# Sofia SIP Stack

## About

Sofia is a SIP stack used by FreeSWITCH. When you see "sofia" anywhere in your configuration, think "This is SIP stuff." It takes a while to master it all, so please be patient with yourself. SIP is a crazy protocol and it will make you crazy too if you aren't careful. Read on for information on setting up SIP/Sofia in your FreeSWITCH configuration.

mod_sofia exposes the Sofia API and sets up the FreeSWITCH SIP endpoint.

---

## Endpoint

A FreeSWITCH endpoint represents a full user agent and controls the signaling protocol and media streaming necessary to process calls. The endpoint is analogous to a physical VoIP telephone sitting on your desk. It speaks a particular protocol such as SIP or Verto, to the outside world and interprets that for the FreeSWITCH core.

### Configuration Files

sofia.conf.xml contains the configuration settings for mod_sofia

See Sofia Configuration Files.

## SIP profiles

See SIP profiles section in Configuring FreeSWITCH.

## What if these commands don't work for me?

Make sure that you are not running another SIP server at the same time as FreeSWITCH. It is not always obvious that another SIP server is running. If you type in Sofia commands such as 'sofia status profile default' and it doesn't work then you may have another SIP server running. Stop the other SIP server and restart FreeSWITCH.

On Linux, you may wish to try, as a superuser (often "root"):

```bash
netstat -lunp | less
```

With the less search facility (usually the keystroke "/"), look for :5060 which is the usual SIP port.

To narrow the focus, you can use grep. In the example configs, port 5060 is the "internal" profile. Try this:

```bash
netstat -lnp | grep 5060
```

See if something other than FreeSWITCH is using port 5060.

## Sofia Recover

sofia recover

You can ask Sofia to recover calls that were up, after crashing (or other scenarios).

Sofia recover can also be used, if your core db uses ODBC to achieve HA / failover.

For FreeSWITCH HA configuration, see Freeswitch HA.

## Flushing and rebooting registered endpoints

You can flush a registration or reboot specific registered endpoint by issuing a flush_inbound_reg command from the console.

```bash
freeswitch> sofia profile <profile_name> flush_inbound_reg [<<call_id>|<user@host>] [reboot]
```

If you leave out <call_id> and/or <user@host>, you will flush/reboot every registered endpoint on a profile.
• Note: For polycom phone, the command causes the phone to check its configuration from the server. If the file is different (you may add extra space at the end of file), the phone will reboot. You should not change the value of volpProt.SIP.specialEvent.checkSync.alwaysReboot="0" to "1" in sip.cfg as that allows potential a DOS attack on the phone.

You can also use the check_sync command:

```
sofia profile <profile_name> check_sync <call_id> | <user@domain>
```

• Note: The polycom phones do not reload <mac>-directory.xml configuration in response to either of these commands, they only reload the configuration. If you want new speed dials to take effect, you'll need to do a full reboot of the phone or enable the alwaysReboot option. (Suggestions from anyone with more detailed PolyCom knowledge would be appreciated here.)

Starting a new profile

If you have created a new profile you need to start it from the console:

```
freeswitch> sofia profile <new_profile_name> start
```

Reloading profiles and gateways

You can reload a specific SIP profile by issuing a rescan/restart command from the console

```
freeswitch> sofia profile <profile_name> [<rescan]|<restart>] reloadxml
```

The difference between rescan and restart is that rescan will just load new config and not stop FreeSWITCH from processing any more calls on a profile. Some config options like IP address and (UDP) port are not reloaded with rescan.

Deleting gateways

You can delete a specific gateway by issuing a killgw command from the console. If you use _all_ as gateway name, all gateways will be killed

```
freeswitch> sofia profile <profile_name> killgw <gateway_name>
```

Restarting gateways

You can force a gateway to restart (good for forcing a re-registration or similar) by issuing a killgw command from the console followed by a profile rescan. This is safe to perform on a profile that has active calls.

```
freeswitch> sofia profile <profile_name> killgw <gateway_name>
freeswitch> sofia profile <profile_name> rescan
```

Adding / Changing Existing Gateways

It will be assumed that you have all your gateways in the /usr/local/freeswitch/conf/sip_profiles/external directory and that you have just created a new entry. You can add a new gateway to FreeSWITCH by issuing a rescan reloadxml command from the console as seen in the example below. This will load the newly created gateway and not affect any calls that are currently up.

```
freeswitch> sofia profile external rescan reloadxml
```

You now realize that you have screwed up the IP address in the new gateway and need to change it. So you edit your gateway file and make any changes that you want. You will then need to issue the following commands to destroy the gateway, and then have FreeSWITCH reload the changes with affecting any existing calls that are currently up.

```
freeswitch> sofia profile external killgw <gateway_name>
freeswitch> sofia profile external rescan reloadxml
```

View SIP Registrations

You can view all the devices that have registered by running the following from the console.

```
freeswitch> sofia status profile <profile name> reg
freeswitch> sofia status profile default reg
freeswitch> sofia status profile outbound reg
```

You can also use the xmlstatus key to retrieve statuses in XML format. This is specially useful if you are using mod_xml_rpc.

Commands are as follows:

```
freeswitch> sofia xmlstatus profile <profile name> reg
freeswitch> sofia xmlstatus profile default reg
freeswitch> sofia xmlstatus profile outbound reg
```
List the status of gateways

For the gateways that are in-service:

`freeswitch> sofia profile <profile> gwlist up`

For the gateways that are out-of-service:

`freeswitch> sofia profile <profile> gwlist down`

Notes:

- It should be used together with `<param name="ping" value="<sec>/">`. See Sofia_Configuration_Files
- It can also be used to feed into mod distributor to exclude dead gateways.

List gateway data

To retrieve the value of an inbound variable:

`sofia_gateway_data <gateway_name> ivar <name>`

To retrieve the value of an outbound variable:

`sofia_gateway_data <gateway_name> ovar <name>`

To retrieve the value of either use:

`sofia_gateway_data <gateway_name> var <name>`

This first checks for an inbound variable, then checks for an outbound variable if there's no matching inbound.

View User Presence Data

Displays presence data from registered devices as seen by the server

Usage:

`sofia_presence_data [list|status|rpid|user_agent] [profile/]<user>@domain`

`sofia_presence_data list */2005`
status|rpid|user_agent|network_ip|network_port
Away|away|Bria 3 release 3.5.1 stamp 69738|192.168.20.150|21368
+OK

It's possible to retrieve only one value

`sofia_presence_data status */2005`
Away

You can use this value in the dialplan, e.g.

```xml
<extension name="12005">
  <condition field="destination_number" expression="^(12005)$" require-nested="false">
    <condition field="$(sofia_presence_data status */2005@$domain)" expression="^(Busy)$">
      <action application="playback" data="ivr/8000/ivr-user_busy.wav"/>
      <action application="hangup"/>
    </condition>
  </condition>
  <action application="bridge" data="user/2005@$domain_name"/>
</extension>
```

Debugging Sofia-SIP

The Sofia-SIP components can output various debugging information. The detail of the debugging output is determined by the debugging level. The level is usually module-specific and it can be modified by module-specific environment variable. There is also a default level for all modules, controlled by environment variable `#SOFIA_DEBUG`.

The environment variables controlling the logging and other debug output are as follows:
The defined debug output levels are:

- 0 SU_DEBUG_0() - fatal errors, panic
- 1 SU_DEBUG_1() - critical errors, minimal progress at subsystem level
- 2 SU_DEBUG_2() - non-critical errors
- 3 SU_DEBUG_3() - warnings, progress messages
- 5 SU_DEBUG_5() - signaling protocol actions (incoming packets, ...)
- 7 SU_DEBUG_7() - media protocol actions (incoming packets, ...)
- 9 SU_DEBUG_9() - entering/exiting functions, very verbatim progress

Starting with 1.0.4, those parameters can be controlled from the console by doing

```
freeswitch> sofia loglevel <all|default|tport|iptsec|nea|nta|nth_client|nth_server|nua|soa|sresolv|stun> [0-9]
```

"all" Will change every component's loglevel

A log level of 0 turns off debugging, to turn them all off, you can do

```
freeswitch> sofia loglevel all 0
```

To report a bug, you can turn on debugging with more verbose

```
sofia global siptrace on
sofia loglevel all 9
sofia tracelevel alert
console loglevel debug
fsctl debug_level 10
```

### Debugging presence and SLA

As of Jan 14, 2011, sofia supports a new debugging command: sofia global debug. It can turn on debugging for SLA, presence, or both. Usage is:

```
sofia global debug sla
sofia global debug presence
sofia global debug none
```

The first two enable debugging SLA and presence, respectively. The third one turns off SLA and/or presence debugging.

### Sample Export (Linux/Unix)

Alternatively, the levels can also be read from environment variables. The following bash commands turn on all debugging levels, and is equivalent to "sofia loglevel all 9"

```
export SOFIA_DEBUG=9
export NUA_DEBUG=9
export SOA_DEBUG=9
export NEA_DEBUG=9
export IPTSEC_DEBUG=9
export NTA_DEBUG=9
export TPORT_DEBUG=9
export TPORT_LOG=9
export TPORT_DUMP=/tmp/transport_sip.log
export SU_DEBUG=9
```

To turn this debugging off again, you have to exit FreeSWITCH and type unset. For example:

```
unset TPORT_LOG
```

### Sample Set (Windows)

The following bash commands turn on all debugging levels.
set SOFIA_DEBUG=9
set NUA_DEBUG=9
set SCO_DEBUG=9
set NEA_DEBUG=9
set IPTSEC_DEBUG=9
set NTA_DEBUG=9
set TPORT_DEBUG=9
set TPORT_LOG=9
set TPORT_DUMP=/tmp/tport_sip.log
set SU_DEBUG=9

To turn this debugging off again, you have to exit FreeSWITCH and type unset. For example:

set TPORT_LOG=

You can also control SIP Debug output within fs_cli, the FreeSWITCH client app.

freeswitch> sofia profile <profilename> siptrace on|off

On newer software release, you can now be able to issue siptrace for all profiles:

sofia global siptrace [on|off]

To have the SIP Debug details put in the /usr/local/freeswitch/log/freeswitch.log file, use

freeswitch> sofia tracelevel info (or any other loglevel name or number)

To have the SIP details put into the log file automatically on startup, add this to sofia.conf.xml:

<global_settings>
  ...
  <param name="tracelevel" value="DEBUG"/>
  ...
</global_settings>

and the following to the sip profile xml file:

<profiles>
  ...
  <profile name="...">
    ...
    <param name="sip-trace" value="yes"/>
    ...
  </profile>
  ...
</profiles>

Profile Configurations

Track Call

<param name="track-calls" value="true"/>

This will make FreeSWITCH track call state using the call database.

This can be stored in your ODBC database if you have that configured, allowing you to share call state between multiple FreeSWITCH instances.

For FreeSWITCH HA configuration, See Freeswitch HA

Sofia SIP Stack Watchdog

As of October 6, 2010, the Sofia profiles now support a watchdog on the Sofia SIP stack.

<param name="watchdog-enabled" value="no"/>
<param name="watchdog-step-timeout" value="30000"/>
<param name="watchdog-event-timeout" value="30000"/>

Sometimes, in extremely rare edge cases, the Sofia SIP stack may stop responding. These options allow you to enable and control a watchdog on the Sofia SIP stack so that if it stops responding for the specified number of milliseconds, it will cause FreeSWITCH to shut down immediately. This is useful if you run in an HA environment and need to ensure automated recovery from such a condition. Note that if your server is idle a lot, the watchdog may fire due to not receiving any SIP messages. Thus, if you expect your system to be idle, you should leave the watchdog disabled.

The watchdog can be toggled on and off through the FreeSWITCH CLI either on an individual profile basis or globally for all profiles. So, if you run in an HA environment with a master and slave, you should use the CLI to make sure the watchdog is only enabled on the master.
Generally, when running in an HA environment, you will have some sort of resource agent that manages which FreeSWITCH node is the master and which is the slave. You will also likely have demote and promote actions that are performed to switch states. When the demote action is called, it should execute the 'sofia global watchdog off' command. When the promote action is called, it should execute the 'sofia global watchdog on' command.

ACL

You can restrict access by IP address for either REGISTERs or INVITEs (or both) by using the following options in the sofia profile.

```xml
<param name="apply-inbound-acl" value="<acl_list|cidr>"/>
<param name="apply-register-acl" value="<acl_list|cidr>"/>
```

See ACL for other access controls

See acl.conf.xml for list configuration

Disabling Hold

Disable all calls on this profile from putting the call on hold:

```xml
<param name="disable-hold" value="true"/>
```

* See also: rtp_disable_hold variable

Using A Single Domain For All Registrations

You can force all registrations in a particular profile to use a single domain. In other words, you can ignore the domain in the SIP message. You will need to modify several sofia profile settings.

challenge realm

```xml
<param name="challenge-realm" value="auto_from"/>
```

Defaults to "auto_to" if not set; default config specifies "auto_from"

Possible Values:

- auto_from - uses the from field as the value for the SIP realm.
- auto_to - uses the to field as the value for the SIP realm.
- <anyvalue> - you can input any value to use for the SIP realm.

force-register-domain

```xml
<param name="force-register-domain" value="${domain}"/>
```

This will force the profile to ignore the domain in the to or from packet and force it to the value listed here for this parameter.

This will store the info into the database with the user@<from domain in SIP packet>

force-subscription-domain

```xml
<param name="force-subscription-domain" value="${domain}"/>
```

This will force the profile to ignore the domain in the to or from packet and force it to the value listed here for this parameter.

This will store the info into the database with the user@<from domain in SIP packet>

force-register-db-domain

```xml
<param name="force-register-db-domain" value="${domain}"/>
```

This will work in conjunction with force-register-domain so that the forced domain is stored in the database also.

Forcing SIP profile to use a static IP address

The default FreeSWITCH configuration will automatically determine the IP address of your local network interface. If you want to manually set the IP (for example, if you have multiple network interfaces on a single machine), you can set the following parameters in the Sofia profile:

```xml
<param name="rtp-ip" value="192.168.1.3"/>
<param name="sip-ip" value="192.168.1.3"/>
```
where 192.168.1.3 is the IP you want it to bind to.

and also in the sofia.conf.xml file:

<param name="auto-restart" value="false"/>

This will prevent the profile from automatically restarting when it detects a network address change.

**Syntax**

**Call an extension on a Remote SIP Server**

Basic form:

sofia/\$\{profile\}/\$\{extension\}@\$\{remoteserver\}

Example 1:

sofia/\$\{profile\}/\$1@example.org

Example 2:

sofia/foo/0@sipphone.com

Where <profile> is the name of one of the profiles defined in sofia.conf.xml. By default, there is one profile defined as name="\$\{domain\}"; where the \$\{domain\} variable is defined in freeswitch.xml, and defaults to mydomain.com.

Therefore, if you have not changed these config files and are dialing an extension on a remote server, the config would be

sofia/mydomain.com/\$\{extension\}@\$\{remoteserver\}

To list all profiles defined, type **sofia status** on the CLI.

**Call a locally registered endpoint**

Basic form:

sofia/\$\{profile\}/\$\{extension\}%\$\{localserver\}

Example 1:

sofia/\$\{profile\}/\$1%\$\{domain\}

Example 2:

sofia/foo/101%192.168.1.1

where foo is the SIP profile, 101 is the userid of the registered endpoint, and 192.168.1.1 is the IP address of FreeSWITCH.

If your SIP profile name is set to your domain, or the domain attribute is set in the profile (e.g., <profile name="internal" domain="\$\{domain\}"/>), you can originate with the following:

sofia/\$\{profile\}/\$\{extension\}

And fore-go the domain when dialing local extensions.

**Multiple Registrations**

Call one extension and ring several phones

You must enable multiple registrations in **conf/sip_profiles/internal-ipv6.xml** and **conf/sip_profiles/internal.xml** (enabling the setting in conf/autoload_configs/switch.conf.xml had no effect).

Valid values for this parameter are "contact", "true", "false", or "call-id".

<param name="multiple-registrations" value="contact"/>

Setting this value to "contact" will remove the old registration based on sip_user, sip_host and contact field as opposed to the call_id.

**Ringing All Registered Extensions**

In your dialplan (**conf/dialplan/default.xml**) you can now use the sofia_contact/lookup function to return all registered sip profiles; for example in the default.xml in sections <extension name="Local_Extension"> and <extension name="extension-intercom"> replace:
With the following:

```xml
<action application="bridge" data="${sofia_contact(${dialed_extension})}"/>
```

This will ring all devices currently registered to the dialed extension. Do note that `sofia_contact` also works at the CLI, so you can test before you add it to your dialplan.

See Function sofia contact for more information.

**Note (this applies to FreeSWITCH 1.0.1 and later):** you can disable multiple registrations on a per-user basis by setting the variable "sip-allow-multiple-registrations" to "false" in the directory. In this case, that single user won’t be allowed to use multiple registrations.

### Registration Count

This api is used to return the count of registration of a user of a domain, or just of a domain.

```
freewatch@internal> sofia_count_reg [user]@domain
```

### sofia_dig

You can use sofia_dig to check DNS A and SRV records for a given domain from the fs_cli. If there isn’t a SRV record freeswitch uses A records. The use of SRV allows graceful fail overs.

**Usage:**

```
sofia_dig voicenetwork.ca
```

<table>
<thead>
<tr>
<th>Preference</th>
<th>Weight</th>
<th>Transport</th>
<th>Port</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.500</td>
<td>udp</td>
<td>5060</td>
<td>74.51.38.15</td>
</tr>
<tr>
<td>1</td>
<td>0.500</td>
<td>tcp</td>
<td>5060</td>
<td>74.51.38.15</td>
</tr>
</tbody>
</table>

### Flushing Inbound Registrations

From time to time, you may need to kill a registration.

You can kill a registration from the CLI, or anywhere that accepts API commands with a command similar to the following:

```
sofia profile <profile_name_here> flush_inbound_reg [optional_callid]
```

### Dial out of a gateway

**Basic form:**

```
sofia/gateway/<gateway>/<number_to_dial>
```

**Example 1:**
Your available gateways (usually configured in conf/sip_profiles/external/*.xml) will show up in sofia status:

```
  sofia status
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Data</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>profile</td>
<td>sip:mod_sofia@2.3.4.5:5060</td>
<td>RUNNING (2)</td>
</tr>
<tr>
<td>mygateway</td>
<td>gateway</td>
<td>sip:username@1.2.3.4</td>
<td>NOREG</td>
</tr>
<tr>
<td>phonebooth.example.com</td>
<td>alias</td>
<td>default</td>
<td>ALIASED</td>
</tr>
</tbody>
</table>

1 profile 1 alias

Modifying the To: header

You can override the To: header by appending `^<toheader>`.

Example 1:

```
sofia/foc/user%192.168.1.1^101@$${domain}
```

Specifying SIP Proxy With fs_path

You can route a call through a specific SIP proxy by using the "fs_path" directive. Example:

```
sofia/foc/user@that.domain;fs_path=sip:proxy.this.domain
```

Safe SIP URI Formatting

As of commit [https://freeswitch.org/stash/projects/FS/repos/freeswitch/commits/76370f4d1767bb0dcf828a3d6cde6e015b2cfa03](https://freeswitch.org/stash/projects/FS/repos/freeswitch/commits/76370f4d1767bb0dcf828a3d6cde6e015b2cfa03) the User part of the SIP URI has been "safely" encoded in the case where spaces or other special characters appear.

Channel Variables

Adding Request Headers

You can add arbitrary headers to outbound SIP calls by prefixing the string `sip_h_` to any channel variable, for example:

```
<action application="set" data="sip_h_X-Answer=42"/>
<action application="bridge" data="sofia/mydomain.com/1000@example.com"/>
```

Note that for BYE requests, you will need to use the prefix `sip_bye_h_` on the channel variable.
Adding Response Headers

There are three types of response header prefixes that can be set:

- Response header
  sip_rh_
- Provisional response header
  sip_ph_
- Bye response header
  sip_bye_h_

Each prefix will exclusively add headers for their given types of requests - there is no "global" response header prefix that will add a header to all response messages.

For example:

```
(action application="set" data="sip_rh_X-Reason=Destination Number Not in Footprint")
(action application="set" data="sip_bye_h_X-Accounting=Some Accounting Data")
```

Adding Custom Headers

For instance, you may need P-Charge-Info to append to your INVITE header, you may do as follows:

```
(action application="set"><!CDATA[sip_h_P-Charge-Info=<sip:${caller_id_number}@${domain_name}>;npi=0;noa=3]]></action>
```

Then, you would see it in SIP message:

```
INVITE sip:1909909099@1.2.3.4 SIP/2.0
Via: SIP/2.0/UDP 5.6.7.8:5080;rport;branch=z9hG4bKyg61X9v3gUD4g
Max-Forwards: 69
From: "DJB" <sip:2132132132@5.6.7.8>;tag=XQKQ322vQF5gK
To: <sip:1909909099@1.2.3.4>
Call-ID: b6c776f6-47ed-1230-0085-000f1f659e58
CSeq: 30776798 INVITE
Contact: <sip:mod_sofia@5.6.7.8:5080>
User-Agent: FreeSWITCH-mod_sofia/1.2.0-rc2+git-20120713T162602Z~0afd7318bd+unclean~20120713T184029Z
Allow: INVITE, ACK, BYE, CANCEL, OPTIONS, MESSAGE, UPDATE, INFO, REGISTER, REFER, NOTIFY
Supported: timer, precondition, path, replaces
Allow-Events: talk, hold, conference, refer
Content-Type: application/sdp
Content-Disposition: session
Content-Length: 229
P-Charge-Info: <sip:2132132132@5.6.7.8>;npi=0;noa=3
X-FS-Support: update_display,send_info.
Remote-Party-ID: "DJB" <sip:2132132132@5.6.7.8>;party=calling;screen=yes;privacy=off
```

Strip Individual SIP Headers

Sometimes a SIP provider will add extra header information. Most of the time they do that for their own use (tracking calls). But that extra information can cause a lot of problems. For example: I get a call from the PSTN via a DIDs provider (provider1). Since im not in the office the call gets bridged to my cell phone (provider2). Provider1 add's extra information to the sip packet like displayed below:
In some scenario, we bridge this call directly to provider2 the calls get dropped since provider2 doesn’t accept the X-voipnow header, so we have to strip off those SIP headers.

To strip them off, use the application UNSET in the dialplan (the inverse of SET):

```xml
<action application="unset" data="sip_h_X-voipnow-did"/>
<action application="unset" data="sip_h_X-voipnow-extension"/>
```

### Strip All custom SIP Headers

If you wish to strip all custom headers while keeping only those defined in dialplan:

```xml
<action application="set" data="sip_copy_custom_headers=false"/>
<action application="set" data="sip_h_X-myCustomHeader=${sip_h_X-myCustomHeader}"/>
```

### Additional Channel variables

Additional variables may also be set to influence the way calls are handled by sofia.

For example, contacts can be filtered by setting the `sip_exclude_contact` variable. Example:

```xml
<anti-action application="set" data="sip_exclude_contact=${network_addr}"/>
```

Or you can perform SIP Digest authorization on outgoing calls by setting `sip_auth_username` and `sip_auth_password` variables to avoid using Gateways to authenticate. Example:

```xml
<action application="bridge" data="[sip_auth_username=secretusername,sip_auth_password=secretpassword]sofia/external/00123456789@sip.example.com"/>
```

Changing the SIP Contact user FreeSWITCH normally uses mod_sofia@ip:port for the internal SIP contact. To change this to foo@ip:port, there is a variable, sip_contact_user:

```xml
{sip_contact_user=foo}sofia/my_profile/1234@192.168.0.1;transport=tcp
```

### sip_renegotiate_codec_on_reinvite

true/false

### sip_recovery_break_rfc

true/false

### Transcoding Issues

G729 and G723 will not let you transcode because of licensing issues. Calls will fail if for example originating endpoint has set G729 with higher priority and receiving endpoint has G723 with highest priority. The logic is to fail the call rather than attempt to find a codec match. If you are having issues due to transcoding you may disable transcoding and both endpoints will negotiate the compatible codec rather than just fail the call.

disable-transcoding will take the preferred codec from the inbound leg of your call and only offer that codec on the outbound leg.

Add the following command `<param name="disable-transcoding" value="true"/>` along with `<param name="inbound-late-negotiation" value="false"/>` to your sofia profile...
Example:

```xml
<configuration name="sofia.conf" description="sofia Endpoint">
  <profiles>
    <profile name="sip">
      <settings>
        <param name="disable-transcoding" value="true"/>
        <param name="inbound-late-negotiation" value="false"/>
      </settings>
    </profile>
  </profiles>
</configuration>
```

Custom Events

The following are events that can be subscribed to via Event Socket

- **Registration**
  * sofia::register
  * sofia::pre_register
  * sofia::register_attempt
  * sofia::register_failure
  * sofia::unregister - explicit unregister calls
  * sofia::expire - when a user registration expires

- **Gateways**
  * sofia::gateway_add
  * sofia::gateway_delete
  * sofia::gateway_state - when a gateway is detected as down or back up

- **Call recovery**
  * sofia::recovery_send
  * sofia::recovery_recv
  * sofia::recovery_recovered

- **Other**
  * sofia::notify_refer
  * sofia::reinvite
  * sofia::error

FAQ

**Does it use UDP or TCP?**

By default it uses both, but you can add ;transport=tcp to the Sofia URL to force it to use TCP.

For example:

```
sofia/profile/foo@bar.com;transport=tcp
```

Also there is a parameter in the gateway config:

```xml
<param name="register-transport" value="tcp"/>
```

That will cause it to use the TCP transport for the registration and all subsequent SIP messages.

Not sure if this is needed or what it does, but the following can also be used in gateway settings:

```xml
<!--extra sip params to send in the contact-->
<param name="contact-params" value="tport=tcp"/>
```

See Also
- mod_sofia
- Debugging Freeswitch
- SIP Protocol Messages
- Authentication params for gateways.
- RTP Issues
- Multi-Homed Example How To

External links