JsocWiki

CVS Initialization

All of the software for the JSOC-SDP is maintained in the "CVS" software configuration management system. In order to make new code using the DRMS you will need to establish a CVS work area, download the JSOC software tree into it, configure it for the machine you are using for code development, and build the libraries and programs. Then you will be able to add your own pipeline modules or other tools.

What is CVS anyway? It's a code manager that allows check-in and check-out of programs from the same directory structure. In a nutshell, there is one official set of source-code files that resides in one or more file directories in a remote CVS server. This set is called the "repository". A developer who wishes to edit these files must first "checkout" (download) the files to a "working directory" on their local work machine. This working directory, which is simply a user's local directory containing a copy of the respository files, is sometimes called a "sandbox". The developer makes changes, then "commits" (uploads) the files back to the CVS server. Commiting is not the same thing as copying, since that would overwrite changes made by other developers who made changes between the checkout and commit of a file by another developer. Instead, CVS will MERGE changes back to the repository, i.e., it will determine exactly what code has changed and insert that change into the respository file. This works as long as two developers do not make conflicting changes (e.g., changing the same line of code in different ways). If a conflict occurs, the developer who is attempting to commit a change that conflicts with a previous change needs to manually edit the file in their working directory (ensuring that both their change and the previous changes are both contained within the working-directory file), and then commit the file. To learn more about CVS, read: a CVS Wiki

The base JSOC system is revised often. The instructions for the most recent release should be consulted to make sure the following is still correct. See the Developer's Guide for a list of recent releases.

Create a CVS Working Directory

This step only needs to be done once. The normal place to put your JSOC working directory is in a "cvs" subdirectory in your home directory. E.g.:

```
% cd¶
% mkdir cvs¶
% mkdir cvs/JSOC¶
```

Many users find it convenient to make a link to the JSOC tree. E.g.:

```
% cd¶
% ln -s cvs/JSOC .¶
```

Set Your Environment

This step also needs to be done but once. To use CVS and JSOC, you must set four environment variables:

- CVSROOT - Path to the CVS repository that contains CVS modules used by JSOC.
- CVS_RSH - Specifies the program used by the CVS client (the code that runs on your machine) to access the CVS repository.
- JSOCROOT - Path to the local JSOC tree (the place on your working machine where you want your copy of the respository files, AKA "working directory" in CVS lingo.). Setting this variable is REQUIRED for full functionality, but without it, the base system (see *Download the JSOC Source-Code Tree below*) will still function.
- JSOC_MACHINE - Specifies the architecture of the machine that will be used to build and run JSOC programs.

In greater detail, a Stanford user should do the following. Set the CVSROOT environment variable to contain ":ext:sunroom.stanford.edu:/home/cvsuser/cvsroot". Set the CVS_RSH environment variable to contain "ssh". Set the JSOCROOT environment variable to contain "$HOME/cvs/JSOC". Set the JSOC_MACHINE environment variable to contain a string that correctly identifies your machine architecture (one of "linux_ia32", "linux_ia64", or "linux_x86_64"). It is best to let a script do this for you (see below).

Regardless of user type, you should create an alias to "$JSOCROOT/base/util/scripts/cvsstatus.pl" named "cvsstatus". This script prints the status of all CVS repository files rooted at the current working directory.

It is best to put all these commands in a script that gets called from one of your initialization scripts (e.g., .login, .csh). Your script should look similar to the following:

```
setenv CVSROOT :ext:sunroom.stanford.edu:/home/cvsuser/cvsroot¶
setenv CVS_RSH ssh¶
setenv JSOCROOT $HOME/cvs/JSOC¶
¶
set OS = `uname -s`¶
switch("$OS")¶
· case "Linux*":¶
··· set CPU = `uname -m`¶
··· breaksw¶
· default:¶
··· set CPU = `uname -p`¶
··· breaksw¶
endsw¶
¶
```

```
if ( ! $?JSOC_MACHINE ) then¶
· if ( $OS == "Linux" ) then¶
··· switch("$CPU")¶
··· case "i686"¶
··· case "i386"¶
··· case "ia32"¶
····· set JSOC_MACHINE = "linux_ia32"¶
····· breaksw¶
··· case "ia64"¶
····· set JSOC_MACHINE = "linux_ia64"¶
····· breaksw¶
··· case "x86_64"¶
··· case "em64t"¶
····· set JSOC_MACHINE = "linux_x86_64"¶
····· breaksw¶
··· default:¶
····· set JSOC_MACHINE = "custom"¶
····· breaksw¶
··· endsw¶
· else¶
···· set JSOC_MACHINE = "custom"¶
· endif¶
endif¶
¶
setenv JSOC_MACHINE $JSOC_MACHINE¶
alias cvsstatus "$JSOCROOT/base/util/scripts/cvsstatus.pl"¶
```

In addition, in order to run JSOC programs and scripts without having to explicitly provide an absolute path to these executables, add JSOCROOT/bin/$JSOC_MACHINE and $JSOCROOT/scripts to your path environment variable.

Download the JSOC Source-Code Tree

The JSOC CVS respository is divided into two sets of files, one for each of two types of user. A Stanford user requires source code for the base DRMS system. This system provides low-level database access to data series. It also provides utility programs that perform common database-access tasks (e.g., querying the database to find information on a particular data series). In addition, a Stanford user requires access to source code that is specific to Stanford projects (e.g., parsing telemetry packets and ingesting the extracted data into data series). The second type of user requires the base DRMS system only. This user is not interested in building or running Stanford-specific code, but instead will be developing or using their own programs. Each set of files is identified by a CVS "module" - "JSOC" for the Stanford user, and "DRMS" for the non-Stanford user.

Each type of user must "check out" the appropriate set of JSOC files (i.e., download the files from the remote CVS server to their working directory on their local machine). This need be done only once (barring major changes to the JSOC CVS repository). Thereafter, the user can "update" their local source tree (see below). To check out the Stanford-specific set of files, a Stanford user does the following:

```
% cd $JSOCROOT/..¶
% cvs checkout -AP JSOC¶
```

To check out the base DRMS system, a user does the following:

```
cd $JSOCROOT/..¶
cvs checkout -AP DRMS¶
```

Note: If CVS asks you for a password or fails to checkout the code, make sure you have created a public and private ssh key under your ~/.ssh directory. You can ask our sys-admin for help (action@sun).

Update Your Existing Working Directory ("Sandbox")

As CVS is a multi-user system, other users will change the repository files after your initial checkout of these files. To get these changes into your working directory, you need to "update" your working directory. Although you can do this using CVS's update command, a couple of scripts that reside inside your working directory are available. **It is highly recommended that these scripts be used to update your working directory.** Improper use of "cvs update" could result in an unstable working directory.

To update the source code in $JSOCROOT and build the default binaries for all supported architectures, use jsoc_update.csh. This works for any user, Stanford and non-Stanford alike. As this builds every executable on every architecture, it takes a lot longer to run than jsoc_sync.csh. However, if you simply want to run JSOC programs, but not develop new code, then this script is sufficent.

```
cd $JSOCROOT¶
jsoc_update.csh¶
```

To update your source-code without building progarms, use the jsoc_sync.csh script. Again, regardless of the type of user, this works.

```
cd $JSOCROOT¶
jsoc_sync.csh¶
```

A good time to run one of these scripts is when a major release occurs (you will be notified by email, provided you are on the jsoc_dev mail list).

Altering the Default 'make' Behavior

Running 'make' will perform a 'default make' which generates a subset of the possible binaries. All compiler warnings will be suppressed and the binaries will be optimized for speed and contain no debug information. To read how to tune this behavior click here.

Notes

A new make system was put in place after Ver_3-5-PreRelease. If you have a CVS working directory based on that version or an earlier version, you'll need to run "make clean" before running *jsoc_update.csh*, *jsoc_sync.csh*, or *cvs update*.

To manually build the JSOC executables (i.e., if you're not using jsoc_update.csh), use the following:

```
% cd $JSOCROOT¶
% jsoc_sync.csh¶
% ./configure¶
% make -j 4¶
%¶
```

If jsoc_update.csh or jsoc_sync.csh indicates a CVS file conflict (check $JSOCROOT/cvsupdate.log, which is generated by these scripts, to see if a line of the form "C <filename>" exists), then you'll need to resolve the conflict before continuing (see the CM if necessary).

If the make file fails because it doesn't find the postgres 'ecpg' preprocessor, then you'll need to install the postgresql-devel package. E.g. "sudo yum install postgresql-devel".

JSOC compilation assumes the following RPMs are installed:

- postgresql
- postgresql-devel
- postgresql-libs
- openssl-devel