

CVS and Software Releases

Ernest L. Williams

For now we must learn to love CVS

Version Control Systems

- CVS [Deprecated but still needed at LCLS]
- Subversion [Deprecated but used by LCLS]
- Mercurial [In use by CSS collaboration]
- Bazaar [In use by EPICS Core collaboration]
- Git [Hardware section used it for a while with Norum]
- Murali will lead an evaluation effort to choose our next version control system. :)
- So, for now we must master CVS.

Learning CVS

- First, read the documentation. There are many good texts on CVS.
 - Essential CVS
 - Pragmatic version control using CVS
 - Open Source Development with CVS (3rd Ed.)
- Request Training from our local experts after doing the above. **[Mike and I are planning a CVS class in 2011]**
- How are we using CVS in the Controls Group?
- Creating a demo CVS Repository in your home directory for practice.

Purpose of CVS for us

- **Track** and **manage** the evolution of our control systems software.
- Integrate the use of CVS into our *software release* procedure.
- Replacement for Tape-Backup? Of course not, but we should be able to quickly revert back to a previous version of a file managed by CVS.

CVS: EPICS Software (1)

- Let's describe how we breakdown EPICS software and use CVS to track the components.
- EPICS = Core + Extensions + Tools + Modules + Applications
 - *We created a cvs module called “base”*
 - *We created a cvs module called “extensions”*
 - *We created a cvs module called “tools”*
 - *We created a cvs module for each epics “module”*
 - *We created a cvs module for each epics “application”.*

CVS: EPICS Software (2)

- Most of the **EPICS packages** we use are developed externally. So, we can treat it as “Third Party”
 - New EPICS packages are then imported or merged into our local CVS Repository.
- **EPICS Base** and **EPICS Extensions** rarely changes locally at our site. We typically modify configuration files to tailor EPICS for our use.
- Most **EPICS Modules** are also developed externally. We simply download and import to our CVS. However, we do develop and maintain some modules at SLAC.
- So most of our heavy development: **EPICS Applications** and **Tools :-)**

CVS: Software Development

- What does software development mean for us?
 - C/C++, EPICS sequencer, Java, python: *code writing*
 - Creating and configuring EPICS Databases and other files.
 - EPICS Databases could involve a lot of complex logic.
 - Creating archiver, alarm, and save/restore configs
 - Software component management: What do I mean?
 - Importing and integrating new software components that are developed outside of SLAC.
 - Example: autosave, asyn, ipac, base, etc...
 - **Releasing** and **Deploying** software components and applications to our *Users*

CVS: Releasing Software

- What does Releasing software mean for us?
 - Testing in Development to increase quality and minimize the time to deliver and deploy in production.
 - Creating software release text with information about what has changed, new features, etc...
 - Coordinating with controls software deputy for deployment and/or testing in production.
 - Communicating “ready for use” and consumption with clear version (cvs tagging) information to our customer.
 - When bugs are reported: only fix bugs on reported release tag (cvs branching).
 - Don't sneak in new bells and whistles

CVS: Tagging and Branching

- All EPICS software delivered to production should be tagged as appropriate. There may be some special cases.
- Stable EPICS Core and EPICS Modules should be **branched** for bug fixes and also to support the **import of newer releases**. Remember, we get most of software modules from outside of SLAC.
- EPICS Applications: Such as IOC Applications should be tagged at a minimum but branching is only optional. IOC software changes more frequently due to the fact we controlling an accelerator with new ideas happening every week. Maybe, even everyday.