Building

You can build SipXcom on the system you are working with or you can create RPM files (CentOS/Fedora/RHEL) to make it installable on another machine. DPKG files (Debian/Ubuntu) are not supported right now.

RPM files are created in a chroot environment called Mock. With this you can build RPM files for other distributions then the one hosting the files. CentOS 6 is not supported anymore since release 19.04, the main distribution is CentOS 7 (== RHEL7).

Disclaimer: the latest modifications to this guide describe creating 64 bit RPM files from SipXcom 19.12 on a CentOS 7 64 bit host system, using a CentOS 7 64 bit Mock environment. So only this part has been verified. Also, local building on CentOS 7 is tried, however this is not finished, the lessons learned however are included as well.

Install prerequisites:

```
yum install git autoconf automake httpd genisoimage sudo
```

Note: It's advised not to build from the root user. This will cause that some scripts wil run as user root, which is not the way things are designed, this will cause looping scripts which will crash your system. You must start SipXcom as root though; this process will start parts of it with the right user.

Create a different user with sudo permissions.

```
useradd -m sipx
passwd sipx
visudo
# add sipx as sudo user
sipx    ALL=(ALL)       NOPASSWD:ALL
```

Login again as the sipx user and checkout source code, specific branch and update submodules. SipXcom consists of a lot of seperate git repositories, which are all submodules. I noticed the .gitmodules file expects a public-private authentication to github.com to be in place, so I changed it to https with sed, so it works without that.

```
git clone https://github.com/sipXcom/sipxecs.git
git checkout -B release-19.12-centos7 origin/release-19.12-centos7
sed -i 's/git@github.com:/https:\/\/github.com\//g' .gitmodules
git submodule init && git submodule update
```

Initialise:

```
autoreconf -if
mkdir build && cd build
```

Configure the build:

```
../configure
```

Or, for creating RPM's (default distro is centos-6-x86_64):

```
../configure --enable-rpm
```

To build for a different DISTRO submit this in the configure like so:

```
../configure --enable-rpm DISTRO="centos-7-x86_64"
```

Finalize setting up your build system. You can edit setup.sh if needed for your specific situation.

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

Before starting building, first some more information about this. In the past SipXcom needed some extra packages from epel (Extra Packages for Enterprise Linux) to build. However these days this tree contain the wrong versions of these packages. However, download.sipxcom.org offers all the right ones. So, to avoid issues with dependencies add this to /etc/yum.conf:

```
[sipxcom]
name=sipxcom
enabled=1
baseurl=http://download.sipxcom.org/19.12-centos7/CentOS_7/x86_64/
gpgcheck=0
```

If any epel settings are in there (or in /etc/yum.conf.d/), remove them.

Now some information about the make process that SipXcom offers for building. It's pretty extensive with a lot of interesting features but also complicated.

First of all run 'make' without any arguments, you'll get:

```
TARGETS
==================
*               - Compile and install from source this a specific project. Assumes all dependencies are already
compiled and installed.
*.check         - Run unit tests on for a single project
*.deps          - List all dependencies to build and run a project.
*.deps-missing  - List all dependencies to build and run a project that are not currently installed on this
system.
*.rpm           - Build single rpm for centos-7-x86_64.
help            - Show this help screen plus more information on more targets and important make variables.
lib.rpm         - Builds all dependencies needed for sipxecs for building for centos-7-x86_64.
repo            - Prepares repository for centos-7-x86_64 for publishing or re-distributing in ISO by removing
extra rpms and moving source rpms to separate directory.
repo-update     - Remove only the packages from chroot that are built. Takes 14 seconds when a full rebuild can
take 2 minutues
sipx            - Build all sipx components starting from autoreconf and finishing with install for each target
sipx.deps       - List all dependencies for all of sipxecs
sipx.deps-missing - List all dependencies for all of sipxecs that are not currently installed on this system.
sipx.list       - List all sipxecs projects considered in this build system. To add or remove, edit .modules-
include or .modules-exclude
sipx.rpm        - Builds all sipxecs rpms for centos-7-x86_64.
```

At the moment of writing some of those commands won't work, and this overview of features is not complete, use make help for that. For more information you can look in the makefile and the included ones: mak/*.mk. You can try the commands to learn something.

If you want to build, install and run on your local machine you'll need the dependencies on it. You should be able to use 'make sipx.deps' and 'make sipx.deps-missing' for that. However the make system checks if you use CentOS, and if that's the case, no matter the version, and old binary included with the sources will be used in those processes. It won't work.

Here's a fix:

```
sed -i '52iRPMSPEC := $(wildcard /usr/bin/rpmspec)' mak/20-list-dependencies.mk
sed -i -e '53,60d' mak/20-list-dependencies.mk
```

Then you can install all dependencies with this (so only if you want to build locally!):

```
make sipx.deps > deps.txt
sudo yum install $(cat deps.txt)
```

Then you should be able to build with:

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

This however will try to install things in different directories then /usr/local/sipx. It might be ok to run 'sudo make sipx' to avoid that, I'm not sure if this won't cause any other issues.

If you want to create RPM files, then you will only need one additional package on the host system which is needed for some generation task when building sipXconfig.

```
sudo yum install dart-sdk
```

Then the Mock environment is required, which is using a centos-7-x86_64.cfg file for setting the repositories to use in the chroot environment. This however also has the previously described epel issues. **So edit mak/mock/centos-7-x86_64.cfg, disable all epel entries and add the sipxcom repo from above.**

Then initialise the Mock environment, which will be used as a chroot environment to build everything.

```
make repo-init
```

This step can be skipped, but you can build all libraries required to run SipXcom. You can put those on your repo server so that all dependencies are there. It also possible to get the dependencies from download.sipxcom.org, which is the preferred method.

```
make lib.rpm
```

Before starting the actual build there's one more issue. SipXcom uses oss_core as a component. I have not succeeded in building this package, so you should use the package provided by download.sipxcom.org and not try to build the rpm:

```
echo oss_core > .modules-exclude
```

As stated before more packages can be excluded if no changes have been made to the original versions. Disabling beyond oss_core has not been tested.

Then make SipXcom RPM's and get a coffee:

```
make sipx.rpm
```

The resulting packages will be in repo/CentOS_7/x86_64/

TIPS

The RPM build process only rebuilds modules that have changes in source code. If you want to force rebuilding a specific module you can remove the checksum files for that module from the build directory:

```
rm build/.centos-6-x86_64.sipXconfig.rpm*
```

If something goes wrong in the mock environment you can enter a shell and look around. If needed, you can install packages to it as well:

```
mock --configdir=mak/mock -r centos-7-x86_64 --resultdir=repo/CentOS_7/x86_64 --shell
mock --configdir=mak/mock -r centos-7-x86_64 --resultdir=repo/CentOS_7/x86_64 --install httpd
```

Related Articles

- Building

- Building RPMS on CentOS or Fedora
- Building RPMs using Docker
- Building sipXecs in Netbeans