# How to Install Asterisk 13 and PJSIP on CentOS 6

Justin Hester

February 24, 2015

With the release of a certified branch of Asterisk 13, the Asterisk training team decided now is the time to provide a brief set of “install from source” instructions. What follows is my three step program to install Asterisk 13. If you are experienced with earlier versions of Asterisk there are some changes to consider, namely the new SIP channel driver powered by the PJSIP SIP stack. The new channel driver is called PJSIP and has been the topic of a few wiki articles and conference presentations already. The original SIP channel driver has been moved to extended support and will not compile by default in Certified Asterisk 13. If you’re still unclear about the impact of the new channel driver, I encourage you to dig into resources linked from this post.

## Things to consider

- These instructions have been tested on a freshly installed CentOS 6.6 x86_64 virtual server.
    - These instructions must be modified to work with the 32-bit version of CentOS.
    - I will point out alternate steps for the 32-bit version of CentOS where appropriate.
- Because the focus of this post is the new SIP channel driver I have not included instructions for DAHDI and LibPRI. Instructions for this process can be found here.
- For the purposes of this installation we recommend changing SELinux configuration from the default of “enforcing” to “permissive”. Permissive mode will not enforce security policy but will log actions which can be helpful when you need to re-enable SELinux. There are many ways to manipulate SELinux settings that are outside the scope of this article. A typical method is to use a text editor to modify the ‘SELINUX=’ line in /etc/selinux/config by changing “enforcing” to “permissive”. After you update and disable SELinux, you may need to reboot and relabel the file system. Click here for more information about SELinux.
- I would also greatly discourage using any part of these instructions on a production server until you have vetted them through your own laboratory setup. If you wish to live dangerously I wish you luck.

Step 1 – Setup the environment

The first step is to install the dependencies required to build the PJSIP libraries and Asterisk 13. Using the CentOS yum package manager we’ll update all currently installed packages to their latest version and then install some of the most common dependencies for Asterisk and PJSIP.

```
# yum update
```

The kernel-devel package we install next could be slightly ahead of the kernel actually in use on your system. This is why we did a *yum update* first. If the kernel has been updated, be sure to reboot before moving forward. More information about the kernel-devel packages available for CentOS can be found here. The following command will install several packages that are needed to compile and install PJSIP and Asterisk.

```
# yum install -y epel-release dmidecode gcc-c++ ncurses-devel libxml2-devel make wget openssl-devel newt-devel kernel-devel sqlite-devel libuuid-devel gtk2-devel jansson-devel binutils-devel
```

NOTE:If you encounter an issue resolving the dependencies check out the fantastic install_prereq tool shipped with the Asterisk tarball. Located in the contrib/scripts directory of the Asterisk source directory that will be unpacked in step 3.

Step 2 – Install pjproject

Next you will download and install the pjproject sip library directly from pjsip.org. But first we'll change directories to work in the /usr/src directory.

```
# cd /usr/src

# wget http://www.pjsip.org/release/2.3/pjproject-2.3.tar.bz2

# tar -jxvf pjproject-2.3.tar.bz2
```

This will create the `pjproject-2.3` directory. Change to this directory and run the following set of commands to build and install the pjproject sip library.

```
# cd pjproject-2.3

# ./configure CFLAGS="-DNDEBUG -DPJ_HAS_IPV6=1" --prefix=/usr --libdir=/usr/lib64 --enable-shared --disable-video --disable-sound --disable-opencore-amr
```

This command must be modified when using a 32-bit operating system. Just remove the `--libdir=/usr/lib64`option from the command. The other options may be different depending on how you want to use Asterisk. More information about these options can be found on the Asterisk wiki or by running the command `./configure --help`. The next four commands will build, install and link the pjsip libraries.

```
# make dep
# make
# make install
# ldconfig
```

And finally this next command will verify the pjsip libraries have been dynamically linked.

```
# ldconfig -p | grep pj
```

Your output should look something like this:

```
[root@a13pj pjproject-trunk]# ldconfig -p | grep pj
        libpjsua2.so.2 (libc6,x86-64) => /usr/lib64/libpjsua2.so.2
        libpjsua2.so (libc6,x86-64) => /usr/lib64/libpjsua2.so
        libpjsua.so.2 (libc6,x86-64) => /usr/lib64/libpjsua.so.2
        libpjsua.so (libc6,x86-64) => /usr/lib64/libpjsua.so
        libpjsip.so.2 (libc6,x86-64) => /usr/lib64/libpjsip.so.2
        libpjsip.so (libc6,x86-64) => /usr/lib64/libpjsip.so
        libpjsip-ua.so.2 (libc6,x86-64) => /usr/lib64/libpjsip-ua.so.2
        libpjsip-ua.so (libc6,x86-64) => /usr/lib64/libpjsip-ua.so
        libpjsip-simple.so.2 (libc6,x86-64) => /usr/lib64/libpjsip-simple.so.2
        libpjsip-simple.so (libc6,x86-64) => /usr/lib64/libpjsip-simple.so
        libpjnath.so.2 (libc6,x86-64) => /usr/lib64/libpjnath.so.2
        libpjnath.so (libc6,x86-64) => /usr/lib64/libpjnath.so
        libpjmedia.so.2 (libc6,x86-64) => /usr/lib64/libpjmedia.so.2
        libpjmedia.so (libc6,x86-64) => /usr/lib64/libpjmedia.so
        libpjmedia-videodev.so.2 (libc6,x86-64) => /usr/lib64/libpjmedia-videodev.so.2
        libpjmedia-videodev.so (libc6,x86-64) => /usr/lib64/libpjmedia-videodev.so
        libpjmedia-codec.so.2 (libc6,x86-64) => /usr/lib64/libpjmedia-codec.so.2
        libpjmedia-codec.so (libc6,x86-64) => /usr/lib64/libpjmedia-codec.so
        libpjmedia-audiodev.so.2 (libc6,x86-64) => /usr/lib64/libpjmedia-audiodev.so.2
        libpjmedia-audiodev.so (libc6,x86-64) => /usr/lib64/libpjmedia-audiodev.so
        libpjlib-util.so.2 (libc6,x86-64) => /usr/lib64/libpjlib-util.so.2
        libpjlib-util.so (libc6,x86-64) => /usr/lib64/libpjlib-util.so
        libpj.so.2 (libc6,x86-64) => /usr/lib64/libpj.so.2
        libpj.so (libc6,x86-64) => /usr/lib64/libpj.so
```

Step 3 – Install Asterisk 13

Now we'll download and install the latest Certified-Asterisk 13.1 branch from source. First we'll change directory up one level to `/usr/src`.

```
# cd ..
```

Then we'll use the wget command to download the tarball from downloads.asterisk.org.

```
# wget http://downloads.asterisk.org/pub/telephony/certified-asterisk/certified-asterisk-13.1-current.tar.gz
```

Next the tar command will unpack the Asterisk source code into a new directory named `certified-asterisk-13.1-certified`. Then we'll go to the new directory.

```
# tar -zxvf certified-asterisk-13.1-current.tar.gz
# cd certified-asterisk-13.1-cert1
```

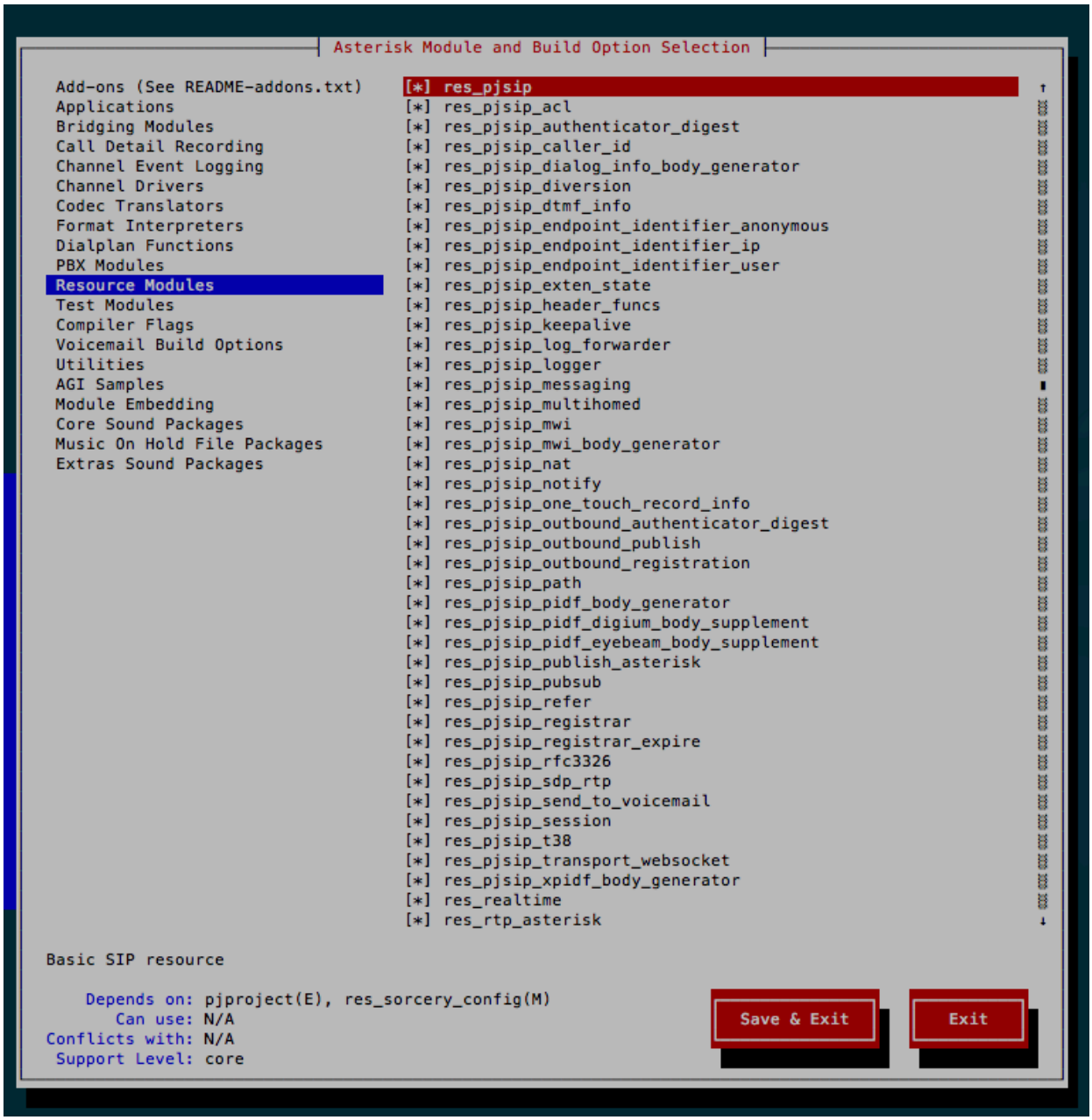
The next set of commands will build and install Asterisk. Remember to skip the `--libdir=/usr/lib64` option for 32-bit versions of CentOS. In that case just run the command `./configure`.

```
# ./configure --libdir=/usr/lib64
```

Next you will run the make menuselect command. This step will verify if the pjsip channel driver dependencies have been successfully installed.

```
# make menuselect
```

Use the arrow keys to navigate to “Resource Modules” in the left column, about halfway down the list. Press the right arrow key and then scroll down until you see the list of modules beginning with “res_pjsip_”. If these modules have “XXX” to the left of their name then the dependencies have not been met. You’ll need to go back to the /usr/src/pjproject directory, run the “*make distclean*” command and start over carefully looking for any error messages and proceed from there. If you see [*] instead of XXX then the res_pjsip module’s dependencies have been met and you can proceed to the next steps. Your menuselect screen should look like this:




After exiting the menuselect screen the next set of commands will build and install Asterisk along with a set of sample configuration files.

```
# make && make install
```

```
# make samples
```

If you want Asterisk to start at boot time use the following command to setup the Asterisk service.

```
# make config
```

And finally, run the command “service asterisk start” to immediately begin the Asterisk service without the need to reboot first. Now you’re ready to begin configuring the PJSIP channel driver on your freshly installed instance of Asterisk 13. If you run into an issue with these instructions feel free to leave a comment on this post, check the official Asterisk forums or reach out to the Asterisk community for help.

Special thanks to the Asterisk Development team for pointing out some helpful improvements to these instructions.

Asterisk 13, CentOS, PJSIP

## About the author

**Justin Hester**

is Digium’s Asterisk Technical Trainer. Before joining Digium in 2013, Justin spent seven years performing various roles related to training and technical support in a large corporate contact center environment. Justin holds a BS from University of North Alabama. When he isn’t grepping security logs, Justin is playing vintage video games with his wife and five-year-old son, or practicing his disc golf drive.

## 12 Responses to “How to Install Asterisk 13 and PJSIP on CentOS 6”

1. 

   Vic

February 25, 2015

You can automate the selection of /usr/lib and /usr/lib64 as follows

ARCH=$(getconf LONG_BIT | grep “64”)
./configure –libdir=/usr/lib${ARCH}

Also, I think it may be good idea to add –prefix=/usr for pjsip. If you do not I believe it install files in /usr/local/ instead of /usr like the rpm package does.

2. 

Justin Hester
February 25, 2015

Hi Vic, thanks for the tip about using getconf and grep to automate which /usr/lib directory to use.

As for the ‘–prefix=/usr’ setting, that already appears in the ./configure command for PJSIP between the CFLAGS and libdir options. After checking the post I can see an unfortunately wrapped line of text makes it a bit difficult to see. In any case, you’re spot on about the “dueling /usr directories”.

3. 

Ambiorix
March 4, 2015

Everything worked fine but when I tried to conect to Asterisk I got the following error :

asterisk: error while loading shared libraries: libasteriskssl.so.1: cannot open shared object file: No such file or directory

I solved running ldcondig in the Asterisk installation directory and then restarting the asterisk service

4. 

Aaron
April 3, 2015

Thanks for the guide, however, a couple of issues.

SELinux should never be disabled. If you have to disable SELinux then you are doing something wrong or the software has not been designed correctly for the RHEL/Centos environment. Security is important.

Installing Asterisk from source on a production machine is a bad idea. The package manager should be able to install and update Asterisk with requiring the development tools. Besides, there is a package repository for Asterisk 13 so why is this even needed? Perhaps you could add a new guide for installing the packages? The Wiki page is not very clear on this.

5. 

Justin Hester
April 7, 2015

Thanks for the comment on this post, I think we agree. I would say security is more than important, it is required. And I agree that SELinux should never be disabled in production. I do recommend setting SELinux to permissive mode when first installing and setting up a new system in the lab. A requisite part of the process (which is outside of the scope of this blog post) is taking the log output from SELinux in permissive mode and configuring SELinux to allow Asterisk to do the job it wants to do. Only after SELinux is enforcing a functional security policy should the system be put into production.

Packages are a wonderful thing that will make the administrator's job much more efficient provided the package is everything you want and lacks nothing you do not want in production. There are cases where customizing an Asterisk install is required. These instructions are meant for a safe, breakable lab environment where the user can get a basic install of Asterisk 13 with the new pjsip channel driver from source. And once the lab build is ready, automate and rapidly deploy.

Thanks for letting us know about an issue with the Asterisk wiki. I like the idea of a more in depth guide for using the package install method. It would help us greatly if you could detail what specifically wasn't clear and perhaps give us suggestions on how to improve the content. In order to increase the clarity of the content we need to know what made it unclear for you in particular.

Thanks!

6. 

Alex Hu
April 26, 2015

FYI. When build for centos 6 on amazon ec2, I used “./configure –libdir=/usr/lib64 CFLAGS=-mtune=native”. Otherwise, the asterisk will not run correctly.

7. 

sean darcy
April 26, 2015

Having just spent the better part of a week trying, and failing, to install asterisk with selinux enforcing, I think almost everyone must be using permissive. I’ve looked all over: there’s nothing describing how to have a usable asterisk server with selinux enforced.

selinux is just too abstruse to use. If not, where are the examples?

8. 

Justin Hester
April 30, 2015

Examples of SELinux configuration tend to be sparse and rare due to the nature of the tool. Because SELinux wants to validate all the interactions between all the processes and all their activity with all the files, configuration becomes much more complex considering Asterisk will often touch files located in a couple of different places in the filesystem. And because Asterisk can be configured to do just about anything the SELinux configuration that works for my Asterisk will very probably not work for yours. Even worse if an example SELinux configuration has the appearance of protecting your Asterisk server but actually leaves an unknown security hole.
The link [1] provided in the blog post has some good introductory information about SELinux and I would also recommend the Red Hat SELinux User Guide [2]. In my opinion the best way to find out what SELinux configuration will work with your Asterisk implementation is to use the logs generated by permissive mode while building and testing in a safe lab environment.

[1] http://wiki.centos.org/HowTos/SELinux
[2] https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/index.html
NOTE: I linked to the RHEL version 6 user guide since this is the version of CentOS discussed in the blog post.

9. 

Terry
May 4, 2015

Congrats for this how-to. Very useful for whom are starting with asterisk
Thanks.

10. 

Michael Newton
October 8, 2015

This has been helpful; just trying my first build of 13, and slowly wrapping my head around changes from 11.

It’s worth mentioning, for those scripting a build, that module selection doesn’t have to be done with a TUI menu. The menuselect command allows you to enable and disable individual modules or categories. From the source directory:

cd menuselect
make menuselect
cd ..
make menuselect-tree
# here of course you would replace –help with your chosen options
menuselect/menuselect –help

11. 

Jose Tapia
October 8, 2015

Good Article, for the people that doesnt have the libjasson can write
git clone https://github.com/akheron/jansson.git
cd ~/jansson
autoreconf -i
./configure
make
make install