Installing from source

If you're goal to to build RPMs, go to Building RPMS on CentOS or Fedora. The instructions on this page are for developers that want to compile and install the source for developing new features OR if binaries are not available for your operating system.

When building from source there are two areas to consider.

- Building non-standard dependencies (e.g. mongodb, freeswitch)
- Building sipXecs components (e.g. sipXproxy, sipXconfig)

For building non-standard dependencies from source, there is no real good way in sipXecs to make this any easier then following the instructions on the individual projects websites. However, if you happen to be working on Fedora, CentOS or Redhat, then there's a good chance the binaries are available for your system and you won't need to build dependencies. After all, you're not developing features for these packages and therefore won't need to recompile often.

Notes

1. If you run into any issue on this page, review Building RPMS on CentOS or Fedora for possible related instructions that work for you.
2. Be sure to check out Release Engineering Notes section for more useful information

Step 1.) Download and initialize source code

```
sudo yum install make automake libtool git rpm-build
git clone git://github.com/sipxcom/sipxecs.git
cd sipxecs
git submodule init
git submodule update
autoreconf -if
```

Here we build in a separate directory. It's not strictly necessary, but you'll appreciate separating build from source when you want to build multiple branches, rpms or change build settings. It will also catch Makefile issues where you assume source and build are in same directory which can be a common mistake. The directory name "build" is arbitrary.

```
mkdir build
cd build
../configure
```

Step 2.) Enabling additional components

By default only core and config modules are compiled. If you wish to add other modules, such as Grandstream phone support, you will need to enable them. You can enable individual projects for compilation or project groups.

Project groups

There are currently 5 project groups that are available for compilation:

- **$(sipx_all)**
  - Compiles all sipX projects
- **$(sipx_core)**
  - Compiles only core sipX projects. Enabled by default
- **$(sipx_extra)**
  - Compiles sipX extra projects
- **$(sipx_config)**
  - Compiles sipXconfig projects. Enabled by default
- **$(sipx_lang)**
  - Compiles sipX language projects

Including/Excluding projects from build.

Now you can define which packages you want to build. For example, if you want to build them all, simply add the following to **.modules-include**

.modules-include

```
echo '$(sipx_all)' > .modules-include
```

But what if you want to exclude a particular package or group of packages? Not to worry, simply add them to the file **.modules-exclude.** We'll disable the language projects and the **sipXgrandstream** project as an example:

.modules-exclude

```
echo '$(sipx_lang) sipXgrandstream' > .modules-exclude
```

To get a full list of the projects that will compile, run

.modules-exclude

```

```

Step 3.) Building

Finalize setting up your build system.

NOTE: You might need EPEL or some required packages will not be available.

```
make setup.sh
sudo ./setup.sh
```

You can edit setup.sh if needed for your specific situation. Then start the build process:

```
sudo mkdir  /usr/local/sipx
sudo chown sipx.sipx /usr/local/sipx
make sipx
```