

# **mitesterforsip\_User\_Guide**

Release Version 2.1

## Table of Contents

Preface .....	3
Foreword.....	3
This Document is for.....	3
Acknowledgement.....	3
Document contains .....	3
[1] Introduction .....	4
[1.1] What is mitester for SIP? .....	4
[1.2] Features of mitester for SIP.....	4
[1.3] Why mitester for SIP .....	4
[1.4] Platform to run.....	4
[1.5] Free License.....	4
[1.6] Mode to get mitester for SIP .....	5
[2] Installation.....	5
[2.1] Pre-requisites for Installation .....	5
[2.2] Installation of mitester for SIP.....	5
[3] USER mode - mitester for SIP .....	7
[3.1] Primary Files and Folders .....	7
[3.2] Run USER mode - mitester for SIP.....	12
[4] ADVANCED mode - mitester for SIP .....	12
[4.1] Primary Files and Folders .....	12
[4.2] SUT Installation.....	16
[4.3] Getting ready for the first run.....	18
[4.4] Run ADVANCED mode - mitester for SIP .....	21
[5] Transition among the screens/windows.....	22
[6] Termination of TestExecution.....	25
[7] Server Modes – Proxy and B2BUA .....	25
[8] Log Files .....	27
[9] Updating Test Results.....	27
[10] Known Issues .....	27
[11] Appendices.....	28
[11.1] How to develop the interface for Automation of any SIP Application.....	28
[11.2] Configuring eclipse.....	29
[12] Wishlist .....	31

## Preface

### Foreword

mitester for SIP is an automation framework to test and verify the call flows from simple to complex for any SIP Application. This document will guide users to use this framework for automating the SIP call flows.

### This Document is for

This document will aid users to install mitester for SIP and test run the system under test on every possible ways of call flows. This tool is confined to the RFC standards of IETF.

Upon reading this document users will be able to install and automate call flows.

### Acknowledgement

We would like to thank the mitester for SIP team members for the assistance and support for releasing the framework.

### Document contains

The document contains few sign boards for users to work with mitester for SIP.



Warning. Care should be taken on working with this process



Note. This point should be noted for future reference



Tip. It is a helpful tip

## **[1] Introduction**

### **[1.1] What is mitester for SIP?**

mitester for SIP is a tool designed and developed for easy testing of any SIP application. It makes the process of development, deployment and maintenance of automated testing more controllable. mitester for SIP works on two modes: USER mode and ADVANCED mode of call flows execution, B2BUA & Proxy.

### **[1.2] Features of mitester for SIP**

mitester for SIP architectural model uses the globally accepted SIP Stack and Server programming that offers a range of constructs from low level to very high level interfaces to control and test every aspect of SIP call flows.

Design of this framework includes achieving the test result from simple to complex call flows for all SIP applications. USER and ADVANCED modes of testing is incorporated in this framework which makes the testing process simple.

Simple syntax of Client scripts and server scripts in XML format makes the simulation of call flows easier. mitester for SIP supports RFC standards –

RFC 3261, RFC 3515, RFC 2976, RFC 3428, RFC 3265, RFC 3262, RFC 3311, RFC 3903, RFC 3455.

### **[1.3] Why mitester for SIP**

mitester for SIP is the right solution for those people who are work effective and time conscious. The remarkable advantage of mitester for SIP is that we can automate the testing of any SIP Application. Once the client and server scripting are done, the test execution can be done multiple times on all releases.

### **[1.4] Platform to run**

mitester for SIP is tested on Windows XP , Ubuntu LINUX, MAC and Solaris platforms.

### **[1.5] Free License**

mitester for SIP is an open source software project, and is released under the GNU General Public License (GPL). All source code is freely available under the GPL. Users are welcome to modify mitester for SIP to suit their own needs, and it would be appreciated if the user wishes to contribute their improvements back to the mitester for SIP team.

## [1.6] Mode to get mitester for SIP

Currently, mitester for SIP runs on Windows XP, Ubuntu Linux, MAC and Solaris environment.

This part of the document will show the user how to obtain source and binary packages, and how to build mitester for SIP from source.

The following are the general steps:

1. Download the relevant binaries for your needs.
2. Build the source into a binary executable, if downloaded the source.  
This may involve building and/or creating package as mentioned thereof.  
Supported extensions and libraries have to be included as per the instructions.

## [2] Installation

### [2.1] Pre-requisites for Installation

mitester for SIP works in both Java JDK 1.5 & JDK 1.6.

Any installed version of SIP application. (system under test).

References:

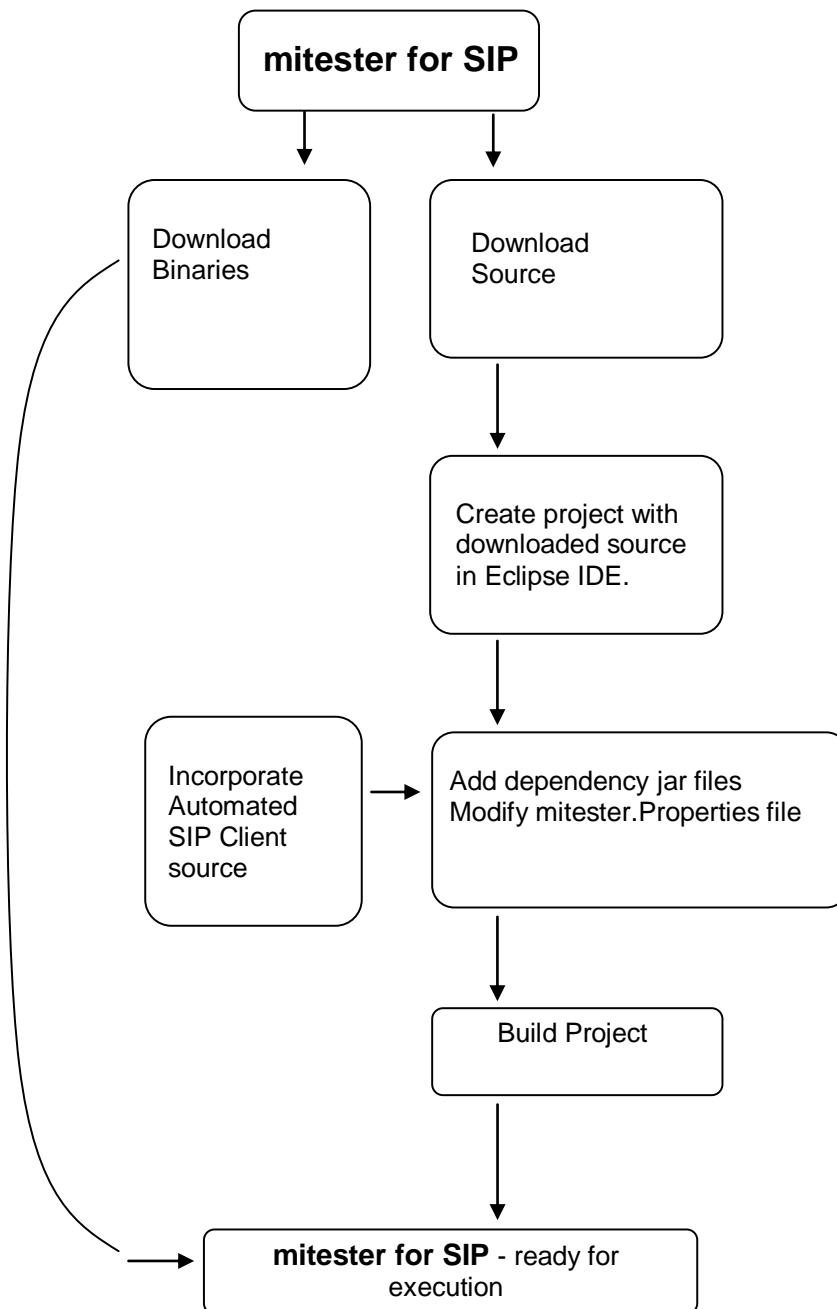
- Java  
<http://java.sun.com/javase/downloads/index.jsp>
- System under test (for example)  
<http://www.sip-communicator.org/index.php/Main/Download>

### [2.2] Installation of mitester for SIP

mitester for SIP installation is very simple. The following are the simple ways to begin with mitester for SIP.

- (i) Using the binary executable
- (ii) Using the source and build to binaries package adding required extension libraries

Flow to start mitester for SIP :



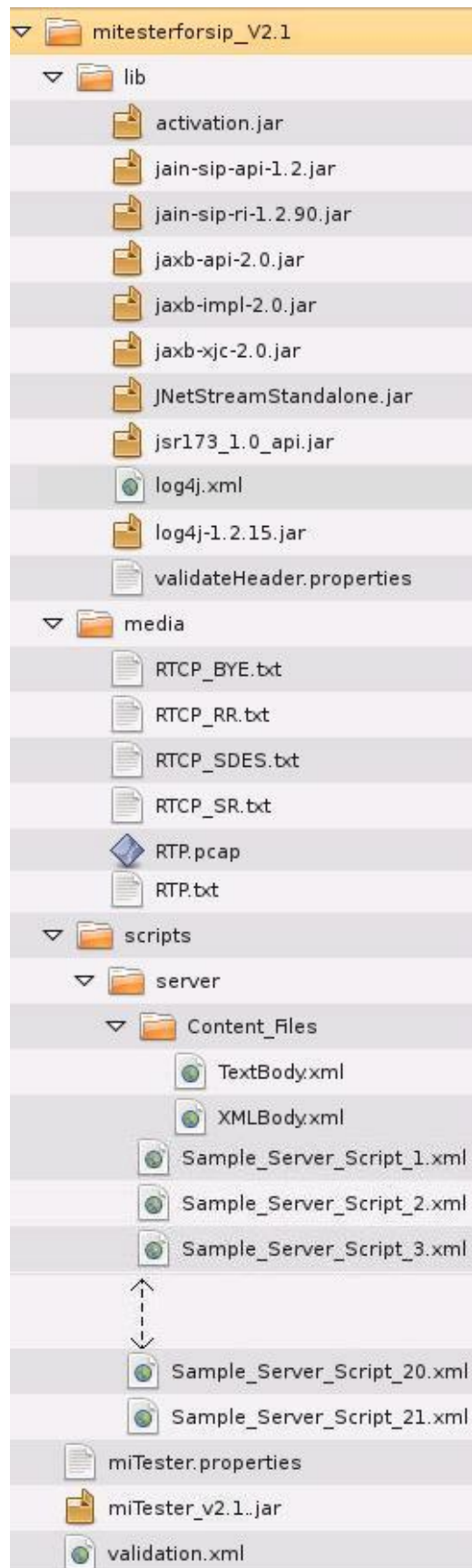
## [3] USER mode - mitester for SIP

mitester for SIP is developed to process, build and control the call flows which simulate the SIP messages to the System Under Test (SUT).

### [3.1] Primary Files and Folders

To work with USER mode of mitester for SIP, download the binaries of your platform and you can find “mitesterforsip\_V2.1” inside the package. Click [here](#) to download. Then click “Download” to move to the target page.

The folder structure of **mitesterforsip\_V2.1** – USER mode is as follows





## mitesterforsip\_V2.1 – USER mode – Folder structure

- 'lib' folder – Holds dependent Jar Files and Properties Files
  - activation.jar
  - jain-sip-api-1.2.jar
  - jain-sip-ri-1.2.90.jar
  - jaxb-api-2.0.jar
  - jaxb-impl-2.0.jar
  - jaxb-xjc-2.0.jar
  - JNetStreamStandalone.jar
  - jsr173\_1.0\_api.jar
  - log4j-1.2.15.jar
  - log4j.xml
  - validateHeader.properties
- 'scripts' folder – Holds Sample server scripts (Sample test scripts for execution)
- 'media' folder – Holds media files, either RTP/RTCP. It is OPTIONAL. It can be included if user adds media test case in the scripts.  
Refer "mitesterforsip\_Developing\_Test\_Scripts-V2.1.pdf", section 3.1.5.
- 'Content\_Files' folder – Holds the message bodies. The message bodies can be either XML body/Text body. It is OPTIONAL. It can be included if user wants to add XML/Text bodies in the scripts.  
Refer "mitesterforsip\_Developing\_Test\_Scripts-V2.1.pdf", section 3.1.1.
- 'validation.xml' – the file to be edited by the user, to validate the headers presence and headers syntax.  
Refer "mitesterforsip\_Developing\_Test\_Scripts-V2.1.pdf", section 4.
- mitester.properties – the file to be edited by the user. This file carries the following primary settings.
  - MITESTER\_MODE: Running mode of mitester. Set as USER. It is mandatory.

MITESTER\_MODE = USER

- SUT\_IP\_ADDRESS: IP address of System Under Test (SIP client). It is mandatory.

SUT\_IP\_ADDRESS = 127.0.0.1

- SCRIPT\_PATH\_MITESTER: This path defines the folder that holds the server scripts with Call flows. Call flows can span to one or more scripts for test execution. It is mandatory.

SCRIPT\_PATH\_MITESTER = ./scripts/server

- MITESTER\_DELAY: The time interval (in seconds) that mitester takes to send successive request / response messages. It is configurable and is optional.

MITESTER\_DELAY = 0.4

- SUT\_RTP\_PORT: The port at which mitester listens RTP packets. It is mandatory while checking media call flows with mitester.

SUT\_RTP\_PORT = 5000

- MITESTER\_RTP\_PORT: System Under Test (SIP client) RTP port. It is mandatory while checking media call flows with mitester.

MITESTER\_RTP\_PORT = 8000

- SUT\_SIP\_PORT: System Under Test (SIP client) SIP port. It is optional, configurable and the default value is 5060. It is optional.

SUT\_SIP\_PORT = 5060

- MITESTER\_SIP\_PORT: The port at which mitester listens SIP messages. It is optional, configurable and the default value is 5070. It is optional.

MITESTER\_SIP\_PORT = 5070

- VALIDATION: On enabling, mitester performs validation of incoming SIP messages. By default the validation is based on regular expression. It is optional.

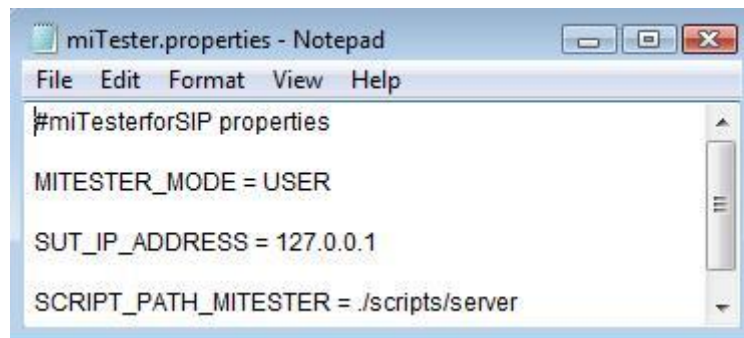
VALIDATION = YES / NO

- CHECK\_PRESENCE\_OF\_HEADER: On enabling, mitester checks the presence of specific headers in the incoming SIP messages. It is optional. The headers to be checked are mentioned in the "validation.xml" file.

CHECK\_PRESENCE\_OF\_HEADER = YES / NO

- VALIDATION\_FILE\_PATH: Path of Validation file. It is mandatory when "VALIDATION " and "CHECK\_PRESENCE\_OF\_HEADER" are enabled.

VALIDATION\_FILE\_PATH = validation.xml



```

#miTesterforSIP properties

MITESTER_MODE = USER

SUT_IP_ADDRESS = 127.0.0.1

SCRIPT_PATH_MITESTER = ./scripts/server

```

**USER mode** - mitester.properties file (mandatory fields only)

quick understanding of the inputs of mitester.properties file.

TEST MODE	INPUTS	MANDATORY	OPTIONAL
USER	MITESTER_MODE	✓	
	SUT_IP_ADDRESS	✓	
	SCRIPT_PATH_MITESTER	✓	
	MITESTER_DELAY		✓
	SUT_RTP_PORT		✓
	MITESTER_RTP_PORT		✓
	SUT_SIP_PORT		✓
	MITESTER_SIP_PORT		✓
	VALIDATION		✓
	CHECK_PRESENCE_OF_HEADER		✓
	VALIDATION_FILE_PATH		✓

- mitester\_V2.1.jar (The Jar file can even be built from the source)



mitester for SIP svn repository URL:  
<https://mitesterforsip.svn.sourceforge.net/svnroot/mitesterforsip>



mitester for SIP can be used to simulate any SIP call flow with any SIP application

### [3.2] Run USER mode - mitester for SIP

Navigate to the folder where mitester\_V2.1.jar resides in the command prompt.  
Provide the following command in the command prompt.

```
java -jar mitester_V2.1.jar
```

where 'mitester\_V2.1.jar' is the name of the binary executable.

For example: g:\mitesterforsip\_V2.1> java -jar mitester\_V2.1.jar



The Test execution results will be displayed on the Command prompt window and also generated as a text file - "TestResult.txt" in the current working directory.



The log information related to the test execution is generated as "mitester.log" in the current working directory.



After starting mitester for SIP, start any SIP application and perform actions to accomplish the expected call flows written in the server scripts.



To write your own server scripts, refer "mitesterforsip\_Developing\_Test\_Scripts-V2.1.pdf".  
Click [here](#) to download.

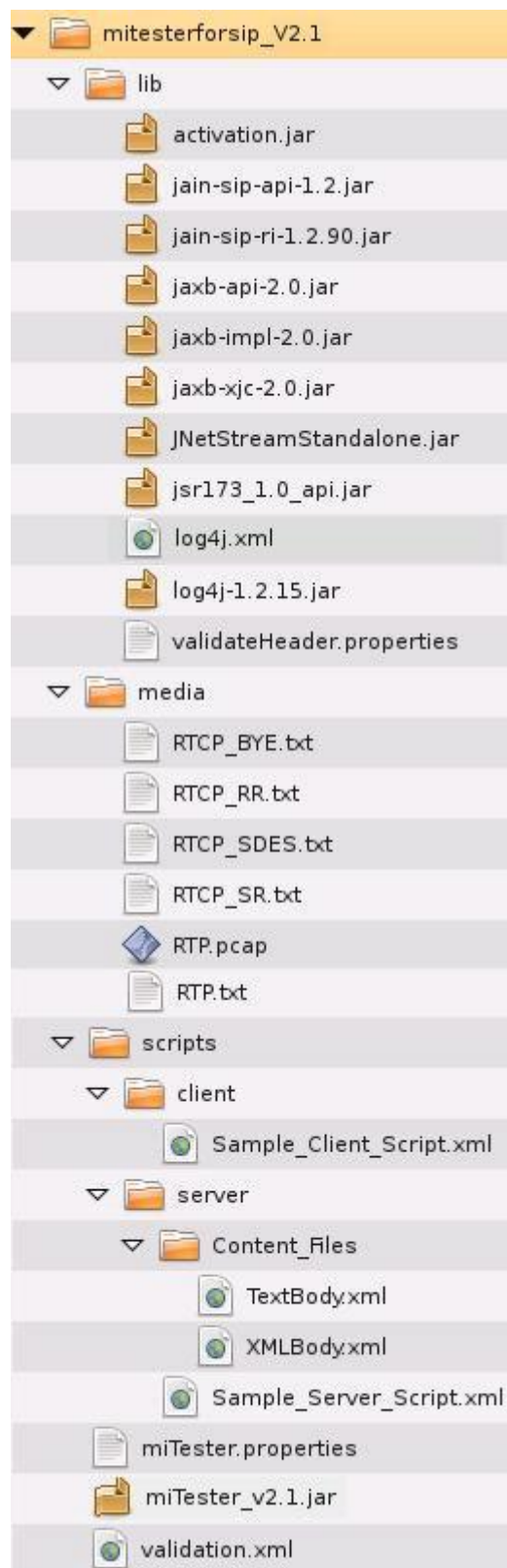
## [4] ADVANCED mode - mitester for SIP

We provide you a case study on ADVANCED mode of testing using SIP Communicator and our mitester for SIP. In this mode, the test executions are automated. At a click of a button, the test executions complete producing the test reports and test logs.

### [4.1] Primary Files and Folders

To work with ADVANCED mode of mitester for SIP, download the binaries of your platform and you can find "mitesterforsip\_V2.1" inside the package. Click [here](#) to download. Then click "Download" to move to the target page.

Slightly alter the folder structure of **mitesterforsip\_V2.1** to run the ADVANCED mode of mitester for SIP.



**mitesterforsip\_V2.1 – ADVANCED mode – Altered Folder structure**

- 'lib' folder – Holds dependent Jar Files
  - activation.jar
  - jain-sip-api-1.2.jar
  - jain-sip-ri-1.2.90.jar
  - jaxb-api-2.0.jar
  - jaxb-impl-2.0.jar
  - jaxb-xjc-2.0.jar
  - JNetStreamStandalone.jar
  - jsr173\_1.0\_api.jar
  - log4j-1.2.15.jar
  - log4j.xml
  - validateHeader.properties
- Folders on any name and at any path can be defined to hold the Client and Server scripts. Any form of uniformity can be maintained on dealing with folders of scripts. 'scripts' folder contains Sample\_Client\_Script.xml inside Client folder and Sample\_Server\_Script.xml inside Server folder.

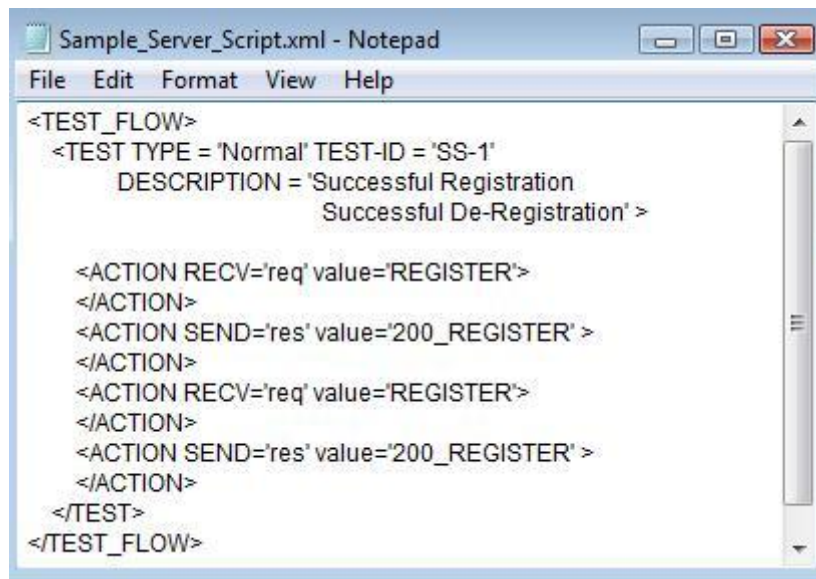


Download the binaries of your platform and you can find "mitesterforsip\_V2.1\_Test\_Scripts" inside the package

```
Sample_Client_Script.xml - Notepad
File Edit Format View Help
<TEST_FLOW>
  <TEST TYPE = 'Normal' TEST-ID = 'SS-1'
    DESCRIPTION = ' Successful Registration
                    Successful De-Registration' >

    <ACTION SEND = 'req' value = 'REGISTER' >
      <PARAM type = 'uri' value = 'Test1@127.0.0.1' />
      <PARAM type = 'password' value = 'Test' />
      <PARAM type = 'expires' value = '3600' />
      <PARAM type = 'serverport' value = '5070' />
      <PARAM type = 'proxyport' value = '5070' />
    </ACTION>
    <ACTION RECV='res' value='REGISTERED' />
    <ACTION SEND = 'req' value = 'DE-REGISTER' />
    <ACTION RECV = 'res' value = 'DE-REGISTERED' />
  </TEST>
</TEST_FLOW>
```

Sample Client Script

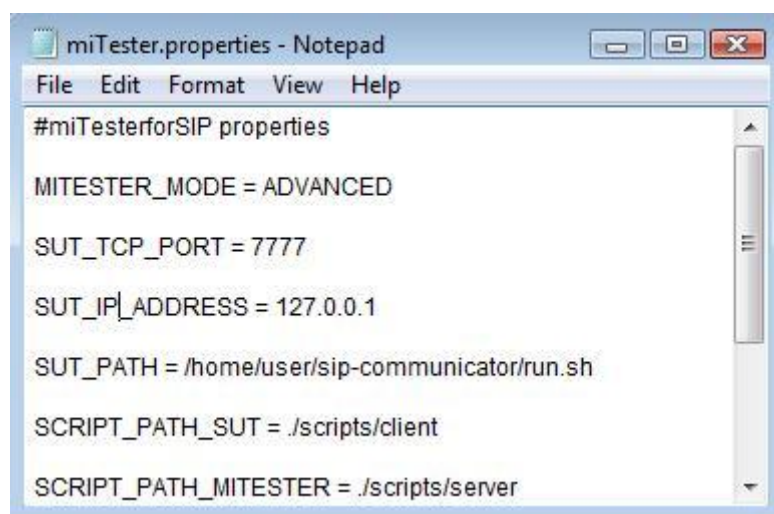


```
<TEST_FLOW>
  <TEST TYPE = 'Normal' TEST-ID = 'SS-1'
    DESCRIPTION = 'Successful Registration
                  Successful De-Registration' >

    <ACTION RECV='req' value='REGISTER'>
    </ACTION>
    <ACTION SEND='res' value='200_REGISTER' >
    </ACTION>
    <ACTION RECV='req' value='REGISTER'>
    </ACTION>
    <ACTION SEND='res' value='200_REGISTER' >
    </ACTION>
  </TEST>
</TEST_FLOW>
```

Sample Server Script

- mitester.properties - file used to define the primary settings of mode of the test execution, port engaged, the path where the client and server scripts of the test run resides, the path of client application.



```
#miTesterforSIP properties

MITESTER_MODE = ADVANCED

SUT_TCP_PORT = 7777

SUT_IP_ADDRESS = 127.0.0.1

SUT_PATH = /home/user/sip-communicator/run.sh

SCRIPT_PATH_SUT = ./scripts/client

SCRIPT_PATH_MITESTER = ./scripts/server
```

ADVANCED mode - mitester.properties file

- mitester\_V2.1.jar (The Jar file can be built from the source)



mitester for SIP svn repository URL:  
<https://mitesterforsip.svn.sourceforge.net/svnroot/mitesterforsip>

## [4.2] SUT Installation

Here the SUT is sip communicator (for example).

For Windows:

Download mitesterforsip\_v2.1\_windows.zip and you can find  
“SUT-sip-communicator-1.0-alpha3-nightly.build.1844.exe” inside the package.  
Click [here](#) to download. Then click “Download” to move to the target page.

For Linux:

Download mitesterforsip\_v2.1\_linux.zip and you can find  
“SUT-sip-communicator-1.0-alpha3-nightly.build.1844-linux.bin” inside the package.  
Click [here](#) to download. Then click “Download” to move to the target page.

For Solaris

Download mitesterforsip\_v2.1\_solaris.zip and you can find  
“SUT-sip-communicator-1.0-alpha3-nightly.build.1844.jar” inside the package.  
Click [here](#) to download. Then click “Download” to move to the target page.

For MAC:

Download mitesterforsip\_v2.1\_mac.zip and you can find  
“SUT-sip-communicator-1.0-alpha3-nightly.build.1844.jar” inside the package.  
Click [here](#) to download. Then click “Download” to move to the target page.

For Solaris Users:



After installing sip-communicator, edit the following line in “run.sh”  
file for successful test execution.

Line to be modified : export PATH=\$PATH:native  
Modified Line : PATH=\$PATH:native export PATH

For MAC Users:



In USER or ADVANCED mode, while using sip-communicator as SUT, you  
need to freshly add the 'native' folder inside the installed SIP communicator  
folder. This should be done after installing the provided jar for MAC. This is  
very essential for successful session completion in a test scenario.

Download mitesterforsip\_v2.1\_mac.zip and you can find  
“SUT\_MAC\_Native\_sip-communicator-1.0-alpha3-nightlybuild.1844”  
inside the package. Click [here](#) to download. Then click “Download” to move to  
the target page.

In MAC, SIP communicator works under **Java JDK 1.5** only





MAC\_Native\_sip-communicator-1.0-alpha3-nightly.build.1844 – folder structure

SUT Interface (For Windows, Linux, MAC and Solaris):

Download the binaries of your platform and you can find

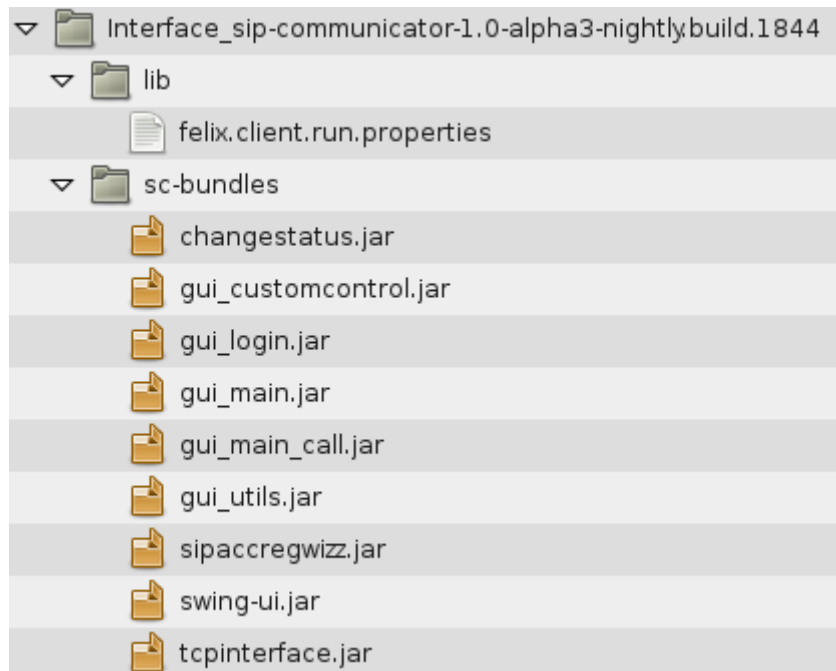
“Interface\_sip-communicator-1.0-alpha3-nightly.build.1844” inside the packages.

Click [here](#) to download. Then click “Download” to move to the target page.

After SUT installation, prepare your SUT for ADVANCED mode - mitester for SIP execution.

Steps to be followed:

1. 'lib' folder – Holds felix.client.run.properties ( File to be replaced inside the lib folder of sip communicator, where it is installed )
2. 'sc-bundles' folder - Holds jar files ( Files to be replaced inside the sc-bundles folder of sip communicator, where it is installed )



Interface\_sip-communicator-1.0-alpha3-nightly.build.1844 – folder structure

### [4.3] Getting ready for the first run

Now the user is ready to execute the first run. Clearly provide input to the mitester.properties file. As said earlier this file will carry the inputs of

MITESTER\_MODE: Running mode of mitester. Set as ADVANCED. It is mandatory.

MITESTER\_MODE = ADVANCED

SUT\_TCP\_PORT: The port assigned to engage SIP transaction during execution of a Test Scenario. It is mandatory and configurable. By default, the value is 7777.

SUT\_TCP\_PORT = 7777

SUT\_IP\_ADDRESS: IP address of System Under Test (SIP client). It is mandatory.

SUT\_IP\_ADDRESS = 127.0.0.1

SUT\_PATH: Path where the system under test binary resides. It is mandatory.

SUT\_PATH = /home/user/sip-communicator/run.sh

SCRIPT\_PATH\_SUT: Path of the folder that holds the client scripts with Call flows. Call flows can span to one or more scripts for test execution. It is mandatory.

SCRIPT\_PATH\_SUT = ./scripts/client

SCRIPT\_PATH\_MITESTER: Path of the folder that holds the server scripts with Call flows. Call flows can span to one or more scripts for test execution. It is mandatory.

SCRIPT\_PATH\_MITESTER = ./scripts/server

MITESTER\_DELAY: The time interval (in seconds) that mitester takes to send successive request / response messages. It is configurable and is optional.

MITESTER\_DELAY = 0.4

SUT\_RTP\_PORT: The port at which mitester listens RTP packets. It is mandatory while checking media call flows with mitester.

SUT\_RTP\_PORT = 5000

MITESTER\_RTP\_PORT: System Under Test (SIP client) RTP port. It is mandatory while checking media call flows with mitester.

MITESTER\_RTP\_PORT = 8000

SUT\_SIP\_PORT: System Under Test (SIP client) SIP port. It is optional, configurable and the default value is 5060. It is optional.

SUT\_SIP\_PORT = 5060

MITESTER\_SIP\_PORT: The port at which mitester listens SIP messages. It is optional, configurable and the default value is 5070. It is optional.

MITESTER\_SIP\_PORT = 5070

SUT\_WAIT\_TIME: The maximum time after which SUT is forcefully closed. It is optional and configurable. By default the value is 35 secs.

SUT\_WAIT\_TIME = 35

MITESTER\_WAIT\_TIME: Optional. The maximum time after which mitester is forcefully closed. It is optional and configurable. By default the value is 60 secs.

MITESTER\_WAIT\_TIME = 60

TEST\_INTERVAL: The time interval (in seconds) taken between the execution of successive test scenarios. It is optional and configurable. By default, the value is 2 secs.

Note: A minimum of 2 secs is recommended.

TEST\_INTERVAL = 2

VALIDATION: On enabling, mitester performs validation of incoming SIP messages. By default the validation is based on regular expression. It is optional.

VALIDATION = YES / NO

CHECK\_PRESENCE\_OF\_HEADER: On enabling, mitester checks the presence of specific headers in the incoming SIP messages. It is optional. The headers to be checked are mentioned in the “validation.xml” file.

CHECK\_PRESENCE\_OF\_HEADER = YES / NO

VALIDATION\_FILE\_PATH: Path of Validation file. It is mandatory when “VALIDATION “ and “CHECK\_PRESENCE\_OF\_HEADER” are enabled.

VALIDATION\_FILE\_PATH = validation.xml

For quick understanding of the inputs of mitester.properties file.

TEST MODE	INPUTS	MANDATORY	OPTIONAL
ADVANCED	MITESTER_MODE	✓	
	SUT_TCP_PORT	✓	
	SUT_IP_ADDRESS	✓	
	SUT_PATH	✓	
	SCRIPT_PATH_SUT	✓	
	SCRIPT_PATH_MITESTER	✓	
	MITESTER_DELAY		✓
	SUT_RTP_PORT		✓
	MITESTER_RTP_PORT		✓
	SUT_SIP_PORT		✓
	MITESTER_SIP_PORT		✓
	SUT_WAIT_TIME		✓
	MITESTER_WAIT_TIME		✓
	TEST_INTERVAL		✓
	VALIDATION		✓
	CHECK_PRESENCE_OF_HEADER		✓
	VALIDATION_FILE_PATH		✓

#### [4.4] Run ADVANCED mode - mitester for SIP

After completing all the previous mentioned steps, navigate to the folder where mitester\_V2.1.jar resides in the command prompt.

Provide the following command in the command prompt.

```
java -jar mitester_V2.1.jar
    where 'mitester_V2.1.jar' is the name of the binary executable.
```

For example: g:\mitesterforsip\_V2.1> java -jar mitester\_V2.1.jar



The Test execution results will be displayed on the Command prompt window and also generated as a text file - "TestResult.txt" in

the current working directory.



The log information related to the test execution is generated as "mitester.log" in the current working directory.

## [5] Transition among the screens/windows

mitester will run in Normal Window by default. Traversal to different windows (Log Window, Call Flow Window, Normal Window, Result Window and Help Window) is possible by using the following keywords.

Logwindow: l + ENTER

```

C:\> Command Prompt - java -jar miTester_1206D.jar
miTesterforSIP running in USER mode
SS-1 Started
waiting for.....REGISTER
l
REGISTER sip:127.0.0.1 SIP/2.0
Call-ID: 42eb2113649649f21ad695081d4e73aa00:0:0:0:0:0:0
CSeq: 3 REGISTER
From: <sip:s0127.0.0.1>;tag=5383eeb7
To: <sip:s0127.0.0.1>
Via: SIP/2.0/UDP 200.201.202.124:5060;branch=z9hG4bK7b146460370383f5ac699f40f1328df4
Max-Forwards: 70
User-Agent: SIP Communicator1.0-alpha3-nightly.build.1844Windows Vista
Expires: 3600
Contact: 's' <sip:s0200.201.202.124:5060;transport=udp;registering_acc=127_0_0_1>;expires=3600
Content-Length: 0

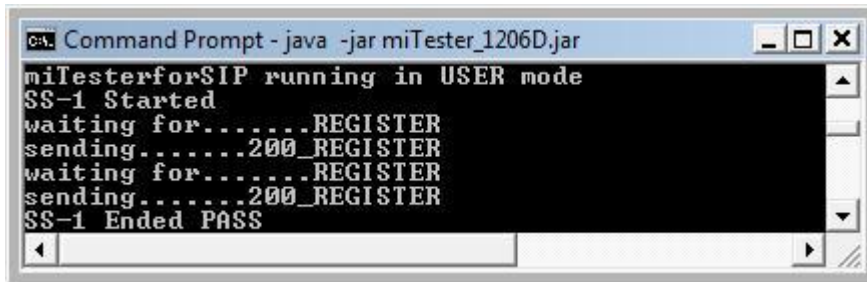
sending.....200_REGISTER
  
```

Call Flow Window: c + ENTER

```

C:\> Command Prompt - java -jar miTester_1206D.jar
miTesterforSIP running in USER mode
SS-1 Started
waiting for.....REGISTER
c
          REGISTER      miTester      SUT
          200_REGISTER  |<-----|
          REGISTER     |----->|
          200_REGISTER  |<-----|
          200_REGISTER  |----->|
SS-1 Ended PASS
  
```

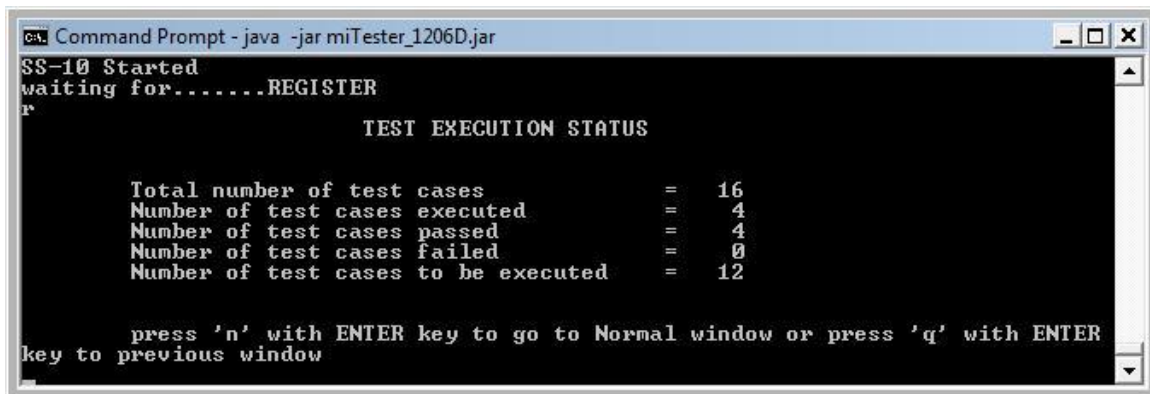
Normal Window: n + ENTER



```

C:\> Command Prompt - java -jar miTester_1206D.jar
miTesterforSIP running in USER mode
SS-1 Started
waiting for.....REGISTER
sending.....200_REGISTER
waiting for.....REGISTER
sending.....200_REGISTER
SS-1 Ended PASS
  
```

Result Window: r + ENTER



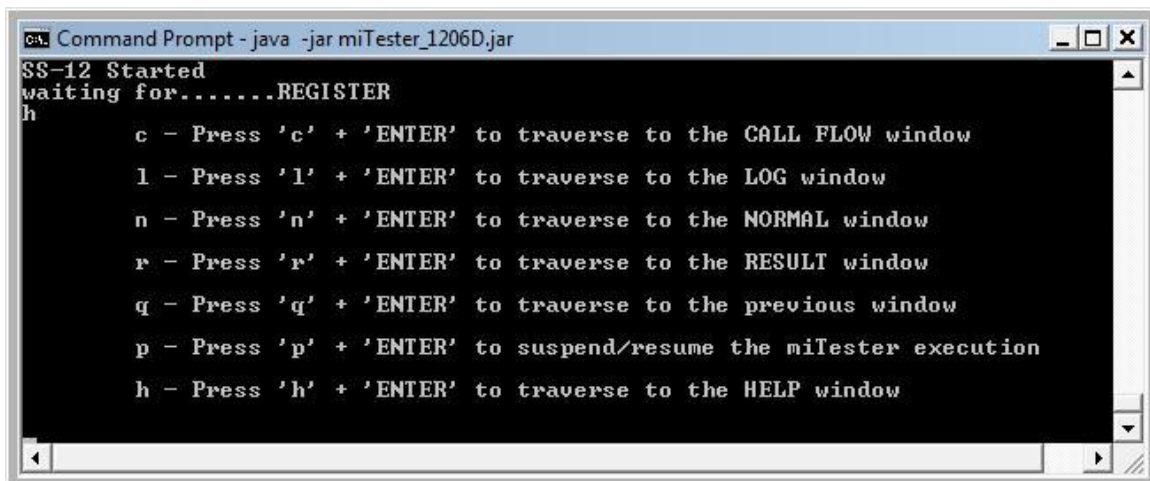
```

C:\> Command Prompt - java -jar miTester_1206D.jar
SS-10 Started
waiting for.....REGISTER
r
                                TEST EXECUTION STATUS

Total number of test cases      = 16
Number of test cases executed   = 4
Number of test cases passed     = 4
Number of test cases failed     = 0
Number of test cases to be executed = 12

press 'n' with ENTER key to go to Normal window or press 'q' with ENTER
key to previous window
  
```

Help Window: h + ENTER



```

C:\> Command Prompt - java -jar miTester_1206D.jar
SS-12 Started
waiting for.....REGISTER
h
c - Press 'c' + 'ENTER' to traverse to the CALL FLOW window
l - Press 'l' + 'ENTER' to traverse to the LOG window
n - Press 'n' + 'ENTER' to traverse to the NORMAL window
r - Press 'r' + 'ENTER' to traverse to the RESULT window
q - Press 'q' + 'ENTER' to traverse to the previous window
p - Press 'p' + 'ENTER' to suspend/resume the miTester execution
h - Press 'h' + 'ENTER' to traverse to the HELP window
  
```

In all these windows by pressing q + ENTER, the previous window will be displayed. In addition to this the test execution can be paused/resumed and terminated by using the following keywords;

p + ENTER – To pause/resume the execution



```
Command Prompt - java -jar miTester_V_2.0.jar
miTesterforSIP running in USER mode
SS-1 Started
waiting for.....REGISTER
sending.....200_REGISTER
waiting for.....REGISTER
sending.....200_REGISTER
SS-1 Ended PASS
SS-2 Started
waiting for.....REGISTER
p
Press 'p' + ENTER key to resume the execution
p
Test execution RESUMED
sending.....200_REGISTER
waiting for.....REGISTER
sending.....200_REGISTER
SS-2 Ended PASS
SS-3 Started
waiting for.....REGISTER
```



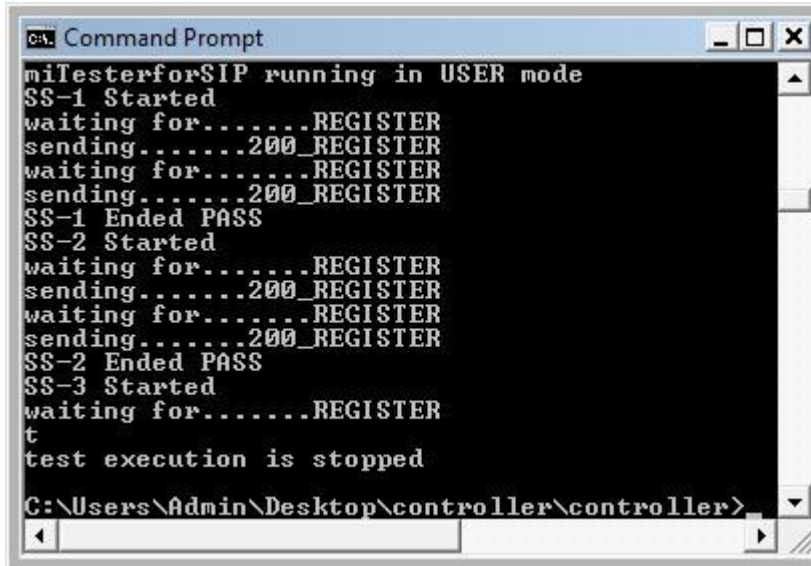
Keywords are Case-Insensitive. All keywords work in dynamic mode except Help keyword, which works in Static mode also.  
To get the Help Window, either h/-h/--help + ENTER can be used.



## [6] Termination of TestExecution

Test execution can be terminated in the middle.

On pressing 't + ENTER', the test execution will be terminated and the results, log will get updated.



```
Command Prompt
mitesterforSIP running in USER mode
SS-1 Started
waiting for.....REGISTER
sending.....200_REGISTER
waiting for.....REGISTER
sending.....200_REGISTER
SS-1 Ended PASS
SS-2 Started
waiting for.....REGISTER
sending.....200_REGISTER
waiting for.....REGISTER
sending.....200_REGISTER
SS-2 Ended PASS
SS-3 Started
waiting for.....REGISTER
t
test execution is stopped
C:\Users\Admin\Desktop\controller\controller>
```

## [7] Server Modes – Proxy and B2BUA

mitesterforSIP can behave as a Proxy and B2BUA. The behavior is achieved by enabling SERVER\_MODE property in the mitester.properties file with the keyword B2BUA or PROXY. Following changes are needed in mitester.properties file and in the folder structure to work as Proxy/B2BUA.

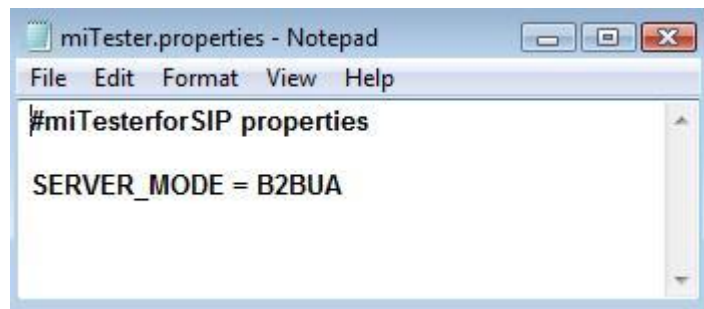
mitester.properties file:

SERVER\_MODE: On enabling, mitester behaves as either PROXY or B2BUA based on the issued keyword. It is Mandatory.

SERVER\_MODE = PROXY/B2BUA

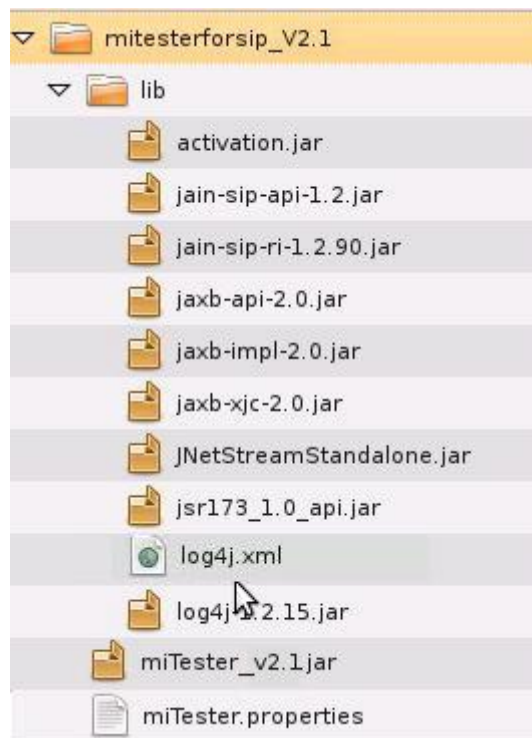
AUTHENTICATION: On enabling, mitester checks for authentication. It is Optional.

AUTHENTICATION = YES / NO



**Server Mode – B2BUA** - mitester.properties file (mandatory fields only)

Folder Structure:



**mitesterforsip\_V2.1** – mitester – B2BUA/PROXY Server Mode – Folder structure

## [8] Log Files

Log files are derived during the execution of tool. Log files play a major role in analyzing the results obtained after test execution of the scenarios completely. A log file shows the transaction information along with relevant exceptions thrown during the course of execution.



Logs can be customizable by log4j.xml which is available in the lib folder.

## [9] Updating Test Results

Another derivative of Test execution is the Test result. Test result shows the outcome of the Test case execution through which one can ascertain the working of the system under test.

## [10] Known Issues

- Currently the mitester for SIP ADVANCED mode will yield a minor setback in consistency of Results. This inconsistency is noticed due to the SUT performance. Overcoming this setback can be done by tuning the time values of transaction in mitester.properties file with below mentioned key.

eg :

**SERVER\_DELAY = 0.4**

( i.e mitester simulator maintains 0.4 seconds of interval between outgoing consecutive requests or responses, which will reduce the message traffic and yields better results.)

- No error is thrown while adding parameter name alone (without value) in the script, for those not maintained in the config file.
- No error is thrown while removing a header which is not present in the generated request / response.

## [11] Appendices

### [11.1] How to develop the interface for Automation of any SIP Application

#### - *ADVANCED MODE*

For automating the execution of any SIP application, an interface layer needs to be placed above the user interface layer of the application. An interface layer acting as a bridge between the Automation framework (which automates the test execution) and system under test through which actions are initiated by the framework.

As a case-study, SIP Communicator is automated and the procedure is described below:

- 1) Created the 'tcpinterface' and 'changestatus' packages under the 'net\java\sip\communicator\impl' directory structure of sip-communicator source.

#### **net.java.sip.communicator.impl.tcpinterface**

This package holds logic in which it initiates the desired actions according to the instruction received from the test scripts and sends back the message on change of status.

#### **TcpInterfaceActivator.java**

This class supports the sip communicator UI, OperationSetBasicTelephony, and AccountRegistrationWizard Services.

#### **TcpInterfaceServiceImpl.java**

This class parses the messages received and initiates the necessary actions. Also sends messages based on the change of status.

#### **CallEventHandler.java**

This class implements CallListener interface and sends the message based on the change of status of incoming and outgoing calls

#### **IncomingHandler.java**

This class implements CallChangeListener interface and sends the message based on the change of status of incoming call

#### **OutgoingCallEventHandler.java**

This class implements CallParticipantListener interface and sends the message based on the change of status of outgoing call

#### **RegisterEventHandler.java**

This class implements RegistrationStateChangeListener interface and sends the message based on the change of status of registration

#### **ListenerException.java**

This is an Exception handles the custom exception

#### **tcpinterface.manifest.mf**

Contains list of supported packages.

#### **net.java.sip.communicator.impl.changestatus**

This package sets the status of user accounts to 'offline' if already exists.

**ChangeStatusActivator.java**

This class retrieves the user account status and sets to 'offline'.

**changestatus.manifest.mf**

Contains list of supported packages.

2) 'net.java.sip.communicator.impl.tcpinterface' requires services from the following packages; bundles created for these packages:

**net.java.sip.communicator.impl.gui.main,**

GuiMainActivator.java and guimain.manifest.mf files are created and included under this directory.

**net.java.sip.communicator.impl.gui.customcontrols,**

GuiCustomControlActivator.java and guicustomcontrol.manifest.mf files are created and included under this directory.

**net.java.sip.communicator.impl.gui.utils,**

GuiUtilsActivator.java and guiutils.manifest.mf files are created and included under this directory

**net.java.sip.communicator.impl.gui.main.call,**

GuiMainCallActivator.java and guimaincall.manifest.mf files are created and included under this directory

**net.java.sip.communicator.impl.gui.main.login,**

LoginManagerActivator.java and guilogin.manifest.mf files are created and included under this directory

3) Exported the services of following packages

**net.java.sip.communicator.impl.gui,**

Added the following in **swing.ui.manifest.mf** file

"Export-Package: net.java.sip.communicator.impl.gui.GuiActivator,

**net.java.sip.communicator.plugin.sipaccregwizz.**

Added the following in **sipaccregwizz.manifest.mf** file

"Export-Package: net.java.sip.communicator.plugin.sipaccregwizz"

4) Modified the **build.xml** and **felix.client.run.properties** file according to bundles created for the above packages.

## [11.2] Configuring eclipse

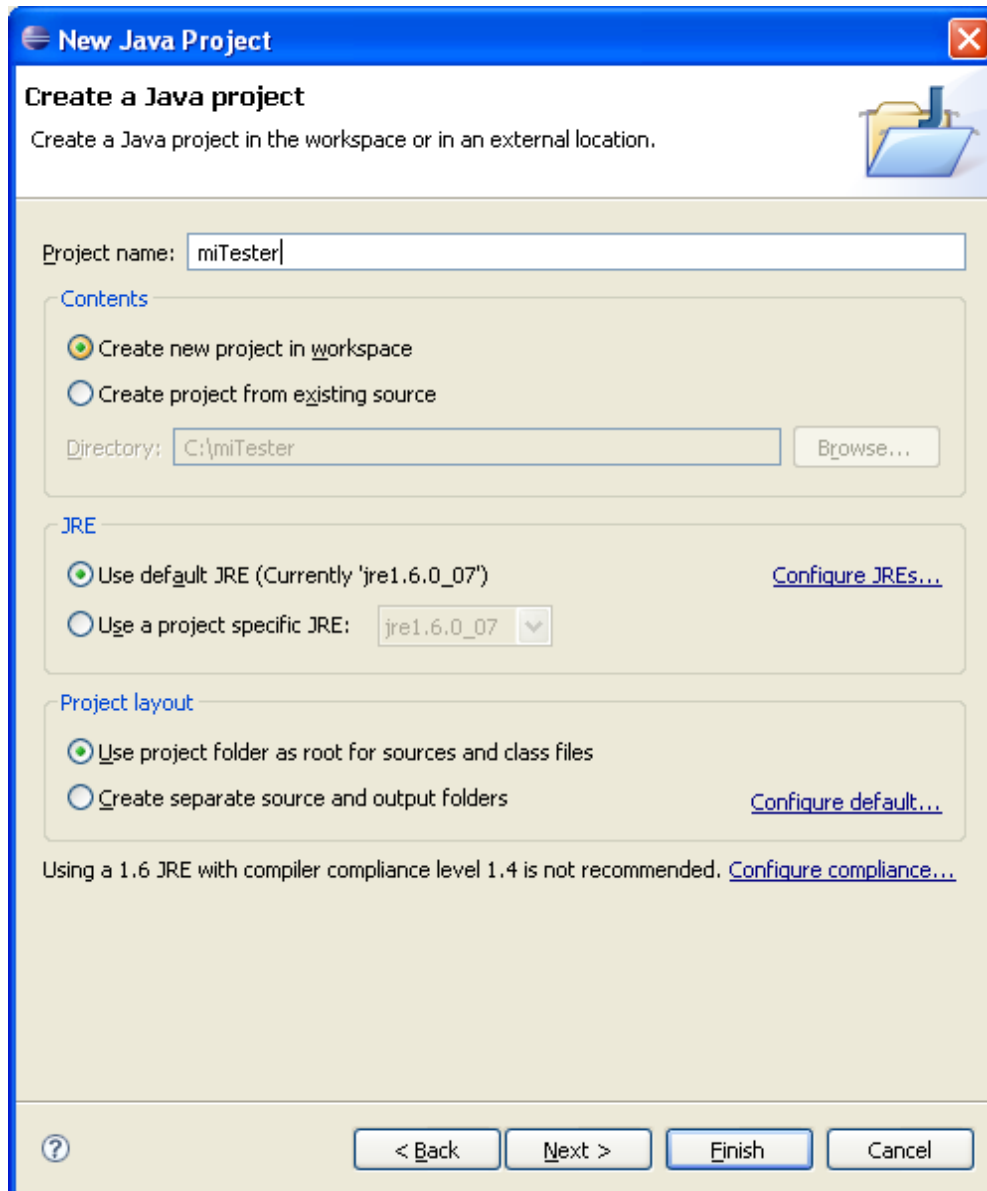
You can get the latest version from <http://www.eclipse.org>.

Note: This document describes the process for Eclipse 3.X

Here are the steps you should follow to create mitester as a new project:

Start Eclipse

1. From the *File* menu select *New* and then click on *Project* in eclipse IDE.
2. Select the Java Project and click 'Next' to enter the project name.
3. Enter the Project name and click 'Finish'.



mitester for SIP svn repository URL:  
<https://mitesterforsip.svn.sourceforge.net/svnroot/mitesterforsip>

- Add the library files from the “lib” folder of the downloaded distribution.
- If error exists, Open the project properties window and select the “*Java Compiler*” in the left panel of Properties window.
- Select the check box “*Enable project specific settings*”.
- Change the default value of “*Compiler Compilation Level*” combo box to the highest level
- Click ‘*Apply*’ and select ‘*Yes*’ for the popup window thrown for the above step.

### Configure Run and Debug through Eclipse

- Open the Run Configurations Window in Eclipse.
- Click the “Java Application” in the left panel of Run Configurations.
- Enter the name of your project and enter the *com.mitester.main.Main* in the project main class.
- Click on “Run” button.

## [12] Wishlist

Test Reports in Excel are also proposed to be developed. SIP Torture test and DTMF are on the tray for future development and enhancement with this tool.