4.3.1 Services

Displays a list of the Services running on this node. A display-only window.

4.3.2 Scripts

Displays a list of the installed Scripts (`.sd` files) and Batch Scripts (`.bd` files).

Refresh
Refreshes the list.

<select> a Script
Displays the Script in the Edit window

4.3.3 Macros

Displays a list of the installed Macros (`.md` files).

Refresh
Refreshes the list.

Scen
Checking this option displays Scripts as well as Macros.

<select> a Macro
Displays the Macro in the Edit window

4.3.4 Readers

For each reader the count of received samples is displayed as well as the QoS and partition. A check box is provided for changing the read state or the show state. When Read is unchecked the reader stops reading samples. When Show is unchecked the topic samples of that topic will not be displayed in the sample list.

Select all
Checks the show state for all topic samples.

Deselect all

Unchecks the show state for all topic samples.

<select> a Topic Instance

Enables you to check/uncheck the Read and Show state.

<right-click> Delete Reader, DELETE

Deletes the selected reader.

<right-click> Recreate Reader, CTRL+R

Recreates the selected reader and as such re-reads any persistent/transient data available.

<right-click> Show First Sample, F3, or double-click on the reader

Shows the first sample for the selected reader.

<right-click> Edit Sample, F4

Opens an Edit Sample window for the selected topic.

F9

Opens the field selection window for the display of fields of the selected topic.

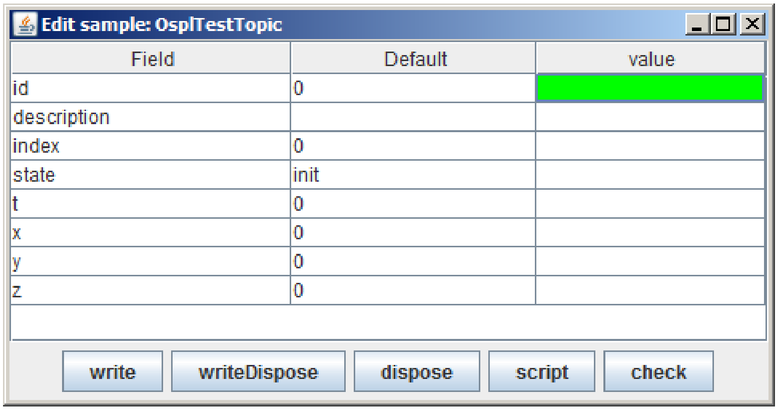
4.3.4.1 Edit Sample Window

The Edit Sample window is used for editing field values of a topic and then writing the sample or dispose the instance. It is also used to insert the topic values as a ‘send’ or ‘check’ entry in the current script (at the cursor position in the script window).

The Edit Sample window can be filled with a topic from both the Topics window and the Sample List window with the F4 key. If the topic write window is filled with a topic from the topics list window then the values are all empty (except for union discriminators which get a default value). If the window is filled from the sample list window then the fields get the values of the selected topic sample in the sample list. The key fields are marked in green and the foreign keys are marked in yellow.

Fields can be edited by selecting the edit field (right most column). If the field is of an enumerated type then a combo box is displayed which provides all possible values. The topmost value is empty for reset to the default value (not set).

The keyboard can be used to navigate the edit fields. The cursor UP and DOWN (arrow) keys move between fields; any other key starts editing the value in the current field.



Field	Default	value
id	0	
description		
index	0	
state	init	
t	0	
x	0	
y	0	
z	0	

Figure 65 Edit sample window

(There is a second form of this window, used when opened from the script with Ctrl+Space, Ctrl+Left-click, or as part of completion. It only has two buttons: OK and Cancel. Pressing Ctrl+Enter or Ctrl+Return is the same as clicking OK.)

write
Write the sample.

writeDispose
Write the sample and Dispose the instance.

dispose
Dispose the instance.

script
Instead of writing the sample this create the script commands to write the sample. These commands are inserted into the current scenario being edited and the user will be taken to this text.

check
Similar to **script** but creates the script command to check the sample values.

F4
Copy the current selected field from the topic in the instance window.

F5
Copy all fields based on an equal name from the topic in the instance window.

F6
Fill all fields with `.sec` in the name with the current time seconds and fields with `.nanosec` in the name with the current tme in nanoseconds.

Ctrl+T
Fills a field of type int with the seconds part of the current time.

CTRL+U
: Fills a field of type `long` with a unique key.

CTRL+V
: Paste a value.

ALT+DOWN
: Opens the enum editor.

ENTER, RETURN
: Commits the current edited value.

ESC
: Discards the current edited value.

Once the desired values have been entered the topic can be written by clicking the Write button, disposed by clicking the Dispose button, or write disposed by clicking the WriteDispose button.

4.3.5 Topics

The topics list displays the list of topics as known in the system.

<select> a Topic
: Selects a Topic.

<right-click> Create Reader
: Create a Reader for the selected Topic.

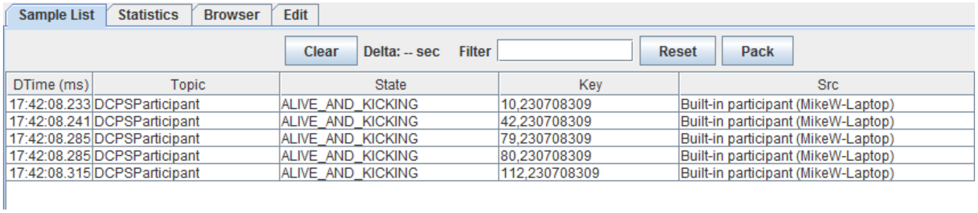
<right-click> Create Default Reader
: Makes the selected Reader the default reader to be displayed in the Samples List.

F2
: The key list definition window will open which allows to change the (foreign) keys. The syntax is the same as in the add topic window or topic file. To support the selection of the keys the primary fields of the topic are displayed and will be inserted at the cursor position in the edit field when clicked.

4.4 Windows

4.4.1 Sample List Window

The Sample List window is used to display samples. By default the delta time, topic name, state, key, and source are displayed. Additional columns can be added and filters defined.



DTime (ms)	Topic	State	Key	Src
17:42:08.233	DCPSParticipant	ALIVE_AND_KICKING	10,230708309	Built-in participant (MikeW-Laptop)
17:42:08.241	DCPSParticipant	ALIVE_AND_KICKING	42,230708309	Built-in participant (MikeW-Laptop)
17:42:08.285	DCPSParticipant	ALIVE_AND_KICKING	79,230708309	Built-in participant (MikeW-Laptop)
17:42:08.285	DCPSParticipant	ALIVE_AND_KICKING	80,230708309	Built-in participant (MikeW-Laptop)
17:42:08.315	DCPSParticipant	ALIVE_AND_KICKING	112,230708309	Built-in participant (MikeW-Laptop)

Figure 66 Sample List window

Clear

Clears the list.

Filter <value>

The current filter value.

Reset

Resets the filter value.

Pack

Adjusts the displayed column widths.

<select> a Sample

Selects a sample to use with **<right-click>**commands. CTRL+LEFT-CLICK selects another sample. If exactly two samples are selected, the difference in source time will be displayed in the top bar of the Sample List window.

<right-click>Select Extra Fields, F9

Opens a dialog box allowing selection of extra fields to display.

<right-click> Display Sample, **<double-click>**

Displays sample details.

<right-click> Display Sample New Window, F3

Displays sample details in new window.

<right-click> Compare Sample, F2

Compares two samples with each other and show differences in red colour.

<right-click> Edit Sample, F4

Allows Tester to edit the selected sample values.

<right-click> Filter on topic, CTRL+F5

Filters on the selected topic value.

<right-click> Filter on topic and key, F5

Filters on both the selected topic and key values.

<right-click> Filter on State, F12

Filters on the State of the selected sample.

<right-click> Filter of Key, F6
Filters on the Key value of the selected sample.

<right-click> Filter on Column Text, F
Sets the filter to be the value of the current column.

<right-click> Filter Reset, F7
Resets the filter value.

<right-click> Delete extra column, DEL
Removes the selected extra column from the list.

<right-click> Add Column as Key to Chart, K
Assigns the selected column as the key field for the chart.

<right-click> Add Column as X to Chart, X
Assigns the selected column as the x-axis for the chart.

<right-click> Add Column as Y to Chart, Y
Assigns the selected column as the y-axis for the chart.

CTRL+F
Finds the next sample containing the search text in any column.

4.4.2 Statistics Window

The Statistics window provides statistics for the topics in use, such as write count, number of alive topics, *etc*. The following values are displayed for each topic:

`Count`	The number of samples currently in the OpenSplice database
`Arrived`	The number of arrived samples
`Takes`	The number of takes by the reader
`Reads`	The number of reads by the reader
`Alive`	The number of alive topics (instances not disposed)
`Writes`	The number of written samples

The left table shows either the participants, the topics, or the statistics of the currently selected reader/writer as indicated by the selected tab.

When the list of participants is shown, a participant can be selected. The second table shows the list of readers with their statistics, the third table show the list of writers with their statistics.

When the list of topics is shown, a topic can be selected. The second table shows the list of participants reading the topics with their statistics, the third table shows the list of participants writing the topic with their statistics.

If a value of `-1` or `-2` is shown then an error occurred during the retrieval of the statistics for the reader/writer.

By selecting a row in the reader or writer list all statistics for that reader or writer will be shown in Stats tab of the left window.

Refresh

Will refresh the content.

Add readers

Will add the topics in the reader list to the list of monitored topics.

Add writers

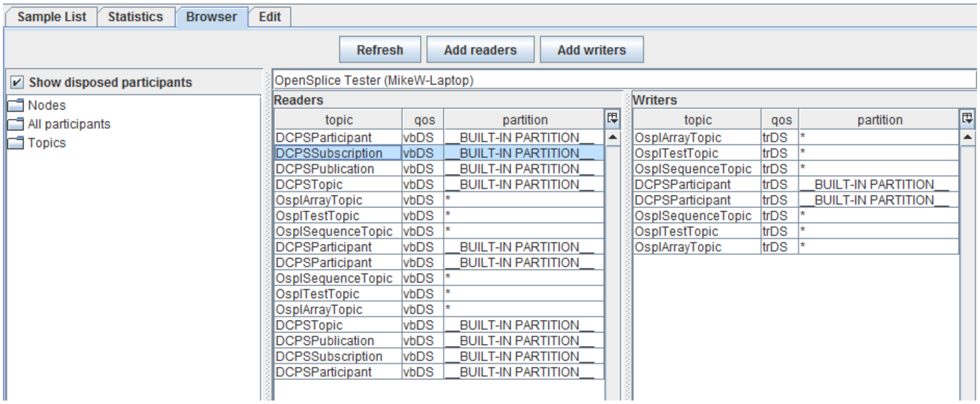
Will add the topics in the writer list to the list of monitored topics.

CTRL+F

Finds the next reader/writer containing the search text in any column.

4.4.3 Browser Window

The Browser window enables you to view the Readers and Writers in the system. You may browse by Node, Participant, or Topic.



OpenSplice Tester (MikeW-Laptop)

Readers

topic	qos	partition
DCPSParticipant	vbDS	__BUILT-IN PARTITION__
DCPSSubscription	vbDS	__BUILT-IN PARTITION__
DCPSPublication	vbDS	__BUILT-IN PARTITION__
DCPSTopic	vbDS	__BUILT-IN PARTITION__
OsplArrayTopic	vbDS	*
OsplTestTopic	vbDS	*
OsplSequenceTopic	vbDS	*
DCPSParticipant	vbDS	__BUILT-IN PARTITION__
DCPSParticipant	vbDS	__BUILT-IN PARTITION__
OsplSequenceTopic	vbDS	*
OsplTestTopic	vbDS	*
OsplArrayTopic	vbDS	*
DCPSTopic	vbDS	__BUILT-IN PARTITION__
DCPSPublication	vbDS	__BUILT-IN PARTITION__
DCPSSubscription	vbDS	__BUILT-IN PARTITION__
DCPSParticipant	vbDS	__BUILT-IN PARTITION__

Writers

topic	qos	partition
OsplArrayTopic	trDS	*
OsplTestTopic	trDS	*
OsplSequenceTopic	trDS	*
DCPSParticipant	trDS	__BUILT-IN PARTITION__
DCPSParticipant	trDS	__BUILT-IN PARTITION__
OsplSequenceTopic	trDS	*
OsplTestTopic	trDS	*
OsplArrayTopic	trDS	*

Figure 67 Browser window

Refresh

Will refresh the browser content.

Add readers

Will create a Tester reader from the list of readers for the selected read-topic. The QoS of the discovered reader will be used to ensure that data read by that reader will be captured in the timeline.

Add writers

Will create a Tester reader from the list of writers for the selected written-topic. The QoS of the discovered writer will be used to ensure that data written by that writer will be captured in the timeline.

Show disposed participants

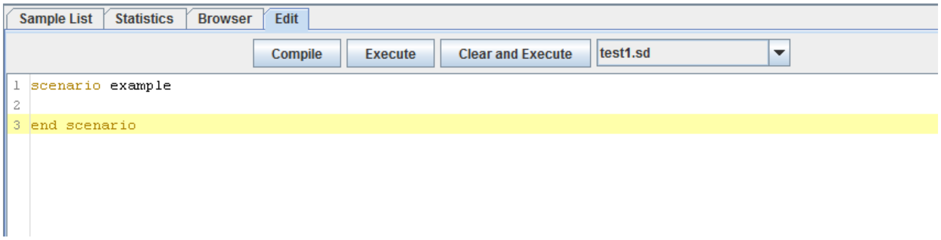
Used to toggle the display of disposed participants.

CTRL+F

Finds the next reader/writer containing the search text in any column.

4.4.4 Edit Window

The Edit window is used to create and modify Scripts and Macros. Refer to Chapter 5, *Scripting*, on page 71, for more details.

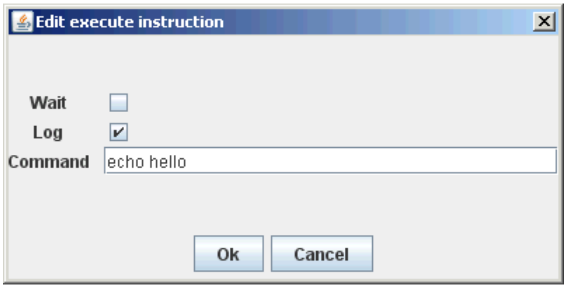


```
scenario example

end scenario
```

Figure 68 Edit window

Traditional text editing commands and standard key combinations (such as CTRL+X and CTRL+C) are recognized. Menu commands and keyboard shortcuts for editing scripts and macros are described in sections 4.2.7, *Editor*, 4.2.8, *Edit*, 4.3.2, *Scripts*, and 4.3.3, *Macros*.

When editing macros, instruction-specific editing dialogs may open; for example, the `send`, `check` and `execute` macro instructions have their own editing dialogs which help to make your entries conform to their syntax.



Wait □
Log ☑
Command echo hello

Figure 69 Editor for *`execute`* instruction

Compile

Compile the current content.

Execute

Run the current script or macro without clearing the sample list.

Clear and Execute

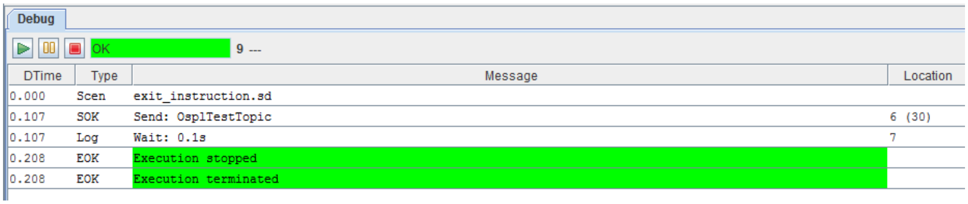
Clears the sample list and then runs the current script/macro and returns the user to the Sample List window.

<drop down>

Allows for quick selection of recently edited scripts/macros.

4.4.5 Debug Window

The Debug window is used for tracing/debugging Script compilation and execution. For each step, the day/time, type of message, and message text is displayed along with the location (line number) in the scenario.



DTime	Type	Message	Location
0.000	Scen	exit_instruction.sd	
0.107	SOK	Send: OsplTestTopic	6 (30)
0.107	Log	Wait: 0.1s	7
0.208	EOK	Execution stopped	
0.208	EOK	Execution terminated	

Figure 70 Debug window

Control execution of the scenario with the buttons at the top left of the window:

▶	Start	Start or resume execution
⏸	Pause	Pause execution
■	Stop	Stop (halt) execution

CTRL+F

Finds the next message containing the search text in any column.

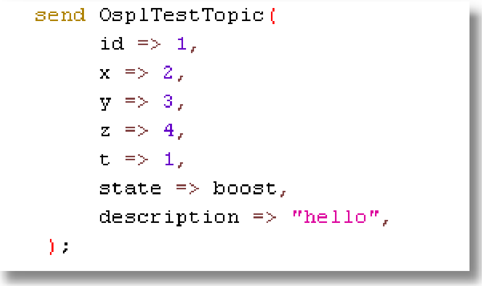
CHAPTER 5 Scripting

The Tester provides automatic testing capabilities by means of scripting. This chapter describes the features of Tester's built-in scripting instructions, and how to install additional script engines.

5.1 The Script Language

The script language as used by the Tester is specifically designed to create readable and easily maintainable scripts.

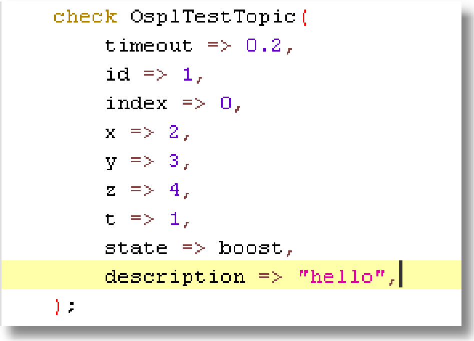
Instructions are simple, with named parameters which enable the Tester to limit the testing to the fields applicable to the test. For example, the `send` instruction is an instruction which sends a topic. The basic syntax is the keyword `send` followed by the `topicname` and a list of named parameters between parentheses (round brackets), terminated with a semicolon.



```
send OsplTestTopic(
     id => 1,
     x => 2,
     y => 3,
     z => 4,
     t => 1,
     state => boost,
     description => "hello",
 );
```

Figure 71 Illustrating send keyword syntax

The `check` instruction is similar to the `send` instruction; it has options to find a specific instance using key fields or a query.



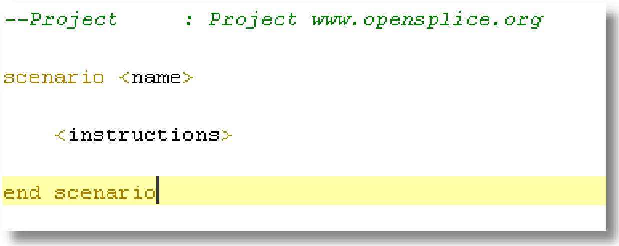
```
check OsplTestTopic(
    timeout => 0.2,
    id => 1,
    index => 0,
    x => 2,
    y => 3,
    z => 4,
    t => 1,
    state => boost,
    description => "hello",
);
```

Figure 72 Illustrating check keyword syntax

In this example a `timeout` is set, which will allow a wait of up to 0.2 seconds for the topic sample for the correct instance to arrive.

5.1.1 A script file

A `scenario` has the following format:



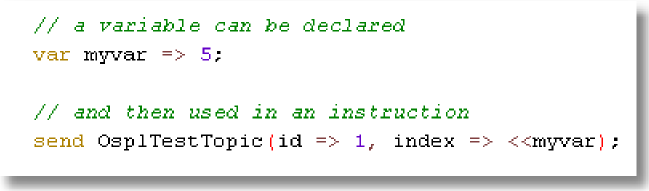
```
--Project      : Project www.opensplice.org

scenario <name>

    <instructions>

end scenario
```

Figure 73 Illustrating `scenario` keyword syntax

The name is for information only, and is not used further.

5.1.2 Variables

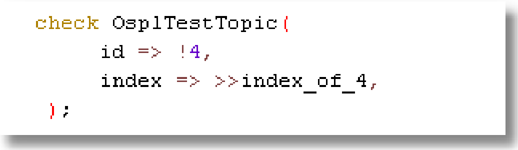
The script language allows the use of *variables*. Variables can be used to store values that can then be used at a later time. A variable is indicated by either a << or >> prefix. Variables may be declared implicitly, or explicitly using the `var` instruction.



```
// a variable can be declared
var myvar => 5;

// and then used in an instruction
send OsplTestTopic(id => 1, index => <<myvar);
```

Figure 74 Example variable

In this example the variable `myvar` is declared and initialized with the value `5`. Within the `send` instruction the variable is used to provide the value for the field `index`. The << prefix indicates the direction of the assignment from the variable to the field.

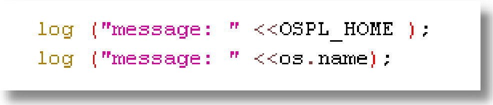


```
check OsplTestTopic(
      id => !4,
      index => >>index_of_4,
 );
```

Figure 75 Variable with >> prefix

Here the variable `index_of_4` is declared implicitly and the value of the field `index` is copied to the variable (the prefix >> points to the variable).

All environment variables and java virtual machine (JVM) properties are also available as variables, and they can be used as shown below:



```
log ("message: " <<OSPL_HOME );
log ("message: " <<os.name);
```

Figure 76 Using environment variables

5.1.2.1 Special variables

There are some special variables which can be useful in scripts.

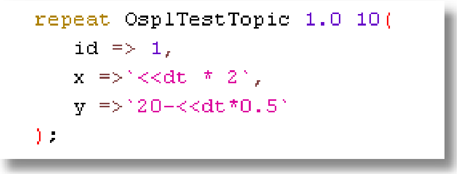
`curtime_sec` and `curtime_nsec` provide the second and nanosecond parts of the current time.

`uniqid` provides a unique number for every call, within the same session of the Tester.

Note that these special values are used without the ‘<<’ prefix.

5.1.3 Embedded Scripts

Inside a scenario any script compatible with the java ScriptEngineFactory can be used to provide calculated values for fields in a `send`, `check` or `var` instruction. Embedded script is enclosed by left single quotes (pink text):

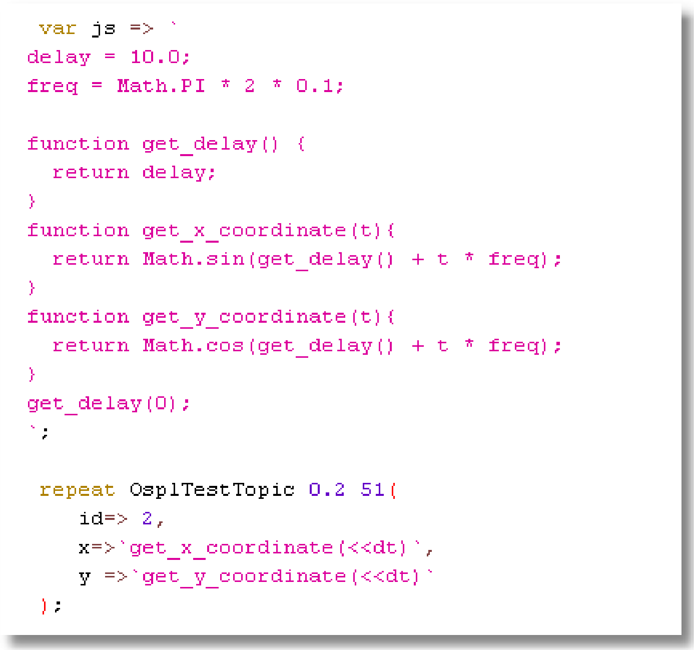


```
repeat OsplTestTopic 1.0 10(
   id => 1,
   x =>`<<dt * 2`,
   y =>`20-<<dt*0.5`
);
```

Figure 77 Embedded javascript

Variables used in the javascript are translated before the evaluation of the script. In this specific case the `<<dt` is the delta time in the repeat function. All javascript in one scenario is executed in the same scope, and functions and variables declared at the beginning of a script are available later in the script.

A specific script language can be selected by providing the name of the script language in the first line of the embedded script: "`#!<language>`", for example "`#!js`". Note that the language description must not be followed by any other text. See section 5.6, *Installing Script Engines*, on page 83, for instructions on installing a scripting language for use with the OpenSplice Tester. If no language descriptor is provided on the first line of a script, the default language is used as set in Preferences.

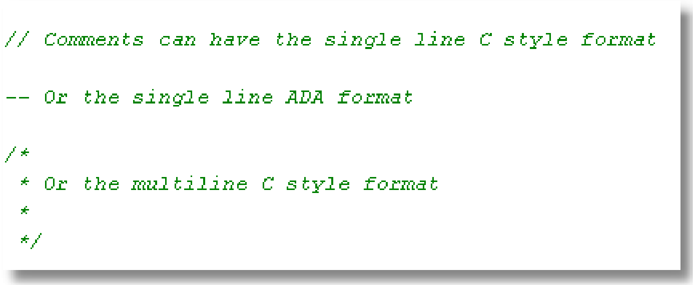


```
 var js => `
delay = 10.0;
freq = Math.PI * 2 * 0.1;

function get_delay() {
  return delay;
}
function get_x_coordinate(t){
  return Math.sin(get_delay() + t * freq);
}
function get_y_coordinate(t){
  return Math.cos(get_delay() + t * freq);
}
get_delay(0);
`;

 repeat OsplTestTopic 0.2 51(
    id=> 2,
    x=>`get_x_coordinate(<<dt)`,
    y =>`get_y_coordinate(<<dt)`
 );
```

Figure 78 Embedded javascript

5.1.4 Comments

Comments can have the following formats:



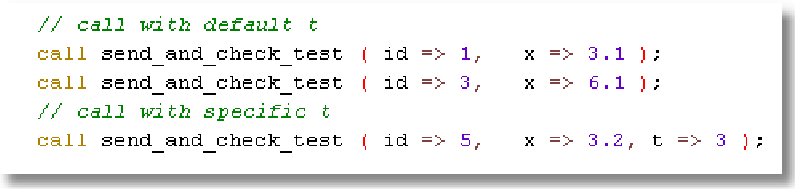
```
// Comments can have the single line C style format

-- Or the single line ADA format

/*
 * Or the multiline C style format
 *
 */
```

Figure 79 Format of comments

Within the scenario editor comments are displayed in green.

5.1.5 Macros

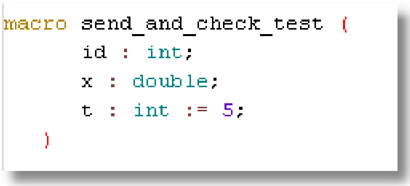
For repeated scenarios a repetitive part can be separated in a separate script file called a macro. Macros can have parameters.



```
// call with default t
call send_and_check_test ( id => 1,   x => 3.1 );
call send_and_check_test ( id => 3,   x => 6.1 );
// call with specific t
call send_and_check_test ( id => 5,   x => 3.2, t => 3 );
```

Figure 80 Calling a macro with parameters

Similarly to `send` and `check` instructions, values for fields can be optional. However, in a macro a default value *must* be provided for a parameter to be optional.



```
macro send_and_check_test (
      id : int;
      x : double;
      t : int := 5;
   )
```

Figure 81 Setting a default value for a macro parameter

In this case `t` is optional, `id` and `x` are mandatory.

It is possible to call a scenario using the `call` instruction. Scenarios do not have parameters.

5.2 The Instructions

5.2.1 Send

Instruction to publish a sample of a topic.

```
send <readername> ( [fieldname => value,]*);
```

5.2.2 Dispose

Instruction to dispose an instance of a topic.

```
dispose <readername> ( [fieldname => value,]*);
```

5.2.3 Writedispose

Instruction to write dispose an instance of a topic.

```
writedispose <readername> ( [fieldname => value,]*);
```

5.2.4 Check

Instruction to check a sample of a topic.

```
Check[_last|_any] <readername> ( [timeout => <timeout in
seconds>,] [<fieldname> => [!]<value>[:deviation],]*);
```

A timeout value can be provided allowing the check to wait for `<timeout in seconds>` for the sample to arrive. If a sample meeting the criteria of the check is available either directly or within `timeout` seconds the fields as provided in the parameter list will be verified for correctness.

When the value of a field is an output variable:

```
>><varname>
```

Then the value will not be checked but entered in the variable with the name `<varname>`.

There are two special fields, `topicReceived` and `topicDisposed`, which when used will provide a `true` or `false` value into a variable.

When no sample is found which meets the criteria of the check then `topicReceived` will be set to `false` (and the check instruction will not fail); if a sample *is* received the value will be set to `true`. When a field `topicDisposed` is found, then the variable will be set to `true` if the sample was disposed and `false` if the sample was not disposed. In this case no fail is reported upon a check instruction when the checked sample was disposed.

The value can be given a possible deviation in the form `<value>:<allowed deviation>`. In this case when the value for the field in the received sample is within the range from `value` *minus* `allowed_deviation` to `value` *plus* `allowed_deviation`, the value is considered correct.

The sample which matches the check can be determined in several ways:

1. The topic does not have a keyfield(s) or the topic has keyfield(s) but no value is provide for all keyfield(s). In this case the oldest not checked or marked sample is checked.

2. The topic has keyfield(s) and the check provides a value for all keyfield(s). In this case the last sample with the key is checked. If no matching sample (within the possible timeout) is found then the check fails.

3. One or more fields of the check are marked as a query by prefixing the value with a '`!`'. The oldest not checked or marked sample which matches the query is checked. If no matching sample is found (within the possible timeout) the check fails.

4. Instead of `check`, the command `check_last` is used. In this case (as for situations 1 and 3) the last sample matching the criteria is checked.

5. Instead of `check`, the command `check_any` is used. In this case also previously-checked or marked samples are considered.

5.2.5 Miss

Instruction to check that no sample of a topic was received since the last checked or marked sample for the given key/query. The same rules apply as for the `check` instruction with respect to finding (or not) the matching topic sample.

```
miss <topicname> ([timeout => <timeout_in_seconds>,] [<fieldname>
=> [!]<value>[:<deviation>],]*);
```

5.2.6 Disposed

Instruction to check that an instance of a topic is disposed for the given key/query. The same rules apply as for the `check` instruction with respect to finding the disposed instance. Note that field values are only provided to find a specific instance (either by key or by query) and not verified for values as part of this instruction.

```
disposed <topicname> ([timeout => <timeout_in_seconds>,]
[<fieldname> => [!]<value>,]*);
```

5.2.7 Mark

Mark all samples (with the given key/query) as read. Any regular miss/check function will not ‘see’ topic samples received before the mark instruction. If no key or query is provided all samples will be marked as read (and therefore not considered for `check` or `check_last` instructions). If a key value or query is provided, all samples matching the key/query will be marked as read.

```
mark <topicname> ( [fieldname => value,]*);
```

5.2.8 Repeat

Instruction to repeatedly send a topic for a specified count or until disposed.

```
repeat <topicname> <period> <count> ( [fieldname => value,]*);
```

If `<count>` is ‘`0`’ then the repeat will continue until the scenario terminates or until a dispose for the same topic and key. The variable `dt` is available for calculating a field value based on time since the repeat was started. The period indicates the period with which the topic will be sent. Note that a repeat command by itself does not extend the execution of a scenario and that when a scenario finishes (*i.e.* all following instructions are executed) the repeat instruction is terminated automatically. In such a case the wait or message instruction can be used to ensure that the repeat instruction is completed.

5.2.9 Set

The `set` instruction allows the call of a macro in a table-like fashion. The command allows a number of static parameters and variable parameters. The command has the following format:

```
set <macroname>
([<fieldname>=><value>]*)((<fieldname>*),[(<value>*),]*);
```

For example the following set instruction:

```
      set send_and_check_test (
           t => 2)
     ((     x,id),
      (   3.1, 1),
      (  2.34, 2),
      ( 3.678, 3),
      (  6.34, 4),
      ( 99.99, 5))
```

In this example the `send_and_check_test` macro is called five times, all five calls will be made with `t` = `2` and the values for `x` and `id` as indicated by each row of values. This can be very usefull for testing of translations.

5.2.10 Execute

The `execute` instruction allows the execution of an application or command line script on the native OS.

```
execute [wait] [log] “<instruction>”;
```

If `wait` is set then the instruction will wait for the execute to complete. If `log` is set then the output of the execute will be logged to the Debug window (and resulting dump file). When `log` is used `wait` should also be used, to avoid overwriting log messages.

5.2.11 Log

The `log` instruction logs a message to the Debug window. Log messages can provide information immediately (*i.e.* a step being made in a script, or a value of some variable) or post-execution as part of the logfile which includes the full content of the Debug window.

```
log (“message” [optional var]);
```

5.2.12 Message

The `message` instruction opens a dialog with the message and allows the operator to provide feedback and a OK/NOK indication. The feedback plus OK/NOK indication are logged to the debug window.

```
message (“message text” [optional var]);
```

This instruction is useful for semi-automatic testing of user interfaces where the GUI part is done manually using message instructions.

5.2.13 Fail

The `fail` instruction fails the execution of the scenario (final result). The execution terminates.

```
fail (“message” [optional var]);
```

The `fail` instruction can be useful in combination with an `if` instruction, for instance when a complex check is executed using javascript.

5.2.14 Call

The `call` instruction calls a macro or scenario. The name of the macro/scenario is the filename without extension. Macros must be on the `macropath` as provided in the configuration file. The Macrolist window displays all available macros. Also note that the macro name *must* be unique throughout all of the available macros because the path is not part of the selection of a macro (just the filename withour extension).

```
call <macroname> ([<parametername> => <value>,]*);
```

5.2.15 Reader

The `reader` instruction allows the creation or deletion of a reader. When the keyword `dispose` is used the reader (if it exists for that topic) will be deleted. When a reader is created the `topicname` is mandatory.

```
reader [dispose] (<topicname> [, <qos> [,<partition>
[,<readername>]]]);
```

The `qos` can be provided in short notation (2 or 4 characters):

```
< v | l | t | p >< b | r >[h][<S|E><D|S>
```

where

`< v	l	t	p >`	Volatile, local transient, transient or persistent
`< b	r >`	Best effort or reliable
`[h]`	History, for a “keep” of 10 which allows for the reception of 10 samples with the same key in one poll interval
`<S|E>`	Shared or exclusive ownership
`<D|S>`	Ordering based on Destination or Source time stamp

5.3 Instructions for Graphs

5.3.1 Graph

The `graph` instruction allows manipulation or save of the graph. It has the following parameters:

```
X
Y
Key
Color
Title
```

```
xUnits
yUnits
save => <name>
show => true|false
reset => true|false
```

Note that all graphs have the same X component; when omitted the X will be the sample time. If the Y parameter is set, then a new trace is created for the current graph. The X, key, color, title and units are used for this trace if provided.

If `reset` is `true`, then the graph is cleared (*i.e.* all existing traces are deleted) before creating any new trace. If `show` is `true` then the graph is made visible after adding the trace; when `false`, then the graph is hidden after adding the trace. When `save` is `true` the graph will be saved to an image file after the trace has been added.

5.3.2 Column

The `column` instruction allows the creation of an extra column from a script for use by the graph instruction.

```
column [clear] (<fieldname> [, <columnname>]);
```

When the optional `clear` is set then the column for the field with name `fieldname` will be removed. When `columnname` is omitted, the `columnname` will be the same as the `fieldname`.

5.4 Instructions for Flow Control

5.4.1 Wait

The `wait` instruction forces a wait in the execution of the script. The time is provided in seconds.

```
wait (<time in seconds>);
```

Value can be a variable.

5.4.2 If

The `if` instruction allows conditional execution of instructions.

```
If (val1 <operator> val2) then
  <true instruction list>
[else
  <false instruction list>]
endif;
```

`<Operator>` is one of ‘==’, ‘!=’, ‘>’, ‘<‘, ‘>=’, ‘<=’, ‘||’, ‘&&’.

Expressions can be layered with brackets:

```
((x>0) && (y>0))
```

5.4.3 For

The `for` instruction allows the execution of a list of instructions multiple times.

```
for ,<var> in 1 .. 10 loop
  <instruction list which can use <<var>
endloop;
```

or

```
for <var> in (a,b,c) loop
  <instruction list which can use <<var>
endloop;
```

5.4.4 Exit

The `exit` instruction exits the scenario

```
exit;
```

5.5 Instructions for the Message Interface

5.5.1 Write

The `write` instruction writes a message to the interface.

```
write <interface>.<message> ([<fieldname> => <value>,]*);
```

5.5.2 Read

The `read` instruction checks a received message from the interface.

```
read <interface>.<message> ([<fieldname> => <value>,]*);
```

5.5.3 Connect

The `connect` instruction calls the `connect` of the interface. The functionality depends on the implementation in the interface.

```
connect <interface>;
```

5.5.4 Disconnect

The `disconnect` instruction calls the `disconnect` of the interface. The functionality depends on the implementation in the interface.

```
disconnect <interface>;
```

5.5.5 Control

The `control` instruction allows the execution of special instructions as provided by the interface.

```
control <interface>.<instruction>[ ([<fieldname> => <value>,]*)];
```

5.6 Installing Script Engines

In order to use additional script languages the appropriate script engines must be added to the Java `classpath`. The Java JRE already comes with a JavaScript engine by default (*i.e.* no specific installation is required). More Java script engines are available and can be used to support different scripting languages for the embedded scripts inside the scenario scripts, or for the additional fields.

When Tester starts, the available script engines will be logged (default log file is `/tmp/OSPLTEST.log`).

5.6.1 Jython

Download and install Jython on the target machine. Include `jython.jar`, which is normally located in the Jython installation directory, in the `classpath` of the OpenSplice Tester. Use this language by adding the following line as the first line of each script using the Jython script language:

```
#!jython
```

5.6.2 Jruby

Download and install Jruby on the target machine. Include `jruby.jar`, which is normally located in the `lib` directory in the Jruby installation, in the `classpath` of the OpenSplice Tester. Use this language by adding the following line as the first line of each script using the Jruby script language:

```
#!jruby
```

5.6.3 Groovy

Download and install Groovy on the target machine. Include `groovy-all-<version>.jar`, which is normally located in the `embeddable` directory in the Groovy installation, in the `classpath` of the OpenSplice Tester. Use this language by adding the following line as the first line of each script using the Groovy script language:

```
#!groovy
```

CHAPTER 6 Message Interfaces

This chapter describes how to test applications with non-DDS interfaces.

6.1 Message interfaces

An important feature of the OpenSplice Tester is the support of additional interfaces. When an application under test only has a DDS interface it is probably easy to test automatically by stimulating it from the OpenSplice Tester with samples and verifying the samples produced by the application under test. When the application under test has a GUI component, the message instruction can be used to perform a semi automated test where the Tester performs manual control of the GUI and/or performs visual inspections of the GUI (as instructed in the message instruction).

When an application under test has a non-DDS interface, then the message interface of OpenSplice Tester can be used. There are a number of constraints on the use of a message interface:

- The interface must consist of a limited number of message types which can be described by a static set of fields with static types.
- It must be possible upon reception of a message over the interface, to determine a message type, and from the message type to interpret the message and determine the value for each field of the message.

If these requirements are met, a message interface can be developed for an specific interface of an application under test. This will allow automated testing where messages are written to the test interface, the message received from the test interface will be added to the sample list and can be checked in the same manner as DDS samples.

6.2 Getting Started with a Message Interface

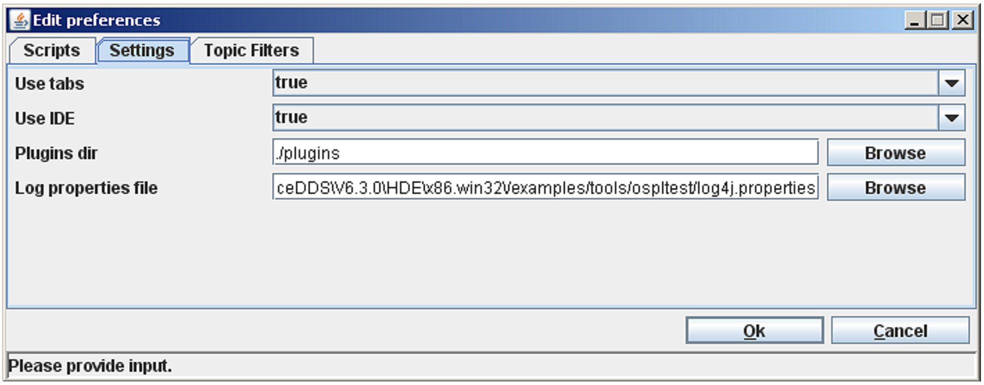
The best way to get started with a message interface is to compile and use the TestInterface. The TestInterface is an example message interface which uses a TCP/IP connection and sends a memory-mapped message with a static structure over this interface. The source for the TestInterface is provided and can be found here:

```
<OSPL_HOME>/examples/tools/ospltest/TestInterface
```

To compile the TestInterface, `ant` and a JDK1.6 must be installed. To build the TestInterface execute `ant` in the `TestInterface` directory. This will compile the testinterface and install the resulting plugin in:

```
<OSPL_HOME>/examples/tools/ospltest/plugins
```

To run with the plugin, make sure the plugin path points to this directory. The plugin path can be set in Preferences:



Use tabs: true

Use IDE: true

Plugins dir: ./plugins

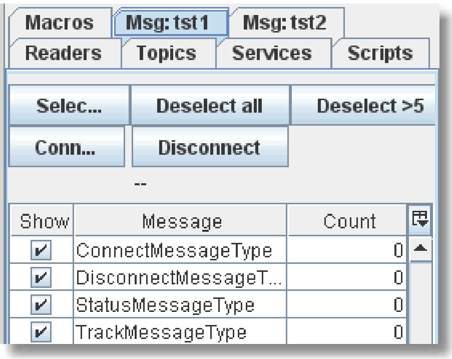
Log properties file: ceDDS\V6.3.0\HDE\x86.win32\examples/tools/ospltest/log4j.properties

Please provide input.

Figure 82 Setting the plugins path in Preferences

If the plugins directory is changed, the Tester needs to be restarted. Once restarted, make sure that OpenSplice is running (the TestInterface registers a topic which will fail is DDS is not running upon startup).

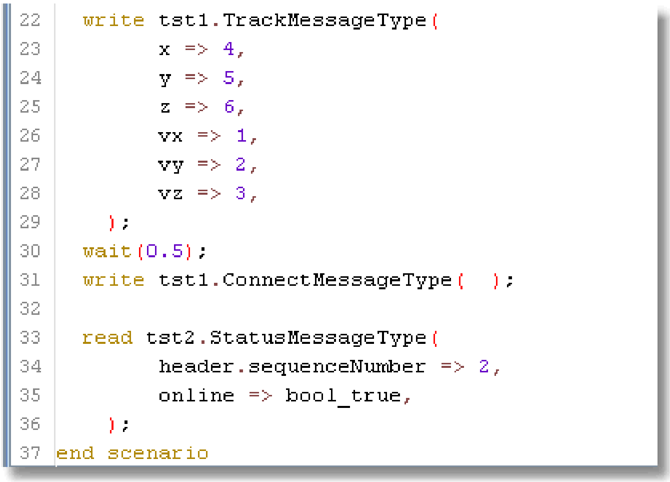
Now two instances of the testinterface should show up in the left tab pane or in separate windows if Use Tabs is `false`. Similar to the Readers pane, the table will show the available messages and the number of received messages per message type. Since there is no application under test, the testinterface is instantiated twice and connected back to back. As a result a message written to the instance “tst1” will be received on the instance “tst2” and *vice versa*. Also the testinterface has created a topic, OsplTestLogTopic, and the test interface will write a sample of this topic for each write and read with the content of the message in hexadecimal format.



Show	Message	Count
☑	ConnectMessageType	0
☑	DisconnectMessageT...	0
☑	StatusMessageType	0
☑	TrackMessageType	0

Figure 83 Messages received on instance `tst1`

Now select the `test_interface.sd` script, which can be found in `examples/tools/ospltest/scripts`:

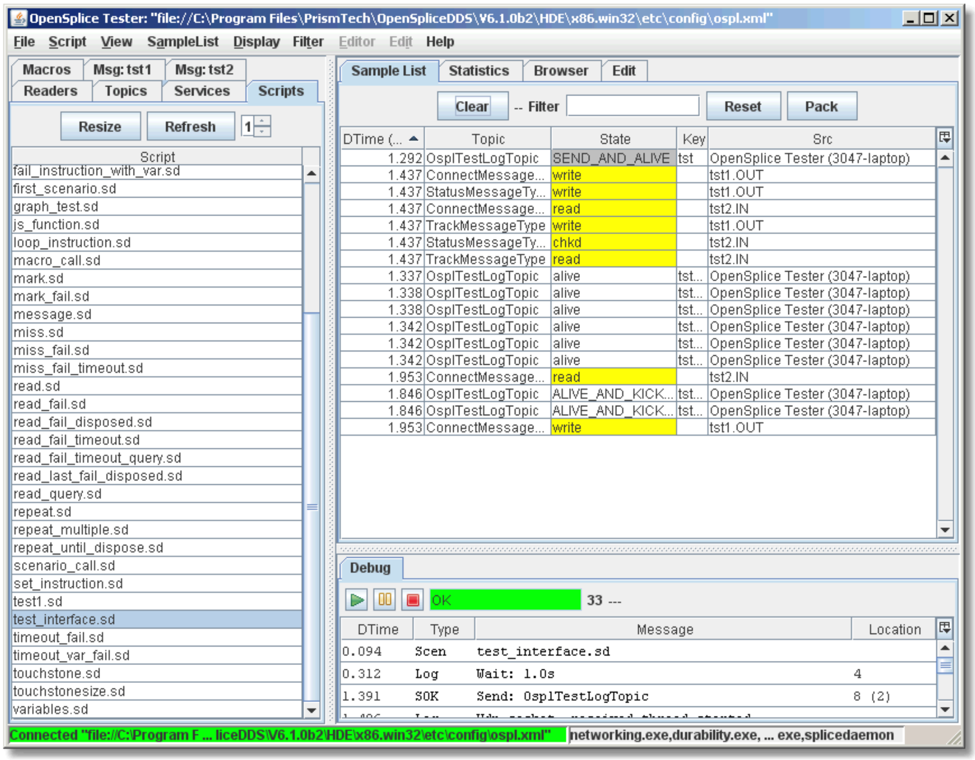


```
  write tst1.TrackMessageType(
        x => 4,
        y => 5,
        z => 6,
        vx => 1,
        vy => 2,
        vz => 3,
    );
  wait(0.5);
  write tst1.ConnectMessageType(  );

  read tst2.StatusMessageType(
        header.sequenceNumber => 2,
        online => bool_true,
    );
end scenario
```

Figure 84 The script `test_interface.sd`

In the script we can see that, similar to the `send` and `check` instruction, the `write` and `read` instructions are used to write a message to the test interface, or read (check) a message received on the test interface.

Execute the script:



OpenSplice Tester: "file://C:\Program Files\PrismTech\OpenSpliceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl.xml"
File Script View SampleList Display Filter Editor Edit Help
Macros Msg: tst1 Msg: tst2
Readers Topics Services Scripts
Resize Refresh 1
Sample List Statistics Browser Edit
Clear -- Filter Reset Pack
Debug
OK 33 ---
Connected "file://C:\Program F ... liceDDS\V6.1.0b2\HDE\x86.win32\etc\config\ospl.xml"
networking.exe,durability.exe, ... exe,splicedaemon

Script
fail_instruction_with_var.sd
first_scenario.sd
graph_test.sd
js_function.sd
loop_instruction.sd
macro_call.sd
mark.sd
mark_fail.sd
message.sd
miss.sd
miss_fail.sd
miss_fail_timeout.sd
read.sd
read_fail.sd
read_fail_disposed.sd
read_fail_timeout.sd
read_fail_timeout_query.sd
read_last_fail_disposed.sd
read_query.sd
repeat.sd
repeat_multiple.sd
repeat_until_dispose.sd
scenario_call.sd
set_instruction.sd
test1.sd
test_interface.sd
timeout_fail.sd
timeout_var_fail.sd
touchstone.sd
touchstonesize.sd
variables.sd

DTime (...	Topic	State	Key	Src
1.292	OsplTestLogTopic	SEND_AND_ALIVE	tst	OpenSplice Tester (3047-laptop)
1.437	ConnectMessage...	write		tst1.OUT
1.437	StatusMessageTy...	write		tst1.OUT
1.437	ConnectMessage...	read		tst2.IN
1.437	TrackMessageType	write		tst1.OUT
1.437	StatusMessageTy...	chkd		tst2.IN
1.437	TrackMessageType	read		tst2.IN
1.337	OsplTestLogTopic	alive	tst...	OpenSplice Tester (3047-laptop)
1.338	OsplTestLogTopic	alive	tst...	OpenSplice Tester (3047-laptop)
1.338	OsplTestLogTopic	alive	tst...	OpenSplice Tester (3047-laptop)
1.342	OsplTestLogTopic	alive	tst...	OpenSplice Tester (3047-laptop)
1.342	OsplTestLogTopic	alive	tst...	OpenSplice Tester (3047-laptop)
1.342	OsplTestLogTopic	alive	tst...	OpenSplice Tester (3047-laptop)
1.953	ConnectMessage...	read		tst2.IN
1.846	OsplTestLogTopic	ALIVE_AND_KICK...	tst...	OpenSplice Tester (3047-laptop)
1.846	OsplTestLogTopic	ALIVE_AND_KICK...	tst...	OpenSplice Tester (3047-laptop)
1.953	ConnectMessage...	write		tst1.OUT

DTime	Type	Message	Location
0.094	Scen	test_interface.sd	
0.312	Log	Wait: 1.0s	4
1.391	SOK	Send: OsplTestLogTopic	8 (2)

Figure 85 Script `test_interface.sd` running

Here we can see that in the sample list, both the DDS samples as well as the `testinterface` samples are available. As a result the interaction is clearly visible.

6.3 Types of interfaces

When integrating a test interface with the OpenSplice Tester, the following functionality is provided:

- Connect/Disconnect with a parameter
- Write of messages based on parameters of a write call
- Read of messages and display received messages in the sample list
- Check received messages
- Display fields of messages (double click in sample list)
- Hooks upon write/read of a message

The OpenSplice Tester provides two ways to create such a message interface:

- Basic message interface

- Buffered message interface

6.3.1 Basic message interface

If it is not possible to describe the content of each message in an ADA interface description (*i.e.* a static memory-mapped definition of each message type) or when the definition of the interface exists in another format, like a MIB for an SNMP interface, then it is possible to derive from the basic message interface class:

```
BaseMsgInterface
```

Similarly for the messages a class must be derived from:

```
MsgSample
```

Note that both `BaseMsgInterface` and `MsgSample` contain a considerable number of abstract function which then must be provided in order to be able to edit and display samples, as well as read and write sample on their interface.

6.3.2 Buffered message interface

The example test interface is a buffered message interface. The OpenSplice Tester provides support for memory-mapped messages and provides all basic functionality for this type of interface. The messages are described using the ADA language type definition for records with a representation clause. This allows to describe message with bit fields, enums, fixed length strings, integer and double values.

On top of the buffered message interface, an implementation using UDP and TCP is available.

When the buffered message interface is used the provided implementation takes care of interpreting the received messages, decode the messages for display in the sample list or display in the sample window. Upon a write instruction a memory buffer will be built using the parameters of the write call and the message definition as provided in the ADA interface description.

6.3.2.1 ADA Syntax for message definition

For each message a record needs to be defined which describes the exact memory layour of the message. See the ADA message description of the test interface for an example of such a message definition.

6.3.2.2 Message ID translation

By default in a buffered message interface a base record is defined with an idfield to determine the type of the message. Then a function is called to translate the value of the idfield to a name of the record type with the definition of a message of the received type.

If a message does not contain a single field which can be used to determine the message type than the method:

```
RecordType determineMsgType(ByteBuffer buf)
```

Can be overwritten to perform the translation of the received buffer to a message type.

If indeed the id can be retrieved from an id field (enum value) then the function:

```
protected static String transformIdToType(String id)
```

is used to translate the enum label to the name of a record definition. The following translation is done:

- ID is changed to TYPE
- Each character following an underscore (‘_’) is capitalized, as well as the first character and the remaining characters are made lowercase.

As a result an enum label: `HEARTBEAT_MESSAGE_ID` is translated to `HeartbeatMessageType`.

Of course if a different convention is used for describing enum labels and message names, then the `transformIdToString` function can be overridden to perform the required translation.

6.3.2.3 Message Hooks

It is possible to override message hooks at several stages in the send and receive process. This allows specific processing, such as:

- Automatic reply to each received message (acknowledge messages)
- Fill in automatic fields like sequence numbers, crcs, or timestamps
- Ignore messages for reception, like acknowledge messages or heartbeats
- Perform specific checks such as crc check

See the example test interface for an example of the hooks and a description of their function.

6.3.2.4 Control functions

The script control function allows implementation specific control functions to be implemented. In the implementation of the derived interface, the following functions can be overridden (note that the base implementation already provides some control commands, overriding these functions must properly include or forward to the base implementation):

```
public String[] getControlCommands()
```

Provide the list of control commands, note that super.getControlCommands should be used to include the list of control commands of the base impementation.

```
public void control(String command, ParameterList params,
ScenarioRuntime runtime, int line, int column)
```

Execute the control command, with the provided parameters and runtime. In case of an error the line and column can be used as the location of the instruction which failed.

Control functions can be used for any specific function as deemed necessary (of course, all must be implemented in the derived interface class):

- Stop sending heartbeats
- Create incorrect crc
- Stop sending acknowledge
- Determine message frequency

Appendix A *Scripting BNF*

This Appendix gives the formal description of the Tester Scripting language.

TOKENS

```
<DEFAULT> SKIP : {
" "
| "\t"
| "\n"
| "\r"
| <"//" (~["\n","\r"])* ("\n" | "\r" | "\r\n")>
| <"--" (~["\n","\r"])* ("\n" | "\r" | "\r\n")>
| <"/*" (~["*"])* "*" ("*" | ~["*","/"] (~["*"])* "*")* "/">
}
```

```
<DEFAULT> TOKEN : {
<INTEGER_LITERAL: <DECIMAL_LITERAL> (["l","L"])? | <HEX_LITERAL> (["l","L"])?
| <OCTAL_LITERAL> (["l","L"])?>
| <#DECIMAL_LITERAL: (["+","-"])? ["0"-"9"] (["0"-"9"])*>
| <#HEX_LITERAL: "0" ["x","X"] (["0"-"9","a"-"f","A"-"F"])+>
| <#OCTAL_LITERAL: "0" (["0"-"7"])*>
| <FLOATING_POINT_LITERAL: (["+","-"])? (["0"-"9"])+ "." (["0"-"9"])*
(<EXPONENT>)? (["f","F","d","D"])? | "." (["0"-"9"])+ (<EXPONENT>)?
(["f","F","d","D"])? | (["0"-"9"])+ <EXPONENT> (["f","F","d","D"])? |
(["0"-"9"])+ (<EXPONENT>)? ["f","F","d","D"]>
| <#EXPONENT: ["e","E"] (["+","-"])? (["0"-"9"])+>
| <CHARACTER_LITERAL: "\'" (~["\'","\\","\n","\r"] | "\\"
(["n","t","b","r","f","\\","\'","\""] | ["0"-"7"] (["0"-"7"])? | ["0"-"3"]
["0"-"7"] ["0"-"7"])) "\'">
| <STRING_LITERAL: "\"" (~["\"","\\","\n","\r"] | "\\"
(["n","t","b","r","f","\\","\'","\""] | ["0"-"7"] (["0"-"7"])? | ["0"-"3"]
["0"-"7"] ["0"-"7"] | ["\n","\r"] | "\r\n"))* "\"">
| <JAVASCRIPT: "`" (~["`"])* "`">
}
```

```
<DEFAULT> TOKEN : {
```

```
<SEND: "send">
| <REPEAT: "repeat">
| <PERIODIC: "periodic">
| <MACRO: "macro">
| <DISPOSE: "dispose">
| <WRITEDISPOSE: "writedispose">
| <WAIT: "wait">
| <WAITABS: "waitabs">
| <CALL: "call">
| <RUN: "run">
| <CHECK: "check">
| <CHECK_LAST: "check_last">
| <CHECK_ANY: "check_any">
| <DISPOSED: "disposed">
| <MARK: "mark">
| <MISS: "miss">
| <MARKMSG: "markmsg">
| <MISSMSG: "missmsg">
| <SCENARIO: "scenario">
| <UNIQID: "uniqid">
| <VAR: "var">
| <END: "end">
| <MSG: "message">
| <LOG: "log">
| <FAIL: "fail">
| <CLEAR: "clear">
| <IF: "if">
| <THEN: "then">
| <ELSE: "else">
| <ENDIF: "endif">
| <FOR: "for">
| <IN: "in">
| <LOOP: "loop">
| <ENDLOOP: "endloop">
| <WHILE: "while">
| <READER: "reader">
| <WRITE: "write">
| <READ: "read">
| <CONNECT: "connect">
| <DISCONNECT: "disconnect">
| <EXEC: "execute">
| <CONTROL: "control">
| <SET: "set">
| <COLUMN: "column">
| <GRAPH: "graph">
| <REVERSE_FAIL: "reverse_fail">
| <EXIT: "exit">
}
```

```
<DEFAULT> TOKEN : {
<IDENTIFIER: <LETTER> (<LETTER> | <DIGIT>)*>
| <#LETTER: ["$","A"-"Z","_","a"-"z"]>
| <DIGIT: ["0"-"9"]>
}
```

NON-TERMINALS

```
       Scenario ::= <SCENARIO> <IDENTIFIER> ( InstructionList )? <END>
                    <SCENARIO>
          Macro ::= <MACRO> <IDENTIFIER> "(" ( ArgumentList )? ")" (
                    InstructionList )? <END> <MACRO>
                  | <SCENARIO> <IDENTIFIER> ( InstructionList )? <END>
                    <SCENARIO>
InstructionList ::= ( Instruction )+
    Instruction ::= SendInstruction
                  | RepeatInstruction
                  | PeriodicInstruction
                  | DisposeInstruction
                  | WriteDisposeInstruction
                  | WaitInstruction
                  | WaitabsInstruction
                  | CheckInstruction
                  | CheckLastInstruction
                  | CheckAnyInstruction
                  | DisposedInstruction
                  | MarkInstruction
                  | MarkMsgInstruction
                  | MissInstruction
                  | MissMsgInstruction
                  | CallInstruction
                  | ForInstruction
                  | WhileInstruction
                  | SetInstruction
```

```
                         | VarDeclaration
                         | IfInstruction
                         | MessageInstruction
                         | ClearInstruction
                         | LogInstruction
                         | FailInstruction
                         | ReaderInstruction
                         | WriteInstruction
                         | ReadInstruction
                         | ConnectInstruction
                         | DisconnectInstruction
                         | ExecuteInstruction
                         | ControlInstruction
                         | ColumnInstruction
                         | GraphInstruction
                         | ReverseFailInstruction
                         | ExitInstruction
    ReaderInstruction :: <READER> ( <DISPOSE> )? "(" Constant ( ","
                         = <IDENTIFIER> ( "," Constant ( "," Constant )? )? )?
                           ");"
    ColumnInstruction :: <COLUMN> ( <CLEAR> )? "(" Constant ( "," Constant )?
                         = ");"
     GraphInstruction :: <GRAPH> "(" ParameterList ");"
                         =
   MessageInstruction :: <MSG> "(" <STRING_LITERAL> ( Constant )? ");"
                         =
       LogInstruction :: <LOG> "(" <STRING_LITERAL> ( Constant )? ");"
                         =
      FailInstruction :: <FAIL> "(" <STRING_LITERAL> ( Constant )? ");"
                         =
   ControlInstruction :: <CONTROL> <IDENTIFIER> "." <IDENTIFIER> ( ( "("
                         = ParameterList ( ( ");" ) | ( ")" ";" ) ) ) | ( ";" ) )
     ClearInstruction :: <CLEAR> ";"
                         =
      ExitInstruction :: <EXIT> ( <IF> <FAIL> )? ";"
                         =
      SendInstruction :: <SEND> <IDENTIFIER> ( ( "." <IDENTIFIER> ) )? "(" (
                         = ParameterList )? ");"
    RepeatInstruction :: <REPEAT> <IDENTIFIER> FloatValue IntValue "(" (
                         = ParameterList )? ");"
  PeriodicInstruction :: <PERIODIC> <IDENTIFIER> <IDENTIFIER> FloatValue
                         = IntValue "(" ( ParameterList )? ");"
```

```
         WriteInstruction :: <WRITE> <IDENTIFIER> "." <IDENTIFIER> "(" (
                          =  ParameterList )? ");"
           VarDeclaration :: <VAR> FieldName "=>" Constant ";"
                          =
       DisposeInstruction :: <DISPOSE> <IDENTIFIER> ( ( "." <IDENTIFIER> ) )? "(" (
                          =  ParameterList )? ");"
  WriteDisposeInstruction :: <WRITEDISPOSE> <IDENTIFIER> ( ( "." <IDENTIFIER> ) )?
                          =  "(" ( ParameterList )? ");"
         CheckInstruction :: <CHECK> <IDENTIFIER> ( ( "." <IDENTIFIER> ) )? "(" (
                          =  ChkParameterList )? ");"
     CheckLastInstruction :: <CHECK_LAST> <IDENTIFIER> ( ( "." <IDENTIFIER> ) )?
                          =  "(" ( ChkParameterList )? ");"
      CheckAnyInstruction :: <CHECK_ANY> <IDENTIFIER> ( ( "." <IDENTIFIER> ) )? "("
                          =  ( ChkParameterList )? ");"
          ReadInstruction :: <READ> <IDENTIFIER> "." <IDENTIFIER> "(" (
                          =  ChkParameterList )? ");"
       MarkMsgInstruction :: <MARKMSG> <IDENTIFIER> "." <IDENTIFIER> "(" (
                          =  ChkParameterList )? ");"
       MissMsgInstruction :: <MISSMSG> <IDENTIFIER> "." <IDENTIFIER> "(" (
                          =  ChkParameterList )? ");"
       ConnectInstruction :: <CONNECT> <IDENTIFIER> ( Constant )? ";"
                          =
    DisconnectInstruction :: <DISCONNECT> <IDENTIFIER> ";"
                          =
      DisposedInstruction :: <DISPOSED> <IDENTIFIER> ( ( "." <IDENTIFIER> ) )? "("
                          =  ( ChkParameterList )? ");"
          MissInstruction :: <MISS> <IDENTIFIER> ( ( "." <IDENTIFIER> ) )? "(" (
                          =  ChkParameterList )? ");"
          MarkInstruction :: <MARK> <IDENTIFIER> ( ( "." <IDENTIFIER> ) )? "(" (
                          =  ChkParameterList )? ");"
          CallInstruction :: <CALL> <IDENTIFIER> ( ( "." <IDENTIFIER> ) )? "(" (
                          =  ParameterList )? ");"
           SetInstruction :: <SET> <IDENTIFIER> "(" ( ParameterList )? ")" "(" "("
                          =  ParamHeaderList ")" ParamSetList ");"
          ParamHeaderList :: <IDENTIFIER> ( "," ParamHeaderList )?
                          =
             ParamSetList :: "," ParamSet ( ParamSetList )?
                          =
                 ParamSet :: "(" ParamValueList ")"
                          =
           ParamValueList :: Constant ( "," ParamValueList )?
                          =
            IfInstruction :: <IF> "(" CompareExpression ")" <THEN> InstructionList
                           = ( <ELSE> InstructionList )? <ENDIF> ";"
        CompareExpression :: CalcExpression ( CompareOperator CompareExpression )?
                           =
           CalcExpression :: PrimaryExpression ( CalcOperator CalcExpression )?
                           =
        PrimaryExpression :: Constant
                           =
                           | "(" CompareExpression ")"
          CompareOperator :: "=="
                           =
                           | "!="
                           | ">"
                           | "<"
                           | ">="
                           | "<="
                           | "||"
                           | "&&"
             CalcOperator :: "|"
                           =
                           | "&"
                           | "+"
                           | "-"
                           | "*"
                           | "/"
           ForInstruction :: <FOR> <IDENTIFIER> <IN> ( ( IntValue ".." IntValue ) |
                           = "(" VarList ")" ) <LOOP> InstructionList <ENDLOOP> ";"
         WhileInstruction :: <WHILE> "(" CompareExpression ")" <LOOP>
                           = InstructionList <ENDLOOP> ";"
                  VarList :: Constant ( "," VarList )?
                           =
          WaitInstruction :: <WAIT> "(" Constant ");"
                           =
       WaitabsInstruction :: <WAITABS> "(" Constant ");"
                           =
       ExecuteInstruction :: <EXEC> ( <WAIT> )? ( <LOG> )? <STRING_LITERAL> ";"
                           =
   ReverseFailInstruction :: <REVERSE_FAIL> ";"
                           =
            ParameterList :: Parameter ( "," Parameter )* ( "," )?
                           =
            Parameter :: FieldName "=>" Constant
                       =
     ChkParameterList :: ChkParameter ( "," ChkParameter )* ( "," )?
                       =
         ChkParameter :: FieldName "=>" ( "!" )? Constant ( ":" Constant )?
                       =
         ArgumentList :: Argument ( Argument )*
                       =
             Argument :: FieldName ":" FieldName ( ":=" Constant )? ";"
                       =
            FieldName :: <IDENTIFIER> ( "[" <INTEGER_LITERAL> "]" )? ( ( "."
                       = FieldName ) )?
             IntValue :: <INTEGER_LITERAL>
                       =
                       | "<<" <IDENTIFIER>
                       | <IDENTIFIER>
           FloatValue :: <FLOATING_POINT_LITERAL>
                       =
                       | "<<" <IDENTIFIER>
                       | <IDENTIFIER>
             Constant :: <INTEGER_LITERAL>
                       =
                       | <FLOATING_POINT_LITERAL>
                       | <CHARACTER_LITERAL>
                       | <STRING_LITERAL>
                       | ">>" <IDENTIFIER>
                       | ">>" <JAVASCRIPT>
                       | "<<" <IDENTIFIER> ( "." <IDENTIFIER> )?
                       | <IDENTIFIER>
                       | <UNIQID>
                       | <JAVASCRIPT>
```

[End]