

Table of Contents

1. Introduction.....	2
1.1 About.....	2
1.2 Copyright.....	2
1.3 Download.....	4
1.4 Mailing List and Forums.....	4
1.5 Sponsors.....	5
2. Compiling and Installing.....	6
2.1 Compiling OpenSBC.....	6
2.2 Installing OpenSBC.....	9
3. Quick Start.....	12
3.1 Basic Configuration.....	12
3.2 Adding User Accounts.....	13
3.3 Adding Basic Routes.....	14
3.4 Sending Traffic.....	15
4. HTTP Admin.....	19
4.1 OpenSBC HTTP Admin.....	19
4.2 OpenSBC General Parameters.....	20
4.2.1 [Logging].....	20
4.2.2 [Modes].....	21
4.2.3 [Listeners].....	23
4.2.4 [Media].....	23
4.2.5 [Call Transfer].....	24
4.2.6 [Encryption].....	25
4.2.7 [Event Processors].....	26
4.2.8 [Timers].....	26

1. Introduction

1.1 About

OpenSBC is an ongoing attempt to create an open-source Session Border Controller that is fully compliant with the mandates of RFC 3261. OpenSBC can be used as a SIP router, media anchor for far-end NAT traversal, SIP egress and ingress trunking among others. OpenSBC functionalities are as follows

Current Functionalities:

- SIP Registrar and Proxy
- Back-to-back UA Router
- Traffic Authentication
- Static Routing Rules
- ENUM Routing
- SIP Privacy Enforcement
- Upstream Registrar Support
- Media Server

Future Functionalities

- Voice XML
- Third Party Call Control (3PCC)
- Dialog Events
- Signaling Compression (SIGCOMP)

1.2 Copyright

OpenSBC is published under the MPL, GPL, LGPL triple license. The default active license is MPL. However, because of the inherent incompatibility between MPL and GNU licenses, we decided to explicitly publish OpenSBC using GPL and LGP as alternative licenses. The full text of the license is as follows

```
* Copyright (c) opensipstack.org.  
*  
* The contents of this file are subject to the Mozilla Public  
License  
* Version 1.0 (the "License"); you may not use this file except  
in  
* compliance with the License. You may obtain a copy of the  
License at  
* http://www.mozilla.org/MPL/  
*  
* Software distributed under the License is distributed on an "AS  
IS"  
* basis, WITHOUT WARRANTY OF ANY KIND, either expressed or  
implied. See  
* the License for the specific language governing rights and  
limitations  
* under the License.  
*  
* The Original Code is OpenSIPStack.  
*  
* The Initial Developer of the Original Code is opensipstack.org.  
*  
* The author of this code is Joegen E. Baclor  
*  
* Alternatively, the contents of this file may be used under the  
terms of  
* either the GNU General Public License Version 2 or later (the  
"GPL"), or  
* the GNU Lesser General Public License Version 2.1 or later (the  
"LGPL"),  
* in which case the provisions of the GPL or the LGPL are  
applicable instead  
* of those above. If you wish to allow use of your version of  
this file only  
* under the terms of either the GPL or the LGPL, and not to allow
```

others to

- * use your version of this file under the terms of the MPL, indicate your

- * decision by deleting the provisions above and replace them with the notice

- * and other provisions required by the GPL or the LGPL. If you do not delete

- * the provisions above, a recipient may use your version of this file under

- * the terms of any one of the MPL, the GPL or the LGPL.

- *

- * The OpenSIPStack Library includes some GPL/LGPL code libraries that MAY be

- * enabled at compile time using the --enable-gpllibs configure switch. If

- * enabled, the content of this file is published under the terms of GNU

- * General Public License Version 2. For a detailed list of the available

- * GPL code, see \$(opensipstack)/gnu/README file.

1.3 Download

The latest development releases and binaries can be downloaded at the following locations

- * <http://www.opensipstack.org>

- * <http://www.opensourcesip.org>

- * [Source Forge](http://sourceforge.net/projects/opensipstack/) (<http://sourceforge.net/projects/opensipstack/>)

1.4 Mailing List and Forums

A [Source Forge Mailing List](https://lists.sourceforge.net/lists/listinfo/opensipstack-devel) (<https://lists.sourceforge.net/lists/listinfo/opensipstack-devel>) is dedicated to discussing OpenSIPStack Development including OpenSBC

Forums are hosted in [Open Source SIP Web site](http://www.opensourcesip.org/) (<http://www.opensourcesip.org/>).

1.5 Sponsors

The chief architect and coordinator of the OpenSBC project is [Joegen E. Baclor](mailto:joegen@opensipstack.org). (joegen@opensipstack.org)

[Solegy Systems](http://www.solegy.com/) (http://www.solegy.com/) sponsored the main functionalities of OpenSBC.

[Ryan Colobong](mailto:rcolobong@solegysystems.com) (rcolobong@solegysystems.com) is a Solegy Architect dedicated to work on OpenSBC.

2. Compiling and Installing

2.1 Compiling OpenSBC

Compiling in Linux/Unix

Most Unix/Linux distribution already goes with all the dependency requirement for OpenSBC and OpenSIPStack to build properly out of the box. Technically, its just a matter of running the 'configure' script before running a 'make bothnoshared'.

Compile `opensipstack` first before compiling `opensbc` using the steps below.

```
unix-shell>#./configure
```

There might be instances when file run permissions are not properly set for the configure script either due to previous commits from a Windows source or your CVS client was not able to properly preserve the file attributes. If you are unable to run the configure script properly, you may do one of two things. Do a 'chmod +x ./configure' or you may regenerate the file by issuing an `autoconf` within the `opensipstack` directory. The latter should regenerate a fresh copy of the configure script from the `configure.ac` `autoconf` template.

If all goes well with the configuration boot strap all that required to be done is:

```
unix-shell>#make bothnoshared
```

or for Solaris and FreeBSD users:

```
unix-shell>#gmake bothnoshared
```

CAVEAT: Most operating systems already have `expat` installed. If the configure script complains about not finding the `expat` library, please install `expat 2.0` (or higher) development package. OpenSIPStack goes with a built-in `expat` source but this is intended for use with Windows builds.

CAVEAT: For OpenSolaris users, you might need to define the `CC` variable to use `gcc` instead of default 'cc' compiler that ships with Solaris.

```
solaris-shell>#gmake bothnoshared CC=gcc
```

OpenSBC Manual

CAVEAT: FreeBSD users may need to install `Linux_base-fc-4_9`.

Compiling In Windows

OpenSIPStack and OpenSBC goes with project files for Visual C++ 7.10 and Visual C++ 8.0. If you don't have access to any of these compilers, Visual C++ Express Edition is available for free download at <http://msdn.microsoft.com/vstudio/express/visualc/download/>. When using Visual Studio 8.0, make sure you open the solution name `opensipstack.sln` and not `opensipstack-7.10.sln` to avoid Visual Studio 8.0 from converting the solution to the recognized format. Compile the Release build first followed by the Debug build. The sequence is important. There are known issues with the Debug build regarding Custom Build Steps.

CAVEAT: Take note that as of version 1.1.4 Visual Studio 7.10 projects files are no longer supported. If you are using these project files, you may need to manually modify the project files to include missing `.cxx` files.

1. Get the latest source code of OpenSBC and OpenSIPStack from CVS.
2. Place the OpenSBC and OpenSIPStack in the same folder.
3. Open the solution file `OpenSBC/OpenSBC.sln`.
4. Make sure the "Solution Explorer" Windows is displayed.
5. Select the build type as "Release" or "Debug" in the build toolbar.
6. Right-click on "OpenSBC" and select "Build" from the menu to build both OpenSIPStack and OpenSBC. Just like OpenSIPStack, OpenSBC has a configure Custom Build Step.

When the compile process has started there will be five Custom Build Steps that will be performed.

```
— Build started: Project: OpenSIPStack, Configuration: Release Win32 —
```

```
Copying Flex.exe ...
Copying Bison.simple ...
Copying Bison.hairy ...
Copying Bison ...
Copying Expat.dll ...
Copying Sqlite3.dll ...
```

These Custom Build Steps will copy the Sqlite and Expat Libraries to `$(SystemRoot)/system32/`. Flex and Bison files would be copied in `C:/Tools`.

OpenSBC Manual

Important Warning!!!: If you already have `sqlite3.dll` and `expat.dll` in `system32` folder and some other applications rely on these two files, it is best that you EXCLUDE these files from the Custom Build Process. All custom build files are located in "Make/Custom Build" folder in the project explorer tab of Visual C++ 8.0. Version conflicts may arise if your copy is of different from the Expat and SQLite3 DLLs that goes with `opensipstack`. You need to either upgrade your local copy or try changing the header files in the `opensipstack/external` folder to your specific version and hope for the best.

After the Custom Build Steps have been performed, the configuration bootstrap will start searching for existing SDK and libraries installed in your system. My compiler displays this information after the configure script has finished searching.

```
Searching C:\
Located IPv6 Support at C:\Program Files\Microsoft Visual Studio
8\SmartDevices\SDK\PocketPC2003\Include\
Located DNS Resolver at C:\Program Files\Microsoft Visual Studio
8\VC\PlatformSDK\
Located QoS Support at C:\Program Files\Microsoft Visual Studio
8\VC\PlatformSDK\Include\
Searching X:\
Located OPAL at X:\dev\opensipstack\
Located MYACMLIB at X:\dev\opensipstack\external\codecs\
Located VOICEAGE at X:\dev\opensipstack\external\codecs\
Located SQLITE at X:\dev\opensipstack\external\CppSQLite\3.1\Common\
Located CPPSQLITE at X:\dev\opensipstack\external\CppSQLite\3.1\Common\
Located Expat XML at X:\dev\opensipstack\external\Expat-2.0.0\
Located LibJingle at X:\dev\opensipstack\external\jingle\talk\
```

If all goes well, with the configuration, OpenSIPStack should start compiling, hopefully, without any problem.

Most recently, a new build configuration for Windows 64-bit has been added to the Visual C++ 8.0 project. Aside from some benign warning about data truncation,

```
1>.\src\pplib\src\ptlib\common\regex\regcomp.c(122) :
warning C4267: '=' : conversion from 'size_t' to 'sopno', possible loss of
data
```

there shouldn't be any trouble compiling OpenSBC in a 64-bit setting.

OpenSBC Manual

CAVEAT: `Windows.h missing` is a compile error commonly reported in the mailing list. This may happen in Visual Studio 2005 Express edition of Visual C++. `Windows.h` goes with the Platform SDK and is available via a separate installation here <http://msdn.microsoft.com/vstudio/express/visualc/usingpsdk/>.

CAVEAT: Windows Vista might not allow the configure script to copy `SQLite.dll` and `Expat.dll` to the system folder. To circumvent this, right click on the Visual Studio 2005 icon in the menu and click "Run as Administrator". This would give Visual studio temporary write access to the system folders.

2.2 Installing OpenSBC

Running OpenSBC in Unix

There is no special install process for OpenSBC. Just copy the binary to a folder of your choice and run OpenSBC. OpenSBC requires '`oss-application.conf.xml`' to be in the same directory as the `opensbc` binary. If this file is not present, `opensbc` will not start.

If you are going to use OpenSBC as a far end media anchor (RTP Proxy), you may need to increase the number of file handles per process using "`ulimit -n`". The default is normally 1024 which will not accommodate much connections. There are operating systems that does not allow limits to be modified via `ulimit`. In Ubuntu, for example, limits can be defined in `/etc/security/limits.conf`.

OpenSBC accepts the following command line switches.

- `-v` –version display version information and exit
- `-d` –daemon run as a daemon
- `-u` –uid uid set user id to run as
- `-g` –gid gid set group id to run as
- `-p` –pid-file name or directory for PID file
- `-t` –terminate orderly terminate process in pid file
- `-k` –kill preemptively kill process in pid file
- `-s` –status check to see if daemon is running
- `-c` –console output messages to stdout rather than syslog
- `-l` –log-file file output messages to file or directory instead of syslog [NO LONGER SUPPORTED]
- `-x` –execute execute as a normal program
- `-i` –ini-file set the ini file to use, may be explicit file or a ':' separated set of directories to search.
- `-H` –handlemax n set maximum number of file handles (set before uid/gid) [MIGHT NOT ALWAYS WORK]

OpenSBC Manual

- `-P` `–http-port n` set the http listener port for the application admin page
- `-a` `–app-name name` set the identifier name that would be display in the http home page

Running OpenSBC in console

```
#./opensbc -xc
```

Running OpenSBC as a daemon

```
#./opensbc -d -p pidfile.txt
```

Running OpenSBC using a specific config file

```
#./opensbc -d -p pidfile.txt -i opensbc.ini
```

Running OpenSBC using a custom HTTP Admin port

```
#./opensbc -d -p pidfile.txt -i opensbc.ini -P 4040
```

Stopping OpenSBC Process

```
#./opensbc -k -p pidfile.txt
```

Checking if OpenSBC is running

```
#./opensbc -s -p pidfile.txt
```

Running OpenSBC in Windows

The best way to run OpenSBC in window is to install it as a Windows service.

OpenSBC accepts the following command line arguments in Windows:

- `Install` – This argument installs OpenSBC as a Windows Service.
- `Remove` – Uninstalls the OpenSBC Service but retains the configuration in the windows Registry. This is usually used when upgrading OpenSBC versions
- `Deinstall` – Uninstalls and erase Registry entries. All previous configuration data will be lost.
- `Start` – Starts the OpenSBC Service
- `Stop` – Stops the OpenSBC Service
- `Debug` – OpenSBC can also run as a “Foreground Process” by using “debug” as the sole

argument.

3. Quick Start

3.1 Basic Configuration

After a successful installation of OpenSBC, the HTTP admin should already be accessible via port 9999 of the server where OpenSBC is running. Figure 1 shows the main HTTP admin page. Pictures speak louder than words!

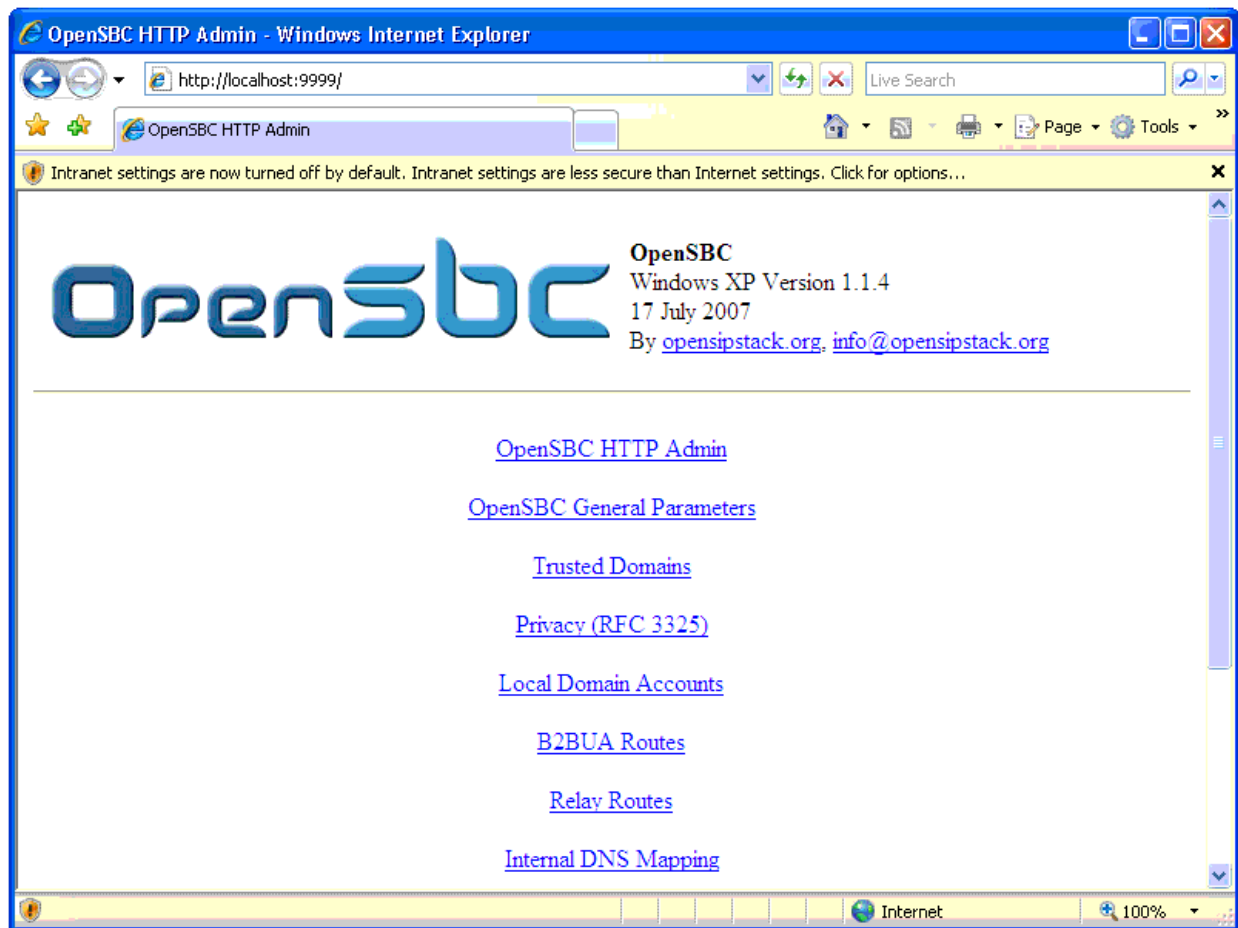


Figure 1. The OpenSBC Admin page.

For a basic configuration, click on the "**OpenSBC General Parameters**" link and look for the "SBC Mode" Parameter.

Using the drop-down list box, choose "B2B Only Mode" as shown in Figure 2.

SIP Log Level 1=Fatal only, 2=Errors, 3=V

PTRACE Log Level 1=Fatal only, 2=Errors, 3=V

SBC Mode

 Full Mode
 Proxy Only Mode
 B2B Only Mode
 B2BUpperReg Mode

Always Proxy Media

Figure 2. SBC Mode.

We need to change the listener port for OpenSBC to allow us to run a softphone client in the same box. To do this, we must add a new entry in the "Interface Address" list. We want to use all available interface bound to port 9000. Figure 3 shows how this is done within the "General Parameters Section":

Interface Address
 Ignore
 Ignore
 Add

Figure 3. Interface address.

Make sure you choose "Keep" in the drop down list-box before clicking the **[Accept]** button.

After this, you **MUST RESTART** OpenSBC so that the changes would take effect. Not all parameters in OpenSBC requires a restart. But these two does!

3.2 Adding User Accounts

The next step would be to add user accounts so that our softphone would be allowed to register and make calls through OpenSBC. User accounts are added using the "**Local Domain Accounts**" link. The format for account entry is using the standard SIP URI format "sip:user:password@domain:port". Let us say our hypothetical account is **sip:john:doe@opensbcrocks.org:9000**. This means the authentication user is "john" and his password is "doe" belonging to the domain opensbcrocks.org:9000. The port is important in this case because OpenSBC is listening at port 9000.

Figure 4 shows how this account is added.

Local Domain Accounts

[Reload page](#) [Home page](#)

Accept All
Registration

Account Ignore Ignore Add

Figure 4. Adding a User account.

Do not forget to choose "Add" from the drop-down list box before clicking the **[Accept]** button.

opensbcrocks.org is not a real domain with either host or SRV records. OpenSBC will not be able to resolve this domain properly. Luckily, there is a way around this problem. You may statically map domains using the "Internal DNS Mapping" section. In our case we want to map `opensbcrocks.org:9999` to the OpenSBC IP address. Figure 5 shows how to add an internal DNS map entry.

Internal DNS Mapping

[Reload page](#) [Home page](#)

Map Ignore Ignore Add

Figure 5. Adding a DNS map entry.

3.3 Adding Basic Routes

After the account for John Doe has been added, the next step would be to add routes so that John Doe can start making calls. Routes are added via "B2BUA Routes" link in the admin page. Figure 6 shows that we want the number 613 be routed to `fwd.pulver.com`. `sip:613@fwd.pulver.com` is a public SIP echo server.

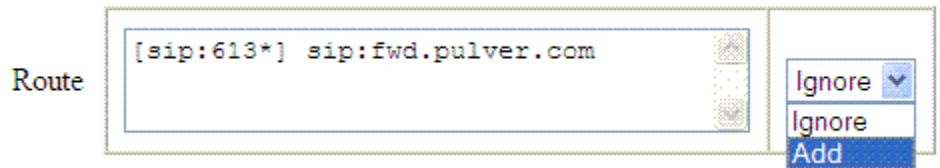


Figure 6. Routing numbers.

OpenSBC is all set! You can now start sending traffic and enjoy the sound of your own voice courtesy of the Free World Dialup echo server.

3.4 Sending Traffic

We need to grab a softphone to send test calls to OpenSBC. For the sake of making life easier, I will be using SJPhone as the test client in this tutorial or else I would be giving you another tutorial on how to compile the equally elegant OSSPhone.

First thing, we need to create a new profile in SJPhone and set the proxy and domain to point to OpenSBC. Figure 7 shows a new SJPhone profile pointing to OpenSBC. *192.168.179.129* is the IP address of OpenSBC while the user domain is *opensbcrocks.org*. Put a check on **Register with Proxy** and **Proxy is Strict Outbound**.

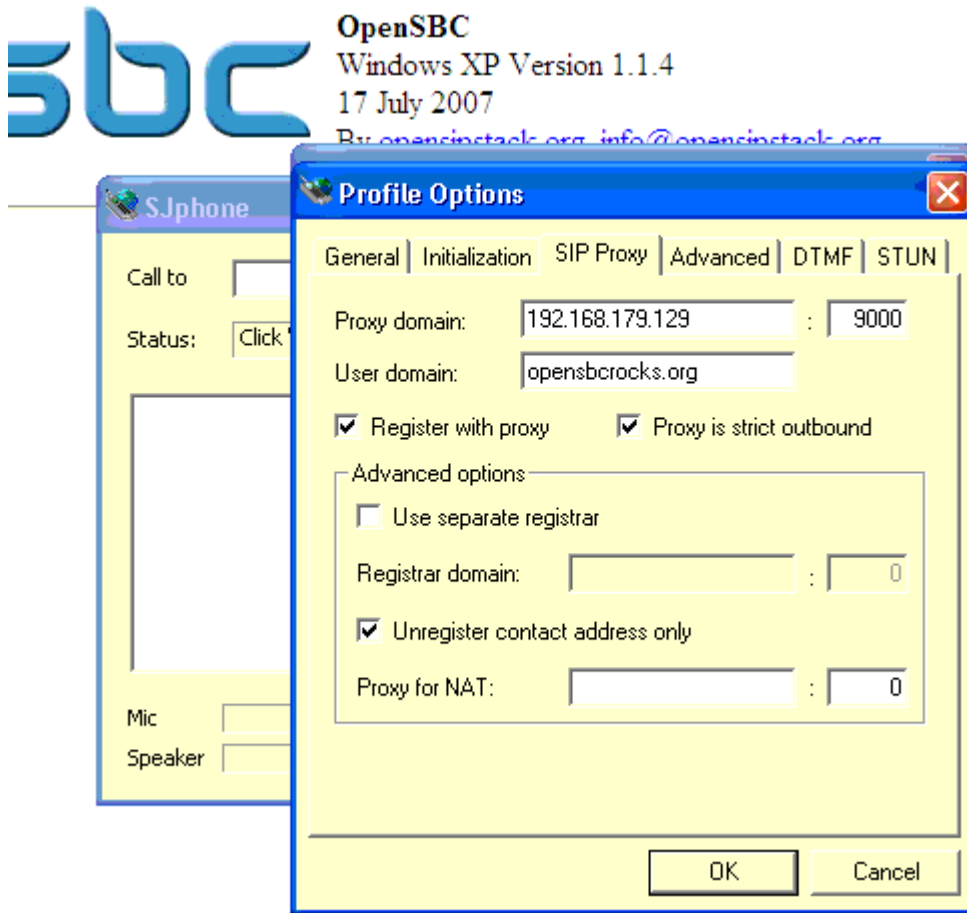


Figure 7. SJphone Profile Options.

OpenSBC Manual

The next step is to provide the user name and password of our hypothetical user John.

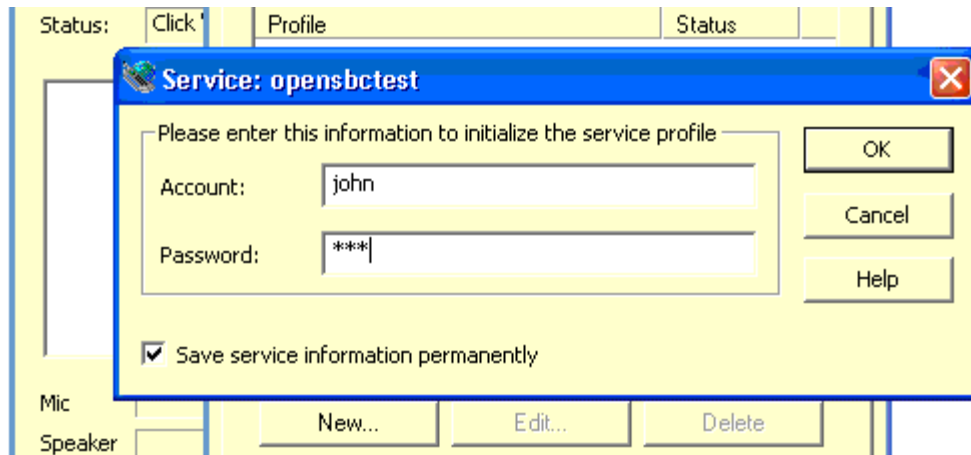


Figure 8. Adding an account to S.Jphone.

Our softphone is all set! Upon closing the new profile dialog, S.Jphone should have already registered with OpenSBC. Figure 9 shows S.Jphone is already registered with OpenSBC as sip:john@opensbcrocks.org:

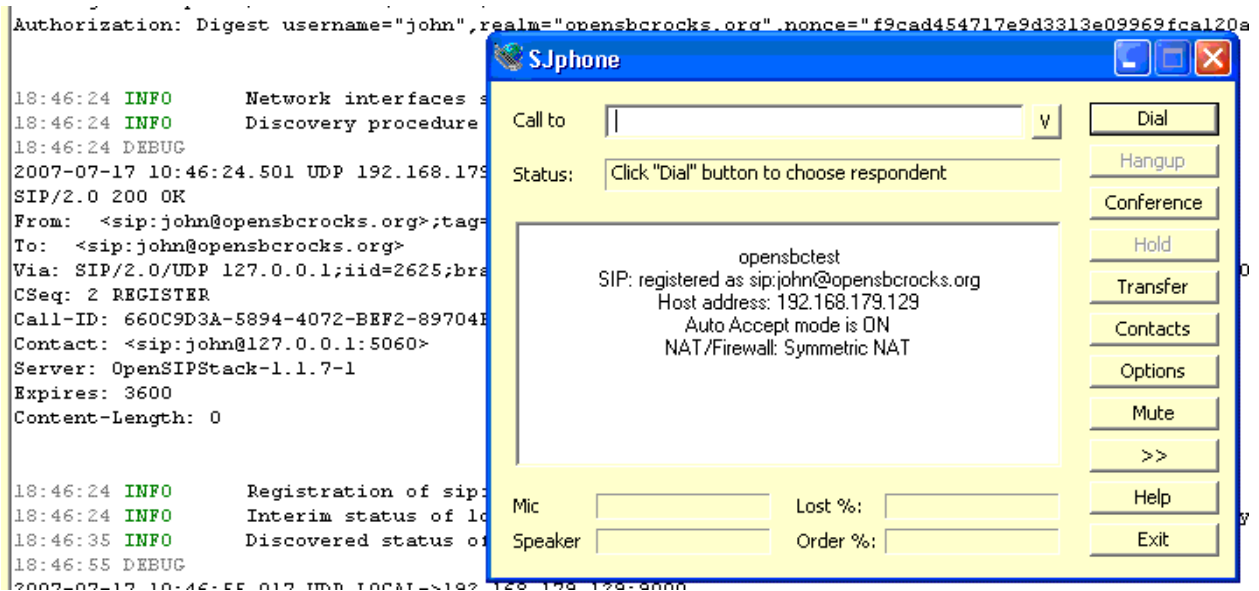


Figure 9. S.Jphone now registered with OpenSBC.

You can view the registration details using the "Registration Status" link in the OpenSBC HTTP admin. Figure 10 shows what the registration detail looks like.

OpenSBC Manual

URI	AOR	Expires	Call-ID	
john@opensbcrocks.org	sip:john@127.0.0.1:5060	3201	660C9D3A-5894-4072-BEF2-89704EEDAE07@192.168.179.129	<input type="button" value="Unregister"/>

Figure 10. Registration details.

On successful registration, we can now call our FWD echo server route by dialing 613.

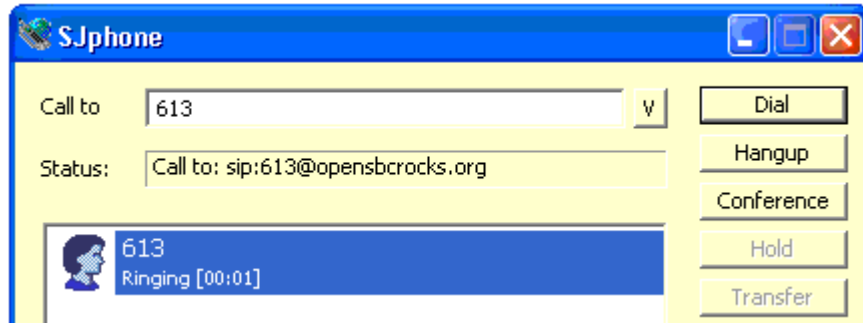


Figure 11. Calling a number.

OpenSBC can also route calls using just plain DNS resolution without having to enter static routes for the destination. The following call shows a call to `proxy01.sipphone.com` which doesn't have a corresponding route. Since this domain is an actual FQDN, OpenSBC should be able to make this call without routes being statically defined.

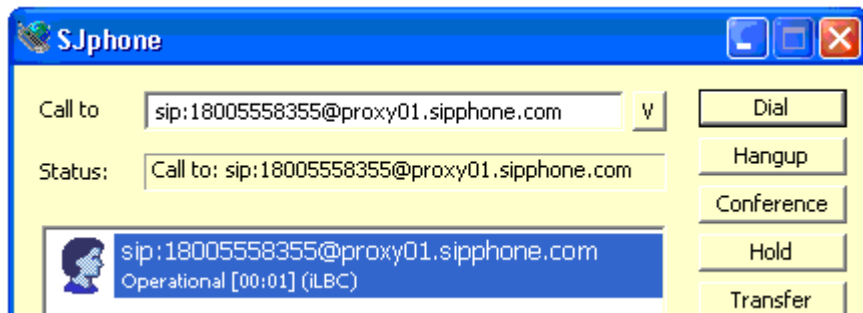


Figure 12. Calling a fully-qualified domain name via DNS resolution.

There you go! This concludes the quick start.

4. HTTP Admin

The HTTP Admin pages houses all configurable parameters to control the behavior of OpenSBC. OpenSBC uses an XML template (oss-application.conf.xml) file for the HTTP admin. This file should be present in the same directory of OpenSBC. The idea behind the XML template is to automate the addition of new parameters in the config pages without the need to code anything.

The HTTP Admin automatically generates a corresponding INI file in Unix (`$(HOME)/.pwwlib_config/opensbc.ini`) and a registry section (`HKEY_LOCAL_MACHINE/Software/opensipstack.org/OpenSBC`) in Windows based on the items listed in `oss-application.conf.xml` template.

4.1 OpenSBC HTTP Admin

OpenSBC HTTP Admin

[Reload page](#) [Home page](#)

HTTP User

HTTP Password

Figure 13. The OpenSBC Administrative Interface login page.

The HTTP Admin link allows for a user name and password to secure the HTTP Admin pages from unauthorized access. We strongly advice that users set the login information to avoid unauthorized access to OpenSBC configuration. In cases of forgotten passwords, you may reset it by deleting the corresponding entries in (`$(HOME)/.pwwlib_config/opensbc.ini` for Unix installations and in `HKEY_LOCAL_MACHINE/Software/opensipstack.org/OpenSBC` registry section for Windows installations. If a password is set for the HTTP Admin, you browser will prompt for the login information every time you attempt to access the admin pages. Figure 14 shows a login dialog in Internet Explorer.

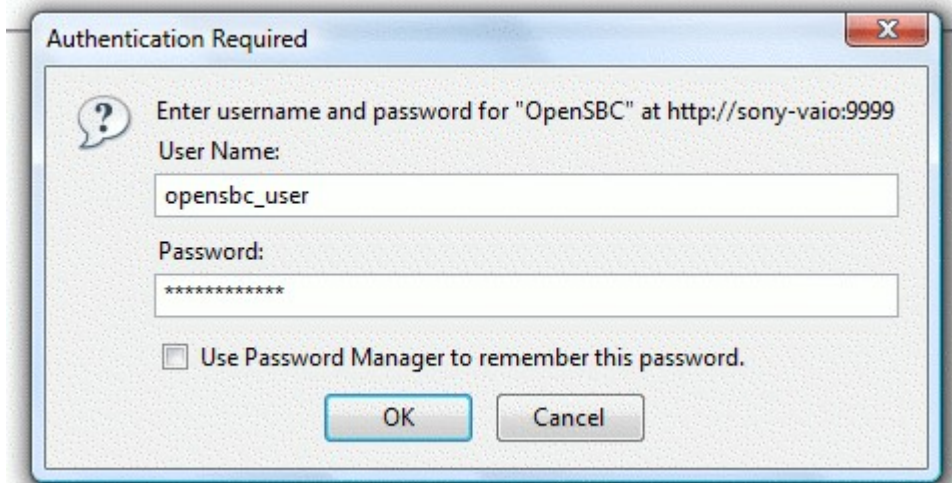


Figure 14. Login dialog for Internet Explorer.

4.2 OpenSBC General Parameters

The General parameters section houses global parameters for OpenSBC. ranging from logging, listeners, modes and signaling behavior.

4.2.1 [Logging]

All OpenSBC logs are stored in `$(OPENSBCDIR)/logs` directory. Logs are created each time OpenSBC is launched and is rotated on a daily basis. The log filename format is `$(prefix)-YYYY-MM-DD-$(process-id).log`; for example, `b2bua-2007-07-17-3464.log`. There are two types of log entries in OpenSBC - SIP Logs and PTRACE Logs. PTRACE is used for debugging purposes while SIP log displays the library and application level activities. Configurable parameters related to logging are listed below:

[SIP Log Level] - Level of verbosity for SIP Logs. Valid Range is 0-5

SIP Log Level 1=Fatal only, 2=Errors, 3=Warnings, 4=Info, 5=Debug

Figure 15. SIP log level.

[PTRACE Log Level] - Level of verbosity for PTRACE Logs. Valid Range is 0-5

PTRACE Log Level 1=Fatal only, 2=Errors, 3=Warnings, 4=Info, 5=Debug

Figure 16. PTRACE log level.

[Log File Prefix] - The prefix is configurable. This is handy during times when its necessary to run multiple instances of OpenSBC and you want to set which log file belongs to which OpenSBC instance.

Log File Prefix

Figure 17. Log file prefix.

4.2.2 [Modes]

The architecture of OpenSBC allows it to act both as a B2BUA, Registrar, and as a pure SIP Proxy. You may control which functionality you like by choosing the mode for OpenSBC. Each mode is explained further below.

[SBC Mode]

SBC Mode
 Full Mode
 Proxy Only Mode
 B2B Only Mode
 B2BUpperReg Mode

Figure 18. SBC Mode.

This is a selection where you can set the run mode of OpenSBC. This parameter would require a restart for the change to take effect.

1. Full Mode - By default OpenSBC runs in full mode exposing its capability both as a relay SIP proxy, Registrar and as a B2B User Agent. When OpenSBC receives an INVITE or a REGISTER request it would follow the following procedure to make a decision how to route a request:

- If the Request-URI resolves to a remote domain, the request will be relayed. If a relay route is available, the request is sent to that route. If a relay route is not available, then the URI is resolved via DNS.
- If the Startline-URI resolves as a local address and port, the **To** URI is checked if it resolves to a local domain and port. If not, the request would be proxied using Relay Routes or via DNS resolution. The Request URI would be rewritten to point to the resolved route.

OpenSBC Manual

- **INVITE:** If both Request URI and To URI resolves to a local listener and port, the B2BUA Route is used to route the INVITE.
- **REGISTER:** If both Request URI and To URI resolves to a local listener and port, the local Registrar will process the registration. This would include Authorization of the user.

2. B2BOnly Mode - This mode removes the relay capability but exposes the Registrar and the B2BUA functionalities. This mode does not do the checks performed by Full Mode. It will always process REGISTER and INVITE as local.

- **INVITE:** This mode always use B2BUA Route to route calls. If there is not corresponding route found, a DNS resolutions is done against the Request URI or the To URI in case the Request-URI resolves to a local address.
- **REGISTER:** Registrations are always handled by the local registrar.

3. Proxy Only Mode - This mode removes the B2BUA functionality but exposes Registrar and the relay SIP Proxy functionalities

- Always uses Relay Routes for all messages including REGISTER. If a relay route is not configured, Requests will be relayed using DNS resolution. If a registrations is resolved as local, the registrar would handle the registration including authorization

4. B2BUpperReg Mode - This is almost the same as the B2BOnly mode but with the additional capability of relaying registrations to upper registrars.

- **INVITE:** This mode always uses B2BUA Route.
- **REGISTER:** For registrations, it performs the Request URI and To URI checking and relay for a remote domain or process the registration locally for local domains.
- **Upper-Registration:** This mode also has the capability to hijack-registrations towards upstream registrars.

4.2.3 [Listeners]

By default, OpenSBC listens on 5060 for UDP in all available Interfaces. OpenSBC also provides a means to configure bindings to a specific address. To do this one must explicitly assign the address of the interface. The ability of OpenSBC to listen on multiple interfaces gave it the powerful capability to bridge calls across distinct networks if installed on a multi-homed host. Which interface to use for sending messages is intelligently determined via the OS routing table. Both SIP and RTP are seamlessly bridged by OpenSBC.

[Interface Address]

Interface	<input type="text" value="sip*:5060"/>	Keep <input type="button" value="v"/>
Address	<input type="text" value="sip*:5070"/>	Ignore <input type="button" value="v"/>

Figure 19. Interface address.

A list of specific address where OpenSBC will listen for incoming connections. By default this is set to 'sip*:5060' where the '*' character means 'all available interfaces'. To make OpenSBC bind to a specific interface and port, one must replace the '*' with an actual interface address, for example 'sip:192.168.0.10:8080'. This will bind OpenSBC to only the IP address 192.168.0.10 on port 8080. Take note that the default listener in 5060 is now replaced by this specific interface binding. Multiple listeners may be defined for a single instance of OpenSBC.

4.2.4 [Media]

Audio NAT traversal is achieved by OpenSBC by proxying media between the NATed user agent (UA) and the Internet. When OpenSBC detects that a call came from a NATed UA based on the via address, it would automatically proxy media for the call. OpenSBC also uses the Registration information of UAs to determine if calls terminating to the UA needs to be media-proxied. It must be noted that proxying media is not always a guarantee of successful NAT traversal for audio. It is required that User Agents uses the same UDP socket to send and receive RTP packets.

[Always Proxy Media]

Always Proxy
Media

Figure 20. Always proxy media option.

This parameter tells OpenSBC to proxy RTP for all calls regardless of whether the the UA is behind NAT or not.

[Static RTP Media Address]

Static RTP
Media
Address

Figure 21. Static RTP media address parameter.

This parameter allows a static IP address to be sent in SDP instead of the actual OpenSBC interface bindings. This is useful when OpenSBC itself is behind a NAT and the router is set to provide it with a one-is-to-one port mapping.

4.2.5 [Call Transfer]

OpenSBC supports blind transfer and attended call transfer. There are two types of call transfer procedure OpenSBC supports. The first is Pass-through REFER and the second is Local REFER. Attended call transfer would only work for Pass-through REFER.

[Pass-through REFER]

Pass-through REFER works by relaying the REFER to the remote UA. The Refer-To header is rewritten to point back to OpenSBC while the original Refer-To header is added as prefix to the User portion of the Refer-To URI. This way, OpenSBC would always stay in between the call legs even if the call is transferred. OpenSBC just reconstructs the original Refer-To URI by extracting the prefix from the User portion of the incoming Request-URI.

[Enable Local Refer]

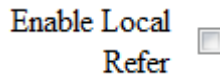


Figure 22. Enable local refer option.

Local REFER will not relay the REFER request. Instead, OpenSBC will create a new call for the REFER. This is advisable in cases where user agents do not support REFER directly. Local REFER would require more processing than Pass-through.

4.2.6 [Encryption]

OpenSBC Supports two types of encryption. These are non-standard encryption supported by some popular hardware vendors of IP Phones and ATA's.

[Encryption Mode]

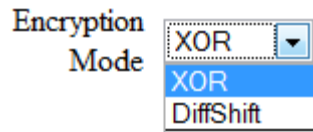


Figure 23. Encryption mode option.

1. XOR encryption uses an extended OR comparison to produce a new hashed REQUEST. Both SIP messages and RTP packets are hashed when XOR encryption is active.
2. DiffShift encryption uses a simple addition and subtraction to the numeric value of each character in SIP Messages to produce the hashed request. RTP will not be hashed if DiffShift mode is in use.

[Encryption Key]

Encryption
Key

Figure 24. Encryption key option.

This is the hash key used by the encryption algorithms.

4.2.7 [Event Processors]

OpenSBC is using an event-based architecture. This means all function calls in OpenSBC are non-blocking. Each transaction unit has its own independent event queue. Each event queue has a static pool of processors.

[Transaction Thread Count]

Transaction
Thread Count

Figure 25. Transaction thread count.

The number of processors to handle the transaction event queue. The default value is 10.

[Session Thread Count]

Session
Thread Count

Figure 26. Session thread count.

The number of processors to handle session events. The default value is 10.

4.2.8 [Timers]

Application timers are exposed to allow custom timeouts for alerting and call seizure. This is used by the failover routing functionality.

[Alerting Timeout]

Alerting
Timeout

Figure 27. Alerting timeout.

The amount of time in seconds the call would wait for an alerting packet. This can be any none 100 provisional response such as 180 and 183.

[Seize Timeout]

Seize Timeout

Figure 28. Seize timeout.

The amount of time in seconds the call would wait for a 200 OK.