miTester

MOBAX

mitesterforsip_User_Guide

Release Version 1.1

Table of Contents

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.

[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
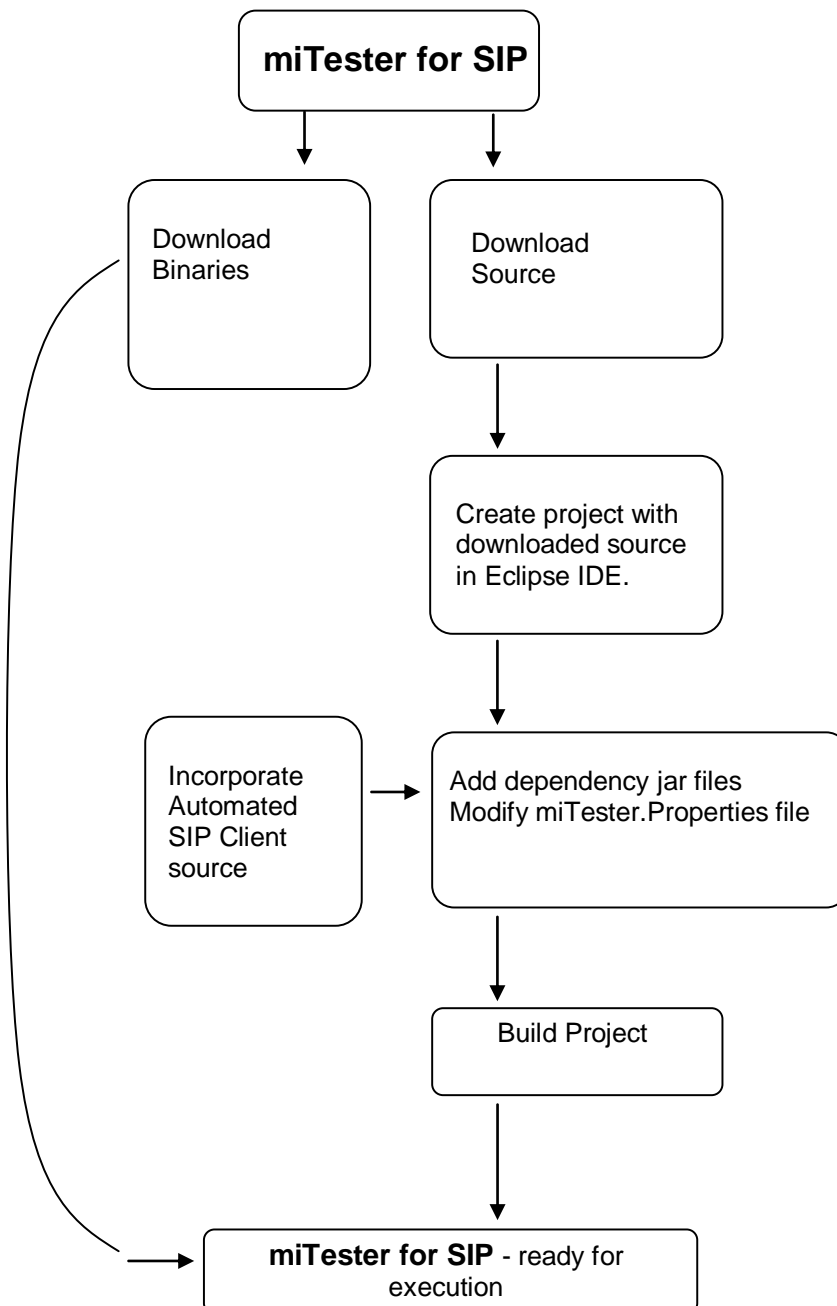

[2.2] Installation of miTester for SIP

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 :

```
                    ┌─────────────────────┐
                    │   miTester for SIP  │
                    └─────────────────────┘
                         │              │
                         ▼              ▼
              ┌──────────────┐   ┌──────────────┐
              │ Download     │   │ Download     │
              │ Binaries     │   │ Source       │
              └──────────────┘   └──────────────┘
                                        │
                                        ▼
                                ┌──────────────────┐
                                │ Create project   │
                                │ with downloaded  │
                                │ source in        │
                                │ Eclipse IDE.     │
                                └──────────────────┘
                                        │
                                        ▼
              ┌──────────────┐   ┌────────────────────────┐
              │ Incorporate  │──▶│ Add dependency jar     │
              │ Automated    │   │ files Modify           │
              │ SIP Client   │   │ miTester.Properties    │
              │ source       │   │ file                   │
              └──────────────┘   └────────────────────────┘
                                        │
                                        ▼
                                ┌──────────────┐
                                │ Build Project│
                                └──────────────┘
                                        │
                                        ▼
              ┌──────────────────────────────────────┐
              │ miTester for SIP - ready for         │
              │ execution                            │
              └──────────────────────────────────────┘
```
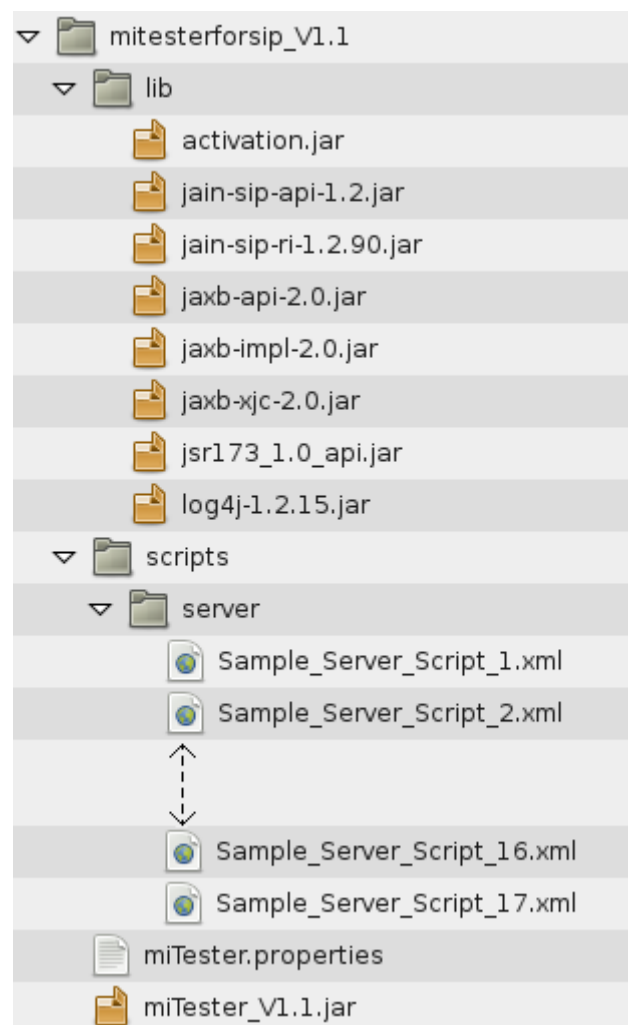
[3] USER mode - miTester for SIP

miTester for SIP is developed to process, build and control the call flows which simulate the SIP messages to the System Under Test (SUT).
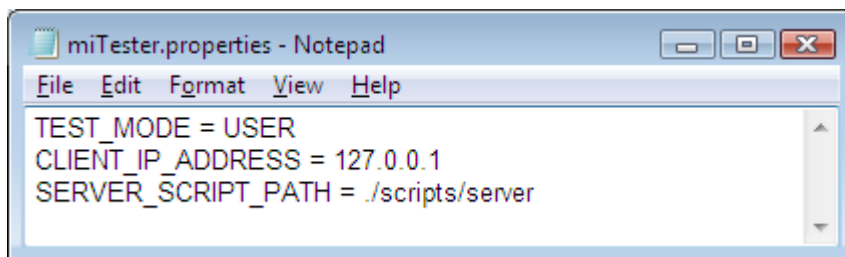
[3.1] Primary Files and Folders

To work with USER mode of miTester for SIP, download the binaries of your platform and you can find "mitesterforsip_V1.1" inside the package. Click here to download. Then click "Download" to move to the target page.

The folder structure of **mitesterforsip_V1.1** – USER mode is as follows

mitesterforsip_V1.1 – USER mode – 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
  - jsr173_1.0_api.jar
  - log4j-1.2.15.jar

- 'Script_Path' folder – Holds Sample server scripts (Sample test scripts for execution)

- miTester.properties – the file to be edited by the user. This file carries the following primary settings.

  - TEST_MODE: Set as USER.
  - CLIENT_IP_ADDRESS: Set the IP address of the System Under Test (SUT).
  - SERVER_SCRIPT_PATH: 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. This is mandatory for test execution in USER mode.
  - SERVER_DELAY: Optional. The time interval (in seconds) that the server takes to send successive request / response messages and is configurable.
  - SERVER_LISTEN_PORT: Optional. The port in which the server listens. By default the value is 5070 and is configurable.

```
miTester.properties - Notepad

File  Edit  Format  View  Help

TEST_MODE = USER
CLIENT_IP_ADDRESS = 127.0.0.1
SERVER_SCRIPT_PATH = ./scripts/server
```

USER mode - miTester.properties file

TEST MODE	INPUTS	MANDATORY	OPTIONAL
USER	TEST_MODE	✓	
	CLIENT_IP_ADDRESS	✓	
	SERVER_SCRIPT_PATH	✓	
	SERVER_DELAY		✓
	SERVER_LISTEN_PORT		✓

- miTester_V1.1.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_V1.1.jar resides in the command prompt.
Provide the following command in the command prompt.

java -jar miTester_V1.1.jar
      where 'miTester_V1.1.jar' is the name of the binary executable.

For example: g:\mitesterforsip_V1.1> java -jar miTester_V1.1.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-V1.1.pdf".
Click here to download.
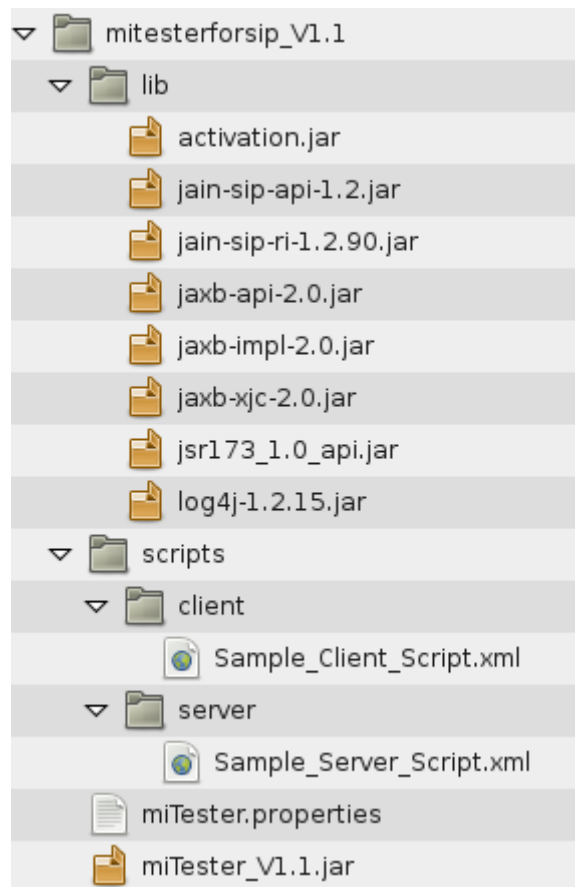
[4] ADVANCED mode - miTester for SIP

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_V1.1" inside the package. Click here to download. Then click "Download" to move to the target page.

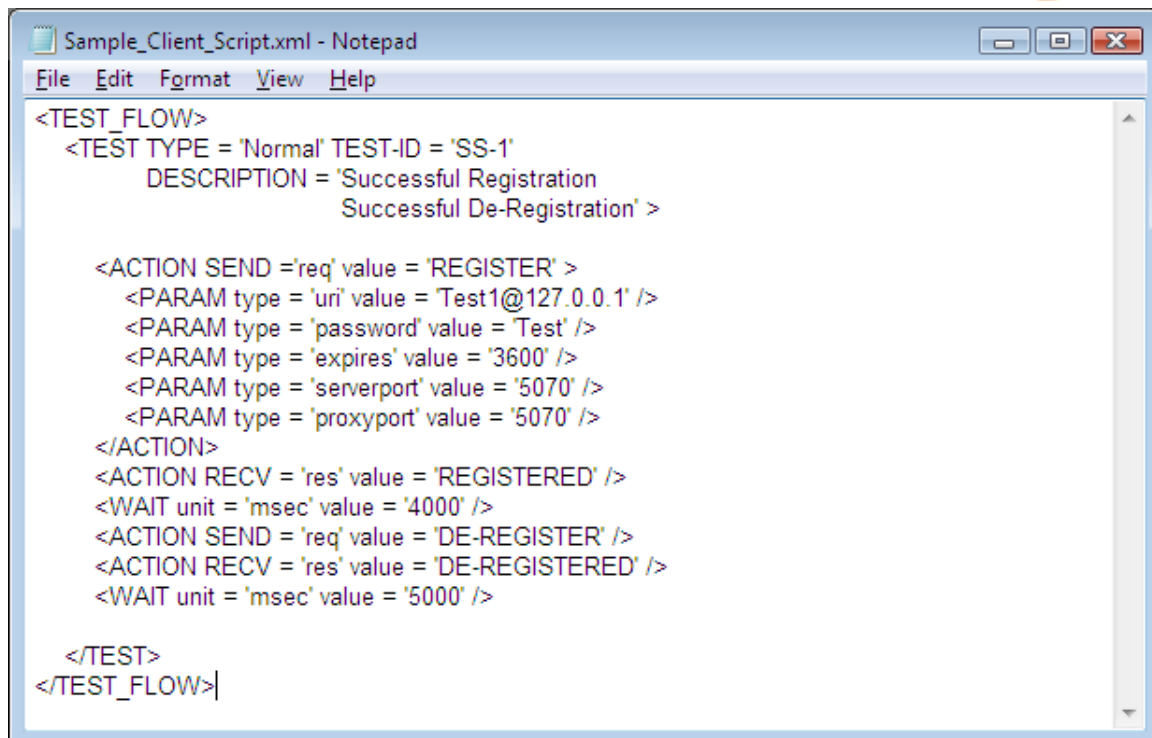Slightly alter the folder structure of **mitesterforsip_V1.1** to run the ADVANCED mode of miTester for SIP.

mitesterforsip_V1.1 – 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
- jsr173_1.0_api.jar
- log4j-1.2.15.jar

- 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. "Script_Path" 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_V1.1_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' />
      <WAIT unit = 'msec' value = '4000' />
      <ACTION SEND = 'req' value = 'DE-REGISTER' />
      <ACTION RECV = 'res' value = 'DE-REGISTERED' />
      <WAIT unit = 'msec' value = '5000' />

   </TEST>
</TEST_FLOW>
```
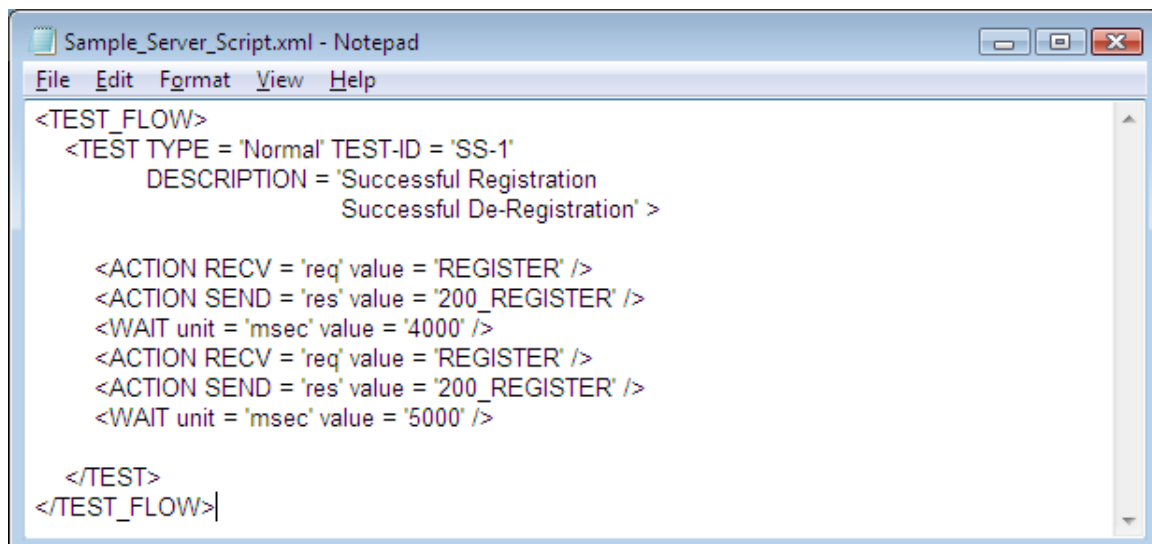
Sample Client Script

```
Sample_Server_Script.xml - Notepad
File   Edit   Format   View   Help

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

      <ACTION RECV = 'req' value = 'REGISTER' />
      <ACTION SEND = 'res' value = '200_REGISTER' />
      <WAIT unit = 'msec' value = '4000' />
      <ACTION RECV = 'req' value = 'REGISTER' />
      <ACTION SEND = 'res' value = '200_REGISTER' />
      <WAIT unit = 'msec' value = '5000' />

   </TEST>
</TEST_FLOW>
```
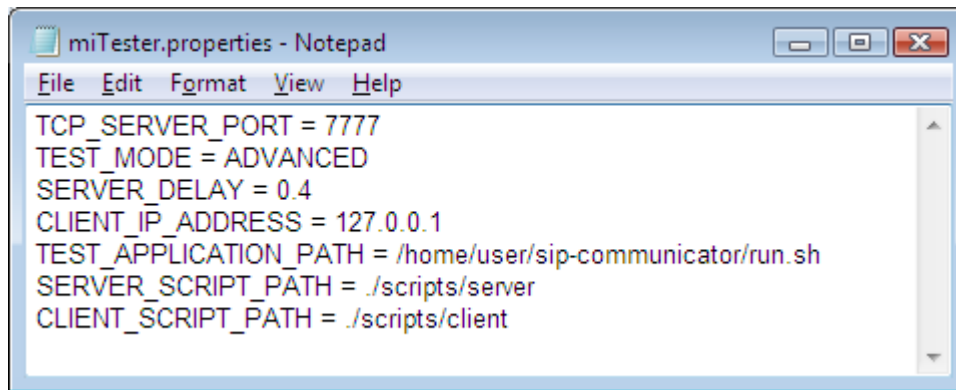
Sample Server Script

- 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. The timer intervals used for smooth execution of the test is mentioned with timers.

ADVANCED mode - miTester.properties file

- miTester_V1.1.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_v1.1_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_v1.1_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_v1.1_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_v1.1_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_v1.1_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-nightlybuild.1844
    ▽  📁  native
            📄  libAEGetURLEventHandlerAgent.jnilib
            📄  libDock.jnilib
            📄  libffmpeg.jnilib
            📄  libjdic.jnilib
            📄  libsparkle_init.dylib
            📄  libtray.jnilib
```

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)
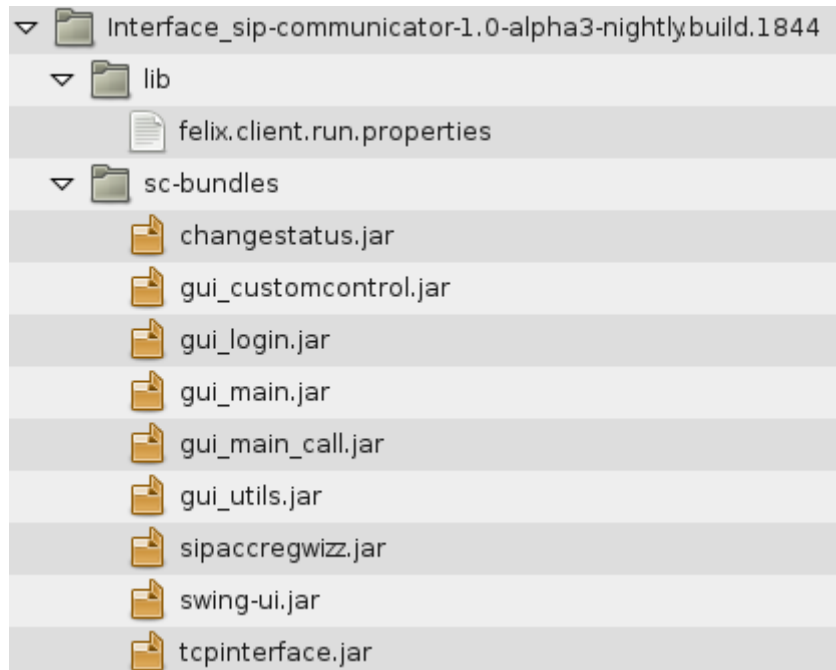
Interface_sip-communicator-1.0-alpha3-nightly.build.1844 – folder structure

[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

TCP_SERVER_PORT: This reads the value of the port assigned to engage for SIP transaction during execution of a Test Scenario. By default, the value is 7777.

TEST_MODE: Set as ADVANCED.

CLIENT_IP_ADDRESS: Set the IP address of the System Under Test (SUT).

TEST_APPLICATION_PATH: This defines where the system under test binary resides. This value is mandatory ADVANCED Test Mode.

CLIENT_SCRIPT_PATH: This path points to the Client scripts with Call flows. Call flows can span to one or more scripts for test execution. This is mandatory for test execution in ADVANCED mode.

SERVER_SCRIPT_PATH: This path points to the Server scripts with Call flows. Call flows can span to one or more scripts for test execution. This is mandatory for test execution in ADVANCED mode.

SERVER_DELAY: Optional. The time interval (in seconds) that the server takes to send successive request / response messages and is configurable.

CLIENT_WAIT_TIME: Optional. The maximum time after which the SUT is forcefully closed. By default the value is 35 secs and is configurable.

SERVER_WAIT_TIME: Optional. The maximum time after which the server is forcefully closed. By default the value is 60 secs and is configurable.

EXECUTION_INTERVAL: Optional. The time interval (in seconds) taken between the execution of successive test cases. By default, the value is 2 secs. The user can assign his own value.
Note: A minimum of 2 secs is recommended.

SERVER_LISTEN_PORT: Optional. The port in which the server listens. By default the value is 5070 and is configurable.

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

TEST MODE	INPUTS	MANDATORY	OPTIONAL
ADVANCED	TCP_SERVER_PORT	✔	
	TEST_MODE	✔	
	CLIENT_IP_ADDRESS	✔	
	TEST_APPLICATION_PATH	✔	
	SERVER_SCRIPT_PATH	✔	
	CLIENT_SCRIPT_PATH	✔	
	SERVER_DELAY		✔
	CLIENT_WAIT_TIME		✔
	SERVER_WAIT_TIME		✔
	EXECUTION_INTERVAL		✔
	SERVER_LISTEN_PORT		✔

[4.4] Run ADVANCED mode - miTester for SIP

After completing all the previous mentioned steps, navigate to the folder where miTester_V1.1.jar resides in the command prompt.
Provide the following command in the command prompt.

java -jar miTester_V1.1.jar
        where 'miTester_V1.1.jar' is the name of the binary executable.

For example: g:\mitesterforsip_V1.1> java -jar miTester_V1.1.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] 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.

[6] 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.

[7] 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.

[8] Appendices

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

- ADVANCED MODE

For automating the execution if any SIP application, an interface layer need 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, OprationSetBasictelephony, and AccountRegistrationWizrd 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.
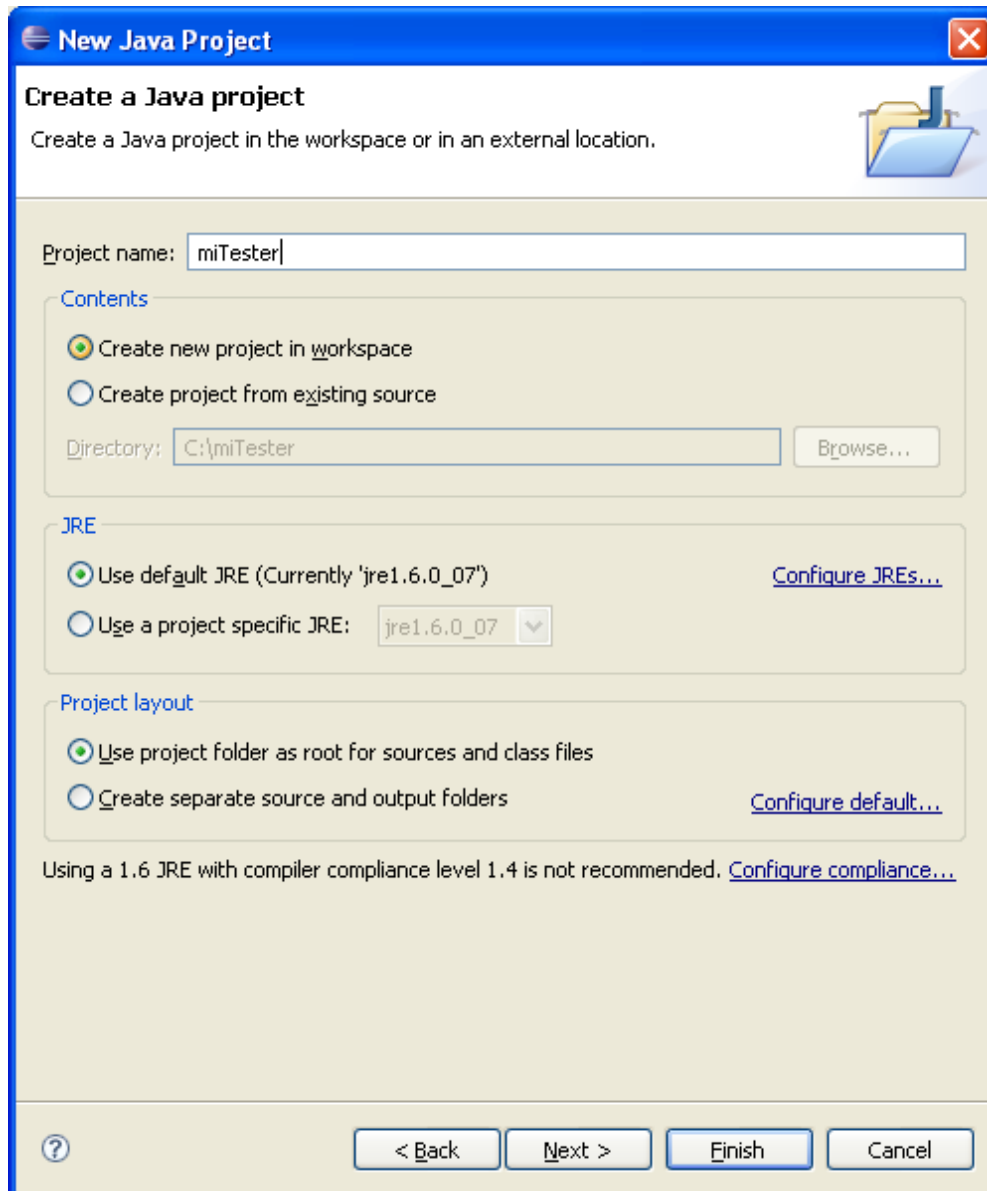
[8.2] Configuring eclipse

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.

[9] Wishlist

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