*SIPp*

## **Introduction**

This document provides guidelines for using SIPp in order to develop the stress/load testing environment for Asterisk platform in different call scenarios i e Inbound and Outbound.

## **Requirements:**

IMPORTANT: In order to develop and execute the examples, you will need:

* SIPp installed on the local system
* Asterisk installed

## **What is SIPp:**

SIPp is a tool which used to send and receive SIP messages between client (UAC -User agent client) and server (UAS - User agent server).

SIPp Relevant Commands

## **Installation:**

* On Linux, SIPp is provided in the form of source code. You will need to compile SIPp to actually use it.
* Pre-requisites to compile SIPp are:
  + C++ Compiler
  + curses or ncurses library
  + For TLS support: OpenSSL >= 0.9.8
  + For pcap play support: libpcap and libnet
  + For SCTP support: lksctp-tools
  + For distributed pauses: [Gnu Scientific Libraries](http://www.gnu.org/software/gsl/)
* You have four options to compile SIPp:
  + Without TLS (Transport Layer Security), SCTP or PCAP support – this is the recommended setup if you don’t need to handle SCTP, TLS or PCAP:
  + Source package can be install from
  + # wget <https://github.com/SIPp/sipp/releases/download/v3.5.0/sipp-3.5.0.tar.gz>
  + # tar -xvzf sipp-xxx.tar   
    # cd sipp  
    # ./configure  
    # make
  + With TLS support, you must have installed [OpenSSL library](http://www.openssl.org/) (>=0.9.8) (which may come with your system). Building SIPp consists only of adding the --with-openssl option to the configure command:
  + tar -xvzf sipp-xxx.tar.gz  
    cd sipp  
    ./configure --with-openssl  
    make
  + With PCAP play support:
  + tar -xvzf sipp-xxx.tar.gz  
    cd sipp  
    ./configure --with-pcap (if pcap.h is missing then use => yum -y install libpcap-devel )  
    Make
  + With SCTP support:
  + tar -xvzf sipp-xxx.tar.gz  
    cd sipp  
    ./configure --with-sctp  
    make
  + You can also combine these various options, e.g.:
  + tar -xvzf sipp-xxx.tar.gz  
    cd sipp  
    ./configure --with-sctp --with-pcap --with-openssl **(RECOMMENDED)**  
    make

## **SIPp relevant command line options:**

-sf <custom.xml> :- load a custom scenario file

-sn <built in scenario> :- Use a built in scenario, e.g (uac , uas)

-d < Durations in mili sec >:- Duration of the calls

-r < count > :- Set the call rate (in calls per seconds).

-l < Max calls >:- 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).

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

-s < extension > :- Set the extension, which user wants to dial when dialing the call.

-i < local ip address > :- local IP address

-set <key> <value> :- Set is used to set the parameter which needs to send along with the sip headers.

-trace\_msg :- Set SIP messages in a file and set the file name as <sceneriao filename>\_<pid>\_messages.log

-trace\_err :- Set error messages in a file and set the file name as <sceneriao filename>\_<pid>\_errors.log

-trace\_err :- Set error messages in a file and set the file name as <sceneriao filename>\_<pid>\_errors.log

## **SIPp scenario file**

SIPp scenario file is used to handle the sip messages, which generates when communication establish between UAC and UAS .

Syntax :

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

<!DOCTYPE scenario SYSTEM "sipp.dtd">

<scenario name="Basic UAS">

// here Need to handle SIP messages by using different scenarios tags.

</scenario>

## 

## **Scenario tags**

This section cover the information regarding the tags which used to develop the scenarios to handle the SIP messages.

* **<send>** : send a SIP message or a response. Important attributes are:
  + **retrans** : set the T1 timer for this message in milliseconds
  + **lost** : show packet loss, value in percentage
* **<recv>** : wait for a SIP message or response. Important attributes are:
  + **response** : indicates what SIP message code is expected
  + **request** : indicates what SIP message request is expected
  + **optional** : Indicates if the message to receive is optional. If optional is set to "global", SIPp will look every previous steps of the scenario
  + **lost** : show packet loss, value in percentage
  + **timeout** : specify a timeout while waiting for a message. If the message is not received, the call is aborted
  + **ontimeout** : specify a label to jump to if the timeout popped regexp\_match: boolean. Indicates if 'request' ('response' is not available) is given as a regular expression.
* The **recv** command can also include the action tag defining the action to execute upon the message reception
* **pause** : pause the scenario execution. Important attributes are:
  + **milliseconds** : time to pause in milliseconds
  + **variable** : scenario variable defining the pause time
* **nop** : the **nop** action doesn’t do nothing at SIP signalling level, is just a tag containing the **action** subtag
* **label** : a label is used when you want to branch to specific parts in your scenarios

## **Scenario Examples**

**UAC Scenario:**

1. UAC starts with send command as per syntax mention below :

<send retrans="500">

<![CDATA[

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

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

From: sipp <sip:sipp@[local\_ip]:[local\_port]>;tag=[pid]SIPpTag00[call\_number]

To: [service] <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\_[$mydest]

Content-Type: application/sdp

Content-Length: [len]

v=0

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

s=-

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

t=0 0

m=audio [media\_port] RTP/AVP 8 101

a=rtpmap:8 PCMA/8000

a=rtpmap:101 telephone-event/8000

a=fmtp:101 0-15

a=ptime:20

]]>

</send>

2) Now scenario waits for answer the call from uas side, 100 Trying , 180 Ringing are optional,

but 200 OK is mandatory as after it the sip dialog will be created.

<recv response="100"

optional="true">

</recv>

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

</recv>

<recv response="183" optional="true">

</recv>

-->

<recv response="200" rtd="true">

</recv>

3) Now After 200 OK, ACK needs to send from UAC to UAS so that the communication between them get establish and media can transmit.

<send>

<![CDATA[

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

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

From: sipp <sip:sipp@[local\_ip]:[local\_port]>;tag=[pid]SIPpTag00[call\_number]

To: [service] <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

Content-Length: 0

]]>

</send>

4) Now we need to send media in the form of DtMF ie pcap file for different digits.

<nop>

<action>

<exec play\_pcap\_audio="../pcap/dtmf\_2833\_1.pcap"/>

</action>

</nop>

<pause milliseconds="3000"/>

<nop>

<action>

<exec play\_pcap\_audio="../pcap/dtmf\_2833\_2.pcap"/>

</action>

</nop>

<pause milliseconds="3000"/>

5) At last we need to send BYE to hang up the call and receive its acknowledgement from UAS

side.

<send retrans="500" crlf="true">

<![CDATA[

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

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

From: sipp <sip:sipp@[local\_ip]:[local\_port]>;tag=[pid]SIPpTag00[call\_number]

To: [service] <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\_[$mydest]

Content-Length: 0

]]>

</send>

<recv response="200" crlf="true">

</recv>

Note : Complete file be shared with on git link.

**UAS Scenario:**

**1)** An UAS scenario starts with a <recv> command, the scenario replies with 180 Ringing.

<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>

2) Now to answer the call send 200 OK and receive acknowledgement for the respective

Message.

<send retrans="500">

<![CDATA[

SIP/2.0 200 OK

[last\_Via:]

[last\_From:]

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

[last\_Call-ID:]

[last\_CSeq:]

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

Content-Type: application/sdp

Content-Length: [len]

v=0

o=sipp 87308505 1 IN IP[local\_ip\_type] [local\_ip]

s=-

t=0 0

m=audio [media\_port] RTP/AVP 8 101

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

a=rtpmap:8 PCMA/8000

a=rtpmap:101 telephone-event/8000

a=fmtp:101 0-15

a=ptime:20

]]>

</send>

<recv request="ACK" crlf="true">

</recv>

3) Send DTMF using the tags as similarly send for uac

<nop>

<action>

<exec play\_pcap\_audio="../pcap/dtmf\_2833\_1.pcap"/>

</action>

</nop>

<!-- This delay can be customized by the -d command-line option -->

<!-- or by adding a 'milliseconds = "value"' option here. -->

<pause milliseconds="3000"/>

<nop>

<action>

<exec play\_pcap\_audio="../pcap/dtmf\_2833\_2.pcap"/>

</action>

</nop>

<pause milliseconds="3000"/>

4) At last we need to receive BYE and send ACK to hang up the call and receive its

acknowledgement from UAC.

<recv request="BYE">

</recv>

<send>

<![CDATA[

SIP/2.0 200 OK

[last\_Via:]

[last\_From:]

[last\_To:]

[last\_Call-ID:]

[last\_CSeq:]

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

Content-Length: 0

]]>

</send>

**Command to run SIPp on CLI :**

**Command to run UAS :**

sudo sipp -sf <scenario file> -i <local ip> -p <local port> -m 1 -trace\_msg -trace\_err

**Command to run UAC :**

sudo sipp -sf <scenario file> -set <key> <value> -s <service> -i <local ip> <remote ip>:<remote

port> -m 10 -trace\_err -trace\_msg

**Note :**

To add the variable using “ -set ” option, corresponding variable need to set as global variable

and reference variable in scenario file. Example added in the respective uac.xml.

## **Setup with Asterisk :**

1) we need to create the IVR flow on which the stress testing will perform.

Filename: /etc/asterisk/extensions.conf

Asterisk context to handle the call (context will work for both inbound and outbound case).

[SIPp]

exten =>\_X!,1,Answer()

exten =>\_X!,n,Verbose(1, SIPp)

; First DTMF

exten =>\_X!,n,Read(dtmf,please-try-again,1,,,10)

exten =>\_X!,n,GotoIf($["${dtmf}"==""]?invalid:)

exten =>\_X!,n,GotoIf($[(${dtmf}>=1) & (${dtmf}<=2)]?:invalid)

exten =>\_X!,n,GotoIf($["${dtmf}"=="1"]?:else)

; second DTMF

exten =>\_X!,n,Read(dtmf\_1,basic-pbx-ivr-main,1,,,10)

exten =>\_X!,n,GotoIf($["${dtmf\_1}"==""]?invalid:)

exten =>\_X!,n,GotoIf($[(${dtmf\_1}>=1) & (${dtmf}<=2)]?:invalid)

exten =>\_X!,n,GotoIf($["${dtmf\_1}"=="2"]?:else)

exten =>\_X!,n,Playback(hello-world)

exten =>\_X!,n,Goto(hangup)

exten =>\_X!,n(else),Playback(auth-thankyou)

exten =>\_X!,n,Goto(hangup)

exten =>\_X!,n(invalid),Playback(/home/local/CORPORATE/naman/AirIndiaFlightStatus/invalidTryAgain)

exten =>\_X!,n(hangup),verbose("call Complete")

;exten => \_X!,n,Hangup()

exten =>\_X!,n,Wait(10)

2) For outbound we need to create sip trunk as to communicate between SIPP and asterisk.

Filename: /etc/asterisk/sip.conf

[sipp\_trunk]

host=127.0.0.1

port=5080

context=SIPp

type=peer

dtmfmode=rfc2833

insecure=very

canreinvite=no

3) Need to check if sip trunk get register successfully :

# sip show peers (on asterisk cli)

**Git Repo :** [**https://github.com/manglikPhonon/SIPp.git**](https://github.com/manglikPhonon/SIPp.git)

Reference Doc :

<http://sipp.sourceforge.net/doc/uac.xml.html>

<https://sipp-wip.readthedocs.io/en/latest/installation.html>

<https://github.com/SIPp/sipp/releases>

<https://github.com/saghul/sipp-scenarios>

### Setup on AWS

1. Installation of SIPp.
2. Add following section in /etc/asterisk/sip\_custom.conf

[sipp\_trunk]

host=172.31.24.188

port=5080

type=peer

context=SIPp

dtmfmode=rfc2833

canreinvite=no

3) Add following section in /etc/asterisk/extensions\_custom.conf

[SIPp\_original]

exten =>\_X!,1,Answer()

exten =>\_X!,n,Verbose(1, SIPp)

exten =>\_X!,n,Read(dtmf,please-try-again,1,,,10)

exten =>\_X!,n,GotoIf($["${dtmf}"==""]?invalid:)

exten =>\_X!,n,GotoIf($[(${dtmf}>=1) & (${dtmf}<=2)]?:invalid)

exten =>\_X!,n,GotoIf($["${dtmf}"=="1"]?:else)

exten =>\_X!,n,Read(dtmf\_1,basic-pbx-ivr-main,1,,,10)

exten =>\_X!,n,GotoIf($["${dtmf\_1}"==""]?invalid:)

exten =>\_X!,n,GotoIf($[(${dtmf\_1}>=1) & (${dtmf}<=2)]?:invalid)

exten =>\_X!,n,GotoIf($["${dtmf\_1}"=="2"]?:else)

exten =>\_X!,n,Playback(hello-world)

exten =>\_X!,n,Goto(hangup)

exten =>\_X!,n(else),Playback(auth-thankyou)

exten =>\_X!,n(invalid),Playback(hello-world)

exten =>\_X!,n(hangup),verbose("call Complete")

exten => \_X!,n,Hangup()

[SIPp]

exten =>\_X!,1,Answer()

exten =>\_X!,n,Verbose(1, extension is checking ${EXTEN})

exten =>\_X!,n,GotoIf($["${EXTEN}"=="919560585295"]?:continue)

exten =>\_X!,n,Verbose(1, extension is ${EXTEN} and going to SIPP\_Out\_Call)

exten =>\_X!,n,Goto(SIPP\_Out\_Call,${EXTEN},1)

exten =>\_X!,n(continue),Verbose(1, SIPp)

exten =>\_X!,n,Verbose(1, ${CALLERID(all)})

exten =>\_X!,n,wait(5)

exten =>\_X!,n,dial(SIP/919560585295@sipp\_trunk)

[SIPP\_Out\_Call]

exten =>919560585295,1,Answer()

exten =>919560585295,n,Verbose(1, SIPP\_Out\_Call)

exten =>919560585295,n,Wait(10)

4) Create outbound and inbound xml files in media folder.

\*) Sample [Inbound XML File](https://drive.google.com/open?id=1qRHPcByQg93WFqDjp53KQ2r4DOJLqVVp)

\*) Sample [Outbound XML File](https://drive.google.com/open?id=1JLo7RyR1FGkstOnDHsBmFoNiFxEDnWnR)

5) Following commands are used for testing flow.

Outbound :-

sudo sipp -sf media/outbound\_load\_test.xml -i 172.31.24.188 -p 5080 -m 1 -trace\_msg -trace\_err

Inbound :-

sudo sipp -sf media/inbound\_load\_test.xml -i 172.31.24.188 172.31.24.188:5061 -s 912238615113 -m 500 -trace\_err -trace\_msg

**Setup of Central Load test (SIPP and Asterisk) :**

**Git Repo :** [**https://github.com/manglikPhonon/SIPp.git**](https://github.com/manglikPhonon/SIPp.git)

### Setup on 192.168.1.42:

**In extensions.conf**

1. [root]

exten => 1001,1,Goto(central\_default\_inbound,2657124249,1)

exten => 10001,1,Goto(central\_default\_inbound,2657124244,1)

exten => 2001,1,Goto(basic\_IVR\_function,12345,1)

exten => 3001,1,Goto(conference,12345,1)

;exten => 4001,1,Goto(SIPp\_outbound,12345,1)

2) [central\_default\_inbound]

include => parkedcalls

;exten => \_X.,1,Answer()

;exten => \_X.,n,Park()

;exten => \_X.,1,dial(sip/out/9739962916,,ToR)

;exten => \_X.,1,wait(3)

;exten => \_X.,n,GotoIfTime(00:00-00:12,sat,30,mar?closed,s,1)

exten => \_X.,1,Set(DisconnectedBy=CALLER)

exten => \_X.,n,Set(CDR(Host)=pbx-1)

exten => \_X.,n,NoOp(${CALLERID(num)})

exten => \_X.,n,GotoIf($["${LEN(${CALLERID(num)})}" = "10"]?setcallerid:continue)

exten => \_X.,n(setcallerid),set(CALLERID(num)=91${CALLERID(num)})

exten => \_X.,n(continue),NoOp(${CALLERID(num)});

exten => \_X.,n,NoOp(${DNID} , ${CALLERID(dnid)})

exten => \_X.,n,NoOp(${dnid} )

exten => \_X.,n,set(dnid=${EXTEN})

exten => \_X.,n,set(CALLERID(dnid)=${EXTEN})

exten => \_X.,n,set(DID=${EXTEN})

exten => \_X.,n,set(Customerno=${CALLERID(num)})

exten => \_X.,n,NoOp(${DNID} , ${CALLERID(dnid)})

exten => \_X.,n,agi(agi://127.0.0.1/inbound.agi?DID=${DID})

exten => \_X.,n,Set(DisconnectedBy=CALLEE)

;exten => \_X.,n,Hangup()

exten=> hold,1,MusicOnHold()

exten=> unhold,1,ConfBridge(${UNIQUEID})

exten => h,1,NoOp(${dnid},${EXTEN},${CALLERID(num)},${UNIQUEID},${DID},${HANGUPCAUSE},${REASON},${Customerno})

exten => h,n,NoOp(DisconnectedBy ${DisconnectedBy})

exten => h,n,agi(agi://127.0.0.1/hangup.agi?DID=${DID})

3) [basic\_IVR\_function]

exten =>\_X!,1,Answer()

exten =>\_X!,n,Verbose(1, Inside basic\_IVR\_function)

exten =>\_X!,n,Playback(hello-world)

exten =>\_X!,n,wait(10)

exten =>\_X!,n,hangup()

4) [conference]

exten => \_X!,1,Answer()

exten => \_X!,n,ConfBridge(${CDR(uniqueid)},basic\_bridge)

;exten => \_X!,n,ConfBridge(1234,basic\_bridge)

;exten => \_X!,n,Wait(300000)

exten => \_X!,n,Wait(5s)

exten => \_X!,n,Hangup()

5) [SIPp\_outbound]

exten =>\_X!,1,Answer()

exten =>\_X!,n,Verbose(1, extension is checking ${EXTEN})

exten =>\_X!,n(continue),Verbose(1, SIPp Outbound)

exten =>\_X!,n,Verbose(1, ${CALLERID(all)})

exten =>\_X!,n,dial(SIP/12345@sipp\_trunk)

In Sip.conf

1. As per the above configuration , **in [general] context, incoming calls context should be root** where all the incoming calls landed.
2. [sipp\_trunk] (sipp outbound trunk)

host=127.0.0.1

port=5080

context=SIPp\_outbound

type=peer

dtmfmode=rfc2833

canreinvite=no

**Commands used :**

**SIPp parameters :**

**-nr : Disable retransmission in UDP mode.**

**-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**

**-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).**

**-mp : Set the local RTP echo port number. Default is 6000.**

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

**is primary host IP address.**

**-s : Service name (extension on which you want to send call in asterisk)**

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

**-r : calls per second**

**-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).**

***For (sipp) uac to asterisk (inbound stress test) :***

**# *sudo sipp -sf scenario/central\_uac.xml -i 127.0.0.1 192.168.1.42:5060 -m 1 -s 1001 -r 1 -rtp\_echo -nr -nd -trace\_err -trace\_msg*** (disable retransmission in UDP using ‘-nr’ option and disable default behaviour of sipp.)

**# *sudo sipp -sf scenario/central\_uac.xml -i 127.0.0.1 192.168.1.42:5060 -m 1 -s 1001 -r 1 -rtp\_echo -nd -trace\_err -trace\_msg*** ( disable default behaviour of sipp only.)

**For sipp asterisk to uas(sipp) (outbound stress test) :**

***# sudo sipp -sf scenario/uas.xml -i 127.0.0.1 -p 5080 -rtp\_echo -trace\_msg -trace\_err***

TroubleShooting Steps:

SIPP :

1. configure: error: SCTP library missing :

# sudo yum install lksctp-tools-devel.x86\_64

2) configure: error: <openssl/md5.h> header missing :

# sudo yum install openssl-devel

3) compile with all packages :

# ./configure --with-sctp --with-pcap --with-openssl

4) Notice: Disconnecting Call due to lack of RTP packets :

# add -rtp\_echo in sipp command

Issues Occurred

1. **Warning: Retransmission Timeout :**
   * Exception occur due to the **retransmission of messages from sipp to asterisk, when the expected response of sip message (from asterisk) get delayed, asterisk throws the retransmission timeout warning.**
   * Due to the **sipp default nature,** after getting the respective warning from asterisk, **sipp hangup calls.**
   * Use of **‘-nr’ parameter in sipp cli will disable all the retransmission of messages in UDP**.
   * Due to the usage of ‘-nr ’ parameter, the number of calls for which retransmission occur will not be processed further and hence not shown in sipp details table.
   * One more sipp parameter ‘-nd’ is used to disable the default behaviour of sipp, due to which the calls get disconnected after getting retransmission timeout from uas side i.e asterisk.
2. **Warning: Exceptionally long voice queue length queuing :**

* Exception occur due to the **multiple conferences created at the time of stress/load testing** done using SIPP and asterisk.
* Observations on respective issue are :
  + Issues occurred when multiple conference created.
  + Independent on the architecture as issue occurred in both the below mention setup. (SIPP to Asterisk and Asterisk to Asterisk)
  + Changes in bridge profile impacts the system by increasing the number of members in confbrige.conf
  + On reaching to 200% (approx) of CPU utilization, system sink and the respective issue occured.
  + Maximum 100 simultaneous calls ( 100 conference rooms ) are successful with both setups (SIPP to Asterisk And Asterisk to Asterisk)
  + Independent on the rate of calls placed. (used rate 20/sec)
  + Independent on the version of Asterisk.
  + Very high configuration of system specifications and capacity is needed.

