# File Systems Source#

The most important thing to do first is to set up a staging directory to allow a) the source to build and b) prevent the build / install from overlaying your "standard" installation. This is always good practise when building QNX source but is particularly important when building the file system tree.

This link, OS Source Build, covers some **Very Important Details**, most notably the description of creating a staging area and a qconf-override.mk file (See steps 2,3 and 4). Just to re-emphasize the importance of this... If you don't properly set things up, you stand a good chance of overwriting your base installation and that would not be a good thing.

Make sure that you install the brand new Momentics 6.4.1 SDP in order to do the build (available here). The upgrade versions of Momentics (6.3.0 SP1, SP2 or SP3 plus the 6.3.2 Core OS upgrade) don't give you everything that you need to build successfully. Of particular importance are updates to some of the make files to properly handle SVN meta-info when you do a "make install" (Yup... If you see all of those "svn" files getting copied into your stage, then you aren't running the proper version of Momentics).

Just a couple of notes on the QCONF_OVERRIDE environment variable. On windows, in a DOS shell, use the SET command (there is no export). You can make the env var "stick" by right clicking on "My Computer" (select Properties). Select the "Advanced" tab and click on the "Environment Variables" button (bottom left in XP). Click on the "New" button and add in QCONF_OVERRIDE and the value and "OK" your way out and, the next time you start a DOS shell, QCONF_OVERRIDE will already be set up for you. If you're running self-hosted (or under Linux), you can add the export command to your .profile script and it will also automatically get set when you start a new terminal session.

## Dependencies#

Some of the files in this project have dependencies upon components in other. The components from the core-os project that some of the file systems components depend upon are ...

- lib/compat
- lib/login
- lib/qnx43
- lib/util
- services/slogger

### Extra dependencies#

As of April 8th 2009, a new macro was added to all filesystem project source. This new macro requires the installation of the srcversion update package. Install this package as root to your build tree.

```
# cd $QNX_TARGET/../..
# tar -xvf <srcversion-patch>
```

Now to re-build the system with the new header. From the trunk directory of your source installation

```
# make install
```

On 6.4.0, it is necessary to exclude building the mips architecture

```
# make install EXCLUDE_CPULIST=mips
```

These dependencies are linked to from our svn repository, so when you checkout the HEAD revision of trunk, you will get all of the dependencies except for services/slogger. The buildscript located in the "How do I build

the source?" section of this page will take care of compiling most of these dependencies for you. For services/slogger, however, you must manually install the headers to your staging dir by using: `make hinstall`

## Source Tree Organization#

The source is laid out with a top level similar to other projects. There is the "trunk" directory which is where the most up to date source code lives that is being worked on. We expect that all code in the trunk directory will work its way into the general product at some point in time. At any point in time, we can't guarantee that the trunk source will a) build and b) if it *does* build, work. While we will be making every effort to ensure that the trunk source will indeed build and operate, we all make mistakes from time to time, so I'm sure that code will slip in from time to time that won't have everyone's best interests in mind. Hey, that's what you get when you sign up to live development!

Some relevant portions of the tree are:

- **/trunk/**
  - **lib** - Libraries
    - **etfs** - Common code for the Embedded Transactional File System
    - **fs-flash3** - Common code for the Flash File System v3
  - **services** - File System processes and shared libraries
    - **blk** - Block file system shared libraries
    - **nfs2** - Network File System v2
    - **nfs3** - Network File System v3
    - **nfsd** - Network File System daemon
  - **utils** - Common file system utilities (grouped alpabetically)

## How do I get the Source?#

The source code is available on the File Systems repository located on http://community.qnx.com/sf/scm/do/listRepositories/projects.filesystems/scm.

To download the source code into your source directory:

**svn checkout --username <userid> http://community.qnx.com/svn/repos/filesystems/trunk** Where **<userid>** is the email address used to create your account on the QNX site.

## How do I build the source?#

**If you don't properly create a staging area and suitable qconf-override.mk file, you risk the chance of corrupting your standard installation.**

This one's simple. Go into the trunk directory and run this script (SourceGuide/fs-build.sh). It does not take any arguments.

That's it! All the associated header files for the build and binaries created will get installed in your staging directory.

## I've built them, but how do I run them?#

One important point to note: To run any of the file system drivers (and some of the utilities) you will need to have root privileges. In order for these programs to be given proper permissions under QNX or Linux you need to change the "op" utility to be "suid" (i.e. run at the privilege level of the owning user). By default in Neutrino, op is owned by root but isn't setuid since this is considered to be a security issue. To do this on Neutrino, log in as root and simply "chmod u+s /usr/bin/op" (on Linux, you need to figure out where the "op" utility lives).

This project does not include the hardware drivers for devb, ETFS or devf. Consequently, you will not be able to build an ETFS or devf executable; however, you can build their static libraries. Once the BSPs and Drivers project is released, you will then be able to link those libraries to the ETFS and devf driver code and create your executable. The block file systems (CD, DOS, QNX4, ...) do not have this limitation. They are compiled into dynamic libraries to by linked at runtime against the devb driver.

Where can I get pre-built binaries? If you don't want the latest and greatest source base, you can grab a set of pre-built binaries from the Project Downloads Page. This page is where we'll be placing integration builds (raw builds that might not have undergone much, if any, testing), Milestone builds (which have had, at a minimum, sanity testing performed), Release Candidate builds (which have had full regression and feature testing done on them) as well as the final GA product builds.