Functional Specification for NetBeans SIP test client

Author(s): elnyvbo@dev.java.net

Version: 0.3

1 Introduction

This document describes a SIP client which integrates with NetBeans. The client offers a graphical user interface for constructing SIP requests and responses. The client was developed as a NetBeans plugin. It should be relatively easy, though, to remove the NetBeans-dependent code and run the client as a standalone Java application.

1.1 Terminology

The following table lists the most important terms and abbrevations used in this document.

EAS	Ericsson Application Server
GUI	Graphical User Interface
Matisse	Framework in NetBeans offering drag & drop GUI development.
NBM	NetBeans Module
NetBeans	Integrated development environment developed by Sun.
SDS	Service Development Studio
SIP	Session Initiation Protocol
TCP	Transport Control Protocol
UDP	Universal Datagram Protocol
UAS	User Agent Server

2 Design Overview

The GUI was developed in Swing and is based on the look and feel of Ericsson's SDS Test Agent. The client is stateless, only basic dialogue awareness was added so subsequent messages automatically copy CallId and tags from preceeding responses.

The GUI operates on a very simple datamodel, consisting of SIPRequest and SIPResponse objects and some additional helper objects to represent the SIPDialogue and the basic state info contained in it.

Only SIP signaling is supported, no media streaming whatsoever. Basic bytestream socket communication is used. The client does not contain a SIP stack as this would inevitably introduce typical behaviour, something you do not want in a barebone testclient.

To parse the bytestream arriving at the socket, the SIP parser from the EAS SIP stack was reused. This is esssentially the same SIP parser that is present in the SailFin code (`com.ericsson.ssa.sip.SipParser`). Socket sending will just mean serializing the SIPrequests and responses to a bytestream.

2.1 Package overview

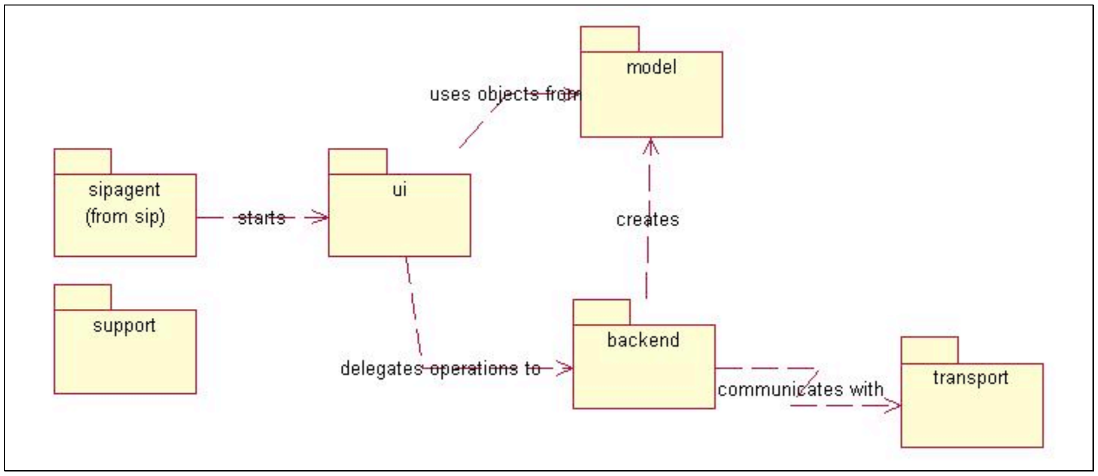


Figure 1 Package overview

The ui package has been developed using the GUI builder framework in NetBeans. The model package offers objects representing SIP requests and SIP responses, and a SIP parser which was reused from the SailFin SIP container.
Model objects are never constructed directly by the frontend, but always via the backend package. The backend package is also capable of receiving raw data from the transport layer and deserializing that data into SIP requests and responses.

The test agent is heavily SIP aware, though it shouldn't be too difficult to refactor it and extend it for use with other (header/value-based) protocols.

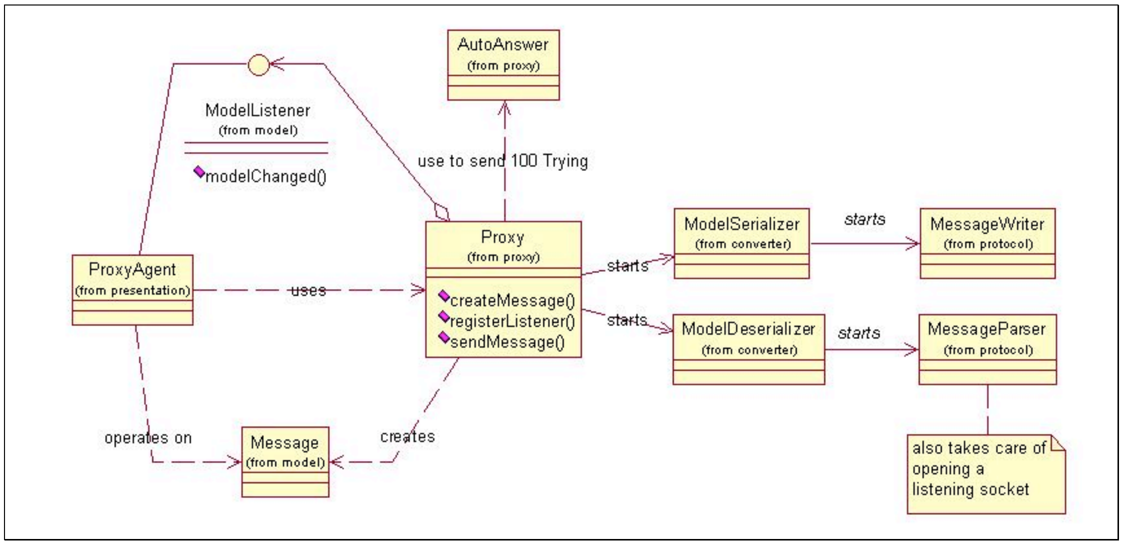


Figure 2 Important classes (obsolete, todo update)

2.2 Usage instructions

The following screenshots were taken from the NetBeans SIP Test Agent. To start the Agent, use either the toolbar button or the popup-menu which is available in the *Files* view, as shown in the following pictures.

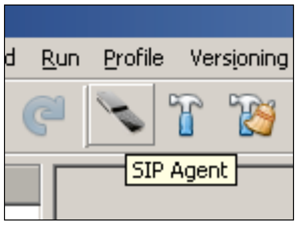


Figure 3 SIP Agent toolbar button

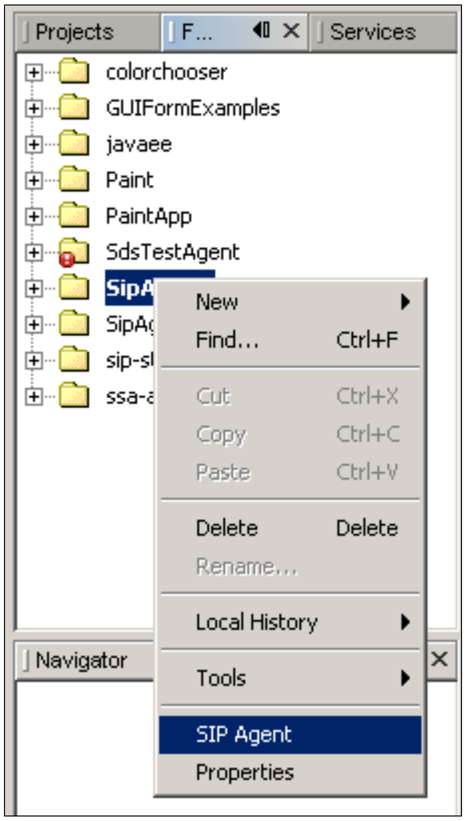


Figure 4 SIP Agent popup menu

Figure 5 shows the main window, offering some basic client configuration options and an overview of the SIP messages sent and received. The following options can be configured:

Agent Name	Just a logical name to distinguish several agent instances.
Listening Port	Port on which the Test Agent receives incoming SIP traffic.
Transport	Specify whether to use UDP or TCP for SIP traffic.
Log File	All SIP traffic can be logged to a log file.
Host	IP address of the UAS.
Port	Port of the UAS.

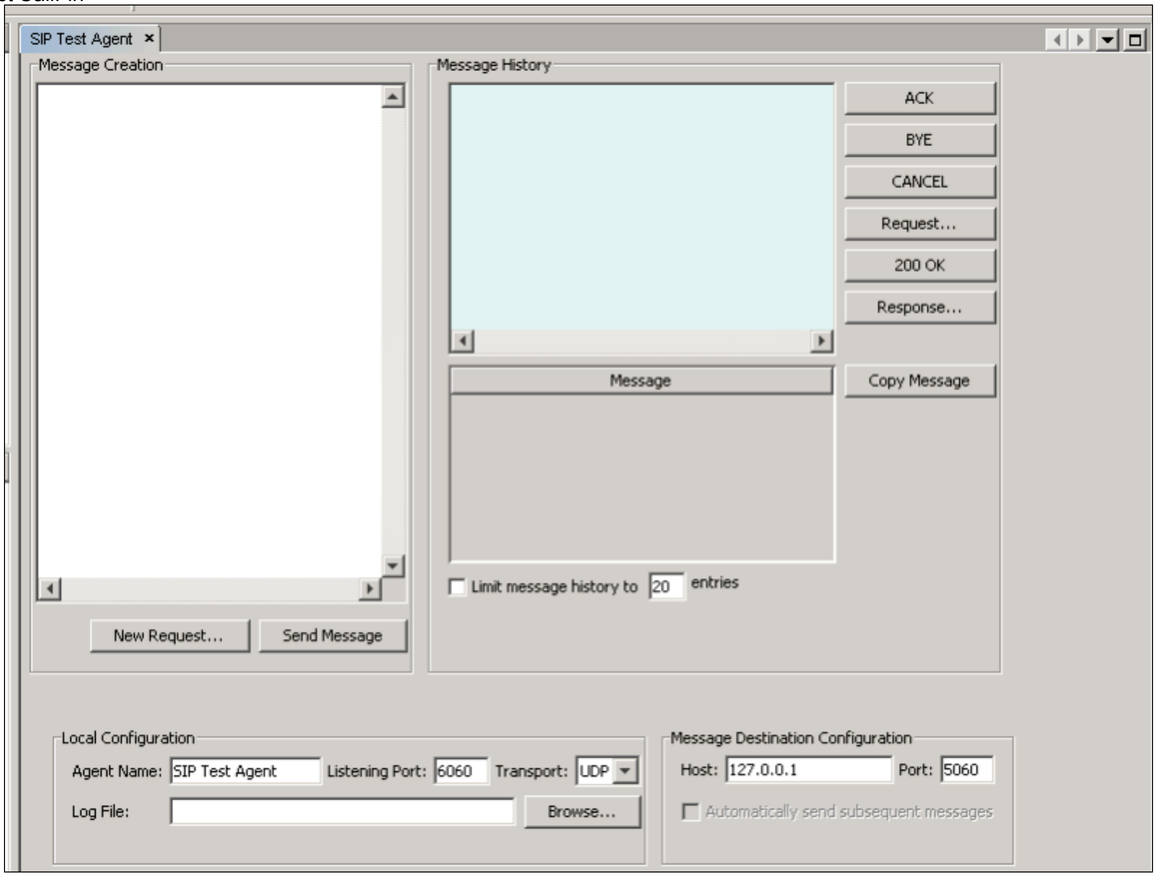
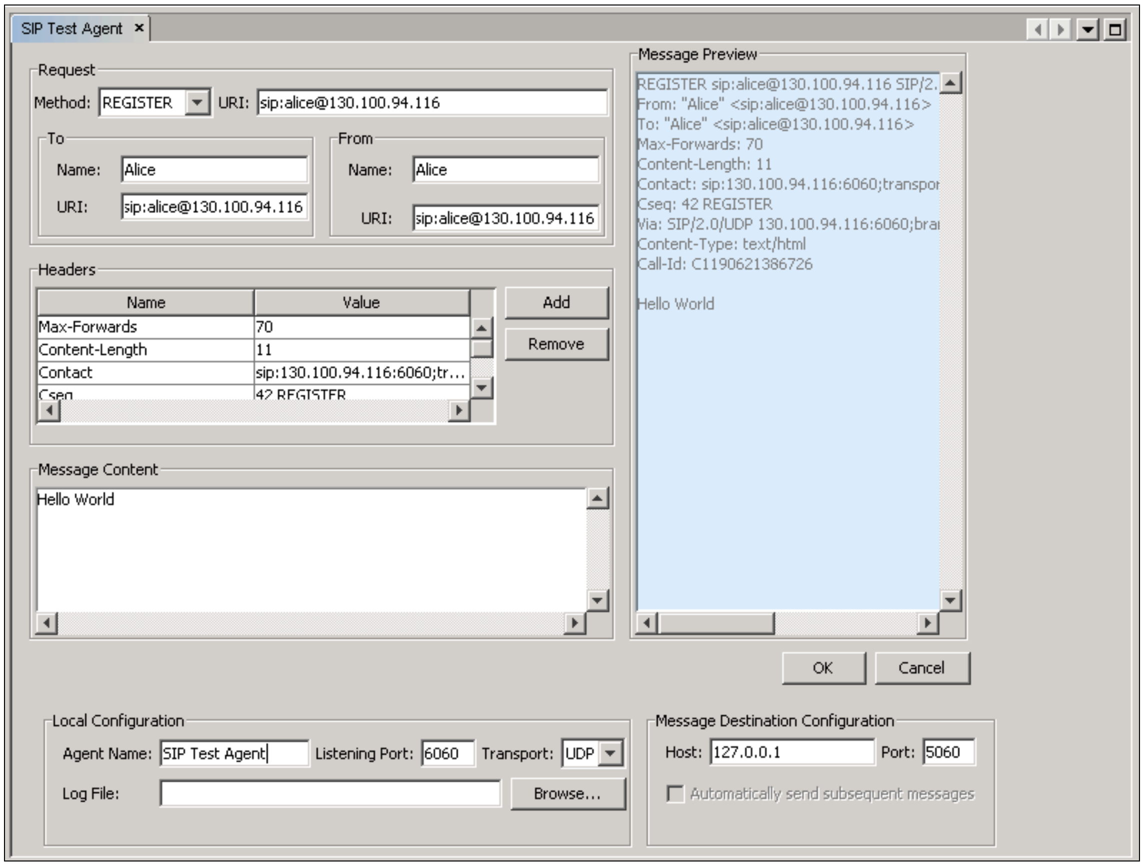


Figure 5 Main screen

The table below describes the functionality of the main screen.

New Request…	Will open the request editor with the text in the *Message Creation* area as a starting point.
Send Message	Will attempt to send the text displayed in the *Message Creation* area to the configured destination.
Message History	Lists messages sent and received. Select one of the messages in the history to display its contents.
ACK	Prepare a default ACK request, based on the response selected in the message history.
BYE	Prepare a default BYE request, based on the response selected in the message history.
CANCEL	Prepare a default CANCEL request, based on the response selected in the message history.
Request…	Open the request editor.
200 OK	Prepare a default 200 OK response, based on the request selected in the message history.
Copy Message	Copy the selected message from the history to the *Message Creation* area.

Figure 6 shows the SIP request editor, which is used to construct SIP requests. The test agent will fill in some header defaults (Cseq, Max-Forwards, etc.) and allow the user to specify additional headers and modify the message content. To accept the SIP request shown in the *Message Preview* area, click *OK*. The test agent will close the request editor and copy the prepared SIP request to the *Message Creation* area of the main screen.



Method: REGISTER URI: sip:alice@130.100.94.116

To — Name: Alice URI: sip:alice@130.100.94.116

From — Name: Alice URI: sip:alice@130.100.94.116

Name	Value
Max-Forwards	70
Content-Length	11
Contact	sip:130.100.94.116:6060;tr...
Cseq	42 REGISTER

Message Content: Hello World

```
REGISTER sip:alice@130.100.94.116 SIP/2.
From: "Alice" <sip:alice@130.100.94.116>
To: "Alice" <sip:alice@130.100.94.116>
Max-Forwards: 70
Content-Length: 11
Contact: sip:130.100.94.116:6060;transpor
Cseq: 42 REGISTER
Via: SIP/2.0/UDP 130.100.94.116:6060;bra
Content-Type: text/html
Call-Id: C1190621386726

Hello World
```

Agent Name: SIP Test Agent Listening Port: 6060 Transport: UDP

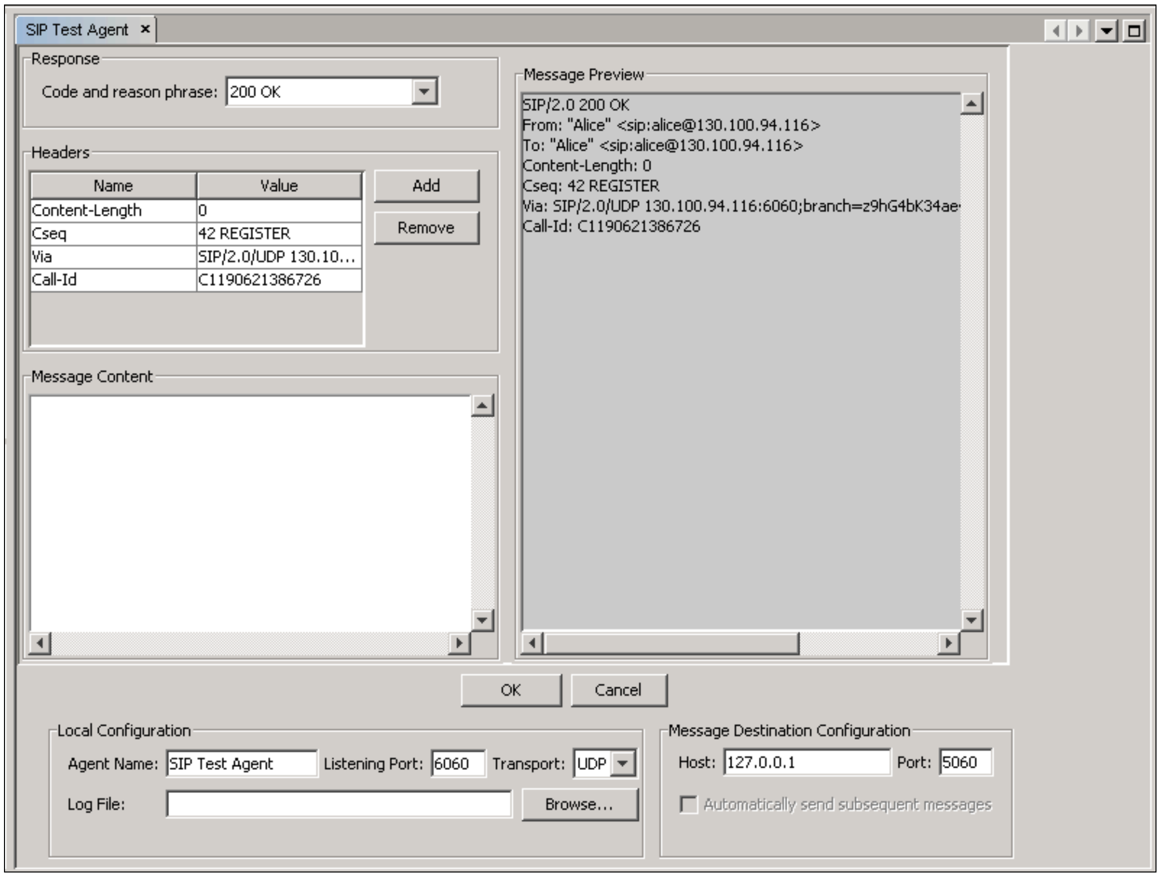
Log File:

Host: 127.0.0.1 Port: 5060

☐ Automatically send subsequent messages

Figure 6 SIP request editor

Figure 7 shows the SIP response editor, which is used to construct SIP responses. Similar to the request editor, the test agent will fill in some header defaults and allow the user to specify additional headers and modify the message content. Again, clicking *OK* closes the editor and the agent returns to the main screen with the prepared SIP responses copied to the *Message Creation* area.



Code and reason phrase: 200 OK

Name	Value
Content-Length	0
Cseq	42 REGISTER
Via	SIP/2.0/UDP 130.10...
Call-Id	C1190621386726

```
SIP/2.0 200 OK
From: "Alice" <sip:alice@130.100.94.116>
To: "Alice" <sip:alice@130.100.94.116>
Content-Length: 0
Cseq: 42 REGISTER
Via: SIP/2.0/UDP 130.100.94.116:6060;branch=z9hG4bK34ae
Call-Id: C1190621386726
```

Agent Name: SIP Test Agent Listening Port: 6060 Transport: UDP
Log File:

Host: 127.0.0.1 Port: 5060
Automatically send subsequent messages

Figure 7 SIP response editor

3 Quality and Availability

Quality is assured by regular prototype deliveries. Testing was mostly carried out by hand. Function test could be automated using robot software. At the time of writing, this has not yet been done.

4 Performance

No specific requirements on performance. GUI responsiveness should be fast, e.g. reception of retransmitted SIP messages should be displayed more or less in real time.
Memory consumption should be measured using basic tools such as Jconsole. It is expected that memory consumption will be low.

5 Management and Monitoring

N/A

6 Packaging, Files, and Location

Test client will be delivered as several NetBeans modules (.nbm archive). The tabel below lists the deliverables:

File	Description
`com-ericsson-sip-sipagent.nbm`	The NetBeans SIP TestAgent plugin
`com-ericsson-ssa.nbm`	Ericsson SSA library module
`javax.nbm`	JavaEE API library module
`javax-servlet-sip.nbm`	javax.servlet servlet library module
`updates.xml`	NetBeans plugin info file.

7 Documentation Requirements

Basic user instructions have been included in this document, see chapter 2.2.

8 Open Issues

The following issues need further attention:

1. Sun/Ericsson licensing issues related to SDS / Eclipse
2. Integration with SIP application wizard developed by Ajay Acharya.
3. Testing