**Foreword**

SIPped is a tool for reproducing SIP signaling scenarios to support functional testing needs, including multiple call flows per session, auto-reregistration, and advanced features such as BLA and SCA. Phone actions can be automated, and the resulting scenarios incorporated into a automation framework for use with regression and smoke testing.

SIPped (the past tense of SIP) is based on the open source SIPp project with significant enhancements to facilitate functional testing, include files, XML-standards compliance and ease of use.

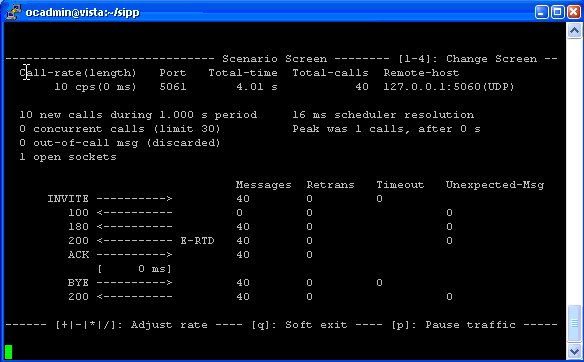
This guide serves as reference documentation. Please refer to the SIPped page for more task-oriented instructions.

**Table of Contents**

* [Foreword](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide" \l "Foreword)
* [Table of Contents](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Table_of_Contents)
* [Using SIPp](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Using_SIPp) 
  + [Main features](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Main_features)
  + [Integrated scenarios](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Integrated_scenarios)
  + [3PCC Extended](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_3PCC_Extended)
  + [Controlling SIPp](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Controlling_SIPp) 
    - [Traffic control](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Traffic_control)
  + [Remote control](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Remote_control)
  + [Running SIPp in background](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Running_SIPp_in_background)
  + [Create your own XML scenarios](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Create_your_own_XML_scenarios) 
    - [Commands](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Commands) 
      * [<send>](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_)
      * [<recv>](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_AN1)
      * [<pause>](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_AN2)
      * [<nop>](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_AN3)
      * [<sendCmd>](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_AN4)
      * [<recvCmd>](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_AN5)
      * [<label>](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_AN6)
      * [<Response Time Repartition>](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_AN7)
      * [<Call Length Repartition>](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_AN8)
      * [<Globals>](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_AN9)
      * [<User>](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_AN10)
      * [<Reference>](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_AN11)
    - [Structure of client (UAC like) XML scenarios](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Structure_of_client_40UAC_like_41_XML_scenarios) 
      * [Keywords](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Keywords) 
        + [[service]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91service_93)
        + [[remote\_ip]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91remote_ip_93)
        + [[remote\_port]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91remote_port_93)
        + [[transport]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91transport_93)
        + [[local\_ip]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91local_ip_93)
        + [[local\_ip\_type]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91local_ip_type_93)
        + [[local\_port]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91local_port_93)
        + [[local\_ip2]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91local_ip2_93)
        + [[local\_ip2\_type]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91local_ip2_type_93)
        + [[len]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91len_93)
        + [[call\_number]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91call_number_93)
        + [[cseq]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91cseq_93)
        + [[cseq\_method]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91cseq_method_93)
        + [[call\_id]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91call_id_93)
        + [[media\_ip]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91media_ip_93)
        + [[media\_ip\_type]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91media_ip_type_93)
        + [[media\_port]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91media_port_93)
        + [[auto\_media\_port]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91auto_media_port_93)
        + [[last\_\*]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91last_42_93)
        + [[field0-n file=<filename> line=<number>]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91field0_45n_file_61_line_61_93)
        + [[file name=<filename>]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91file_name_61_93)
        + [[timestamp]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91timestamp_93)
        + [[last\_message]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91last_message_93)
        + [[$n]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91_36n_93)
        + [[authentication]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91authentication_93)
        + [[pid]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91pid_93)
        + [[routes]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91routes_93)
        + [[next\_url]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91next_url_93)
        + [[branch]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91branch_93)
        + [[msg\_index]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91msg_index_93)
        + [[clock\_tick]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91clock_tick_93)
        + [[sipp\_version]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91sipp_version_93)
        + [[tdmmap]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91tdmmap_93)
        + [[fill]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91fill_93)
        + [[users]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91users_93)
        + [[userid]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91userid_93)
        + [[remote\_tag\_param]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91remote_tag_param_93)
        + [[remote\_tag]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91remote_tag_93)
        + [[local\_tag\_param]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91local_tag_param_93)
        + [[local\_tag]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91local_tag_93)
        + [[contact\_uri]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91contact_uri_93)
        + [[contact\_name\_and\_uri]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91contact_name_and_uri_93)
        + [[to\_uri]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91to_uri_93)
        + [[to\_name\_and\_uri]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91to_name_and_uri_93)
        + [[from\_uri]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91from_uri_93)
        + [[from\_name\_and\_uri]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91from_name_and_uri_93)
      * [[last\_cseq\_number]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91last_cseq_number_93) 
        + [[last\_branch]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91last_branch_93)
        + [[last\_Request\_URI]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91last_Request_URI_93)
        + [[client\_cseq]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91client_cseq_93)
        + [[client\_cseq\_method]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91client_cseq_method_93)
        + [[server\_cseq]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91server_cseq_93)
        + [[server\_cseq\_method]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91server_cseq_method_93)
        + [[received\_cseq]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91received_cseq_93)
        + [[received\_cseq\_method]](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#A_91received_cseq_method_93)
      * [Encoding](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Encoding)
      * [Receive](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Receive)
    - [Structure of server (UAS like) XML scenarios](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Structure_of_server_40UAS_like_41_XML_scenarios)
    - [Actions](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Actions) 
      * [Regular expressions](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Regular_expressions)
      * [Log a message](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Log_a_message)
      * [Execute a command](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Execute_a_command)
      * [Internal commands](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Internal_commands)
      * [External commands](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#External_commands)
      * [PCAP (media) commands](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#PCAP_40media_41_commands)
    - [Including files (and building reusable SIPp modules)](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Including_files_40and_building_reusable_SIPp_modules_41) 
      * [Variable Manipulation](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Variable_Manipulation)
      * [String Variables](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#String_Variables)
      * [Variable Testing](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Variable_Testing)
      * [lookup](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#lookup)
      * [Updating In-Memory Injection files](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Updating_In_45Memory_Injection_files)
      * [Jumping to an Index](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Jumping_to_an_Index)
      * [gettimeofday](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#gettimeofday)
      * [setdest](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#setdest)
      * [verifyauth](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#verifyauth)
    - [Variables](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Variables)
    - [Injecting values from an external CSV during calls](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Injecting_values_from_an_external_CSV_during_calls)
    - [Conditional branching](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Conditional_branching)
    - [SIP authentication](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#SIP_authentication)
    - [Initialization Stanza](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Initialization_Stanza)
  + [Screens](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Screens)
  + [Transport modes](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Transport_modes)
  + [Handling media with SIPp](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Handling_media_with_SIPp) 
    - [RTP echo](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#RTP_echo)
    - [PCAP Play](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#PCAP_Play)
  + [Exit codes](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Exit_codes)
  + [Statistics](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Statistics)
  + [Error handling](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Error_handling) 
    - [Unexpected messages](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Unexpected_messages)
    - [Retransmissions (UDP only)](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Retransmissions_40UDP_only_41)
    - [Log files](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Log_files)
  + [Options (-h)](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Options_40_45h_41) 
    - [Message Sequencing](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Message_Sequencing)
* [Performance testing with SIPp](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Performance_testing_with_SIPp) 
  + [Advices to run performance tests with SIPp](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Advices_to_run_performance_tests_with_SIPp)
  + [SIPp's internal scheduling](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#SIPp_39s_internal_scheduling)
* [IPv6 and SIPp](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#IPv6_and_SIPp)
* [Useful tools aside SIPp](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Useful_tools_aside_SIPp) 
  + [JEdit](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#JEdit)
  + [Wireshark/tshark](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Wireshark_47tshark)
  + [SIP callflow](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#SIP_callflow)
* [Getting support](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Getting_support)
* [Contributing to SIPp](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Contributing_to_SIPp)

**Want to see it?**

Here is a screenshot



And here is a video (Windows Media Player 9 codec or above required) of SIPp in action:

[sipp-01.wmv](https://twiki.polycom.com/twiki/bin/viewauth/Main/images/sipp-01.wmv)

**Using SIPp**

**Main features**

SIPp allows to generate one or many SIP calls to one remote system. The tool is started from the command line. In this example, two SIPp are started in front of each other to demonstrate SIPp capabilities.

Run sipp with embedded server (uas) scenario: # ./sipp -sn uas

On the same host, run sipp with embedded client (uac) scenario # ./sipp -sn uac 127.0.0.1

**Integrated scenarios**

Go [here](https://twiki.polycom.com/twiki/bin/view/Main/SIPpedReferenceGuide?topic=IntegratedScenarios) for the full article.

**3PCC Extended**

Go [here](https://twiki.polycom.com/twiki/bin/view/Main/SipP3pccExtended) for the full article.

**Controlling SIPp**

SIPp can be controlled interactively through the keyboard or via a UDP socket. SIPp supports both 'hot' keys that can be entered at any time and also a simple command mode. The hot keys are:

| [**Key**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=1;up=0#sorted_table) | [**Action**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=1;up=0#sorted_table) |
| --- | --- |
| + | Increase the call rate by 1 \* rate\_scale |
| \* | Increase the call rate by 10 \* rate\_scale |
| - | Decrease the call rate by 1 \* rate\_scale |
| / | Decrease the call rate by 10 \* rate\_scale |
| c | Enter command mode |
| q | Quit SIPp (after all calls complete, enter a second time to quit immediately) |
| Q | Quit SIPp immediately |
| s | Dump screens to the log file (if -trace\_screen is passed) |
| p | Pause traffic |
| 1 | Display the scenario screen |
| 2 | Display the statistics screen |
| 3 | Display the repartition screen |
| 4 | Display the variable screen |
| 5 | Display the TDM screen |
| 6-9 | Display the second tdrough fifth repartition screen. |

In command mode, you can type a single line command that instructs SIPp to take some action. Command mode is more versatile than the hot keys, but takes more time to input some common actions. The following commands are available:

|  |  |  |
| --- | --- | --- |
| [**Command**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=2;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=2;up=0#sorted_table) | [**Example**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=2;up=0#sorted_table) |
| dump tasks | Prints a list of active tasks (most tasks are calls) to the error log. | dump tasks |
| set rate X | Sets the call rate. | set rate 10 |
| set rate-scale X | Sets the rate scale, which adjusts the speed of '+', '-', '\*', and '/'. | set rate-scale 10 |
| set users X | Sets the number of users (only valid when -users is specified). | set rate 10 |
| set limit X | Sets the open call limit (equivalent to -l option) | set limit 100 |
| set hide <true or false> | Should the hide XML attribute be respected? | set hide false |
| set index <true or false> | Display message indexes in the scenario screen. | set index true |
| set display <main or ooc> | Changes the scenario that is displayed to either the main or the out-of-call scenario. | set display main set display ooc |
| trace <log> <on or off> | Turns log on or off at run time. Valid values for log are "error", "logs", "messages", and "shortmessages". | trace error on |

**Traffic control**

SIPp generates SIP traffic according to the scenario specified. You can control the number of calls (scenario) that are started per second. If you pass the -users option, then you need to control the number of instantiated users. You can control the rate through:

* Interactive hot keys (described in the previous section)
* Interactive Commands
* Startup Parameters

There are two commands that control rates: set rate X sets the current call rate to X. Additionally, set rate-scale X sets the rate\_scale parameter to X. This enables you to use the '+', '-', '\*', and '/' keys to set the rate more quickly. For example, if you do set rate-scale 100, then each time you press '+', the call rate is increased by 100 calls and each time you press '\*', the call rate is increased by 1000 calls. Similarly, for a user based benchmark you can run set users X.

At starting time, you can control the rate by specifying parameters on the command line:

* "-r" to specify the call rate in number of calls per seconds
* "-rp" to specify the "**r**ate **p**eriod" in milliseconds for the call rate (default is 1000ms/1sec). This allows you to have n calls every m milliseconds (by using -r n -rp m).

Example

run SIPp at 7 calls every 2 seconds (3.5 calls per second):

./sipp -sn uac -r 7 -rp 2000 127.0.0.1

You can also **pause** the traffic by pressing the 'p' key. SIPp will stop placing new calls and wait until all current calls go to their end. You can **resume** the traffic by pressing 'p' again.

To **quit** SIPp, press the 'q' key. SIPp will stop placing new calls and wait until all current calls go to their end. SIPp will then exit.

You can also force SIPp to **quit** immediatly by pressing the 'Q' key. Current calls will be terminated by sending a BYE or CANCEL message (depending if the calls have been established or not). The same behaviour is obtained by pressing 'q' twice. **TIP:** you can place a defined number of calls and have SIPp exit when this is done. Use the -m option on the command line.

**Remote control**

SIPp can be "remote-controlled" through a UDP socket. This allows for example

* To automate a series of actions, like increasing the call rate smoothly, wait for 10 seconds, increase more, wait for 1 minute and loop
* Have a feedback loop so that an application under test can remote control SIPp to lower the load, pause the traffic, ...

Each SIPp instance is listening to a UDP socket. It starts to listen to port 8888 and each following SIPp instance (up to 60) will listen to base\_port + 1 (8889, 8890, ...).

It is then possible to control SIPp like this:

echo p >/dev/udp/x.y.z.t/8888 -> put SIPp in pause state (p key)

echo q >/dev/udp/x.y.z.t/8888 -> quit SIPp (q key)

All keys available through keyboard are also available in the remote control interface

You could also have a small shell script to automate a serie of action. For example, this script will increase the call rate by 10 more new calls/s every 5 seconds, wait at this call rate for one minute and exit SIPp:

#!/bin/sh

echo "\*" >/dev/udp/127.0.0.1/8889

sleep 5

echo "\*" >/dev/udp/127.0.0.1/8889

sleep 5

echo "\*" >/dev/udp/127.0.0.1/8889

sleep 5

echo "\*" >/dev/udp/127.0.0.1/8889

sleep 60

echo "q" >/dev/udp/127.0.0.1/8889

To send a command to SIPp, preface it with 'c'. For example: echo "cset rate 100" >/dev/udp/127.0.0.1/8888 sets the call rate to 100.

**Running SIPp in background**

SIPp can be launched in background mode (-bg command line option).

By doing so, SIPp will be detached from the current terminal and run in the background. The PID of the SIPp process is provided. If you didn't specify a number of calls to execute with the -m option, SIPp will run forever.

There is a mechanism implemented to stop SIPp smoothly. The command kill -SIGUSR1 [SIPp\_PID] will instruct SIPp to stop placing any new calls and finish all ongoing calls before exiting.

When using the background mode, the main sipp instance stops and a child process will continue the job. Therefore, the log files names will contain another PID than the actual sipp instance PID.

**Create your own XML scenarios**

Of course embedded scenarios will not be enough. So it's time to create your own scenarios. A SIPp scenario is written in XML (a DTD that may help you write SIPp scenarios does exist and has been tested with jEdit - this is described in a later section). A scenario will always start with:

<?xml version="1.0" encoding="ISO-8859-1" ?> <!DOCTYPE scenario SYSTEM "sipp.dtd">

Followed by the scenario tag:

<scenario name="Generated Scenario for 172.24.44.75:5060" parameters="-mc -aa">

The scenario tag must include:

* **name** : name of the scenario
* **parameters** : the parameters required to run the scenario (all parameter options are listed [here](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Options_h))
* **xmlns:xi="http://www.w3.org/2001/XInclude** : this is only required if the scenario includes files containing other scenarios

Optional attributes the scenario tag may contain:

* **id** : this attribute is only present if the scenario is generated by [snipp](https://twiki.polycom.com/twiki/bin/view/Main/Snipp). It corresponds to the bug ID that that was being tested
* **source** : this attribute is only present if the scenario is generated by [snipp](https://twiki.polycom.com/twiki/bin/view/Main/Snipp). It is the command line arguments that were used to run snipp to generate the scenario
* **config** :

The scenario must end with the closing tag: </scenario>

**Example**

<scenario xmlns:xi="http://www.w3.org/2001/XInclude" name="Generated Scenario for 172.24.44.75:5060" parameters="-mc" source="snipp.pl -i 172.24.44.75 -f PlaceCall.pcap -d TESTING" id="TESTING" config="">

Easy, huh? Ok, now let's see what can be put inside. You are not obliged to read the whole table now! Just go in the next section for an example.

**Commands**

There are many common attributes used for flow control and statistics, that can be used for all of the message commands (i.e., **<send>**, **<recv>**, **<nop>**, **<pause>**, **<sendCmd>**, and **<recvCmd>**).

| [**Attribute(s)**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=3;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=3;up=0#sorted_table) | [**Example**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=3;up=0#sorted_table) |
| --- | --- | --- |
| start\_rtd | Starts one of the "**R**esponse **T**ime **D**uration" timer. (see [statistics section](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Response+times)). | <send start\_rtd="invite">: the timer named "invite" will start when the message is sent. |
| rtd | Stops one of the 5 "**R**esponse **T**ime **D**uration" timer. | <send rtd="2">: the timer number 2 will stop when the message is sent. |
| repeat\_rtd | Used with a rtd attribute, it allows the corresponding "**R**esponse **T**ime **D**uration" timer to be counted more than once per call (useful for loop call flows). | <send rtd="1" repeat\_rtd="true">: the timer number 1 value will be printed but the timer won't stop. |
| crlf | Displays an empty line **after** the arrow for the message in main SIPp screen. | <send crlf="true"> |
| next | You can put a "next" in any command element to go to another part of the script when you are done with sending the message. For optional receives, the next is only taken if that message was received. See [conditional branching](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#branching) section for more info. | Example to jump to label "12" after sending an ACK:  <send next="12">  <![CDATA[   ACK sip:[service]@[remote\_ip]:[remote\_port] SIP/2.0  Via: ...  From: ...  To: ...  Call-ID: ...  Cseq: ...  Contact: ...  Max-Forwards: ...  Subject: ...  Content-Length: 0   ]]>  </send>   Example to jump to label "5" when receiving a 403 message:  <recv response="100" optional="true"> </recv>  <recv response="180" optional="true">  </recv>  <recv response="403" optional="true" next="5">  </recv>  <recv response="200">  </recv> |
| test | You can put a "test" next to a "next" attribute to indicate that you only want to branch to the label specified with "next" if the variable specified in "test" is set (through [regexp](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_regexp) for example). See [conditional branching](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#branching) section for more info. | Example to jump to label "6" after sending an ACK only if variable 4 is set:  <send next="6" test="4">  <![CDATA[  ACK sip:[service]@[remote\_ip]:[remote\_port] SIP/2.0  Via ...  From ...  To ...  Call-ID ...  Cseq ...  Contact ...  Max-Forwards ...  Subject ...  Content-Length 0  ]]>  </send> |
| chance | In combination with "test", probability to actually branch to another part of the scenario. Chance can have a value between 0 (never) and 1 (always). See [conditional branching](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#branching) section for more info. | <recv response="403" optional="true" next="5" test="3" chance="0.90"> </recv>  90% chance to go to label "5" if variable "3" is set. |
| condexec | Executes an element only if the variable in the condexec attribute is set. This attribute allows you to write complex XML scenarios with fewer next attributes and labels. | <nop condexec="executethis"> |
| condexec\_inverse | If condexec is set, condexec\_inverse inverts the condition in condexec. This allows you to execute an element only when a variable is **not** set. | <nop condexec="skipthis" condexec\_inverse="true"> |
| counter | Increments the counter given as parameter when the message is sent. The counters are saved in the [statistic file](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Available+counters). | <send counter="MsgA">: Increments counter "MsgA" when the message is sent. |

Each command also has its own unique attributes, listed here:

**<send>**

| [**Attribute(s)**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=4;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=4;up=0#sorted_table) | [**Example**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=4;up=0#sorted_table) |
| --- | --- | --- |
| retrans | Used for UDP transport only: it specifies the T1 timer value, as described in SIP RFC 3261, section 17.1.1.2. | <send retrans="500">: will initiate T1 timer to 500 milliseconds (RFC3261 default). |
| lost | Emulate packet lost. The value is specified as a percentage. | <send lost="10">: 10% of the messages are not actually sent. |
| start\_txn | Records the transaction state include branch ID and cseq of this sent message so that responses can be properly matched and/or reused (without this element the transaction matching is done based on the CSeq method, which is imprecise).  start\_txn can only be used with requests (excluding ACK or Cancel).  The [cseq] keyword will produce an auto-incremented value as described in the [cseq] section. The actual cseq value used in the cseq header is the vaue stored.  [branch] will generate a unique branch value.  Transaction-aware keywords are:  [cseq\_method], [cseq], [client\_cseq\_method], [client\_cseq], [server\_cseq\_method], [server\_cseq], [received\_cseq\_method], [received\_cseq], [last\_cseq\_number], [last\_branch], [last\_Request\_URI], [last\_message] | <send start\_txn="invite">: Stores the transaction state of this message in the transaction named "invite". |
| use\_txn | Make stored transaction state available via keywords such as [cseq] and [branch]. This allows simulation of retransmissions and allows automatically correct branch generation.  use\_txn may also be used with ACK messages and [branch] will be a unique value if the last-received response code was 2xx and will be the stored value if the last-received response code as 300 or greater.  NOTE: There is no need to use ack\_txn or response\_txn: the use\_txn can be used with ACKs and responses too. | <send use\_txn="invite">: References the branch ID and cseq of the transaction named "invite". |
| ack\_txn | Indicates that the ACK being sent corresponds to the transaction started by a start\_txn attribute. Behavior is identical to use\_txn except that ack\_txn can only be used with ACK messages. | <send ack\_txn="invite">: References the branch ID and cseq of the transaction named "invite". |
| response\_txn | Indicates that this is a response to a transaction that was previously started. Behavior is identical to use\_txn except that response\_txn can only be used with response messages. | <send response\_txn="invite">: References the branch ID and cseq of the transaction named "invite". |

**<recv>**

| [**Attribute(s)**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=5;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=5;up=0#sorted_table) | [**Example**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=5;up=0#sorted_table) |
| --- | --- | --- |
| response | Indicates what SIP message code is expected. | <recv response="200">: SIPp will expect a SIP message with code "200". |
| request | Indicates what SIP message request is expected. | <recv request="ACK">: SIPp will expect an "ACK" SIP message. |
| optional | Indicates if the message to receive is optional. In case of an optional message and if the message is actually received, it is not seen as a unexpected message. When an unexpected message is received, Sipp looks if this message matches an optional message defined in the previous step of the scenario. If optional is set to "global", Sipp will look every previous steps of the scenario. | <recv response="100" optional="true">: The 100 SIP message can be received without being considered as "unexpected". |
| rrs | **R**ecord **R**oute **S**et. if this attribute is set to "true", then the "Record-Route:" header of the message received is stored and can be recalled using the **[routes]** keyword. | <recv response="100" rrs="true">. |
| auth | [Authentication](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#authentication). if this attribute is set to "true", then the "Proxy-Authenticate:" or "WWW-Authenticate" header of the message received is stored and is used to build the **[authentication]** keyword. | <recv response="407" auth="true">. |
| lost | Emulate packet lost. The value is specified as a percentage. | <recv lost="10">: 10% of the message received are thrown away. |
| timeout | Specify a timeout while waiting for a message. If the message is not received, the call is aborted, unless an ontimeout label is defined. | <recv timeout="100000"> |
| ontimeout | Specify a label to jump to if the timeout popped before the message to be received. | Example to jump to label "5" when not receiving a 100 message after 100 seconds:  <recv response="100" timeout="100000" ontimeout="5"> </recv> |
| action | Specify an action when receiving the message. See [Actions section](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#actions) for possible actions. | Example of a "regular expression" action:  <recv response="200">  <action>  <ereg regexp="([0-9]{1,3}\.){3}[0-9]{1,3}:[0-9]\*"  search\_in="msg" check\_it="true"  assign\_to="1,2"/>  </action>  </recv> |
| regexp\_match | Boolean. Indicates if 'request' ('response' is not available) is given as a regular expression. If so, the recv command will match against the regular expression. This allows to catch several cases in the same receive command. | Example of a recv command that matches MESSAGE or PUBLISH or SUBSCRIBE requests:  <recv request="MESSAGE | PUBLISH | SUBSCRIBE" crlf="true" regexp\_match="true"> </recv> |
| start\_txn | Records the transaction state include branch ID and cseq of this message so that responses are correctly generated and subsequent requests (such as ACK, Cancel and retransmissions) can be matched. Without this element the transaction matching is done based on the CSeq method, which is imprecise).  start\_txn can only be used with requests (excluding ACK or Cancel).  Transaction-specific state including the most-recently received message, the branch, cseq number and the cseq method is stored. | <recv request="INVITE" start\_txn="invite" />: Stores transaction state for use in responses and identifying subsequent requests (such as ACK, Cancel and retransmissions). |
| use\_txn | Ensure packet matches existing transaction by comparing branch.  In the case of an ACK to a successfully established INVITE dialog (in which branch does not match), SIPp will accept a unique branch for the first ACK in the transaction and will require all subsequent (retransmissions of that) ACK to have the same branch as the original ACK.  NOTE: There is no need to use ack\_txn or response\_txn: the use\_txn can be used with ACKs and responses too. | <recv response="200" use\_txn="invite" />: Verifies that response belongs to correct transaction by examinig branch.  <recv request="ACK" use\_txn="invite" />: Matches only ACK responses to the message sent with start\_txn="invite" attribute, possibly with different branch due to previous 2xx response code. |
| response\_txn | Indicates that this is a response to a transaction that was previously started. To match, the branch ID of the first via header must match the stored transaction ID. Behavior is identical to use\_txn except that response\_txn can only be used with response messages. | <recv response="200" response\_txn="invite" />: Matches only responses to the message sent with start\_txn="invite" attribute. |
| ack\_txn | Indicates that the corresponds to the transaction started by a start\_txn attribute.  Behavior is identical to use\_txn except that ack\_txn can only be used with ACK messages. | <recv request="ACK" ack\_txn="invite" />: Matches only ACK responses to the message sent with start\_txn="invite" attribute. |

**<pause>**

|  |  |  |
| --- | --- | --- |
| [**Attribute(s)**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=6;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=6;up=0#sorted_table) | [**Example**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=6;up=0#sorted_table) |
| milliseconds | Specify the pause delay, in milliseconds. When this delay is not set, the value of the -d command line parameter is used. | <pause milliseconds="5000"/>: pause the scenario for 5 seconds. |
| variable | Indicates which call variable to use to determine the length of the pause. | <pause variable="1" /> pauses for the number of milliseconds specified by call variable 1. |
| distribution | Indicates which statistical distribution to use to determine the length of the pause. Without GSL, you may use uniform or fixed. With GSL, normal, exponential, gamma, lambda, lognormal, negbin, (negative binomial), pareto, and weibull are available. Depending on the distribution you select, you must also supply distribution specific parameters. | The following examples show the various types of distributed pauses:   * <pause distribution="fixed" value="1000" /> pauses for 1 second. * <pause distribution="uniform" min="2000" max="5000"/> pauses between 2 and 5 seconds.   The remaining distributions require GSL. In general The parameter names were chosen to be as consistent with Wikipedia's distribution description pages.   * <pause distribution="normal" mean="60000" stdev="15000"/> provides a normal pause with a mean of 60 seconds (i.e. 60,000 ms) and a standard deviation of 15 seconds. The mean and standard deviation are specified as integer milliseconds. The distribution will look like: http://sipp.sourceforge.net/doc/images/dist_normal.gif * <pause distribution="lognormal" mean="12.28" stdev="1" /> creates a distribution's whose natural logarithm has a mean of 12.28 and a standard deviation of 1. The mean and standard deviation are specified as double values (in milliseconds). The distribution will look like: http://sipp.sourceforge.net/doc/images/dist_lognormal.gif * <pause distribution="exponential" mean="900000"/> creates an exponentially distributed pause with a mean of 15 minutes. The distribution will look like: http://sipp.sourceforge.net/doc/images/dist_exponential.gif * <pause distribution="weibull" lambda="3" k ="4"/> creates a Weibull distribution with a scale of 3 and a shape of 4 (see [Weibull on Wikipedia](http://en.wikipedia.org/wiki/Weibull) for a description of the distribution). * <pause distribution="pareto" k="1" x\_m="2"/> creates a Pareto distribution with k and xm of 1 and 2, respectively (see [Pareto on Wikipedia](http://en.wikipedia.org/wiki/Pareto_distribution) for a description of the distribution). * <pause distribution="gamma" k="3" theta="2"/> creates a Gamma distribution with k and theta of 9 and 2, respectively (see [Gamma on Wikipedia](http://en.wikipedia.org/wiki/Gamma_distribution) for a description of the distribution). * <pause distribution="negbin" p="0.1" n="2"/> creates a Negative binomial distribution with p and n of 0.1 and 2, respectively (see [Negative Binomial on Wikipedia](http://en.wikipedia.org/wiki/Negative_binomial_distribution) for a description of the distribution). |
| sanity\_check | By default, statistically distributed pauses are sanity checked to ensure that their 99th percentile values are less than INT\_MAX. Setting **sanity\_check** to false disables this behavior. | <pause distribution="lognormal" mean="10" stdev="10" sanity\_check="false"/> disables sanity checking of the lognormal distribution. |

**<nop>**

| [**Attribute(s)**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=7;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=7;up=0#sorted_table) | [**Example**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=7;up=0#sorted_table) |
| --- | --- | --- |
| action | The nop command doesn't do anything at SIP level. It is only there to specify an action to execute. See [Actions section](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#actions) for possible actions. | Execute the play\_pcap\_audio/video action:  <nop>  <action>  <exec play\_pcap\_audio="pcap/g711a.pcap"/>  </action>  </nop> |
| display | By default the nop command prints [ NOP ] to the sequence diagram. If you want to print a more meaningful message you can use the display attribute. | <nop display="This tag does nothing" /> |

**<sendCmd>**

| [**Attribute(s)**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=8;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=8;up=0#sorted_table) | [**Example**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=8;up=0#sorted_table) |
| --- | --- | --- |
| <![CDATA[]]> | Content to be sent to the twin [3PCC](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#ThreePCC) SIPp instance. The Call-ID must be included in the CDATA. In 3pcc extended mode, the From must be included to. | <sendCmd>  <![CDATA[  Call-ID: [call\_id]  [$1]   ]]>  </sendCmd> |
| dest | 3pcc extended mode only: the twin sipp instance which the command will be sent to | <sendCmd dest="s1">: the command will be sent to the "s1" twin instance |

**<recvCmd>**

| [**Attribute(s)**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=9;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=9;up=0#sorted_table) | [**Example**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=9;up=0#sorted_table) |
| --- | --- | --- |
| action | Specify an action when receiving the command. See [Actions section](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#actions) for possible actions. | Example of a "regular expression" to retrieve what has been send by a sendCmd command:  <recvCmd>  <action  <ereg regexp="Content-Type:.\*  " search\_in="msg"  assign\_to="2"/>  </action> &  </recvCmd> |
| src | 3pcc extended mode only: indicate the twin sipp instance which the command is expected to be received from | <recvCmd src = "s1">: the command will be expected to be received from the "s1" twin instance |

**<label>**

| [**Attribute(s)**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=10;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=10;up=0#sorted_table) | [**Example**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=10;up=0#sorted_table) |
| --- | --- | --- |
| id | A label is used when you want to branch to specific parts in your scenarios. The "id" attribute is an integer where the maximum value is 19. See [conditional branching](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#branching) section for more info. | Example: set label number 13:  <label id="13"/> |

**<Response Time Repartition>**

| [**Attribute(s)**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=11;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=11;up=0#sorted_table) | [**Example**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=11;up=0#sorted_table) |
| --- | --- | --- |
| value | Specify the intervals, in milliseconds, used to distribute the values of response times. | <ResponseTimeRepartition value="10, 20, 30"/>: response time values are distributed between 0 and 10ms, 10 and 20ms, 20 and 30ms, 30 and beyond. |

**<Call Length Repartition>**

| [**Attribute(s)**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=12;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=12;up=0#sorted_table) | [**Example**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=12;up=0#sorted_table) |
| --- | --- | --- |
| value | Specify the intervals, in milliseconds, used to distribute the values of the call length measures. | <CallLengthRepartition value="10, 20, 30"/>: call length values are distributed between 0 and 10ms, 10 and 20ms, 20 and 30ms, 30 and beyond. |

**<Globals>**

| [**Attribute(s)**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=13;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=13;up=0#sorted_table) | [**Example**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=13;up=0#sorted_table) |
| --- | --- | --- |
| variables | Specify the name of globally scoped variables. | <Globals variables="foo,bar" />. |

**<User>**

| [**Attribute(s)**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=14;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=14;up=0#sorted_table) | [**Example**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=14;up=0#sorted_table) |
| --- | --- | --- |
| variables | Specify the name of user-scoped variables. | <User variables="foo,bar" />. |

**<Reference>**

| [**Attribute(s)**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=15;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=15;up=0#sorted_table) | [**Example**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=15;up=0#sorted_table) |
| --- | --- | --- |
| variables | Suppresses warnings about unused variables. | <Reference variables="dummy" /> |

There are not so many commands: send, recv, sendCmd, recvCmd, pause, ResponseTimeRepartition, CallLengthRepartition, Globals, User, and Reference. To make things even clearer, nothing is better than an example...

**Structure of client (UAC like) XML scenarios**

A client scenario is a scenario that starts with a "send" command. So let's start:

<scenario name="Basic Sipstone UAC">

<send> <![CDATA[

INVITE sip:[service]@[remote\_ip]:[remote\_port] SIP/2.0

Via: SIP/2.0/[transport][local\_ip]:[local\_port]

From: sipp <sip:sipp@[local\_ip]:[local\_port]>;tag= [call\_number]

To: sut <sip:[service]@[remote\_ip]:[remote\_port]>

Call-ID: [call\_id]

Cseq: 1 INVITE

Contact: sip:sipp@[local\_ip]:[local\_port]

Max-Forwards: 70

Subject: Performance Test

Content-Type: application/sdp

Content-Length: [len]

v=0

o=user1 53655765 2353687637 IN IP[local\_ip\_type][local\_ip]

s=-

t=0 0

c=IN IP[media\_ip\_type][media\_ip]

m=audio [media\_port] RTP/AVP 0

a=rtpmap:0 PCMU/8000

]]>

</send>

**Keywords**

Inside the "send" command, you have to enclose your SIP message between the "<![CDATA" and the "]]>" tags. Everything between those tags is going to be sent toward the remote system. You may have noticed that there are strange keywords in the SIP message, like **[service], [remote\_ip], ...**. Those keywords are used to indicate to SIPp that it has to do something with it.

Here is the list:

**[service]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=16;up=0#sorted_table) | service |
| **Description** | Service field, as passed in the **-s service\_name** |

**[remote\_ip]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=17;up=0#sorted_table) | - |
| **Description** | Remote IP address, as passed on the command line.  optional attribute [remote\_ip no\_square\_bracket] to specify that  address is not to be surrounded by square brackets. Default is for local\_ip to be surrounded by square brackets. Attribute has no impact on ipv4 addresses |

**[remote\_port]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=18;up=0#sorted_table) | 5060 (5061 if TLS is specified as the transport protocol) |
| **Description** | Remote IP port, as passed on the command line. You can add a computed offset [remote\_port+3] to this value. |

**[transport]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=19;up=0#sorted_table) | UDP |
| **Description** | Depending on the value of **-t** parameter, this will take the values "UDP", "TCP", or "TLS". |

**[local\_ip]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=20;up=0#sorted_table) | Primary host IP address |
| **Description** | Will take the value of **-i** parameter.  optional attribute [local\_ip no\_square\_bracket] to specify that ipv6 address is not to be surrounded by square brackets. Default is for local\_ip to be surrounded by square brackets. Attribute has no impact on ipv4 addresses |

**[local\_ip\_type]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=21;up=0#sorted_table) | - |
| **Description** | Depending on the address type of **-i** parameter (IPv4 or IPv6), local\_ip\_type will have value "4" for IPv4 or "6" for IPv6. |

**[local\_port]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=22;up=0#sorted_table) | Chosen by the system (5061 if TLS is specified as the transport protocol) |
| **Description** | Will take the value of **-p** parameter. You can add a computed offset [local\_port+3] to this value. |

**[local\_ip2]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=23;up=0#sorted_table) | Secondary host IP address |
| **Description** | Will take the value of **-i2** parameter.  optional attribute [local\_ip no\_square\_bracket] to specify that ipv6 address is not to be surrounded by square brackets. Default is for local\_ip2 to be surrounded by square brackets. Attribute has no impact on ipv4 addresses |

**[local\_ip2\_type]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=24;up=0#sorted_table) | - |
| **Description** | Depending on the address type of **-i2** parameter (IPv4 or IPv6), local\_ip2\_type will have value "4" for IPv4 or "6" for IPv6. |

**[len]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=25;up=0#sorted_table) | - |
| **Description** | Computed length of the SIP body. To be used in "Content-Length" header. You can add a computed offset [len+3] to this value. |

**[call\_number]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=26;up=0#sorted_table) | - |
| **Description** | Index. The call\_number starts from "1" and is incremented by 1 for each call. |

**[cseq]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=27;up=0#sorted_table) | Initialized to 1 |
| **Description** | Automatically generates the 'correct' CSeq number. How this is calculated by SIPp depends on two factors: client vs server transaction, and if the start\_txn/use\_txn/ack\_txn attributes are specified.  For client transactions the initial value is 1 by default. It can be changed by using the -base\_cseq command line option.  For client transactions iith start\_txn or that do not specify use\_txn/ack\_txn, it is incremented for each sent transaction request except when sending an ACK or Cancel. In this case the value of [cseq] is identical to the value of [client\_cseq].  With use\_txn/ack\_txn the cseq number associated with the existing transaction is used. This allows simulated retransmission of requests and overlapping transactions.  For server transactions (ie response codes) it is the value associated with the last received request (in that dialog and transaction, if specified), and is equivelent to [server\_cseq] |

**[cseq\_method]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=28;up=0#sorted_table) | - |
| **Description** | CSeq Method, tracked separately for client and server transactions.  The actual value depends on two factors: client vs server transaction, and if the start\_txn/use\_txn/ack\_txn attributes are specified.  For client transactions the initial value is empty by default and contains the cseq method once a message has been generated or received. The value is identical in this case to [client\_cseq\_method]. For server transactions it is the cseq method of the last received message and is identical in this case to [server\_cseq\_method] and [received\_cseq\_method].  When used with use\_txn/ack\_txn/response\_txn, it is the value associated with the named transaction.  When used with start\_txn it is the value associated with the PREVIOUS transaction of the same name (or blank initially). |

**[call\_id]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=29;up=0#sorted_table) | - |
| **Description** | A call\_id identifies a call and is usually generated by SIPp for each new call.SIPp will generate a unique call-id for each distinct dialog.  In functional mode (-mc) all calls are routed to the one instance of SIPp regardless of call-id and are identified by the 'dialog' attribute.  In scalability mode (without -mc), when in client mode, it is mandatory to use the value generated by SIPp in the "Call-ID" header. Otherwise, SIPp will not recognise the answer to the message sent as being part of an existing call. Note: [call\_id] can be pre-pended with an arbitrary string using '///'. Example: Call-ID: ABCDEFGHIJ///[call\_id] - it will still be recognized by SIPp as part of the same call. |

**[media\_ip]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=30;up=0#sorted_table) | - |
| **Description** | Depending on the value of **-mi** parameter, it is the local IP address for RTP echo.  optional attribute [media\_ip no\_square\_bracket] to specify that ipv6 address is not to be surrounded by square brackets. Default is for local\_ip to be surrounded by square brackets. Attribute has no impact on ipv4 addresses |

**[media\_ip\_type]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=31;up=0#sorted_table) | - |
| **Description** | Depending on the address type of **-mi** parameter (IPv4 or IPv6), media\_ip\_type will have value "4" for IPv4 and "6" for IPv6. Useful to build the SDP independently of the media IP type. |

**[media\_port]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=32;up=0#sorted_table) | - |
| **Description** | Depending on the value of **-mp** parameter, it set the local RTP echo port number. Default is none. RTP/UDP packets received on that port are echoed to their sender. You can add a computed offset [media\_port+3] to this value. |

**[auto\_media\_port]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=33;up=0#sorted_table) | - |
| **Description** | Only for pcap. To make audio and video ports begin from the value of **-mp** parameter, and change for each call using a periodical system, modulo 10000 (which limits to 10000 concurrent RTP sessions for play\_pcap) |

**[last\_\*]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=34;up=0#sorted_table) | - |
| **Description** | The '[last\_\*]' keyword is replaced automatically by the specified header if it was present in the last message received (except if it was a retransmission). If the header was not present or if no message has been received, the '[last\_\*]' keyword is discarded, and all bytes until the end of the line are also discarded. If the specified header was present several times in the message, all occurences are concatenated (CRLF separated) to be used in place of the '[last\_\*]' keyword.  Several keywords have compact forms (e.g.the via header can be represented by "Via:" or "v:", see [RFC 3261](http://www.ietf.org/rfc/rfc3261.txt) for more details). For these headers you may use either the compact form or the the normal form of the header as the \* in '[last\_\*]', SIPp will search for both in the message. SIPp will then use "\*:value\_of\_header" in place of '[last\_\*]', where "\*" is the inputed value in '[last\_\*]', and "value\_of\_header" is the actual value of the header.  '[last\_\*]' also has an optional "value\_only" attribute. If you set this attribute to true then '[last\_\*]' will be replaced with the value of the header, rather that the entire header (eliminating the header name). |
| **Examples** | **Compact Form:**  Incoming message contains: "Call-ID: 1234"  Scenario contains: "[last\_i:]"  SIPp sends: "i: 1234"  **Value Only:**  Incoming message contains: "Call-ID: 1234"  Scenario contains: "HEADER\_NAME: [last\_Call-id: value\_only="true"]"  SIPp sends: "HEADER\_NAME: 1234" |

**[field0-n file=<filename> line=<number>]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=35;up=0#sorted_table) | - |
| **Description** | Used to inject values from an external CSV file. See ["Injecting values from an external CSV during calls"](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#inffile) section. The optional file and line parameters allow you to select which of the injection files specified on the command line to use and which line number from that file. |

**[file name=<filename>]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=36;up=0#sorted_table) | - |
| **Description** | Inserts the entire contents of filename into the message. Whitespace, including carriage returns and newlines at the end of the line in the file are not processed as with other keywords; thus your file must be formatted exactly as you would like the bytes to appear in the message. |

**[timestamp]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=37;up=0#sorted_table) | - |
| **Description** | The current time using the same format as error log messages. |

**[last\_message]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=38;up=0#sorted_table) | - |
| **Description** | The last received message. |

**[$n]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=39;up=0#sorted_table) | - |
| **Description** | Used to inject the value of call variable number n. See "[Actions](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#actions)" section |

**[authentication]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=40;up=0#sorted_table) | - |
| **Description** | Used to put the authentication header. This field can have parameters, in the following form: [authentication username=myusername password=mypassword]. If no username is provided, the value from -s command line parameter (service) is used. If no password is provided, the value from -ap command line parameter is used. See "[Authentication](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#authentication)" section |

**[pid]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=41;up=0#sorted_table) | - |
| **Description** | Provide the process ID (pid) of the main SIPp thread. |

**[routes]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=42;up=0#sorted_table) | - |
| **Description** | If the "rrs" attribute in a recv command is set to "true", then the "Record-Route:" header of the message received is stored and can be recalled using the [routes] keyword |

**[next\_url]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=43;up=0#sorted_table) | - |
| **Description** | If the "rrs" attribute in a recv command is set to "true", then the [next\_url] contains the contents of the Contact header (i.e within the '<' and '>' of Contact) as in as in: **sip:Extension@IP\_Address** **:Port**.  [contact\_uri] is similr except contains contact header from previous message and does not require the "rrs" attribute. |

**[branch]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=44;up=0#sorted_table) | - |
| **Description** | Provide a branch value which is a concatenation of magic cookie (z9hG4bK) + call number + message index in scenario. An offset (like [branch-N]) can be appended if you need to have the same branch value as a previous message. |

**[msg\_index]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=45;up=0#sorted_table) | - |
| **Description** | Provide the message number in the scenario. |

**[clock\_tick]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=46;up=0#sorted_table) | - |
| **Description** | Includes the internal SIPp clock tick value in the message. |

**[sipp\_version]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=47;up=0#sorted_table) | - |
| **Description** | Includes the SIPp version string in the message. |

**[tdmmap]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=48;up=0#sorted_table) | - |
| **Description** | Includes the tdm map values used by the call in the message (see -tdmmap option). |

**[fill]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=49;up=0#sorted_table) | - |
| **Description** | Injects filler characters into the message. The length of the fill text is equal to the call variable stored in the variable=N parameter. By default the text is a sequence of X's, but can be controlled with the text="text" parameter. |

**[users]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=50;up=0#sorted_table) | - |
| **Description** | If the -users command line option is specified, then this keyword contains the number of users that are currently instantiated. |

**[userid]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=51;up=0#sorted_table) | - |
| **Description** | If the -users command line option is specified, then this keyword containst he integer identifier of the current user (starting at zero and ending at [users-1]). |

**[remote\_tag\_param]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=52;up=0#sorted_table) | - |
| **Description** | The remote tag as per RFC 3261 including the tag= portion (identical to peer\_tag\_param). This value is updated whenver a message is reeived. It represents the 'from' tag in the most recent reply response or the 'to' tag in the most recent request. |

**[remote\_tag]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=53;up=0#sorted_table) | - |
| **Description** | The remote tag as per RFC 3261 without the tag= portion. This value is updated whenver a message is received. It represents the 'from' tag in the most recent reply response or the 'to' tag in the most recent request. |

**[local\_tag\_param]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=54;up=0#sorted_table) | - |
| **Description** | The local tag as per RFC 3261 including the tag= portion. This value is updated whenver a message is received. It represents the 'to' tag in the most recent reply response or the 'from' tag in the most recent request. If used before a message specifying a local tag has been received for the dialog SIPp will auto-generate the value 'local-PID-NUMBER-IDX' (with PID, NUMBER and IDX substituted to create a unique value). |

**[local\_tag]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=55;up=0#sorted_table) | - |
| **Description** | The local tag as per RFC 3261. This value is updated whenver a message is received. It represents the 'to' tag in the most recent reply response or the 'from' tag in the most recent request. If used before a message specifying a local tag has been received for the dialog SIPp will auto-generate the value 'local-PID-NUMBER-IDX' (with PID, NUMBER and IDX substituted to create a unique value). |

**[contact\_uri]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=56;up=0#sorted_table) | - |
| **Description** | The URI from the most-recently received 'Contact' headeras in: **sip:Extension@IP\_Address** **:Port** |

**[contact\_name\_and\_uri]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=57;up=0#sorted_table) | - |
| **Description** | The name and URI portion of the most-recently received 'Contact' header as in: **"SIPp" <sip:Extension@IP\_Address:Port>** |

**[to\_uri]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=58;up=0#sorted_table) | - |
| **Description** | The URI from the most-recently received 'To' header as in: **sip:Extension@IP\_Address** **:Port** |

**[to\_name\_and\_uri]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=59;up=0#sorted_table) | - |
| **Description** | The name and URI from the most-recently received 'To' header as in: **"SIPp" <sip:Extension@IP\_Address** **:Port>** |

**[from\_uri]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=60;up=0#sorted_table) | - |
| **Description** | The URI from the most-recently received 'From' header as in: **sip:Extension@IP\_Address** **:Port** |

**[from\_name\_and\_uri]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=61;up=0#sorted_table) | - |
| **Description** | The name and URI from the most-recently received 'From' header as in: **"SIPp" <sip:Extension@IP\_Address** **:Port>** |

**[last\_cseq\_number]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=62;up=0#sorted_table) | - |
| **Description** | The CSeq value of the last message received. This value can be incremented (e.g. [cseq+1] adds 1 to the CSeq value of the last message). |

**[last\_branch]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=63;up=0#sorted_table) | - |
| **Description** | The branch id of the last message received. |

**[last\_Request\_URI]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=64;up=0#sorted_table) | - |
| **Description** | The request URI as obtained from the To header of the previously received message. |

**[client\_cseq]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=65;up=0#sorted_table) | - |
| **Description** | The CSeq number of the last client transaction. For client transactions, identical to [cseq].  NOTE: [cseq] is usually preferable, but is available for those strange situations when you need to access the client\_cseq number from a server transaction.  Can not be used within a named client transactions. |

**[client\_cseq\_method]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=66;up=0#sorted_table) | - |
| **Description** | The CSeq method of the last client transaction. For client transactions, identical to [cseq\_method].  NOTE: [cseq] is usually preferable, but is available for those strange situations when you need to access the client\_cseq method from a server transaction.  Can not be used within a named server transactions. |

**[server\_cseq]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=67;up=0#sorted_table) | - |
| **Description** | The CSeq number of the last received request. Identical to [cseq] for server transactions. Always identical to [received\_cseq].  NOTE: [cseq] is usually preferable, but is available for those strange situations when you need to access the server cseq from a client transaction.  Can not be used within a named client transactions. |

**[server\_cseq\_method]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=68;up=0#sorted_table) | - |
| **Description** | The CSeq method of the last received request. Identical to [cseq\_method] for server transactions. Always identical to [received\_cseq\_method].  NOTE: [cseq] is usually preferable, but is available for those strange situations when you need to access the server cseq method from a client transaction.  Can not be used within a named client transactions. |

**[received\_cseq]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=69;up=0#sorted_table) | - |
| **Description** | The CSeq number of the last received request. Identical to [cseq] for server transactions. Always identical to [server\_cseq].  NOTE: [cseq] is usually preferable, but is available for those strange situations when you need to access the the server cseq from a client transaction.  Can not be used within a named client transactions. |

**[received\_cseq\_method]**

|  |  |
| --- | --- |
| [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=70;up=0#sorted_table) | - |
| **Description** | The CSeq method of the last received request. Identical to [cseq\_method] for server transactions. Always identical to [server\_cseq\_method] .  NOTE: [cseq] is usually preferable, but is available for those strange situations when you need to access the server cseq method from a client transaction.  Can not be used within a named client transactions. |

**Encoding**

SIPp automatically decodes strings of the form "\x##" in the body of any message being sent as the corresponding ASCII character, where ## is the corresponding hexidecimal ASCII code. This is useful if you wish to send characters which can not be easily typed into the message.

SIPp also decodes XML protected characters (i.e. < > & ' ") from their standard encoded form (e.g. < = <), whenever the encoding is encountered inside a attribute. See [wikipedia](http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references#Predefined_entities_in_XML) for the complete list of encodings.

The 'encoding' attribute is used to tell SIPp to send keyword data out encoded in a different form than normal. For example if you want to send possible deliminators in a form such that they will be taken literally you can use URI-encoding. URI-encoding is currently avaliable for the following parameters:

* cseq\_method
* cseq
* client\_cseq\_method
* client\_cseq
* server\_cseq\_method
* server\_cseq
* received\_cseq\_method
* received\_cseq
* last\_cseq\_number
* last\_branch
* last\_Request\_URI
* last\_message
* call\_id
* next\_url
* routes
* peer\_tag\_param
* remote\_tag\_param
* remote\_tag
* local\_tag\_param
* local\_tag
* contact\_uri
* contact\_name\_and\_uri
* to\_uri
* to\_name\_and\_uri
* from\_uri
* from\_name\_and\_uri

**Example**:

Incoming Call-Id is [123@89098-09](mailto:123@89098-09)

In scenario file message has [call\_id encoding="uri"]

Output: Call-Id: 123%6089098-09

**Receive**

Now that the INVITE message is sent, SIPp can wait for an answer by using the "[recv](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#recv)" command.

<recv response="100" optional="true">

</recv>

<recv response="180" optional="true">

</recv>

<recv response="200">

</recv>

100 and 180 messages are optional, and 200 is mandatory. **In a "recv" sequence, there must be one mandatory message**.

Now, let's send the ACK:

<send>

<![CDATA[

ACK sip:[service]@[remote\_ip]:[remote\_port] SIP/2.0

Via: SIP/2.0/[transport] [local\_ip]:[local\_port]

From: sipp <sip:sipp@[local\_ip]:[local\_port]>;tag= [call\_number]

To: sut <sip:[service]@[remote\_ip]:[remote\_port]>[peer\_tag\_param]

Call-ID: [call\_id]

Cseq: 1 ACK

Contact: sip:sipp@[local\_ip]:[local\_port]

Max-Forwards: 70

Subject: Performance Test

Content-Length: 0

]]>

</send>

We can also insert a pause. The scenario will wait for 5 seconds at this point. <pause milliseconds="5000"/>

And finish the call by sending a BYE and expecting the 200 OK:

<send retrans="500">

<![CDATA[

BYE sip:[service]@[remote\_ip]:[remote\_port] SIP/2.0

Via: SIP/2.0/[transport] [local\_ip]:[local\_port]

From: sipp <sip:sipp@[local\_ip]:[local\_port]>;tag= [call\_number]

To: sut <sip:[service]@[remote\_ip]:[remote\_port]>[peer\_tag\_param]

Call-ID: [call\_id]

Cseq: 2 BYE

Contact: sip:sipp@[local\_ip]:[local\_port]

Max-Forwards: 70

Subject: Performance Test

Content-Length: 0

]]>

</send>

<recv response="200">

</recv>

And this is the end of the scenario:

</scenario>

Creating your own SIPp scenarios is not a big deal. If you want to see other examples, use the -sd parameter on the command line to display embedded scenarios.

**Structure of server (UAS like) XML scenarios**

A server scenario is a scenario that starts with a "[recv](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#recv)" command. The syntax and the list of available commands is the same as for "client" scenarios.

But you are more likely to use [last\_\*] keywords in those server side scenarios. For example, a UAS example will look like:

<recv request="INVITE">

</recv>

<send>

<![CDATA[

SIP/2.0 180 Ringing [last\_Via:]

[last\_From:]

[last\_To:];tag= [call\_number]

[last\_Call-ID:]

[last\_CSeq:]

Contact: <sip:[local\_ip]:[local\_port];transport=[transport]>

Content-Length: 0

]]>

</send>

The answering message, 180 Ringing in this case, is built with the content of headers received in the INVITE message.

**Actions**

In a "[recv](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#recv)" or "[recvCmd](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#recvCmd)" command, you have the possibility to execute an action. Several actions are available:

* [Regular expressions](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_regexp) (ereg)
* [Log something in aa log file](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_log) (log)
* [Execute an external (system), internal (int\_cmd) or play\_pcap\_audio/play\_pcap\_video/play\_pcap\_application command](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#Execute_a_command) (exec)
* [Manipulate double precision variables using arithmetic](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_variables)
* [Assign string values to a variable](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_strings)
* [Compare double precision variables](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_test)
* [Jump to a particular scenario index](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_jump)
* [Store the current time into variables](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_gettimeofday)
* [Lookup a key in an indexed injection file](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_lookup)
* [Verify Authorization credentials](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_verifyauth)
* [Change a Call's Network Destination](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_setdest)

**Regular expressions**

Using regular expressions in SIPp allows to

* Extract content of a SIP message or a SIP header and store it for future usage (called re-injection)
* Check that a part of a SIP message or of an header is matching an expected expression

Regular expressions used in SIPp are defined per [Posix Extended standard (POSIX 1003.2)](http://www.opengroup.org/onlinepubs/007908799/xbd/re.html). If you want to learn how to write regular expressions, I will recommend this [regexp tutorial](http://analyser.oli.tudelft.nl/regex/index.html.en).

Here is the syntax of the regexp action:

| [**Keyword**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=0;table=71;up=0#sorted_table) | [**Default**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=1;table=71;up=0#sorted_table) | [**Description**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide?sortcol=2;table=71;up=0#sorted_table) |
| --- | --- | --- |
| regexp | None | Contains the regexp to use for matching the received message or header. MANDATORY. |
| search\_in | msg | can have four values: "msg" (try to match against the entire message); "hdr" (try to match against a specific SIP header); "body" (try to match against the SIP message body); or "var" (try to match against a SIPp string variable). |
| header | None | Header to try to match against. Only used when the search\_in tag is set to hdr. MANDATORY IF search\_in is equal to hdr. Note that SIPp will look for the header as specified, and will NOT check the short form of the header. |
| variable | None | Variable to try to match against. Only used when the search\_in tag is set to var. MANDATORY IF search\_in is equal to var. |
| case\_indep | false | To look for a header ignoring case . Only used when the search\_in tag is set to hdr. |
| occurence | 1 | To find the nth occurence of a header. Only used when the search\_in tag is set to hdr. |
| start\_line | false | To look only at start of line. Only used when the search\_in tag is set to hdr. |
| check\_it | false | if set to true, the call is marked as failed if the regexp doesn't match. Can not be combined with check\_it\_inverse. |
| check\_it\_inverse | false | Inverse of check\_it. iff set to true, the call is marked as failed if the regexp does match. Can not be combined with check\_it. |
| assign\_to | None | contains the variable id or a list of variable id which will be used to store the result(s) of the matching process between the regexp and the message. Those variables can be re-used at a later time either by using '[$n]' in the scenario to inject the value of the variable in the messages or by using the content of the variables for [conditional branching](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#branching). The first variable in the variable list of assign\_to contains the entire regular expression matching. The following variables contain the sub-expressions matching. |

Note that you can have several regular expressions in one action.

The following example is used to:

* First action:
  + Extract the first IPv4 address of the received SIP message
  + Check that we could actually extract this IP address (otherwise call will be marked as failed)
  + Assign the extracted IP address to call variables 1 and 2.
* Second action:
  + Extract the Contact: header of the received SIP message
  + Assign the extracted Contract: header to variable 6.

<recv response="200" start\_rtd="true">

<action>

<ereg regexp="([0-9]{1,3}\.){3}[0-9]{1,3}:[0-9]\*" search\_in="msg" check\_it="true" assign\_to="1,2" />

<ereg regexp=".\*" search\_in="hdr" header="Contact:" check\_it="true" assign\_to="6" />

</action>

</recv>

**Log a message**

The "log" action allows you to customize your traces. Messages are printed in the file. Any [keyword](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#keyword) is expanded to reflect the value actually used. Logs are generated only if -trace\_logs option is set on the command line.

Example:

<recv request="INVITE" crlf="true" rrs="true">

<action>

<ereg regexp=".\*" search\_in="hdr" header="Some-New-Header:" assign\_to="1" />

<log message="From is [last\_From]. Custom header is [$1]"/>

</action>

</recv>

You can use the alternative "warning" action to log a message to SIPp's error log. For example:

<warning message="From is [last\_From]. Custom header is [$1]"/>

**Execute a command**

The "exec" action allows you to execute "internal", "external", "play\_pcap\_audio" or "play\_pcap\_video" commands.

**Internal commands**

**Internal** commands (specified using int\_cmd attribute) are stop\_call, stop\_gracefully (similar to pressing 'q'), stop\_now (similar to ctrl+C).

Example that stops the execution of the script on receiving a 603 response:

<recv response="603" optional="true">

<action>

<exec int\_cmd="stop\_now"/>

</action>

</recv>

**External commands**

**External** commands (specified using command attribute) are anything that can be executed on local host with a shell.

Example that execute a system echo for every INVITE received:

<recv request="INVITE"> <action> <exec command="echo [last\_From] is the from header received >> from\_list.log"/> </action> </recv>

**PCAP (media) commands**

**PCAP play** commands (specified using play\_pcap\_audio / play\_pcap\_video attributes) allow you to send a pre-recorded RTP stream using the [pcap library](http://www.tcpdump.org/pcap3_man.html). Only udp based media is supported.

Choose **play\_pcap\_audio** to send the pre-recorded RTP stream using the "m=audio" SIP/SDP line port as a base for the replay.

Choose **play\_pcap\_video** to send the pre-recorded RTP stream using the "m=video" SIP/SDP line port as a base.

Choose **play\_pcap\_application** to send the pre-recorded RTP stream using the "m=application" SIP/SDP line port as a base.

The play\_pcap\_audio/video/application command can have the following format: play\_pcap\_audio="[file\_to\_play]" media\_port\_offset="[offset]" with:

* file\_to\_play: the pre-recorded pcap file to play
* media\_port\_offset: if you wish to send media before SDP negotiation takes place, or stream media from several different ports you may use an offset from the number in media\_port. If no SDP messages have been sent the offset will be applied to a default value of 6000 (note that SIPp will use port 0 if no offset is specified). The offset is formatted as +/- followed by some number of digits, if a port has been specified in some SDP message the port used for the streaming will be media\_port + offset (where offset may be negative). The action is non-blocking. SIPp will start a light-weight thread to play the file and the scenario with continue immediately. If needed add a pause to wait for the end of the pcap play.

Example that plays a pre-recorded RTP stream:

<nop>

<action>

<exec play\_pcap\_audio="pcap/g711a.pcap"/>

</action>

</nop>

Example that plays a pre-recorder RTP stream, on port media\_port - 10:

<nop>

<action>

<exec play\_pcap\_audio="pcap/g711b.pcap" media\_port\_offset="-10"/>

</action>

</nop>

For handling of multiple simultaneous media streams (multiple m= lines in the sdp), the play\_pcap\_[audio|video|application] can have the index and/or source\_ip attributes. The index is one based per media type counter (ie: 1st audio stream, 2nd audio stream, 1st video stream). The index is automatically inserted by snipp processed pcap files and ties the scenario outgoing sdp (for 'from ports') to the incoming sdp during sipp playback (for 'to' ip address and port) to the specific pcap playback file. If the index attribute is not present, a value of 1 is used (only the first occurence in an sdp of each media type is used).

For use of multiple source ip addresses for media streams, the command line arg -i2 sets [local\_ip2] value that can be refered to inside a sipp scenario file. That address can also be used by play\_pcap to change the source ip address to the second interface by assigning attribute source\_ip="2". This places the onus on the user to ensure that the -i2 address is valid on the sipp playback host and is compatible with the far end expected address family. The index for given media type within the far ends sdp message sent to sipp must have a contact line for this media that matches the address family of the local\_ip2. SDP with multiple contact lines use the session level contact line as the default(if any) and overrides it with any per media contact line addresses (if any). -i and -i2 addresses do not have to be from the same address family so that ipv6 media can be played back from a sipp host using ipv4 for sip signalling, If source\_ip attribute is not present, default value is 1 (-i address is used).

if source\_ip="0" then sipp will try to bind to any local host address that it can find in the same address family as the destination.

<nop>

<action>

<exec play\_pcap\_audio="pcap/g711b.pcap" index="1" source\_ip="2"/>

</action>

</nop>

**Including files (and building reusable SIPp modules)**

<xi:include href="afile.xml" dialogs="2,3"/>

This tag will perform **Substitution** and insert the contents of afile.xml at the location of this tag. The 'dialogs' tag may be used to specify a list of dialog IDs to substitute in the included file, allowing the creation of re-usable libraries.

The format must precisely match the the first 18 chars (exactly one space between include and href) as the opening of the tag and exactly match the last two characters as the closing of the tag (no whitespace permitted). If the dialogs attribute is present, no whitespace is permitted within the dialog attribute value assignment. Variable amount of whitespace is only permitted before the dialogs attribute.

**Include File Search Order**

1. The file is first attempted to be opened using exactly the name specified (which will search relative to the current working directory).
2. relative path specification to the file will be recognized, eg the double dot notation for parent directory (../filename.xml) and paths relative to the sipp file location ( dirname/filename.xml )
3. environment variables can be used to specify file locations where the environment variable is surrounded by % characters (eg <xi:include href="%TA\_DIR%/SIPped/SIPped/src/test/include\_file.xml dialogs="99,3,1"/> )

**Debugging**

1. Use the **-dump\_xm**l parameter to print to console the final XML (with all includes expanded).
2. Use the **-dump\_sequence\_diagram** to print to console the entire sequence diagram (with all includes expanded).

**Include Files**

The include files themselves should end in .xml and must contain the <?xml ...?>, <!DOCTYPE ...> and <scenario> tags. They will be inserted into the including script as if the file contents were literally copy & pasted in place of the <xi:include> line (except for dialog="NN" substitution). An include file may contain another include file, and substitution rules will apply in the included files as well.

For example, if main.sipp contains

<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="Main" parameters="-mc">

<xi:include href="first.xml" dialogs="1,2"/>

</scenario>

first.xml contains:

<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="Main" parameters="-mc">

<recv request="REGISTER" dialog="AA" />

<xi:include href="second.xml" dialogs="BB"/>

</scenario>

second.xml contains:

<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="Main" parameters="-mc">

<recv request="NOTIFY" dialog="AA" />

</scenario>

This would execute the same as if main.sipp contained:

<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="Main" parameters="-mc">

<recv request="REGISTER" dialog="1" />

<recv request="NOTIFY" dialog="2" />

</scenario>

**dialogs=NN substitutions**

The format is dialogs="9,6,1" with no whitespace permitted. The numbers correspond to the dialog IDs referenced in the included file. The first dialog ID specified will replace the all instances of dialog="AA" (dialog="09" in the above example). The second dialog ID will replace dialog="BB" (dialog="06"), etc up to a maximum of 26 entries in which dialog="ZZ" would be substituted. The maximum dialog ID allowed is 99.

It is an error for an file using the dialog="NN" syntax to be included with an insufficient number of dialog IDs passed in. The intention is to allow the creation of reusable modules to perform common tasks, such as BLA\_Register\_And\_Subscribe\_AA\_BB\_CC.xml (which uses 3 dialog IDs and performs registration and bi-directional BLA subscription and can be re-used by any test requiring this functionality).

**Variable Manipulation**

You may also perform simple arithmetic (add, subtract, multiply, divide) on floating point values. The "assign\_to" attribute contains the first operand, and is also the destination of the resulting value. The second operand is either an immediate value or stored in a variable, represented by the "value" and "variable" attributes, respectively.

SIPp supports call variables that take on double-precision floating values. The actions that modify double variables all write to the variable referenced by the **assign\_to** parameter. These variables can be assigned using one of three actions: assign, sample, or todouble. For assign, the double precision value is stored in the "value" parameter. The sample action assigns values based on statistical distributions, and uses the same parameters as a [statistically distributed pauses](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#pause_distributions). Finally, the todouble command converts the variable referenced by the "variable" attribute to a double before assigning it.

For example, to assign the value 1.0 to $1 and sample from the normal distribution into $2:

<nop>

<action>

<assign assign\_to="1" value="1" />

<sample assign\_to="2" distribution="normal" mean="0" stdev="1"/>

<!-- Stores the first field in the injection file into string variable $3.

You may also use regular expressions to store string variables. -->

<assignstr assign\_to="3" value="[field0]" />

<!-- Converts the string value in $3 to a double-precision value stored in $4. -->

<todouble assign\_to="4" variable="3" />

</action>

</nop>

Simple arithmetic is also possible using the **<add>**, **<subtract>**, **<multiply>**, and **<divide>** actions, which add, subtract, multiply, and divide the variable referenced by **assign\_to** by the value in **value**. For example, the following action modifies variable one as follows:

<nop>

<action>

<assign assign\_to="1" value="0" /> <!-- $1 == 0 -->

<add assign\_to="1" value="2" /> <!-- $1 == 2 -->

<subtract assign\_to="1" value="3" /> <!-- $1 == -1 -->

<multiply assign\_to="1" value="4" /> <!-- $1 == -4 -->

<divide assign\_to="1" value="5" /> <!-- $1 == -0.8 -->

</action>

Rather than using fixed values, you may also retrieve the second operand from a variable, using the **<variable>** parameter. For example:

<nop>

<action>

<!-- Multiplies $1 by itself -->

<multiply assign\_to="1" variable="1" />

<!-- Divides $1 by $2, Note that $2 must not be zero -->

<divide assign\_to="1" variable="2" />

</action>

</nop>

**String Variables**

You can create string variables by using the **<assignstr>** command, which accepts two parameters: **assign\_to** and **value**. The value may contain any of the same substitutions that a message can contain. For example:

<nop>

<action>

<!-- Assign the value in field0 of the CSV file to a $1. -->

<assignstr assign\_to="1" value="[field0]" />

</action>

</nop>

A string variable and a value can be compared using the **<strcmp>** action. The result is a double value, that is less than, equal to, or greater than zero if the variable is lexographically less than, equal to, or greater than the value. The parameters are assign\_to, variable, check\_it, check\_it\_inverse and value. check\_it="true" will abort the script and report FAIL immediately if the strings are not equal.

For example:

<nop>

<action>

<!-- Compare the value of $strvar to "Hello" and assign it to $result.. -->

<strcmp assign\_to="result" variable="strvar" value="Hello" check\_it="true" />

</action>

</nop>;

The <**trim**> action can be used to remove any excess whitespace from the variable.

<nop>

<action>

<trim assign\_to="strvar" />

</action>

</nop>;

**Variable Testing**

Variable testing allows you to construct loops and control structures using call variables. THe **test** action takes serveral arguments. In the looping case you have **variable** which is the variable that to **compare** against **value** (or **variable2**), and **assign\_to** which is a boolean call variable that the result of the test is stored in. Compare may be one of the following tests: **equal**, **not\_equal**, **greater\_than**, **less\_than**, **greater\_than\_equal**, or **less\_than\_equal**.

Example that sets $2 to true if $1 is less than 10:

<nop>

<action>

<test assign\_to="2" variable="1" compare="less\_than" value="10" />

</action>

</nop>;

You can also use variable testing to check whether or not certain condition is met, and if it is not the program will fail. The arguments are similar to above, except instead of **assign\_to** use **check\_it** set to true to tell SIPp to check the result of the test immediately.

Example testing that CSEQ numbers increment correctly (the majority of the content of the sent messages has been removed to simplify the example):

<send dialog="1" start\_txn="C1">

<![CDATA[

...

CSeq: [cseq] INVITE

...

]]>

</send>

<recv response="100" dialog="1" use\_txn="C1">

<action>

<ereg regexp="[0-9]+" search\_in="hdr" header="CSeq:" check\_it="true" assign\_to="firstcseq"/>

</action>

</recv>

<send dialog="1" start\_txn="C2">

<![CDATA[

...

CSeq: [cseq] INVITE

...

]]>

</send>

<recv response="100" dialog="1" use\_txn="C2">

<action>

<add assign\_to="firstcseq" value="1"/>

<ereg regexp="[0-9]+" search\_in="hdr" header="CSeq:" check\_it="true" assign\_to="secondcseq"/>

<test variable="firstcseq" compare="equal" variable2="secondcseq" check\_it="true"/>

</action>

</recv>

**lookup**

The lookup action is used for indexed injection files (see [indexed injection files](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#infindex)). The lookup action takes a file and key as input and produces an integer line number as output. For example the following action extracts the username from an authorization header and uses it to find the corresponding line in users.csv.

<recv request="REGISTER">

<action>

<ereg regexp="Digest .\*username=\"([^\"]\*)\"" search\_in="hdr" header="Authorization:" assign\_to="junk,username" />

<lookup assign\_to="line" file="users.csv" key="[$username]" />

</action>

</nop>

**Updating In-Memory Injection files**

Injection files, particularly when an [index](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#infindex) is defined can serve as an in-memory data store for your SIPp scenario. The <insert> and <replace> actions provide a method of programmatically updating SIPp's in-memory version of an injection file (there is presently no way to update the disk-based version). The insert action takes two parameters: file and value, and the replace action takes an additional line value. For example, to inserting a new line can be accomplished as follows:

<nop display="Insert User">

<action>

<insert file="usersdb.conf" value="[$user];[$calltype]" />

</action>

</nop>

Replacing a line is similar, but a line number must be specified. You will probably want to use the lookup action to obtain the line number for use with replace as follows:

<nop display="Update User">

<action>

<lookup assign\_to="index" file="usersdb.conf" key="[$user]" />

<!-- Note: This assumes that the lookup always succeeds. -->

<replace file="usersdb.conf" line="[$index]" value="[$user];[$calltype]" />

</action>

</nop>

**Jumping to an Index**

You can jump to an arbitrary scenario index using the <jump> action. This can be used to create rudimentary subroutines. The caller can save their index using the [msg\_index] substitution, and the callee can jump back to the same place using this action. If there is a special label named "\_unexp.main" in the scenario, SIPp will jump to that label whenever an unexpected message is received and store the previous address in the variable named "\_unexp.retaddr".

Example that jumps to index 5:

<nop>

<action>

<jump value="5" />

</action>

</nop>

Example that jumps to the index contained in the variable named \_unexp.retaddr:

<nop>

<action>

<jump variable="\_unexp.retaddr" />

</action>

</nop>

**gettimeofday**

The gettimeofday action allows you to get the current time in seconds and microseconds since the epoch. For example:

<nop>

<action>

<gettimeofday assign\_to="seconds,microseconds" />

</action>

</nop>

**setdest**

The setdest action allows you to change the remote end point for a call. The parameters are the transport, host, and port to connect the call to. There are certain limitations baed on SIPp's design: you can not change the transport for a call; and if you are using TCP then multi-socket support must be selected (i.e. -t tn must be specified). Also, be aware that frequently using setdest may reduce SIPp's capacity as name resolution is a blocking operation (thus potentially causing SIPp to stall while looking up host names). This example connects to the value specified in the [next\_url] keyword.

<nop>

<action>

<assignstr assign\_to="url" value="[next\_url]" />

<ereg regexp="sip:.\*@([0-9A-Za-z\.]+):([0-9]+);transport=([A-Z]+)" search\_in="var" check\_it="true" assign\_to="dummy,host,port,transport" variable="url" />

<setdest host="[$host]" port="[$port]" protocol="[$transport]" />

</action>

</nop>

**verifyauth**

The verifyauth action checks the Authorization header in an incoming message against a provided username and password. The result of the check is stored in a boolean variable. This allows you to simulate a server which requires authorization. Currently only simple MD5 digest authentication is supported. Before using the verifyauth action, you must send a challenge. For example:

<recv request="REGISTER" />

<send>

<![CDATA[

SIP/2.0 401 Authorization Required

[last\_Via:]

[last\_From:]

[last\_To:];tag=[pid]SIPpTag01[call\_number]

[last\_Call-ID:] [last\_CSeq:]

Contact:

WWW-Authenticate: Digest realm="test.example.com", nonce="47ebe028cda119c35d4877b383027d28da013815"

Content-Length: [len]

]]>

</send]]>

After receiving the second request, you can extract the username provided and compare it against a list of user names and passwords provided as an injection file, and take the appropriate action based on the result:

<recv request="REGISTER" />

<action>

<ereg regexp="Digest .\*username=\"([^\"]\*)\"" search\_in="hdr" header="Authorization:" assign\_to="junk,username" />

<lookup assign\_to="line" file="users.conf" key="[$username]" />

<verifyauth assign\_to="authvalid" username="[field0 line=\"[$line]\"]" password="[field3 line=\"[$line]\"]" />

</action>

</recv>

<nop hide="true" test="authvalid" next="goodauth" />

<nop hide="true" next="badauth" />

**Variables**

For complex scenarios, you will need to store bits of information that can be used across messages or even calls. Like other programming languages, SIPp's XML scenario definition allows you to use variables for this purpose. A variable in SIPp is referenced by an alphanumeric name. In past versions of SIPp, variables names were numeric only; thus in this document and the embedded scenarios, you are likely to see lots of variables of the form "1", "2", etc.; although when creating new scenarios you are encouraged to assign meaningful names to your variables.

Aside from a name, SIPp's variables are also loosely typed. The type of a variable is not explicitly declared, but is instead inferred from the action that set it. There are four types of variables: string, regular expression matches, doubles, and booleans. All mathematical operations take place on doubles. The **<test>** and **<verifyauth>** actions create boolean values. String variables and regular expression matches are similar. When a string's value is called for, a regular expression match can be substituted. The primary difference is related to the **test** attribute (see [Conditional Branching](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#branching)). If a string has been defined, a test is evaluated to true. However, for a regular expression variable, the regular expression that set it must match for the test to evaluated to true. Values can be converted to strings using the [**<assignstr>**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_strings) action. Values can be converted to doubles using the [**<todouble>**](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_variables) action.

Variables also have a scope, which is one of global to all calls, per-user, or the default per-call. A global variable can be used, for example to store scenario configuration parameters or to keep a global counter. A user-variable when combined with the -users option allows you to keep per-user state across calls (e.g., if this user has already registered). Finally, the default per-call variables are useful for copying values from one SIP message to the next or controlling branching. Variables can be declared globally or per-user using the following syntax:

<Global variables="foo,bar" />

<User variables="baz,quux" />

Local variables need not be declared. To prevent programming errors, SIPp performs very rudimentary checks to ensure that each variable is used more than once in the scenario (this helps prevent some typos from turning into hard to debug errors). Unfortunately, this can cause some complication with [regular expression matching](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_regexp). The regular expression action must assign the entire matched expression to a variable. If you are only interested in checking the validity of the expression (i.e. the check\_it attribute is set) or in capturing a sub-expression, you must still assign the entire expression to a variable. As this variable is likely only referenced once, you must inform SIPp that you are knowingly using this variable once with a Reference clause. For example:

<recv request="INVITE">

<action>

<ereg regexp="<sip:([^;@]\*)" search\_in="hdr" header="To:" assign\_to="dummy,uri" />

</action>

</recv>

<Reference variables="dummy" />;

**Injecting values from an external CSV during calls**

Go [here](https://twiki.polycom.com/twiki/bin/view/Main/SippInjectingValues) for the full article.

**Conditional branching**

Go [here](https://twiki.polycom.com/twiki/bin/view/Main/SippInjectingValues#Conditional_branching) for the full article.

**SIP authentication**

SIPp supports SIP authentication. Two authentication algorithm are supported: Digest/MD5 ("algorithm="MD5"") and Digest/AKA ("algorithm="AKAv1-MD5"", as specified by 3GPP for IMS). To enable authentication support, SIPp must be compiled in a special way. See [SIPp installation](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#installing) for details

To enable authentication add auth="true" in the <recv> command when receiving a 401 (Unauthorized) or a 407 (Proxy Authentication Required) to take the challenge into account. Then, the authorization header can be re-injected in the next message by using [authentication] keyword. Currently authentication credentials can only be calculated once in a conversation. Note that the message containing the [authentication] should be identical to the request that was challenged by the server, whatever the request was.

**Example:**

<send dialog="1">

<![CDATA[

REGISTER sip:[remote\_ip]:[remote\_port] SIP/2.0

Via: SIP/2.0/UDP [local\_ip]:[local\_port];branch=[branch]

From: "9000" <sip:9000@[local\_ip]:[local\_port]>[local\_tag\_param]

To: <sip:9000@[remote\_ip]:[remote\_port]>

CSeq: [cseq] REGISTER

Call-ID: [call\_id]

Contact: <sip:9000@[local\_ip]:[local\_port];transport=[transport]>;methods="INVITE, ACK, BYE, CANCEL, OPTIONS, INFO, MESSAGE, SUBSCRIBE, NOTIFY, PRACK, UPDATE, REFER"

User-Agent: PolycomSoundPointIP-SPIP\_650-UA/3.3.2.0104

Accept-Language: en

Max-Forwards: 70

Expires: 3600

Content-Length: [len]

]]>

</send>

<recv response="401" dialog="1" auth="true" />

<send dialog="1">

<![CDATA[

REGISTER sip:[remote\_ip]:[remote\_port] SIP/2.0

Via: SIP/2.0/UDP [local\_ip]:[local\_port];branch=[branch]

From: "9000" <sip:9000@[local\_ip]:[local\_port]>[local\_tag\_param]

To: <sip:9000@[remote\_ip]:[remote\_port]>

CSeq: [cseq] REGISTER

Call-ID: [call\_id]

Contact: <sip:9000@[local\_ip]:[local\_port];transport=[transport]>;methods="INVITE, ACK, BYE, CANCEL, OPTIONS, INFO, MESSAGE, SUBSCRIBE, NOTIFY, PRACK, UPDATE, REFER"

User-Agent: PolycomSoundPointIP-SPIP\_650-UA/3.3.2.0104

Accept-Language: en

[authentication username=9000]

Max-Forwards: 70

Expires: 3600

Content-Length: [len]

]]>

</send>

Computing the authorization header is done through the usage of the "[authentication]" keyword. Depending on the algorithm ("MD5" or "AKAv1-MD5"), different parameters must be passed next to the authentication keyword:

* Digest/MD5 (example: [authentication username=joe password=schmo])
  + **username**: username: if no username is specified, the username is taken from the '-s' (service) command line parameter
  + **password**: password: if no password is specified, the password is taken from the '-ap' (authentication password) command line parameter
* Digest/AKA: (example: [authentication username=HappyFeet aka\_OP=0xCDC202D5123E20F62B6D676AC72CB318 aka\_K=0x465B5CE8B199B49FAA5F0A2EE238A6BC aka\_AMF=0xB9B9])
  + **username**: username: if no username is specified, the username is taken from the '-s' (service) command line parameter
  + **aka\_K**: Permanent secret key. If no aka\_K is provided, the "password" attributed is used as aka\_K.
  + **aka\_OP**: OPerator variant key
  + **aka\_AMF**: Authentication Management Field (indicates the algorithm and key in use)

In case you want to use authentication with a different username/password or aka\_K for each call, you can do this:

* Make a CSV like this: SEQUENTIAL User0001;[authentication username=joe password=schmo] User0002;[authentication username=john password=smith] User0003;[authentication username=betty password=boop]
* And an XML like this (the [field1] will be substituted with the full auth string, which is the processed as a new keyword):

<send retrans="500 >

<![CDATA[

REGISTER sip:[remote\_ip] SIP/2.0

Via: SIP/2.0/[transport] [local\_ip]:[local\_port]

To: <sip:[field0]@sip.com:[remote\_port]>

From: <sip:[field0]@[remote\_ip]:[remote\_port]>

Contact: <sip:[field0]@[local\_ip]:[local\_port]>;transport=[transport]

Expires: 300

Call-ID: [call\_id]

CSeq: 2 REGISTER

Content-Length: 0

]]>

</send>

**Initialization Stanza**

Some complex scenarios require setting appropriate global variables at SIPp startup. The initialization stanza allows you do do just that. To create an initialization stanza, simply surround a series of <nop> and <label> commands with <init> and </init>. These <nop>s are executed once at SIPp startup. The variables within the init stanza, except for globals, are not shared with calls. For example, this init stanza sets $THINKTIME to 1 if it is not already set (e.g., by the -set command line parameter).

<init>

<!-- By Default THINKTIME is true. -->

<nop>

<action>

<strcmp assign\_to="empty" variable="THINKTIME" value="" />

<test assign\_to="empty" compare="equal" variable="empty" value="0" />

</action>

</nop>

<nop condexec="empty">

<action>

<assignstr assign\_to="THINKTIME" value="1" />

</action>

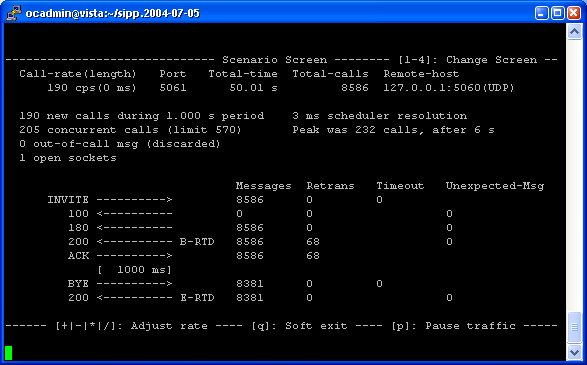
</nop>

</init>

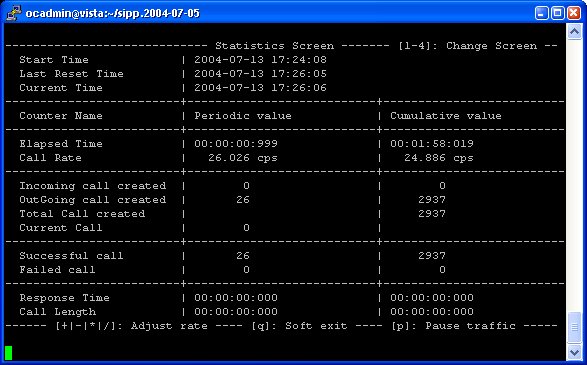
**Screens**

Several screens are available to monitor SIP traffic. You can change the screen view by pressing 1 to 9 keys on the keyboard.

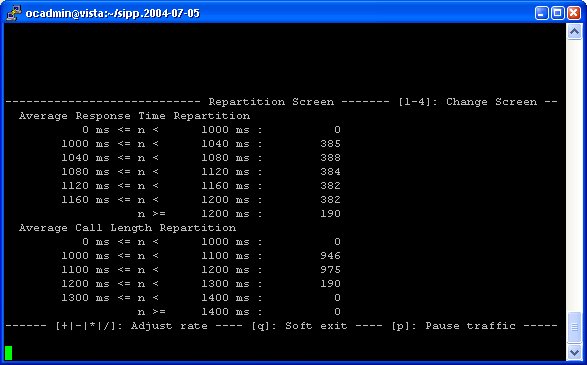
* Key '1': Scenario screen. It displays a call flow of the scenario as well as some important informations.



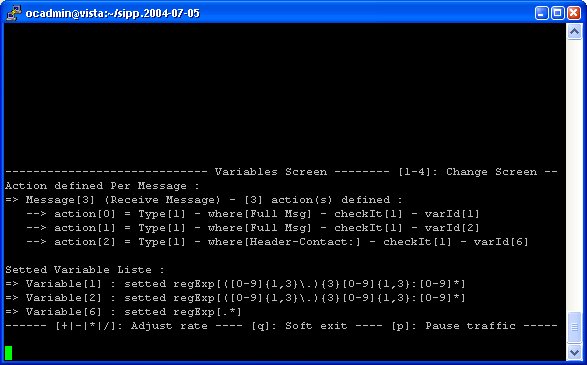
* Key '2': Statistics screen. It displays the main statistics counters. The "Cumulative" column gather all statistics, since SIPp has been launched. The "Periodic" column gives the statistic value for the period considered (specified by -f frequency command line parameter).



* Key '3': Repartition screen. It displays the distribution of response time and call length, as specified in the scenario.



* Key '4': Variables screen. It displays informations on actions in scenario as well as scenario variable informations.



**Transport modes**

Go [here](https://twiki.polycom.com/twiki/bin/view/Main/SIPpedReferenceGuide?topic=IntegratedScenarios) for the full article.

When using one of the "multi-socket" transports, the maximum number of sockets that can be opened (which corresponds to the number of simultaneous calls) will be determined by the system (see [how to increase file descriptors section](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#filedesc) to modify those limits). You can also limit the number of socket used by using the -max\_socket command line option. Once the maximum number of opened sockets is reached, the traffic will be distributed over the sockets already opened.

For TCP connections, phones will tear down idle connections at 60 seconds. The next message to be sent by sipp will fail indicating that the remote end has closed the connection. To override this behaviour, specify options for multisocket (tn) and for call to survive remote disconnects (reconnect\_close). eg sipp -sf ReceiveCall01.sipp  -i 172.23.2.49 172.23.7.156 -t tn -reconnect\_close false -mc -aa

**Handling media with SIPp**

SIPp is originally a signalling plane traffic generator. There is a limited support of media plane (RTP).

**RTP echo**

The "RTP echo" feature allows SIPp to listen to one or two local IP address and port (specified using -mi and -mp command line parameters) for RTP media. Everything that is received on this address/port is echoed back to the sender.

RTP/UDP packets coming on this port + 2 are also echoed to their sender (used for sound and video echo).

**PCAP Play**

The PCAP play feature makes use of the [PCAP library](http://www.tcpdump.org/pcap3_man.html) to replay pre-recorded RTP streams towards a destination. RTP streams can be recorded by tools like [Wireshark](http://www.wireshark.org/) (formerly known as Ethereal) or [tcpdump](http://www.tcpdump.org/). This allows you to:

* Play any RTP stream (voice, video, voice+video, out of band DTMFs/RFC 2833, T38 fax, ...)
* Use any codec as the codec is not handled by SIPp
* Emulate precisely the behavior of any SIP equipment as the pcap play will try to replay the RTP stream as it was recorded (limited to the performances of the system).
* Reproduce exactly what has been captured using an IP sniffer like [Wireshark](http://www.wireshark.org/).
* the -mi command line parameter can be used to set the source address for media streams, the -mp command line parameter can be used to set the base source port for media playback. Destination address/port are determined from the incoming sip sdp message during playback.
* the play\_pcap media\_port\_offset attribute can be used to set the playback source port offset and can have a positive or negative value.

As of v3.2.66, sipp also has the ability to support

* per media contact lines (that is each 'm=' line in the sdp can have their own 'c=' that can override the session level contact line).
* play\_pcap\_audio, play\_pcap\_video, play\_pcap\_application are valid exec commands that retrieve their src/dest information from scenario/received sdp for each m=audio, m=video, or m=application line. There is no intellienge in the play back of media streams : sipp is not aware of the signalling logic found within application media streams.
* multiple simultaneous media streams is supported so sipp can send multiple of different or same media types (eg 2 video streams)
* optional index attribute to play\_pcap\_[audio|video|application] is used to associate src (scenario sipp file) dest (incoming sdp during playback) to the pcap file (scenario sipp file). index attribute is automatically inserted by pcap files processed by snipp.
* optional source\_ip attribute to allow the use of multiple source ip addresses, or auto selection of source ip. source\_ip="2" will allow for the use of -i2 command line option to specify a second ip address (which can be refered to within sipp scenario files by [local\_ip2]) . Primary -i and secondary -i2 addresses do not have to be from the same address family (ipv4 or ipv6). Setting source\_ip="0" will allow for auto selection where the destination address family is used to scan for any available local host address that can sipp can bind, to send stream to destination address.

A good example is the [UAC with media](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#uac_with_media) (uac\_pcap) embedded scenario.

SIPp comes with a G711 alaw pre-recorded pcap file and out of band (RFC 2833) DTMFs in the pcap/ directory. The PCAP play feature uses pthread\_setschedparam calls from pthread library. Depending on the system settings, you might need to be root to allow this. Please check "man 3 pthread\_setschedparam" man page for details

More details on the possible PCAP play actions can be found in the [action reference section](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_exec).

The latest info on this feature, tips and tricks can be found on [SIPp wiki](http://sipp.sourceforge.net/wiki/index.php/Pcapplay).

**Exit codes**

To ease automation of testing, upon exit (on fatal error or when the number of asked calls (-m command line option) is reached, sipp exits with one of the following exit codes:

* 0: All calls were successful
* 1: At least one call failed
* 95: Test ended manually by user (ie by pressing q)
* 96: Test exited because it recieved some sort of kill signal (ie SIGINT, SIGKILL, SIGABRT, etc)
* 97: Exit on global timeout (see -timeout\_global option)
* 98: Test exited before completion, possibly due to tcp errors or excessive call rate
* 99: Normal exit without calls processed
* -1 (255): Fatal error (ie an ERROR message was encountered)
* -2 (254): Bind error
* -3 (253): System error
* -4 (252): Error parsing parameters

Depending on the system that SIPp is running on, you can echo this exit code by using "echo ?" command.

**Statistics**

Go [here](https://twiki.polycom.com/twiki/bin/view/Main/SipPStatistics) for the full article.

**Error handling**

SIPp has advanced feature to handle errors and unexpected events. They are detailed in the following sections.

**Unexpected messages**

* When a SIP message that **can** be correlated to an existing call (with the Call-ID: header) but is not expected in the scenario is received, SIPp will send a CANCEL message if no 200 OK message has been received or a BYE message if a 200 OK message has been received. The call will be marked as failed. If the unexpected message is a 4XX or 5XX, SIPp will send an ACK to this message, close the call and mark the call as failed.
* When a SIP message that **can't** be correlated to an existing call (with the Call-ID: header) is received, SIPp will send a BYE message. The call will not be counted at all.
* When a SIP "PING" message is received, SIPp will send an ACK message in response. This message is not counted as being an unexpected message. But it is counted in the "AutoAnswered" [statistic counter](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#stats).
* An unexpected message that is not a SIP message will be simply dropped.

**Retransmissions (UDP only)**

|  |  |
| --- | --- |
| -nr | No Retransmissions: either incoming or outgoing. |
| -yr | retransmissions enabled.   * If the last incoming message is a retransmission , the last sent message is resent. Note that this applies only the most recent incoming/outgoing message, retransmission of earlier messages will not be handled as retransmissions, but as unexpected messages. * Also ,for outgoing messages that have the retrans attribute in the scenario file, outgoing messages will be retransmitted if the response is not received within the retrans specified time. eg " <send retrans="500" dialog="1">" |
| -ar | absorb retransmission. This only impacts incoming retransmissions, Counters will be incremented but no outgoing responses to received retransmission will be sent. Note that this allows incoming retransmissions of *any previously received message* in the current call scenario. This option has no impact outgoing messages with retrans attrribute. |

A retransmission mechanism exists in UDP transport mode. To activate the retransmission mechanism, the "send" command must include the "retrans" attribute.

When it is activated and a SIP message is sent and no ACK or response is received in answer to this message, the message is re-sent. The retransmission mechanism follows RFC 3261, section 17.1.1.2. Retransmissions are differentiated between INVITE and non-INVITE methods.

<send retrans="500">: will initiate the T1 timer to 500 milliseconds.

Even if retrans is specified in your scenarios, you can override this by using the -nr command line option to globally disable the retransmission mechanism.

A third option retransmission option exists :-ar to absorb retransmits. Unlike the -yr, retransmissions of any prior received packet will be detected and the retransmission counter incremented but nothing will be sent in resonse to the retransmission. This option shall have no impact on the retrans attribute behaviour for sending packets (the second impact described above) but shall override the -yr behaviour of incoming packet retransmit detection if both options are present. By specifying both -yr and -ar, retrans timer will continue function on outgoing retransmissions initiated by response timeouts but incoming retransmits will not trigger any sipp retransmissions (as they would without -ar option).

-ar processing only takes place after matching messages against scenario files, so if a scenario file expects duplicate messages, they will be correctly matched and processed against the scenario file and not counted as a retransmit.

-yr processing takes place before matching messages against scenario files. Scenarios containing retransmissions in them may never see the retransmitted message as -yr processing will automatically detect and respond to retransmissions of current message before any matching against scenario messages is done. When -ar and -yr are both present, -ar takes precedence and prevents auto response to incoming retransmitted messages.

The -mc command automatically set -nr option to prevent retransmissions and -ar to absorb retransmissions.

**Log files**

There are several ways to trace what is going on during your SIPp runs.

* You can log sent and received SIP messages in <name\_of\_the\_scenario>\_<pid>\_messages.log by using the command line parameter -trace\_msg. The messages are time-stamped so that you can track them back.
* You also can trace it using the -trace\_shortmsg parameter. This logs the most important values of a message as CSV into one line of the <scenario file name>\_<pid>\_shortmessages.log
* You can trace all unexpected messages or events in <name\_of\_the\_scenario>\_<pid>\_errors.log by using the command line parameter -trace\_err.
* You can trace the counts from the main scenario screen in <name\_of\_the\_scenario>\_<pid>\_counts.csv by using the command line parameter -trace\_counts.
* You can trace the messages and state transitions of failed calls in <name\_of\_the\_scenario>\_<pid>\_calldebug.log using the -trace\_calldebug command line parameter. This is useful, because it has less overhead than -trace\_msg yet allows you to debug call flows that were not completed successfully.
* You can save in a file the statistics screens, as displayed in the interface. This is especially useful when running SIPp in background mode.  
  This can be done in different ways:
  + When SIPp exits to get a final status report (-trace\_screen option)
  + On demand by using USR2 signal (example: kill -SIGUSR2 738)
  + By pressing 's' key (if -trace\_screen option is set)
  + If the -trace\_logs option is set, you can use the <log> action to print some scenario traces in the file. See the [Log action section](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#action_log)

SIPp can treat the messages, short messages, logs, and error logs as ring buffers. This allows you to limit the total amount of space used by these log files and keep only the most recent messages. To set the maximum file size use the -ringbuffer\_size option. Once the file exceeds this size (the file size can be exceeded up to the size of a single log message), it is rotated. SIPp can keep several of the most recent files, to specify the number of files to keep use the -ringbuffer\_files option. The rotated files have a name of the form <name\_of\_the\_scenario>\_<pid>\_<logname>\_<date>.log, where <date> is the number of seconds since the epoch. If more than one log file is rotated during a one second period, then the date is expressed as <seconds.serial>, where serial is an increasing integer identifier.

**Options (-h)**

sipp remote\_host[:remote\_port] [options]

Available options:

-v : Display version and copyright information.

-aa : Enable automatic 200 OK answer for INFO, UPDATE, NOTIFY and REGISTERmessages.

-aa\_expires : Unit in seconds, Default value is 3600 seconds

-auth\_uri : Force the value of the URI for authentication. By default, the URI is composed of remote\_ip:remote\_port.

-base\_cseq : Start value of [cseq] for each call.

-bg : Launch SIPp in background mode.

-bind\_local : Bind socket to local IP address, i.e. the local IP address is used as the source IP address. If SIPp runs in server mode it will only listen on the local IP address instead of all IP addresses.

-buff\_size : Set the send and receive buffer size.

-calldebug\_file : Set the name of the call debug file.

-calldebug\_overwrite

Overwrite the call debug file (default true). -cid\_str : Call ID string (default %u-%p@%s). %u=call\_number, %s=ip\_address, %p=process\_number, %%=% (in any order).

-ci : Set the local control IP address

-cp : Set the local control port number. Default is 8888.

-d : Controls the length of calls. More precisely, this controls the duration of 'pause' instructions in the scenario, if they do not have a 'milliseconds' section. Default value is 0 and default unit is milliseconds.

-deadcall\_wait : How long the Call-ID and final status of calls should be kept to improve message and error logs (default unit is ms). Default is 33 seconds, unless -mc is specified in which case it is 0.

-default\_behaviors

Set the default behaviors that SIPp will use. Possbile values are: - all Use all default behaviors - none Use no default behaviors - bye Send byes for aborted calls, - abortunexp Abort calls on unexpected messages, - pingreply Reply to ping requests. If a behavior is prefaced with a -, then it is turned off. Example: all,-bye -dump\_xml : Dump expanded XML to screen. Useful for debugging includes  
  
-dump\_sequence\_diagram: Dump sequence diagram.

-debug\_file : Set the name of the call debug file.  
  
-debug\_overwrite : Overwrite the call debug file (default true).

-error\_file : Set the name of the error log file.

-error\_overwrite : Overwrite the error log file (default true).

-exec\_file : Set the name of the exec log file.

-exec\_overwrite : Overwrite the call debug file (default true).

-f : Set the statistics report frequency on screen. Default is 1 and default unit is seconds.

-fd : Set the statistics dump log report frequency. Default is 60 and default unit is seconds.

-force\_client\_mode/force\_server\_mode: Set creation/send mode to client/server regardless of the contents of the scenario/

-i : Set the local IP address for 'Contact:','Via:', and 'From:' headers. Default is primary host IP address.

-i2 : Set the secondary local IP address, It doesn not ave to be the same address family as the primary address but must be a valid local host address.

-inf : Inject values from an external CSV file during calls into the scenarios. First line of this file say whether the data is to be read in sequence (SEQUENTIAL), random (RANDOM), or user (USER) order. Each line corresponds to one call and has one or more ';' delimited data fields. Those fields can be referred as [field0], [field1], ... in the xml scenario file. Several CSV files can be used simultaneously (syntax: -inf f1.csv -inf f2.csv ...)

-infindex : file field Create an index of file using field. For example -inf users.csv -infindex users.csv 0 creates an index on the first key.

-ip\_field : Set which field from the injection file contains the IP address from which the client will send its messages. If this option is omitted and the '-t ui' option is present, then field 0 is assumed. Use this option together with '-t ui'

-l : Set the maximum number of simultaneous calls. Once this limit is reached, traffic is decreased until the number of open calls goes down. Default: (3 \* call\_duration (s) \* rate).

-log\_file : Set the name of the log actions log file.

-log\_overwrite : Overwrite the log actions log file (default true).

-lost : Set the number of packets to lose by default (scenario specifications override this value).

-rtcheck : Select the retransmisison detection method: full (default) or loose.

-m : Stop the test and exit when 'calls' calls are processed

-mi : Set the local media IP address (default: local primary host IP address)

-master : 3pcc extended mode: indicates the master number

-max\_recv\_loops : Set the maximum number of messages received read per cycle. Increase this value for high traffic level. The default value is 1000.

-max\_sched\_loops : Set the maximum number of calsl run per event loop. Increase this value for high traffic level. The default value is 1000.

-max\_reconnect : Set the the maximum number of reconnection.

-max\_retrans : Maximum number of UDP retransmissions before call ends on timeout. Default is 5 for INVITE transactions and 7 for others.

-max\_invite\_retrans

Maximum number of UDP retransmissions for invite transactions before call ends on timeout.

-max\_non\_invite\_retrans

Maximum number of UDP retransmissions for non-invite transactions before call ends on timeout. -max\_log\_size : What is the limit for error and message log file sizes.

-max\_socket : Set the max number of sockets to open simultaneously. This option is significant if you use one socket per call. Once this limit is reached, traffic is distributed over the sockets already opened. Default value is 50000

-mb : Set the RTP echo buffer size (default: 2048).

-mc : Enable multiple-dialog support by directing all messages to one scenario regardless of call-id. Only 1 concurrent call is possible, stop calls (-m) defaults to 1.

-message\_file : Set the name of the message log file.

-message\_overwrite

Overwrite the message log file (default true) -mp : Set the local RTP echo port number. Default is 6000.

-nd : No Default. Disable all default behavior of SIPp which are the following: - On UDP retransmission timeout, abort the call by sending a BYE or a CANCEL - On receive timeout with no ontimeout attribute, abort the call by sending a BYE or a CANCEL - On unexpected BYE send a 200 OK and close the call - On unexpected CANCEL send a 200 OK and close the call - On unexpected PING send a 200 OK and continue the call - On any other unexpected message, abort the call by sending a BYE or a CANCEL

-nr : Disable retransmission in UDP mode. Retransmissions are enabled by default unless -mc options is used. See -yr to enable them with -mc.

-nostdin : Disable stdin.

-p : Set the local port number. Default is a random free port chosen by the system.

-pause\_msg\_ign : Ignore the messages received during a pause defined in the scenario

-periodic\_rtd : Reset response time partition counters each logging interval.

-r : Set the call rate (in calls per seconds). This value can bechanged during test by pressing '+','\_','\*' or '/'. Default is 10. pressing '+' key to increase call rate by 1 \* rate\_scale, pressing '-' key to decrease call rate by 1 \* rate\_scale, pressing '\*' key to increase call rate by 10 \* rate\_scale, pressing '/' key to decrease call rate by 10 \* rate\_scale. If the -rp option is used, the call rate is calculated with the period in ms given by the user.

-rp : Specify the rate period for the call rate. Default is 1 second and default unit is milliseconds. This allows you to have n calls every m milliseconds (by using -r n -rp m). Example: -r 7 -rp 2000 ==> 7 calls every 2 seconds. -r 10 -rp 5s => 10 calls every 5 seconds.

-rate\_scale : Control the units for the '+', '-', '\*', and '/' keys.

-rate\_increase : Specify the rate increase every -fd units (default is seconds). This allows you to increase the load for each independent logging period. Example: -rate\_increase 10 -fd 10s ==> increase calls by 10 every 10 seconds.

-rate\_max : If -rate\_increase is set, then quit after the rate reaches this value. Example: -rate\_increase 10 -rate\_max 100 ==> increase calls by 10 until 100 cps is hit.

-no\_rate\_quit : If -rate\_increase is set, do not quit after the rate reaches -rate\_max.

-recv\_timeout : Global receive timeout. Default unit is milliseconds. If the expected message is not received, the call times out and is aborted.

-send\_timeout : Global send timeout. Default unit is milliseconds. If a message is not sent (due to congestion), the call times out and is aborted.

-sleep : How long to sleep for at startup. Default unit is seconds.

-reconnect\_close : Should calls be closed on reconnect?

-reconnect\_sleep : How long (in milliseconds) to sleep between the close and reconnect?

-ringbuffer\_files

How many error/message files should be kept after rotation? -ringbuffer\_size : How large should error/message files be before they get rotated?

-rsa : Set the remote sending address to host:port for sending the messages.

-rtp\_echo : Enable RTP echo. RTP/UDP packets received on port defined by -mp are echoed to their sender. RTP/UDP packets coming on this port + 2 are also echoed to their sender (used for sound and video echo).

-rtt\_freq : freq is mandatory. Dump response times every freq calls in the log file defined by -trace\_rtt. Default value is 200.

-s : Set the username part of the resquest URI. Default is 'service'.

-sd : Dumps a default scenario (embeded in the sipp executable)

-sf : Loads an alternate xml scenario file. To learn more about XML scenario syntax, use the -sd option to dump embedded scenarios. They contain all the necessary help.

-shortmessage\_file

Set the name of the short message log file.

-shortmessage\_overwrite

Overwrite the short message log file (default true). -oocsf : Load out-of-call scenario.

-oocsn : Load out-of-call scenario.

-skip\_rlimit : Do not perform rlimit tuning of file descriptor limits. Default: false.

-slave : 3pcc extended mode: indicates the slave number

-slave\_cfg : 3pcc extended mode: indicates the file where the master and slave addresses are stored

-sn : Use a default scenario (embedded in the sipp executable). If this option is omitted, the Standard SipStone[?](https://twiki.polycom.com/twiki/bin/edit/Main/SipStone?topicparent=Main.SIPpedReferenceGuide) UAC scenario is loaded. Available values in this version:

- 'uac' : Standard SipStone[?](https://twiki.polycom.com/twiki/bin/edit/Main/SipStone?topicparent=Main.SIPpedReferenceGuide) UAC (default). - 'uas' : Simple UAS responder. - 'regexp' : Standard SipStone[?](https://twiki.polycom.com/twiki/bin/edit/Main/SipStone?topicparent=Main.SIPpedReferenceGuide) UAC - with regexp and variables. - 'branchc' : Branching and conditional branching in scenarios - client. - 'branchs' : Branching and conditional branching in scenarios - server.

Default 3pcc scenarios (see -3pcc option):

- '3pcc-C-A' : Controller A side (must be started after all other 3pcc scenarios) - '3pcc-C-B' : Controller B side. - '3pcc-A' : A side. - '3pcc-B' : B side.

-stat\_delimiter : Set the delimiter for the statistics file

-stf : Set the file name to use to dump statistics

-t : Set the transport mode: - u1: UDP with one socket (default), - un: UDP with one socket per call, - ui: UDP with one socket per IP address The IP addresses must be defined in the injection file. - t1: TCP with one socket, - tn: TCP with one socket per call, - l1: TLS with one socket, - ln: TLS with one socket per call, - c1: u1 + compression (only if compression plugin loaded), - cn: un + compression (only if compression plugin loaded). This plugin is not provided with sipp.

-timeout : Global timeout. Default unit is seconds. If this option is set, SIPp quits after nb units (-timeout 20s quits after 20 seconds).

-timeout\_error : SIPp fails if the global timeout is reached is set (-timeout option required).

-timer\_resol : Set the timer resolution. Default unit is milliseconds. This option has an impact on timers precision.Small values allow more precise scheduling but impacts CPU usage.If the compression is on, the value is set to 50ms. The default value is 10ms.

-sendbuffer\_warn : Produce warnings instead of errors on SendBuffer[?](https://twiki.polycom.com/twiki/bin/edit/Main/SendBuffer?topicparent=Main.SIPpedReferenceGuide) failures.

-trace\_msg : Displays sent and received SIP messages in <scenario file name>\_\_messages.log

-trace\_shortmsg : Displays sent and received SIP messages as CSV in \_\_shortmessages.log

-trace\_screen : Dump statistic screens in the \_\_cenaris.log file when quitting SIPp. Useful to get a final status report in background mode (-bg option).

-trace\_err : Trace all unexpected messages in <scenario file name>\_\_errors.log.

-trace\_debug : Dumps debugging information about SIPp execution and ALL  
other messages to <scenario\_name>\_<pid>\_DEBUG.log file.  
  
-trace\_exec : Redirects output from <exec> commands to <scenario\_name>\_<pid>\_EXEC.log file.  
-trace\_calldebug : Dumps debugging information about aborted calls to \_\_calldebug.log file.

-trace\_stat : Dumps all statistics in \_.csv file. Use the '-h stat' option for a detailed description of the statistics file content.

-trace\_counts : Dumps individual message counts in a CSV file.

-trace\_rtt : Allow tracing of all response times in <scenario file name>\_\_rtt.csv.

-trace\_logs : Allow tracing of actions in <scenario file name>\_\_logs.log.

-users : Instead of starting calls at a fixed rate, begin 'users' calls at startup, and keep the number of calls constant.

-watchdog\_interval

Set gap between watchdog timer firings. Default is 400, unless -mc option is used (in which case watchdogs are disabled by default). -watchdog\_reset : If the watchdog timer has not fired in more than this time period, then reset the max triggers counters. Default is 10 minutes.

-watchdog\_minor\_threshold

If it has been longer than this period between watchdog executions count a minor trip. Default is 500.

-watchdog\_major\_threshold

If it has been longer than this period between watchdog executions count a major trip. Default is 3000.

-watchdog\_major\_maxtriggers

How many times the major watchdog timer can be tripped before the test is terminated. Default is 10.

-watchdog\_minor\_maxtriggers

How many times the minor watchdog timer can be tripped before the test is terminated. Default is 120. -yr : Enable retransmission in UDP mode. Retransmissions are enabled by default unless -mc options is used. See -nr to disable them.

-ap : Set the password for authentication challenges. Default is 'password'

-tls\_cert : Set the name for TLS Certificate file. Default is 'cacert.pem'. The certificate file must be placed in either the directory from which the scenario is being run, or the directory referenced by the SIPPED environment variable.

-tls\_key : Set the name for TLS Private Key file. Default is 'cakey.pem'. The private key file must be placed in either the directory from which the scenario is being run, or the directory referenced by the SIPPED environment variable.

-tls\_crl : Set the name for Certificate Revocation List file. If not specified, X509 CRL is not activated.

-3pcc : Launch the tool in 3pcc mode ("Third Party call control"). The passed ip address is depending on the 3PCC role. - When the first twin command is 'sendCmd' then this is the address of the remote twin socket. SIPp will try to connect to this address:port to send the twin command (This instance must be started after all other 3PCC scenarii). Example: 3PCC-C-A scenario. - When the first twin command is 'recvCmd' then this is the address of the local twin socket. SIPp will open this address:port to listen for twin command. Example: 3PCC-C-B scenario.

-tdmmap : Generate and handle a table of TDM circuits. A circuit must be available for the call to be placed. Format: -tdmmap {0-3}{99}{5-8}{1-31}

-key : keyword value Set the generic parameter named "keyword" to "value".

-set : variable value Set the global variable parameter named "variable" to "value".

Signal handling:

SIPp can be controlled using posix signals. The following signals are handled:

USR1

Similar to press 'q' keyboard key. It triggers a soft exit of SIPp. No more new calls are placed and all ongoing calls are finished before SIPp exits.

Example

kill -SIGUSR1 732

USR2

Triggers a dump of all statistics screens in \_\_screens.log file. Especially useful in background mode to know what the current status is.

Example

kill -SIGUSR2 732 Example: Run sipp with embedded server (uas) scenario: ./sipp -sn uas

On the same host, run sipp with embedded client (uac) scenario ./sipp -sn uac 127.0.0.1

**Message Sequencing**

sipp has the ability to handle out of order messages if the messages are from different dialogs. This capability is automatically turned on with the -mc option. The exception is if the scenario files contains jump or next commands, in which case strict sequencing is required and no out of order messaging is allowed. More details on how it works can be found here <https://twiki.polycom.com/twiki/bin/view/Sandbox/LooseMessageSequence>

**Performance testing with SIPp**

**Advices to run performance tests with SIPp**

SIPp has been originally designed for SIP performance testing. Reaching high call rates and/or high number of simultaneous SIP calls is possible with SIPp, provided that you follow some guidelines:

* Use an HP-UX, Linux or other \*ix system to reach high performances. The Windows port of SIPp (through CYGWIN) cannot handle high performances.
* Limit the traces to a minimum (usage of -trace\_msg, -trace\_logs should be limited to scenario debugging only)
* To reach a high number of simultaneous calls in multi-socket mode, you must increase the number of filedescriptors handled by your system. Check "[Increasing File Descriptors Limit](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#filedesc)" section for more details.
* Understand [internal SIPp's scheduling mechanism](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#scheduling) and use the -timer\_resol, -max\_recv\_loops and -up\_nb command line parameters to tune SIPp given the system it is running on.

Generally, running performance tests also implies measuring response times. You can use SIPp's timers (start\_rtd, rtd in scenarios and -trace\_rtt command line option) to measure those response times. The precision of those measures are entirely dependent on the timer\_resol parameter (as described in "[SIPp's internal scheduling](https://twiki.polycom.com/twiki/bin/viewauth/Main/SIPpedReferenceGuide#scheduling)" section). You might want to use another "objective" method if you want to measure those response times with a high precision (a tool like [Wireshark](http://www.wireshark.org/) will allow you to do so).

**SIPp's internal scheduling**

SIPp has a single-threaded event-loop architecture, which allows it to handle high SIP traffic loads. SIPp's event loop tracks various tasks, most of which are the calls that are defined in your scenario. In addition to tasks that represent calls there are several special tasks: a screen update task, a statistics update task, a call opening task, and a watchdog task. SIPp's main execution loop consists of:

1. Waking up tasks that have expired timers.
2. Running up to max\_sched\_loop tasks that are in a running state (each call is executed until it is no longer runnable).
3. Handling each of the sockets in turn, reading max\_recv\_loops messages from the various sockets.

SIPp executes this loop continuously, until some condition tells it to stop (e.g., the user pressing the 'q' key or the global call limit or timeout being reached).

Several parameters can be specified on the command line to fine tune this scheduling.

* timer\_resol: during the main loop, the management of calls (management of wait, retransmission ...) is done for all calls, every "timer\_resol" ms at best. The delay of retransmission must be higher than "timer\_resol". The default timer resolution is 1 millisecond, and that is the most precise resolution that SIPp currently supports. If you increase this parameter, SIPp's traffic will be burstier and you are likely to encounter retransmissions at high load. If you have too many calls, or each call takes too long, the timer resolution will not be respected.
* max\_recv\_loops and max\_sched\_loops: received messages are read and treated in batch. "max\_recv\_loops" is the maximum number of messages that can be read at one time. "max sched loops" is the maximum number of processing calls loops. These limits prevent SIPp from reading and processing new messages from sockets to the exclusion of processing existing calls, and vice versa. For heavy call rate, increase both values. Be careful, those two parameters have a large influence on the CPU occupation of SIPp.
* watchdog\_interval, watchdog\_minor\_threshold, watchdog\_major\_threshold, watchdog\_minor\_maxtriggers, and watchdog\_major\_maxtriggers: The watchdog timer is designed to provide feedback if your call load is causing SIPp's scheduler to be overwhelmed. The watchdog task sets a timer that should fire every watchdog\_interval milliseconds (by default 400ms). If the timer is not serviced for more than watchdog\_minor\_threshold milliseconds (by default 500s), then a "minor" trigger is recorded. If the number of minor triggers is more than watchdog\_minor\_maxtriggers; the watchdog task terminates SIPp. Similarly, if the timer is not serviced for more than watchdog\_major\_threshold milliseconds (by default 3000ms), then a major trigger is recorded; and if more than watchdog\_major\_maxtriggers are recorded SIPp is terminated. If you only see occasional messages, your test is likely acceptable, but if these events are frequent you need to consider using a more powerful machine or set of machines to run your scenario.

**IPv6 and SIPp**

./sipp -sn uas -i [fe80::204:75ff:fe4d:19d9] -p 5063

./sipp -sn uac -i [fe80::204:75ff:fe4d:19d9] [fe80::204:75ff:fe4d:19d9]:5063

creates a server and client instance over ipv6. sipp expects ipv6 addresses to be specified with square brackets surrounding the address. In IPv6, square brackets are used to disambiguate the internal : vs the external : sepearator to specify address:port. By default, sipp will show local\_ip address with square brackets. If there are some fields where you specifically want no square brackets, you can specify the optional attribute no\_square\_bracket.

**[local\_ip no\_square\_bracket]**

in the sipp scenario file where required. This optional attribute has no impact if the local\_ip is a ipv4 address. Similarly, the attribute is available for other addresses (local\_ip remote\_ip media\_ip server\_ip).

Some other reference ipv6 addresses that you might find usefull

* ::1 is localhost
* fe8x: fe9x: feax:febx: nonroutable link local address prefixes
* ff02::1, all-hosts multicast address to receive Neighbor Advertisements.
* :: unspecified address like 0.0.0.0
* ::ffff:a.b.c.d/96 is a 96bit prefix that flags ipv4 mapped ipv6 where a.b.c.d is the ipv4 address. note also the double colon is a short hand notattion for bunch of zeros.
* ::a.b.c.d/96 is a 96bit prefix that flags ipv4 compatible ipv6 address
* 2002: tunnelling of 6 over 4 clouds.
* ff*x*y: multicast
* note that the double colon :: is used as a shorthand notation for a bunch of zeros(as many as necessary to complete ipv6 address)

sipp has been run against early polycom loads that support ipv6 for simple call scenario.

**Useful tools aside SIPp**

**JEdit**

JEdit (<http://www.jedit.org/>) is a GNU GPL text editor written in Java, and available on almost all platforms. It's extremely powerful and can be used to edit SIPp scenarios with syntax checking if you put the DTD (sipp.dtd) in the same directory as your XML scenario.

**Wireshark/tshark**

Wireshark (<http://www.wireshark.org/>) is a GNU GPL protocol analyzer. It was formerly known as Ethereal. It supports SIP/SDP/RTP.

**SIP callflow**

When tracing SIP calls, it is very useful to be able to get a call flow from an wireshark trace. The "callflow" tool allows you to do that in a graphical way: <http://callflow.sourceforge.net/>

An equivalent exist if you want to generate HTML only call flows [http://www.iptel.org/~sipsc/](http://www.iptel.org/%7Esipsc/)

**Getting support**

You can likely get email-based support from the sipp users community. The mailing list address is [sipp-users@lists.sourceforge.net](mailto:sipp-users@lists.sourceforge.net). To protect you from SPAM, this list is restricted (only people that actually subscribed can post). Also, you can browse the SIPp mailing list archive: <http://lists.sourceforge.net/lists/listinfo/sipp-users>

**Contributing to SIPp**

Of course, we welcome contributions! If you created a feature for SIPp, please send the "diff" output (diff -bruN old\_sipp\_directory new\_sipp\_directory) on the [SIPp mailing list](http://lists.sourceforge.net/lists/listinfo/sipp-users), so that we can review and possibly integrate it in SIPp.

-- [EdwardEstabrook](https://twiki.polycom.com/twiki/bin/view/Main/EdwardEstabrook) - 02 Mar 2011

[Edit](https://twiki.polycom.com/twiki/bin/edit/Main/SIPpedReferenceGuide?t=1365031230" \o "Edit this topic text) | [Attach](https://twiki.polycom.com/twiki/bin/attach/Main/SIPpedReferenceGuide) | [Print version](https://twiki.polycom.com/twiki/bin/view/Main/SIPpedReferenceGuide?cover=print) | [History](https://twiki.polycom.com/twiki/bin/oops/Main/SIPpedReferenceGuide?template=oopshistory): r58 [<](https://twiki.polycom.com/twiki/bin/compare/Main/SIPpedReferenceGuide?rev1=57;rev2=58) [r57](https://twiki.polycom.com/twiki/bin/view/Main/SIPpedReferenceGuide?rev=57) [<](https://twiki.polycom.com/twiki/bin/compare/Main/SIPpedReferenceGuide?rev1=56;rev2=57) [r56](https://twiki.polycom.com/twiki/bin/view/Main/SIPpedReferenceGuide?rev=56) [<](https://twiki.polycom.com/twiki/bin/compare/Main/SIPpedReferenceGuide?rev1=55;rev2=56) [r55](https://twiki.polycom.com/twiki/bin/view/Main/SIPpedReferenceGuide?rev=55) | [Backlinks](https://twiki.polycom.com/twiki/bin/view/Main/SIPpedReferenceGuide?template=backlinksweb) | [View wiki text](https://twiki.polycom.com/twiki/bin/view/Main/SIPpedReferenceGuide?raw=on) | [Edit wiki text](https://twiki.polycom.com/twiki/bin/edit/Main/SIPpedReferenceGuide?t=1365031230;nowysiwyg=1) | [More topic actions](https://twiki.polycom.com/twiki/bin/view/Main/SIPpedReferenceGuide?template=more&maxrev=58&currrev=58)

Topic revision: r58 - 05 Oct 2012 - 21:20:18 - [EdwardEstabrook](https://twiki.polycom.com/twiki/bin/view/Main/EdwardEstabrook)