

Siproxd Users Guide

Thomas Ries

Siproxd Users Guide

by Thomas Ries

Copyright © 2005-2007 Thomas Ries

This document can be freely redistributed according to the terms of the GNU General Public License.

Revision History

Revision 0.1 2005-04-10 Revised by: tries@users.sourceforge.net

Initial version

Revision 0.2 2006-07-28 Revised by: tries@users.sourceforge.net

Comment on Asterisk Scenario

Revision 0.3 2007-05-15 Revised by: tries@users.sourceforge.net

New Asterisk Config Files

Table of Contents

README.....	1
Important / Warning	1
1. Overview	2
2. Building and Installation.....	3
2.1. Prerequisites	3
2.2. Compiling and Installing.....	3
3. Configuration	5
3.1. The configuration file 'siproxd.conf'	5
3.2. Command Line Options	9
4. Features.....	10
4.1. Custom Firewall Module.....	10
4.2. Chroot() Jail	10
4.3. Quick Dial	10
5. Troubleshooting	11
5.1. Problem Reporting	11
5.2. Create a Debug Log	11
5.3. Siproxd crashes	11
6. Sample Configurations	13
6.1. The "Standard Scenario"	13
6.2. GS BT-100 behind NAT Router running Siproxd	13
6.3. GS BT-100 with Siproxd running "in front of" a NAT router	15
6.4. Transparent SIP Proxy	16
6.5. Masquerading an Asterisk box.....	17

README

Important information, please read me!

Important / Warning

As it still happens that people try to mix different NAT traversal technologies together with siproxd I'll put some words here:

- Do NOT USE anything like an STUN Server together with siproxd.
- Do NOT USE any additional technologies trying to help in NAT traversal (additional firewall modules like ip_nat_sip.ko or whatever fancy stuff may tempt you).

If you do not follow the above rules, those other "helping technologies" WILL DO CONFLICT with siproxd and result in a mess.

Chapter 1. Overview

Siproxd is an proxy/masquerading daemon for the SIP protocol. It handles registrations of SIP clients on a private IP network and performs rewriting of the SIP message bodies to make SIP connections possible via an masquerading firewall. It allows SIP clients (like kphone, linphone) to work behind an IP masquerading firewall or router.

SIP (Session Initiation Protocol, RFC3261) is used by Softphones and Hardphones (Voice over IP) to initiate communication. By itself, SIP does not work via masquerading firewalls as the transfered data contains IP addresses and port numbers.

There exist so called STUN servers that allow a SIP client to figure out its public visible IP address and use this one instead. As a drawback, usually on the masquerading firewall a very wide port range must be opened up for the incoming RTP traffic. The SIP client must support STUN (which most of them do).

Siproxd uses another approach (application layer proxy) and places itself as outbound proxy in between the local SIP client and the remote client or registrar. It does rewrite the SIP traffic on the fly and also includes a RTP proxy for incoming and outgoing RTP traffic (the actual audio data). The port range to be used for receiving RTP data is configurable, so the firewall only must allow incoming traffic for a small port range.

A standard scenario would look like:

```
private IP address range      :      Internet
10.0.0.x                      :      (public IP address range)
                              :
                              :      foo.bar.org
+-----+                    +-----+
!          !.10          .1 ! masquerading ! publicIP
! IntHost  !-----! Firewall !----->>
!          !          !          !
+-----+                    +-----+
                              :      eth0      :      ppp0
```

- The Firewall does IP masquerading and is running siproxd
- IntHost is running an SIP softphone (like linphone, kphone)
- The SIP address used by the softphone is sip:johndoe@foo.bar.org
- The softphone is configured to register itself at siproxd running on the firewall host (10.0.0.1) as sip:johndoe@foo.bar.org
- foo.bar.org is the domain name corresponding to the public IP address of the firewall (e.g. use some dynamic DNS service [1])

Chapter 2. Building and Installation

2.1. Prerequisites

Operating system of either:

- Linux (should work with any kernel)
- FreeBSD
- Solaris (porting is still being worked on but you may try it)

Additional required Packages:

- Libosip2 package (<http://www.gnu.org/software/osip>)

2.2. Compiling and Installing

It is quite simple. If you have a more-or-less standard installation and libosip2 installed at a standard location, it should be sufficient to do:

```
./configure
make
make install
```

This will install siproxd into /usr/local/. If you wish to install it into another location, specify **--prefix=<myprefix>** when running **./configure**. If you have installed libosip2 in a non-standard location use **--with-libosip-prefix=<libosipprefix>** to tell configure where to find libosip2 (e.g. **--with-libosip-prefix=\$HOME/lib**).

Common features for ./configure:

--enable-static	build statically linked executable
--with-libosip-prefix=DIR	use libosip2 from DIR/include and DIR/lib
--with-extra-includes=DIR	adds non standard include paths
--with-extra-libs=DIR	adds non standard library paths

Edit `/usr/etc/siproxd.conf` according to your situation, at least configure `if_inbound` and `if_outbound`. They must represent the interface names (e.g. on Linux: `ppp0`, `eth1`) for the inbound and outbound interfaces.

Edit `/usr/etc/siproxd_passwd.cfg` if you enable client authentication.

Start `siproxd`:

```
# siproxd
```

Chapter 3. Configuration

3.1. The configuration file 'siproxd.conf'

Siproxd by default searches for its configuration file in the following locations:

- \$HOME/.siproxdrc
- <buildingprefix>/etc/siproxd.conf
- /etc/siproxd.conf
- /usr/etc/siproxd.conf
- /usr/local/etc/siproxd.conf

The following is a list of directives that do exist. Note that string values **MUST NOT** contain spaces or tabs. Also read the explanations included in the supplied example configuration file for more explanation. Items with a # in front are normally disabled / not defined.

To start with siproxd in the first run, just adapt the interface definition for the inbound and outbound network interfaces (*if_inbound* and *if_outbound*).

Definition of network interfaces for the inbound network (local network where your SIP client is connected, this network normally uses IP addresses from one of the private IP ranges like 10.x.x.x, 192.168.x.x) and outbound network (your connection to the Internet, normally this interface has a public IP assigned by your provider).

```
if_inbound = eth0
if_outbound = ppp0
```

Usually only the *if_inbound* and *if_outbound* directives will be used. The *host_outbound* directive comes into play when running siproxd "in front of" a NAT router. Please check the configuration examples in this document for more details.

```
# host_outbound = <my_public_ip_address>
```

Access control lists for incoming SIP registrations and SIP traffic in general. These are comma separated lists of the form <IP>/<mask>, note that no spaces are allowed within the list (the configuration file parser cannot yet handle spaces).

```
# hosts_allow_reg = 192.168.1.0/24,192.168.2.0/24
# hosts_allow_sip = 123.45.0.0/16,123.46.0.0/16
# hosts_deny_sip = 10.0.0.0/8,11.0.0.0/8
```


Port to listen for incoming SIP messages. 5060 is usually the correct choice, don't change this unless you have a reason to.

```
sip_listen_port = 5060
```

Shall siproxd run as daemon? Usually 1 is the correct choice. If you want siproxd not to daemonize and keep running in foreground and writing its output to the terminal set this to 0.

```
daemonize = 1
```

Siproxd does log using the syslog() facility when running a daemon. This setting controls how much logging is done:

- 0 - DEBUGs, INFOs, WARNINGs and ERRORs
- 1 - INFOs, WARNINGs and ERRORs
- 2 - WARNINGs and ERRORs
- 3 - only ERRORs
- 4 - absolutely nothing

```
silence_log = 0
```

Siproxd can log call establishment to syslog.

```
log_calls = 1
```

If siproxd is started as root, it can drop the root privileges and change its user ID at startup. It also can put itself into a chroot() jail (see 4.2 for details)

```
user = nobody  
# chrootjail = /var/lib/siproxd/
```

Where to store the current registrations. This allows siproxd to remember registration across a restart. An empty value means we do not save registrations. The specified directory path must exist.

```
registration_file = /var/lib/siproxd/siproxd_registrations
```

Where to create the PID file.

```
pid_file = /var/run/siproxd/siproxd.pid
```

Enable/disable the RTP proxy. This must always be enabled. In some future release this directive may become obsolete.

```
rtp_proxy_enable = 1
```

Port range (UDP) that siproxd will use for incoming and outgoing RTP traffic. A firewall must be configured to allow traffic from and to these ports (UDP only). By default the range 7070 up to (and including) 7089 is used. This allows up to 10 simultaneous calls (2 ports per call). If you need more simultaneous calls, increase the range.

```
rtp_port_low  = 7070
rtp_port_high = 7089
```

Timeout for an RTP stream. If for the specified number of seconds no data is relayed on an active stream, it is considered dead and will be killed.

```
rtp_timeout = 300
```

If a REGISTER request does not contain an Expires header or expires= parameter in the Contact header, this number of seconds will be used and reported back to the UA in the answer.

```
default_expires = 600
```

If siproxd is used as registration server and authentication is wanted, define the following directive. If *proxy_auth_realm* is defined (a string), clients will be forced to authenticate themselves to the proxy (for registration only). To disable Authentication, simply comment out this line. Default is disabled.

```
# proxy_auth_realm = Authentication_Realm
```

The password to be used for authentication may be a global one

```
# proxy_auth_passwd = some_password
```

or on a per user base, stored in its own file. *proxy_auth_pwfile* takes precedence over *proxy_auth_passwd*

```
# proxy_auth_pwfile = /etc/mysiproxd_passwd.cfg
```

To enable additional debug output of siproxd. This is a bit pattern representing the following items. Default is 0x0 - disabled. See below in this document for information on how to create a debug log file.

- DBCLASS_BABBLE 0x00000001 // babble (like entering/leaving fnc)
- DBCLASS_NET 0x00000002 // network
- DBCLASS_SIP 0x00000004 // SIP manipulations
- DBCLASS_REG 0x00000008 // Client registration
- DBCLASS_NOSPEC 0x00000010 // non specified class

- DBCLASS_PROXY 0x00000020 // proxy
- DBCLASS_DNS 0x00000040 // DNS stuff
- DBCLASS_NETTRAF 0x00000080 // network traffic
- DBCLASS_CONFIG 0x00000100 // configuration
- DBCLASS_RTP 0x00000200 // RTP proxy
- DBCLASS_ACCESS 0x00000400 // Access list evaluation
- DBCLASS_AUTH 0x00000800 // Authentication

```
debug_level = 0x00000000
```

You may connect to this port from a remote machine and receive the debug output. This allows better creation of debug output on embedded systems that do not have enough memory for large disk files. Port number 0 means this feature is disabled.

```
debug_port = 0
```

Some UAs (SIP clients) will always use the host/ip they register TO as host part in the registration record (which will be the inbound ip address/hostname of the proxy) and can not be told to register a different host (public IP address). This Mask feature allows to force such a UA to be masqueraded to a different host. Siemens SIP Phones seem to need this feature. Normally disabled.

```
# mask_host=local.ip.of.sipphone
# masked_host=public.domaind.org
```

Siproxd itself can be told to send all traffic to another outbound proxy. You can use this feature to 'chain' multiple siproxd proxies if you have several masquerading firewalls to cross. Normally disabled.

```
# outbound_proxy_host = my.outboundproxy.org
# outbound_proxy_port = 5060
```

Outbound proxies can be specified on a per-domain base. This allows to use an outbound proxy needed for ProviderA and none (or another) for ProviderB. Multiple domain specific proxies may be specified, each one with one set of the following directives. Note: These directives must always be specified as a triple, skipping one of them will affect later definitions.

```
#outbound_domain_name = freenet.de
#outbound_domain_host = proxy.for.domain.freende.de
#outbound_domain_port = 5060
```

Quick-Dial "Plug-in": ability to define quick dial numbers that can be accessed by dialing "*nn" from a local phone. 'nn' corresponds to the entry number (pi_shortcode_entry) below. The '*' character can be chosen freely (pi_shortcode_key). Note: To call a real number like "*1234" you would have to dial "***1234"

```
pi_shortdial_enable = 1
pi_shortdial_akey = *
#
# *01 sipphone echo test
pi_shortdial_entry = 17474743246
# *02 sipphone welcome message
pi_shortdial_entry = 17474745000
```

3.2. Command Line Options

Siproxd knows the following command line options:

-h, --help	help
-d, --debug <pattern>	set debug-pattern
-c, --config <cfgfile>	use the specified config file

These options take precedence over the values configured in the configuration file.

Chapter 4. Features

4.1. Custom Firewall Module

The API

make your library

example code

```
./configure --with-custom-fwmodule=LIBRARY.a
```

4.2. Chroot() Jail

Create chroot jail

What files must be present?

4.3. Quick Dial

Since 0.5.12, Siproxd includes a Quick-Dial feature. This allows you to define SIP numbers that can be accessed by using a shortctu (like "*nn") from any local SIP phone.

For example, the following lines in your siproxd.conf will configure 2 Quick-Dial numbers:

```
# *01 sipphone echo test
pi_shortcode_entry = 17474743246
# *02 sipphone welcome message
pi_shortcode_entry = 17474745000
```

The numbering starts with "1" ("*01") and every following "pi_shortcode_entry" entry will allocate the following position. Curently it is not possible to freely assign the positions.

Chapter 5. Troubleshooting

5.1. Problem Reporting

If you encounter problems/crashes and ask for support, please include as much information as possible. Very helpful is a debug log that has been recorded at the time of the misbehavior. Also include the exact versions of the siproxd package and libosip2 that you are using. You should also include your `siproxd.conf`.

5.2. Create a Debug Log

The easiest way to generate a debug log is:

1. make sure siproxd is not started as daemon ('daemonize = 0' in the config file)
2. start siproxd: `$./siproxd -d -l 2>debug.log`
3. reproduce the error
4. include the generated `debug.log` in your error report

Another possibility of to use TCP logging. This method is recommended if you run siproxd on a router with limited disk space (e.g. an embedded system). To enable TCP logging:

1. Edit the configuration file and set `debug_port` to 5050 (or any other TCP port number you like).
2. Restart siproxd
3. `$ telnet <IP_of_siproxd> 5050 > debug.log`

You may prefer to use netcat instead of telnet. Note: The TCP debug port is bound to all available interfaces on the system, make sure no unauthorized people (like from the outbound network) can connect.

5.3. Siproxd crashes

If siproxd crashes, a stack back trace usually is helpful to me:

1. start siproxd in the debugger (daemonize set to 0):

```
$ gdb ./src/siproxd
```

```
(gdb) set args -c /path/to/siproxd.conf
```

```
(gdb) run
```

2. reproduce the crash

3. use gdb to print the stack backtrace:

```
(gdb) info thread
```

```
...
```

```
(gdb) bt
```

```
#0  0x400ec9ee in __select ()
```

```
#1  0xbffff6f8 in ?? ()
```

```
#2  0x804a5c2 in main (argc=3, argv=0xbfffc54) at siproxd.c:186
```

```
#3  0x4005bcb3 in __libc_start_main (main=0x804a30c <main>, argc=3,  
    argv=0xbfffc54, init=0x8049a08 <_init>, fini=0x804edac <_fini>,  
    rtdl_fini=0x4000a350 <_dl_fini>, stack_end=0xbfffc4c)  
    at ../sysdeps/generic/libc-start.c:78
```

```
(gdb)
```

4. copy-paste all the output and include it in your problem report.

Chapter 6. Sample Configurations

Check also the FAQ in the siproxd package.

6.1. The "Standard Scenario"

Scenario:

```
private IP address range      :      Internet
10.0.0.x                      :      (public IP address range)
                              :
                              :      foo.bar.org

+-----+                     +-----+
!          !.10             .1 ! masquerading ! publicIP
! IntHost  !-----! Firewall  !----->>
!          !               !               !
+-----+                     +-----+
                              :      eth0      :      ppp0
```

The Firewall does IP masquerading and is running siproxd. IntHost is running an SIP softphone (like linphone, kphone). The SIP address used by the softphone is sip:johndoe@foo.bar.org. The softphone is configured to register itself at siproxd running on the firewall host (10.0.0.1) as sip:johndoe@foo.bar.org. Foo.bar.org is the domain name corresponding to the public IP address of the firewall (e.g. use some dynamic DNS service like DynDNS).

Firewall configuration (iptables):

```
# allow incoming SIP and RTP traffic
iptables -A INPUT -m udp -p udp -i ppp0 --dport 5060 -j ACCEPT
iptables -A INPUT -m udp -p udp -i ppp0 --dport 7070:7089 -j ACCEPT
```

Firewall configuration (ipchains):

```
# allow incoming SIP and RTP traffic
ipchains -A input --proto udp --dport 5060 -j ACCEPT
ipchains -A input --proto udp --dport 7070:7089 -j ACCEPT
```

The first line will allow incoming SIP traffic. The second line will allow incoming RTP traffic on the ports 7070 - 7089 (the default port range used by siproxd for incoming RTP traffic).

6.2. GS BT-100 behind NAT Router running Siproxd

Scenario:

```

private IP address range      :      Internet
10.0.0.x                      :      (public IP address range)
                              :
                              :      foo.bar.org
+-----+                    +-----+
!          !.10              .1 ! masquerading ! publicIP
! SIP UA   !-----! Firewall !----->>
! BT-100   !          ! siproxd !
+-----+                    +-----+
                              :
                              :      eth0      :      ppp0

```

Siproxd is running on the same host as the masquerading firewall. The SIP phone is a Grandstream BudgeTone-100. In this example the external SIP registrar used is sipphone.com (<http://www.sipphone.com/>).

siproxd.conf:

```

if_inbound  = eth0
if_outbound = ppp0
hosts_allow_reg = 10.0.0.0/24
sip_listen_port = 5060
daemonize = 1
silence_log = 1
log_calls = 1
user = siproxd
registration_file = /var/lib/siproxd/registrations
pid_file = /var/run/siproxd/siproxd.pid
rtp_proxy_enable = 1
rtp_port_low  = 7070
rtp_port_high = 7089
rtp_timeout = 300
default_expires = 600
debug_level = 0
debug_port = 0

```

Firewall configuration (iptables):

```

# allow incoming SIP and RTP traffic
iptables -A INPUT -m udp -p udp -i ppp0 --dport 5060 -j ACCEPT
iptables -A INPUT -m udp -p udp -i ppp0 --dport 7070:7089 -j ACCEPT

```

Phone configuration (only the relevant items are listed):

```

IP Address:      10.0.0.10
Subnet Mask:     255.255.255.0

```

```

Default Router: 10.0.0.1
DNS Server 1:      <DNS Server of your Internet provider>
SIP Server:        proxy01.sipphone.com
Outbound Proxy: 10.0.0.1
SIP User ID:       1747669xxxx
Authenticate ID:   1747660xxxx
Authenticate Passwd: *****
Name:              Your Name Here
Use DNS SRV:       no
User ID is phone #: no
Sip Registration:  yes
Unregister on reboot: no
Register expiration: 60
Early Dial:        no
local SIP port: 5060
local RTP port: 5004
Use random port:   yes
NAT traversal:     no
Use NAT IP:        <empty>
Subscribe for MWI: No
Send DTMF:         via RTP (RFC2833)

```

6.3. GS BT-100 with Siproxd running "in front of" a NAT router

Scenario:

```

private IP address range      :      Internet
10.0.0.x                      :      (public IP address range)
                              :
                              :      foo.bar.org
+-----+                     +-----+
!          !.10              .1 ! masquerading ! publicIP
! SIP UA   !-----! NAT router !----->>
! BT-100   !          !          !
+-----+         !          +-----+
                  !      eth0      :      ppp0
                  !          :
                  !          :
eth0 !.2
+-----+
!   siproxd   !
!           !
+-----+

```

Siproxd is running on 10.0.0.2. The masquerading NAT router (e.g. a ADSL NAT router that cannot run any user applications).

siproxd.conf:

```

if_inbound  = eth0
if_outbound = eth0
host_outbound = foo.bar.org
hosts_allow_reg = 10.0.0.0/24
sip_listen_port = 5060
daemonize = 1
silence_log = 1
log_calls = 1
user = siproxd
registration_file = /var/lib/siproxd/registrations
pid_file = /var/run/siproxd/siproxd.pid
rtp_proxy_enable = 1
rtp_port_low  = 7070
rtp_port_high = 7089
rtp_timeout = 300
default_expires = 600
debug_level = 0
debug_port = 0

```

NAT router configuration:

```

forward all incoming traffic on 5060/udp to 10.0.0.2
forward all incoming traffic from 7070/udp - 7089/udp to 10.0.0.2

```

Phone configuration:

```

IP Address:          10.0.0.10
Subnet Mask:         255.255.255.0
Default Router: 10.0.0.1
DNS Server 1:        <DNS Server of your Internet provider>
SIP Server:          proxy01.sipphone.com
Outbound Proxy: 10.0.0.2
SIP User ID:         1747669xxxx
Authenticate ID:     1747660xxxx
Authenticate Passwd: *****
Name:                Your Name Here
Use DNS SRV:         no
User ID is phone #:  no
Sip Registration:    yes
Unregister on reboot: no
Register expiration: 60
Early Dial:          no
local SIP port: 5060
local RTP port: 5004
Use random port:     yes
NAT traversal:       no
Use NAT IP:          <empty>
Subscribe for MWI:   No
Send DTMF:           via RTP (RFC2833)

```

6.4. Transparent SIP Proxy

Scenario:

```

private IP address range      :      Internet
10.0.0.x                      :      (public IP address range)
                              :
                              :      foo.bar.org

+-----+                     +-----+
!         !.10                .1 ! masquerading ! publicIP
! SIP UA   !-----! Firewall   !----->>
!         !         ! siproxd   !
+-----+                     +-----+
                              :
                              :      eth0      :      ppp0

```

You may have a SIP UA (Phone) that does not allow the specification of an outbound proxy. If siproxd is running on the masquerading router, the following configuration will do so called transparent proxying. The firewall will redirect outgoing SIP messages to siproxd, however the local Client is not aware of it.

siproxd.conf:

```

if_inbound  = eth0
if_outbound = ppp0
hosts_allow_reg = 10.0.0.0/24
sip_listen_port = 5060
daemonize = 1
silence_log = 1
log_calls = 1
user = siproxd
registration_file = /var/lib/siproxd/registrations
pid_file = /var/run/siproxd/siproxd.pid
rtp_proxy_enable = 1
rtp_port_low  = 7010
rtp_port_high = 7019
rtp_timeout = 300
default_expires = 600
debug_level = 0
debug_port = 0

```

Firewall configuration (iptables):

```

# redirect outgoing SIP traffic to siproxd (myself)
iptables -t nat -A PREROUTING -m udp -p udp -i eth0 \
--destination-port 5060 -j REDIRECT

# allow incoming SIP and RTP traffic
iptables -A INPUT -m udp -p udp -i ppp0 --dport 5060 -j ACCEPT
iptables -A INPUT -m udp -p udp -i ppp0 --dport 7070:7089 -j ACCEPT

```

6.5. Masquerading an Asterisk box

Scenario:

```

private IP address range      :      Internet
10.0.0.x                      :      (public IP address range)
                              :
                              :      foo.bar.org

+-----+                     +-----+
!         !.10                .1 ! masquerading ! publicIP
! Asterisk !-----! Firewall !----->>
!         ! SIP trunk    ! siproxd !
+-----+                     +-----+
! ! ! ! !                     eth0      :      ppp0
...!!!!!!!!!!!!
extensions
(local SIP clients)

```

Siproxd can also be used to masquerade an Asterisk server. The Asterisk server will register itself as a SIP UA (Client) to an external SIP registrar. In this example this would be again sipphone.com. As Asterisk does not allow to specify an SIP outbound proxy we use the same setup for transparent proxying. The context values of the asterisk configuration probably must be adapted to fit your needs.

siproxd.conf:

```

if_inbound  = eth0
if_outbound = ppp0
hosts_allow_reg = 10.0.0.0/24
sip_listen_port = 5060
daemonize = 1
silence_log = 1
log_calls = 1
user = siproxd
registration_file = /var/lib/siproxd_registrations
pid_file = /var/run/siproxd/siproxd.pid
rtp_proxy_enable = 1
rtp_port_low  = 7070
rtp_port_high = 7089
rtp_timeout = 300
default_expires = 600
debug_level = 0
debug_port = 0

```

Firewall configuration (iptables):

```

# redirect outgoing SIP traffic to siproxd (myself)
iptables -t nat -A PREROUTING -m udp -p udp -i eth0 \
        --source 10.0.0.11 --destination-port 5060 -j REDIRECT
# allow incoming SIP and RTP traffic
iptables -A INPUT -m udp -p udp -i ppp0 --dport 5060 -j ACCEPT

```

```
iptables -A INPUT -m udp -p udp -i ppp0 --dport 7070:7080 -j ACCEPT
```

Asterisk configuration (SIP related part):

Note: Very important are the `fromuser` and `fromdomain` keywords in the client section. They are required to have Asterisk send the correct From headers in SIP dialogs. The used Asterisk version is 'SVN-branch-1.4-r62331M'.

With newer Asterisk versions, it is no longer required to have a separate REGISTER definition, this can be made implicit in the SIP trunk config.

```
; sip.conf:

[general]
port = 5060           ; Port to bind to (SIP is 5060)
bindaddr = 0.0.0.0    ; Address to bind to (all addresses on machine)
context = from-sip-external ; Send unknown SIP callers to this context

useragent = PBX       ; NOTE: some providers (e.g sipcall.ch) do simply
                      ;       not work with the default "AsteriskPBX"
                      ;       UA String.

; Network Settings
nat=never
localnet = 10.0.0.0/24
domain = 10.0.0.10

; Codecs
disallow=all
allow=gsm      ; 13 Kbps
allow=ulaw     ; 64 Kbps
allow=alaw     ; 64 Kbps
autoframing = yes

; SIP Settings
canreinvite = no      ; important!

; the following are just my settings I use, however
; I dont' consider them critical
allowexternaldomains = yes
allowexternalinvites = yes
allowguest = yes
allowsubscribe = no
allowtransfer = yes
alwaysauthreject = no
automain = yes
callevts = no
compactheaders = no
dumphistory = no
g726nonstandard = no
```

```

ignoreregexpire = no
jbenable = no
jbf force = no
jblog = no
maxcallbitrate = 384
maxexpiry = 3600
minexpiry = 180
notifyringing = no
pedantic = no
promiscredirect = no
recordhistory = no
relaxdtmf = no
rtcache friends = no
rtsavesysname = no
rtupdate = no
sendrpid = yes
sipdebug = no
tlmin = 100
progressinband = no
;register =
t38pt_udptl = no
trustrpid = no
usereqphone = no
videosupport = no

```

The Trunk definition looks like:

```

; users.conf:

[general]
;
; Full name of a user
;
fullname = New User
userbase = 200
;
; Create voicemail mailbox and use use macro-stdexten
;
hasvoicemail = yes
;
; Set voicemail mailbox 6000 password to 1234
;
vmsecret = 1234
;
; Create SIP Peer
;
hassip = yes
hasiax = no
;
; Create H.323 friend
;
;hash323 = yes

```

```

;
; Create manager entry
;
hasmanager = no
;
; Remaining options are not specific to users.conf entries but are general.
;
callwaiting = yes
threewaycalling = yes
callwaitingcallerid = yes
transfer = yes
canpark = yes
cancallforward = yes
callreturn = yes
callgroup = 1
pickupgroup = 1
host = dynamic
localextenlength = 3
allow_aliasextns = no
allow_an_extns = no
hasagent = no
hasdirectory = no

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Local SIP UAs
; = locally connected phones. nothing special here.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
[201]
callwaiting = yes
cid_number = 201
context = local_sip
email = e@mail
fullname = Full Name
group =
hasagent = yes
hasdirectory = yes
hasiax = no
hasmanager = no
hassip = yes
hasvoicemail = yes
host = dynamic
mailbox = 201
secret = sip_password
threewaycalling = yes
zapchan =
registeriax = no
registersip = yes
vmsecret = 1234

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; SIP Trunks

```



```

; these are masqueraded via siproxd
;;;;;;;;;;;;;
[trunk_1]
disallow = all
allow = gsm,ulaw,alaw,adpcm,speex,g729,g723
callerid =
contact = 17476691234      ; IMPORTANT
context = DID_trunk_1
dialformat = ${EXTEN:1}
fromdomain = proxy01.sipphone.com
fromuser = 17476691234      ; IMPORTANT
group =
hasexten = no
hasiax = no
hassip = yes
host = proxy01.sipphone.com
insecure = very
port = 5060
provider =
registeriax = no
registersip = yes
secret = sip_password
trunkname = Custom - sipphone1234
trunkstyle = customvoip
username = 17476691234

```