SIPp cheatsheet

SIPp is a free test tool and traffic generator for the SIP protocol. It uses XML format files to define test scenarios.

```
General usage: sipp remote_host[:remote_port] [options]
```

Some important command-line options:

```
-sf filename
     Load test scenario from specified file.
-inf filename
     Use CSV file to insert data substituted for [field0], [field1], etc
     into XML scenario. First line of file describes order of inserting
     field sets (SEQUENTIAL/RANDOM/USE).
-sn name
     Use one of the embedded, predefined scenarios like "uac", "uas".
-r rate
     Scenario execution rate, default value = 10 times per period, default
     period = 1000 ms.
-rp period
     Scenario execution period [ms], combined with execution rate.
     Execution rate is combined of rate and period parameters, i.e. if
     period = 3500 and rate = 7 there will be 7 calls in 3.5 s.
-t transport mode
     Set the transport mode: "u1" - UDP, one socket (default), "un" - UDP,
     one socket per call, other modes (TCP and with compression)
     available.
-max_socket max
     Set the limit for simultaneously used sockets (for one socket per
     call mode). If limit is reached, sockets are reused.
-m calls
     Stop and exit after specified tests count.
-s service
     Set user part of the request URI (default: 'service'). Replaces
```

Sitemap

- PROJEKTY
- PROGRAMY
- NARZĘDZIA ONLINE
- INNE
  - Code Snippets
  - **SIPp**
  - SIPp/MinGW
  - Linux command line
  - LPT WinXP
  - Avrdude/MinGW
  - Dell D600
  - JY-MCU/HC-06 BT
  - ESP8266
- Linki
- Co nowego?
- Kontakt

Google™ Custom Search

Search

```
     [service] tag in XML scenario file.
-ap pass
     Set password used for auth challenges (default: 'password').
-l limit
     Limit simultaneous calls (default: 3 * call_duration (s) * rate).
-recv_timeout
     Global receive timeout (miliseconds). By default call is aborted, use
     ontimeout attribute to take other action.
-trace_msg
     Log sent and received SIP messages (file: scenario_pid_messages.log).
-trace_err
     Log error message to file (like "Discarding message which can't be
     mapped to a known SIPp call").
-sd
     Dumps one of the default scenarios. Usage example: sipp -sd uas >
     uas.xml.
```


Simple scenario files with usage

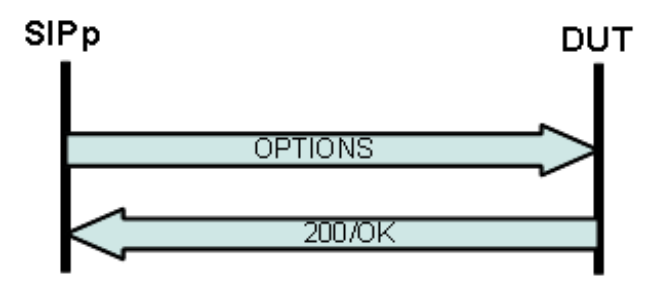
These scenario files were tested with sipp-win32-2009-06-06.

OPTIONS

Send OPTIONS message 5 times to 30@192.168.1.211.

```
sipp 192.168.1.211 -sf OPTIONS.xml -m 5 -s 30
```

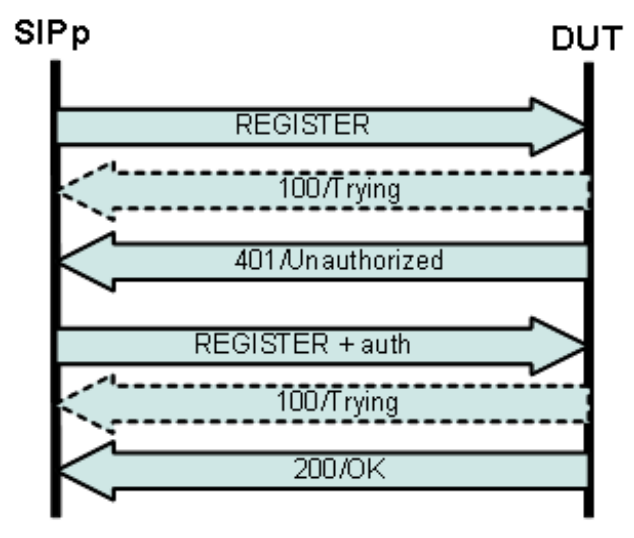
Send OPTIONS message 30 times to 30@192.168.1.211 waiting 200 ms for 200/OK reply each time.



```
sipp 192.168.1.211 -sf OPTIONS_recv_200.xml -m 30 -s 30
```

REGISTER

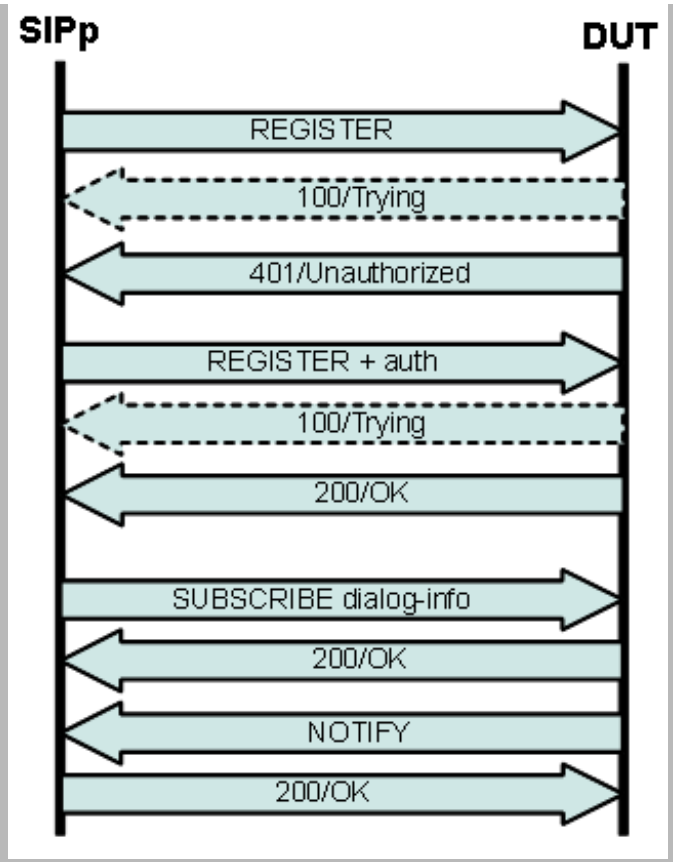
Register to 192.168.1.106 using parameters from CSV file. If CSV file has more than one entry you can increase simultaneous call limit (-l option).




```
sipp 192.168.1.106 -sf REGISTER_client.xml
  -inf REGISTER_client.csv -m 1 -l 1 -trace_msg -trace_err
```

REGISTER + SUBSCRIBE application/dialog-info+xml (BLF)

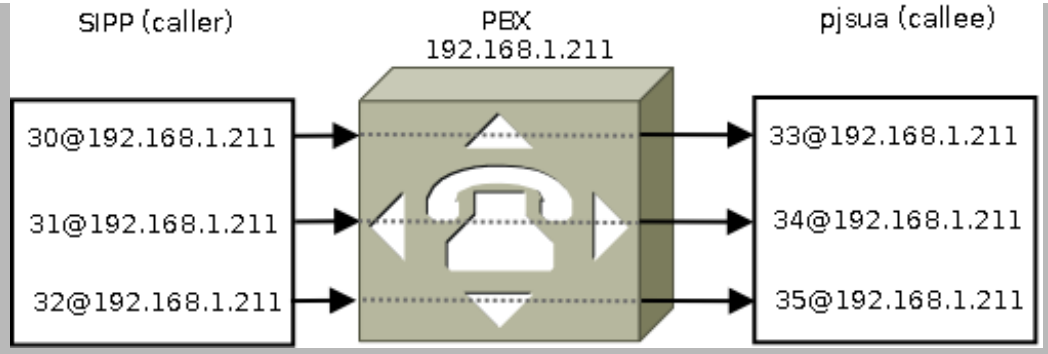
Register to 192.168.1.211 using parameters from CSV file and start dialog-info subscription (RFC4235).




```
sipp 192.168.1.211 -sf REGISTER_SUBSCRIBE_client.xml
    -inf REGISTER_SUBSCRIBE_client.csv -m 100 -l 2 -r 2
```

REGISTER + INVITE

SIPp is simulating 3 UACs, each one of them is making outgoing call. This scenario expects calls to be answered. Call targets are 3 other UACs configured to auto answer and play wav file (single pjsua instance with 3 accounts).




```
pjsua_vc6d --local-port=5068
    --id sip:33@192.168.1.211 --registrar sip:192.168.1.211
    --proxy sip:192.168.1.211 --realm * --username 33 --password 33
    --next-account --id sip:34@192.168.1.211 --registrar sip:192.168.1.211
    --proxy sip:192.168.1.211 --realm * --username 34 --password 34
    --next-account --id sip:35@192.168.1.211 --registrar sip:192.168.1.211
    --proxy sip:192.168.1.211 --realm * --username 35 --password 35
    --play-file file.wav --auto-answer 200 --auto-play
```

Each call is disconnected after 30 s. Call limit is this time smaller than number of CSV entries to avoid multiple calls to single target.

```
sipp 192.168.1.211 -sf REGISTER_INVITE_client.xml
    -inf REGISTER_INVITE_client.csv -m 100 -l 2 -r 1 -rp 10000
```

REGISTER + INVITE (2)

Some modification may be needed when calling operator that is using more complex proxy infrastructure.
1) Handling SIP/407 after INVITE.
2) Using rrs="true" and [routes] to keep Record-Route header set supplied by the operator.
3) Using [next_url] in ACK and BYE messages.

```
sipp 192.168.1.211 -sf REGISTER_INVITE_client2.xml
    -inf REGISTER_INVITE_client.csv -recv_timeout 10000 -m 1 -l 1
```

INVITE + CANCEL immediately after SIP/100

```
sipp 192.168.1.211 -sf INVITE_CANCEL.xml -recv_timeout 10000 -m 1 -l 1
```

INVITE with video stream SDP (H.263, H.264, AS/TIAS bandwidth modifiers)

```
sipp 192.168.1.211 -sf INVITE_SDP_video.xml -recv_timeout 30000 -m 1 -l 1
```

INVITE + re-INVITE with T38 offer

When detecting FAX tone 1st party sends re-INVITE with T38/image offer. Second party rejects offer with 488/Not Acceptable Here but call should not be disconnected.

```
sipp 192.168.0.192 -sf INVITE_T38_reINVITE.xml -s 30 -r 1 -l 12 -m 1
```

REGISTER UAS + SUBSCRIBE application/dialog-info+xml (BLF) UAS

Little tricky scenario that requires two actual scenario files. Sipp is simulating registration and BLF subscription server that immediately terminates subscription with reason=noresource.
1) With tested UAC create registration account 30@your_pc_ip_address, no password. Create dialog-info+xml subscription for 108@your_pc_ip_address.
2) Run UAS REGISTER scenario and wait for the phone to log in. 3) Break REGISTER scenario by hitting Ctrl+C and run UAS SUBSCRIBE scenario.

```
sipp -sf uas_register.xml
sipp -sf uas_subscribe.xml
```

REGISTER UAS sending unsolicited MWI NOTIFY messages

Sending unsolicited message-summary events to registered phone (31@192.168.0.228)

```
sipp 192.168.0.228 -s 31
  -sf NOTIFY_MWI_unsolicited.xml -m 1 -l 1 -r 1 -rp 1000
```

Session audit using UPDATE message

DUT is expected to send 200/OK with SDP offer but not changing session parameters.

```
sipp 192.168.0.228
  -sf INVITE_UPDATE_session_audit.xml -m 1 -l 1
```

REGISTER UAC + INVITE + DTMF INFO

1) SIP registration with authorization.
2) Calling number 110. In my test extension 110 is FXS and is looped back to FXO port and call is answered by DISA.
3) Dialing another 3-digit number using SIP INFO DTMF (Content-Type: application/dtmf-relay in this scenario). Call is not answered.
4) Disconnecting.

Note: if using "application/dtmf" Content-Type (message body consisting of dtmf only) sign "*" should be encoded as "10" and "#" as "11".

```
sipp 192.168.1.211 -sf REGC_INVITE_INFO.xml
    -inf REGC_INVITE_INFO.csv -m 5 -l 1 -r 1 -rp 10000
```

SIP digest leak test

SIP digest leak is a SIP phone vulnerability that allows attacker to get digest response from a phone and use it to guess password using brute-force method described first on enablesecurity.com page. Here are required steps:

1. attacker calls phone (direct IP call) sending INVITE frame,
2. callee picks up phone, connection is confirmed by both sites,
3. attacker does not send any RTP frames (at least does not have to) and just waits,
4. callee hangs up phone sending BYE request (probably throwing some profanities),
5. in response to BYE attacker sends SIP/401 or 407 message (authentification request),
6. if attack is successfull callee is sending BYE again with Authorization: Digest header added.

At this point attacker has authentification challenge (sent by him with 401/407 message) and response (received with last BYE). Most likely there will be simplest SIP/2.0 authentification scheme used (RFC2069):

```
ha1 = MD5(username ":" realm ":" password)
ha2 = MD5(method ":" req_uri)
response = MD5(ha1 ":" nonce ":" ha2)
```

Assuming that username and realm are known attacker can now use brute-force method to guess user password.

There are few conditions that have to be met to make this scheme work:

- SIP phone has to respond to authentication challenges sent by other sources than registration server(s) it is using (as a note it works with one hardware phone and one softphone I've tested (and those were all user agents I've tested)),
- phone SIP port has to accessible to attacker, usally phone will be placed behind the Restricted Cone NAT and port would not be forwarded,
- attacker would most likely have to know username and authentication realm used by target; for better security you probably should not leave "realm" configuration field of SIP phone empty (it could respond to challenges with any realm then making it easier to prepare attack),
- guessing password through brute-fource would be time consuming or almost impossible for more complex passwords.

```
sipp 192.168.1.211 -sf uac_digest_leak.xml -s 30 -m 1
```

Example result: digest_leak.log.

Generating calls using G.729 codec

If you ever had to make high-load call or even single-call tests with G.729(a) codec then you may find out that finding a free softphone with G729a capability is not an easy task. Obviously pjsua would be good choice, but it require downloading DirectX SDK, Intel Performance Primitives package and rebuilding from sources, so it would take few hours to get working binary.

Another option is capturing RTP stream using Wireshark and playing it back when generating or receiving calls with SIPp. Here is .pcap file with 2 minutes of G.729 RTP stream. This is actually recorded connection with some voice mail system. Extract this file to \pcap subdirectory of SIPp. Included scenario is UAC call, to get credible load test results you can call i.e. some DISA or auto-attendant lines that will play some announcement back.

```
sipp 192.168.1.211 -sf uac_pcap_G729.xml -l 1 -m 10
```

Call with video payload from captured RTP (H.264)

.pcap file with H.264 RTP stream

```
sipp 127.0.0.1 -sf uac_pcap_H264.xml -l 1 -m 1
```

Blind transfer

Registration with authorization, call and blind transfer

```
REGISTER ---------->
     100 <---------- (optional)
     401 <----------
REGISTER ---------->
     100 <---------- (optional)
     200 <----------
  INVITE ---------->
     100 <---------- (optional)
     180 <---------- (optional)
     183 <---------- (optional)
     200 <----------
     ACK ---------->
   Pause [   5000ms]
   REFER ---------->
     200 <----------
  NOTIFY <----------
     200 ---------->
  NOTIFY <----------
     200 ---------->
   Pause [   1000ms]
     BYE ---------->
     200 <----------


sipp 192.168.1.211 -sf REGISTER_INVITE_REFER.xml
    -inf REGISTER_INVITE_REFER.csv -m 1 -l 1 -r 1 -rp 10000
```

UAS with T38 reinvite

Simulating UAS that sends re-INVITE with image/t38 after detecting fax tone / preamble on fax reception. Since there is no actual tone/preamble detection script assumes that all calls are fax calls and reinvites all calls after short delay.
Note: actual fax transmission would fail after timeout due to no UDPTL endpoint presence.
Contains From/To headers content swapping using variables and ereg action.

```
sipp -sf uas_T38_reinvite.xml
```

For similar tests: rtp_pcma_fax_cng.pcap - CNG signal recorded from PCMA RTP stream (can be used with scenario similar to uac_pcap_G729.xml to test switching to fax extension on CNG detection).

Short announcement, CED signal and modem preable/training sequence as generated by software bundled with PC fax/modem I bought: rtp_pcma_fax_ced_training.pcap.

Running two or more scripts same time

Simulating two endpoints with single script seems difficult if not impossible, so some tests requre running two or more separate scripts (e.g. one for caller: REGISTER + outgoing INVITE, second for callee) same time. To synchronization between scripts create batch file:

```
start cmd /K sipp [parameters1]
start cmd /K sipp [parameters2]
[Enter]
```

With /K switch new cmd window will stay open after sipp exit. /C switch closes cmd window on application exit. If delay between scripts is needed (alternatively pause can be used inside scripts), e.g. callee must register before caller executes use ping as follows:

```
start cmd /K sipp [parameters1]
ping -n 3 127.0.0.1 > nul         // pauses for 3 - 1 = 2 seconds total
start cmd /K sipp [parameters2]
[Enter]
```

Pjsua as a scripted call generator

Pjsua sleep command allows to pipe commands from prepared text file to pjsua in a timely

manner making it possible to use it as limited but very easy to use call generator.

```
pjsua < commands.txt
```

Tools to generate scenario files from Wireshark traces

sniff2sipp - hosted by Digium, Perl script
Sippie (Sourceforge), Java based

Sipsak

For simple tasks such as sending single SIP message to remote destination sipsak may be handy.

Sending OPTION message:

```
sipsak -vv -s sip:192.168.1.10:5060
```

Sending custom message (NOTIFY Event: check-sync;reboot=true causing yealink phone to reboot):

```
sipsak -f reboot_yealink.sipfile -s sip:1234@192.168.0.195
```