Overall Requirements and Design of the Unix Development Environment

31-Mar-2003, V1.0 (summary, reworked release directories)

ESD Software Engineering Group

Table 1: Table of Contents

1 Introduction

This document describes a proposal for the design of the Unix software development environment of the SLAC accelerator control system. This system is intended to encompass the following areas:

Source code directories, and file version management - which are the important directories for such things as scripts, where application source code goes, where packages go (Ch 4), how it's managed by CVS (Ch 5), and file protections (Ch 6)
Release Escalation - how executables and associated files are identified as in "test" or "development" or "production", and moved from one such stage to the next (Ch 7)
Distribution – how executables, and associated files, which must be run on a particular host when in production, are moved to that computer's file-system (Ch 8)
Session environment definition – how the environment variables, such as those identifying common directories (eg CD_SOFT) or execution environment (like "PATH") get defined for each user, and for the control system accounts (cddev) on both development and production hosts (Ch 9)
Documentation and web support (Ch 10)
Filesystem cleanup and reorganization. Miscellaneous items to get the unix file system more streamlined (Ch 11).

1.1 Document Format

Each of these functional areas constitutes a chapter in this document. Within each chapter are sections on major topics of the development system. For most topics we describe the present system, if necessary some design objectives, then the planned system, and finally some notes for the transition from one to the other – how to get from A to B.

Chapter
    Topic
        Present System
        [Design References].
        Planned System
        Transition Plan
    Topic
        …
        …

1.2 Scope

This document concentrates on the development system as it pertains to development hosts – it's weak on recommendations for the production control system.

1.2.1 Not covered in this document

1. Process Monitoring, starting, restarting, or process privilege management (who can start and stop processes).
2. Plan for EPICS cvs. Specifically, section 5.3 (Planned Directory Layout) does not include whether EPICS is in the CD_SOFT/cvs repository, or has its own repository, nor whether extensions should be part of our release mechanism and conform to our makefile structure. We propose that the basic directory structure, permissions, and release procedure is implemented first – then we will be in a better position to decide what to do with EPICS base, site and extensions.

2 Summary

If you don't read any more of this document, at least check this out:

1. Host machines (non-IOC) will be split into three "execution-types", defined on account basis: DEV (a developer's login name on an AFS machine), PRODONDEV (cddev on an AFS machine), and PROD (cddev on an NFS machine).
2. Software will be released through an "escalation procedure", implemented as a sequence of directories (tst, dev, new, prod, bck). The 1st 4 of these directories will be effectively "mirrored" on all hosts.
    a. Later, tst, dev, and new will only be on AFS, and AFS will be "mounted" on PROD machines. The "prod" dir on PROD machines will remain on NFS.
3. Release Escalation and distribution will not be supported for IOC software yet.
4. Executable scripts, display files, matlab, python, etc, will all be put through the release escalation system.
5. Non-executable scripts will not go through the escalation system yet, they will remain in CD_SOFT/dev/script/ until we decide on whether to implement a "path" for them too.
6. The EPICS IOC makefile system will be extended for use as the framework for host side software development. We will create a makefile template to help new projects get started. The main extension will be to support the escalation procedure.
7. The existing CVS structure remains largely but not completely unchanged. A new CVS dir will be added, named "util". This will contain packages that whose makefiles use the new makefile template and guarantee to deliver to our escalation procedure. The existing "app" CVS directory will continue to contain both source code and config files, but should in the future do this only for applications as opposed to libraries, packages or suites (which go into util instead).
8. Three versions of EPICS IOC software will be maintained in CVS, and a scheme in which all three generations are in one CVS module but have 3 different reference areas will be used. This supports transition from one version of EPICS to another.
9. The existing AFS cd/soft directory system will be significantly cleaned up. CD_SOFT/support/package will be removed altogether!
10. ACLs on AFS will be extended to protect individual software projects, similar to the VMS protections scheme.
11. Execute permissions will be handled on PROD machines by a combination of accounts and directories through unix permission bits, and on PRODONDEV machines in combination with ACLs.
12. The existing login setup for EPICS development and production, will be unified with HEPiX, so that HEPiX logins on DEV and PRODONDEV use a modified version of the existing files (such as pathStup.csh etc).
13. Web support will be significantly upgraded, to support mirroring should AFS become temporarily unavailable, to support very easy web publication using new tools which can tunnel security, cache passwords, and know their file-system location; and the group's web site directories will be re-organized to make it clear where new documentation should go. There will be a new group home page at www.slac.stanford.edu/grp/cd/soft/; the home page at www-group/slac.satnford.edu/cdsoft/home.html will be retired.

3 Production and Development Systems

This chapter deals with the distinction between computers on which we develop software, and those on which we run it to control the accelerator. These each have different requirements, but sometimes the same machine is used for both.

3.1 Taxonomy of Machine Types

Code shall be developed on the (so-called) Development System. The Development System is basically the set of "Taylored" machines such as tersk, flora, slcs5 and so on, on which software shall be written, debugged and version controlled. Once developed, the binary and necessary supporting files, such as scripts, shall be moved to the Production System. The Production System is the set of Unix machines which run the operational aspects of the accelerator complex.

Call the host on which software is run for the operational accelerator a "Production host". The production hosts of most EPICS related host software are called "gateways"; the production host of IOC software is the IOC on which that software will be run. However, some significant body of software for the control system is presently run from Taylored (AFS) machines, for instance gateways 4 and 5, which run 8-pack and NLCDEV software, and DM displays. Aida running on slcs6 will be another example. So, those machines are in some sense Development Hosts which run production software – "production on development", or "production on Taylor".

3.1.1 Systematic Machine Definition

We will need to create a global environment variable which declares for each host which type it is, equivalent to the VMS "SLC$EXECUTION_MODE". But as opposed to the VMS case it will have 3 possible values, like DEV, PRODONDEV, and PROD[1].

Function	"Execution Mode"
"dev" – software which is in a development directory on a development system host.	DEV
"development on production" – software which is in a development directory on a production control system host. Similar to software which is being used from "DEVSHR" on MCC.	None – this is a temporary class-ification.
"production on development" – software which is in a production directory on a development control system host, and is being used operationally to run the accelerator.	PRODONDEV
"prod" – software which is in a production directory on a production control system host.	PROD

Table 2: Taxonomy of control system software "execution mode"

If the AFS system, or our connection to it, goes down, "production on development" software will not be available to run the accelerator complex (see Chapter 13).

3.2 Nomenclature

In this document, the term Production Host is used, where frequently the term "gateway" may be more familiar. The word gateway has been avoided here though partly because its definition isn't clear – it's roughly understood to mean a host running the gateway EPICS CA proxy server for purposes of network isolation, and partly because we do have hosts running production software which are not gateways in this narrow sense.

[1] "Development on Production" is only a designation of software, not of the host on which it is running, so this would not be included in the possible machine types.

See the Glossary (Chapter 12) for further definitions of terms.

4 Primary Directories of the Unix Environment

This chapter describes the important top-level directories used in the Unix control system environment. The Development System and Production System directories are described separately, and under each, first the presently used directories are outlined, and then how we plan to add to those.

4.1 Two File-systems, one directory organization

Files on the development system are in the AFS file-system, rooted at /afs/slac/. Files on the production systems are on the NFS file-system, rooted at /usr/. Since EPICS software involves many files in support of each application that must be available on production, there must be an organizing principle used for the directory structure on production. The principle we'll use is that we keep the files on production in a directory structure which largely mirrors the file-system on development. The two equivalent file-systems will be rooted at:

Table 3: Production and Development DirTree Roots

Host Type	Root directory	Environment Variable
Development	/afs/slac/g/cd/soft/	CD_SOFT
Production	/usr/local/cd/soft/	CD_SOFT

See Ch 8 Distribution, for how these two sets of directories are kept up to date with respect to each other.

4.2 Primary Directories of Development System

The directories presently, and planned, to be employed in the Development System are described here. These are rooted at /afs/slac/g/cd/soft/.

4.2.1 Present Primary Directories on the Development System

The following describes the top-level directories on the Development System after we started collecting our software in the CVS repository in /afs/slac/g/cd/soft/cvs/ last year:

Table 4: Present Primary Directories under /afs/slac/g/cd/soft/

Directory	Important Subdirectories	Function
`cvs/`		The CVS repository
`ref/`		What's in CVS, plus the output of the gmake operation.
`dev/`	`script/` `@sys/bin/`	Release directory. "gmake" copies script files to dev/script and dev/@sys/bin/ here. The only directories in the PATH are here.
`ioc/`	`<epics-version>@`	IOC software. Symlink into ../ref/epics/<epics-version>/ioc

4.2.2 Planned Primary Directories on the Development System

The basic top level directories that shall be employed in the development system are outlined below. These are basically as those we use presently (see above), plus some more for release escalation.

4.2.2.1 Constraints

Presently "dev/" is the name of the directory for Host side software (in contrast to IOC) and also the intended name for the directory for the 2nd level of release. So there are a couple of problems.

Table 5: Planned Primary Directories on Development, under /afs/slac/g/cd/soft

Directory	Important Subdirectories	Function
cvs/	See Chap 7	The CVS repository
ref/	See Chap 7	Only what's in cvs/. This will be used for searching and browsing only, not building, and scripts should not be run from directories under ref/. Described in Chapter 5.
lib/		Archive (non-dynamic) libs (.a), these are not escalated.
tst/	 @sys/lib/ @sys/bin/ @sys/pbin/ javalib/ disp/ matlab/ python/ ora/ include/	Where software will be built for the first stage of its release. Not in the default PATH on any host type. Not in the default PATH on any host type. Not in the default PATH on any host type. Not in the default CLASSPATH on any host type
dev/	 script/ @sys/script/ @sys/lib/ @sys/bin/ @sys/pbin/ javalib/ disp/ matlab/ python/ ora/ include/	First level of public release. Platform independent non-executable script directory. Platform dependent non-executable script directory. First lib, bin, pbin, directories in the PATH on DEV hosts. First directory in the CLASSPATH on DEV hosts.
new/	 @sys/lib/ @sys/bin/ @sys/pbin/ javalib/ disp/ matlab/ python/ ora/ include/	Second level of public release. First lib, bin, pbin, in PATH on PRODONDEV hosts. Second lib/ and bin/ directories in the PATH on DEV hosts. First javalib on CLASSPATH on PRODONDEV hosts. Second javalib directory in the CLASSPATH on DEV hosts.
prod/	 @sys/lib/ @sys/bin/ @sys/pbin/ javalib/ disp/ matlab/ python/ ora/ include/	Final level of public release. Last lib and bin and pbin directories in PATH on PRODONDEV hosts. Last javalib directory in CLASSPATH on PRODONDEV hosts.
bck/	 @sys/lib/ @sys/bin/ @sys/pbin javalib/ disp/ matlab/ python/ ora/	When items are moved to prod/, the existing item in prod/ will be moved to bck/. "oldsoft" prepends bck/@sys/lib, /bin, /pbin to PATH. "oldsft" prepends bck/javalib to CLASSPATH.

1. There are presently many scripts which refer to $CD_SOFT/dev/script, so it will be difficult to rename the subdir off CD_SOFT to anything else, such as “host/” or “opi/”.
2. If development system software is all put under $CD_SOFT/dev, then what would we call the directory for the 2nd level of release beneath that – can’t be “dev” because that’s taken?

Given these constraints, here’s an imperfect solution: use the names $CD_SOFT/{tst/, dev/, new/ and prod/} to mean the names of the release directories for Host side (non-IOC) software only. $CD_SOFT/dev will be then mean host-side software (only) in the development stage of release. When we come to do a release procedure for IOC software (which we are not going to do at present –Kristi thinks we’re not ready), we will then decide whether the dev,new,and prod directories for ioc software shall be subdirectories of $CD_SOFT/ioc, or whether ioc/ shall be made a subdirectrory of $CD_SOFT/dev, $CD_SOFT/new and $CD_SOFT/prod.

4.2.2.2 Planned Top level Directories on the Development System

Given the constraints then, Table 5 outlines the top level directories to be used on the development system. This table shows that the plan is largely to keep $CD_SOFT/dev as it is, and that the release escalation directories for IOC software will be added only later.

lib/,tst/,dev/,new/,prod,bck/ contain ONLY host side software, all IOC software is under ioc/.

4.3 Primary Directories on the Production System

The system of directories on the production system is in transition. Until recently, the directories used on the production system for most EPICS related software were rooted at /usr/local/pepii/. The primary directories in that file-tree will not be described here further. The file-tree for the planned directories, to which we have been recently moving, and whose structure is intended to mirror that found on Development, is rooted at /usr/local/cd/soft/. The planned top-level directories in this new location are in Table 6: Planned Primary Directories on Production under /usr/local/cd/soft/.

Table 6: Planned Primary Directories on Production under /usr/local/cd/soft/

Directory	Important Subdirectories	Function
ref/		What's in CVS, plus the output of the gmake operation (See notes below).
tst/	solaris/lib/ solaris/bin/ solaris/pbin/ javalib/ disp/ matlab/ python/ ora/ include/	First release, but not in the default PATHs. User can choose to caddpath tst.
dev/	script/ solaris/script/ solaris/lib/ solaris/bin/ solaris/pbin/ javalib/ disp/ matlab/ python/ ora/ include/	First level of public release Lib,bin,pbin not in the default PATHs on PROD hosts. Not in default CLASSPATH on PROD hosts.
new/	solaris/lib/ solaris/bin/ solaris/pbin/ javalib/ disp/ matlab/ python/ ora/ include/	Second level of public release. First lib, bin, pbin directories in PATH on PROD hosts. First directory in CLASSPATH on PROD hosts.
prod/	solaris/lib solaris/bin/ solaris/pbin/ javalib/ disp/ matlab/ python/ ora/ include/	Final level of public release. Last lib, bin, pbin directories in PATH on PROD hosts. Last javalib directory in CLASSPATH on PROD host.
ioc/	<epics-version>/ioc/	IOC software.

Note that the only directory for non-executable scripts right now is CD_SOFT$/dev/script/. Non-executable scripts won't be moved through the release directories because Kristi indicates the effort in re-pathing all the calls made to existing scripts on production would just be too great.

5 CVS, Reference Areas, and /afs/slac/package.

This chapter deals with the question of the CVS directory layout, and where the reference areas for CVSed items can be found.

5.1 File Version Management on the Development System

File version management will use CVS. The CVS repository will be in /afs/slac/g/cd/soft/cvs/. A "reference" area will be created, in /afs/slac/g/cd/soft/ref/, in which we keep a permanently checked-out copy of the latest of each file in the repository. As opposed to on VMS, the reference directory will also be used for building), see 7.4.

5.2 Present Directory layout

Presently the reference area contains many directories and files that are not in cvs. That is, although the files are in $CD_SOFT/ref, they have not yet been added as modules to $CD_SOFT/cvs. The following tables summarize the existing contents of the reference area, the CVS repository, and the other commonly used areas for EPICS and Aida, and where they exist, the links between them.

Table 7: Present state of CVS Repository at $CD_SOFT/cvs/

Repository Directory Area	What's there now
$CD_SOFT/cvs/package/	Aida, CmdSrv/, Err/, Except/
$CD_SOFT/cvs/app/	Alh/, channelArchiver/, cmdSrv/, err/, fwd_server/
$CD_SOFT/cvs/ioc/	IOC configs and applications

Table 8: Present state of Reference directories at $CD_SOFT/ref/

Reference Area subdirectory	Interpretation (from CD_SOFT/ref/README)	What's there now
$CD_SOFT/ref/package/	Self-Contained Software Packages that we either get from the outside and change or develop in-house and make available to the outside	Aida/ CmdSrv/, Err/, Cmlog@-> /afs/slac/package/cmlog Epics@ -> /afs/slac/package/epics Except/
$CD_SOFT/ref/app/	Applications shared across multiple ESD projects	Alh/, artemis/, channelArchiver/, channelWatcher/, gateway/, cmdSrv/, err/, fwd_server/
$CD_SOFT/ref/epics/	IOC EPICS Applications	The reference directory of cvs/ioc, containing separate sub-dirs for each EPICS version: R3.13.1/ioc/, R3.13.2/ioc/, R3.13.6/ioc/

$CD_SOFT/ref/common/	Common files shared across multiple ESD applications	Include/ Script/ Ora/ Tool/ …	Include files Scripts Oracle scripts Handy tools …
$CD_SOFT/ref/gui/	GUI Files	cud/ disp/ stripTool/	CUD scripts and links. DM/DM2K/EDM displays and scripts. StripTool configs and scripts
$CD_SOFT/ref/matlab/	Matlab Applications	Configs/, script/, src/	

We additionally keep some major components of the control system software in the SLAC-wide "package" area.

Table 9: Present state of Control System components in /afs/slac/package

/afs/slac/package/ subdirectory	What's there now
epics/	???
aida/	All the stuff for Aida except the NewLabour code base, which is in $CD_SOFT/ref/package/aida
cmlog/	???

Additionally, we keep some 3rd party software suites in our own "package" area under $CD_SOFT.

Table 10: Present state of non-CVS related directories in $CD_SOFT

$CD_SOFT (non-CVS) subdirectories	What's there now
support/	Only contains subdirectory package/
support/package	X11R6/ chimera@ -> /afs/slac/package/chimera image_lib/ lesstif-0.93.18/ netbeans/ openMotif/ OpenMotif-2.2.1 …

5.2.1 Problems with the Present Directory Layout

1. No consistency under ref/package. Some directories are pointers to /afs/slac/package, some are local. Some packages here must clearly deliver into our release escalation directories (those which were developed here), but others not.
2. Confusion and inconsistency over CVS and release practices for packages in /afs/slac/package/. For instance, aida should clearly deliver into our escalation procedure directories because we produce it, but cmlog is an outside package with outside makefiles – should it deliver into our escalation procedures? The same question goes for EPICS extensions.
3. The guideline in $CD_SOFT/ref/README for what goes in ref/package is incompletely quantified: what about packages we develop in house but don't make available to the outside?
4. The guideline in $CD_SOFT/ref/README for what goes in ref/app is incompletely quantified: what about applications that are not shared across multiple ESD projects?
5. 3 different areas named "package", all with different (but not absolutely clear) uses.

6. External package suites in 2 places, /afs/slac/package and $CD_SOFT/support/package. One item in support/package points to /afs/slac/package, but not others.
7. Inconsistent versioning scheme in CD_SOFT/support/package: some have a version number in the directory name, others don't.
8. No obvious relationship between PATH and items in $CD_SOFT/ref/package, or $CD_SOFT/support/package.

5.2.2 Design References for Directory Layout.

The planned directory layout will disambiguate "packages" in the following way:

1. $CD_SOFT/cvs/package. Software packages which are in our CVS repository in the module cvs/package, will be those which specifically will *not* be required to follow our Makefile structure and release procedures (an example may be cmlog, if we choose that its makefiles are too complicated for us to re-arrange to work into our release procedures). For such cases, $CD_SOFT/cvs/package will be used as a convenient CVS repository for packages which are otherwise in /afs/slac/package. That is, the "reference" area of $CD_SOFT/cvs/package will be /afs/slac/package. The inode $CD_SOFT/ref/package will be made a symlink to /afs/slac/package. PATH will have to contain separate entries for each needed package in /afs/slac/package.
2. Software packages we create which will be required to use our makefile scheme and release procedures, will be placed in a new repository module $CD_SOFT/cvs/util. These software suites' reference areas will be under $CD_SOFT/ref/util. These suites will deliver to the release directories (see 7.4), and therefore the standard PATH will find their executables.

5.3 Planned Directory Layout

1. Given the design reference above, the following tables summarize the plan for the repository, reference areas and other directories. Items unchanged structurally from what we today from what we plan, are in grey.

Table 11: Planned CVS Repository Directories at $CD_SOFT/cvs/

CVS Repository Directory Area	What will go there	
$CD_SOFT/cvs/package/	cmlog/	
$CD_SOFT/cvs/util/	Aida/ CmdSrv/, Err/, Except/	
$CD_SOFT/cvs/app/	Alh/, artemis/, channelArchiver/, cmdSrv/, err/, fwd_server/, just as presently.	
$CD_SOFT/cvs/ioc/	IOC db and applications, just as presently.	
$CD_SOFT/cvs/common/	Include/ Script/ Ora/ Tool/ …	Include files Scripts Oracle scripts Handy tools …
$CD_SOFT/cvs/gui/	cud/ disp/ stripTool/	CUD scripts and links. DM/DM2K/EDM displays and scripts. StripTool configs and scripts

$CD_SOFT/cvs/matlab/	Configs/, script/, src/

Table 12: Planned Reference Directories of $CD_SOFT/cvs/

Reference Directory	Interpretation	What will go there	
$CD_SOFT/ref/package/	Packages that do not conform to our release procedures (but that we want to cvs).	This is just a soft link to /afs/slac/package. That is, reference areas are directories under /afs/slac/package.	
$CD_SOFT/ref/util/	Packages that do conform to our release procedures. Similar to a "util" shareable on VMS.	Aida/ CmdSrv/, Err/, Except/	
$CD_SOFT/ref/app/	Applications (interactive programs).	Alh/, artemis/, channelArchiver/, cmdSrv/, err/, fwd_server/, just as presently.	
$CD_SOFT/ref/epics[2]/	IOC programs and dbs.	The reference directory of cvs/ioc, containing separate sub-dirs for each EPICS version: /R3.13.1/ioc/, /R3.13.2/ioc/, /R3.13.6/ioc/, just as presently.	
$CD_SOFT/ref/common/	Common files shared across multiple ESD applications	Include/ Script/ Ora/ Tool/ …	Include files Scripts Oracle scripts Handy tools …
$CD_SOFT/ref/gui/	GUI Files	cud/ disp/ stripTool/	CUD scripts and links. DM/DM2K/EDM displays and scripts. StripTool configs and scripts
$CD_SOFT/ref/matlab/	Matlab things.	Configs/, script/, src/, just as presently.	

Table 13: Planned Control System components in /afs/slac/package

/afs/slac/package/ subdirectory	What will be there
epics/	???
aida/	Cvs code base will move from $CD_SOFT/cvs/package to $CD_SOFT/cvs/util
cmlog/	???

Table 14: Planned non-CVS related directories in $CD_SOFT

$CD_SOFT (non-CVS) subdirectories	What will be there
support/	Will be deleted. See Ch 11 for plan for each item now in this directory.
support/package	

[2] The reference dir can not be called "ioc" (to match the cvs module) because of the need to create different subdirectories for each version of EPICS. Under each version directory there is a checked out version of the module called "ioc/".

The Reference directory of any CVS module in $CD_SOFT/cvs/package shall be rooted at /afs/slac/package (so reference directories won't silently be lost in some subdirectory of a package directory, like /afs/slac/package/cmlog/mysecondtry/version3/whatisincvs. This will be enforced by the commit script which does the cvs update, since hopefully it need simply perform the single statement "cvs update $CD_SOFT/ref/package/<cvs/package subdirectoryname>"[3].

5.4 File Versioning Update

Both the cvs repository directory /afs/slac/g/cd/soft/cvs/, and the reference area /afs/slac/g/cd/soft/ref/, must be protected against accidental write or deletion. Having worked out an appropriate scheme using ACLs and protection bits, the cvs commit and update command must be modified, so that:

1. Only a cvs commit operation can write into the CVS repository (any subdirectory of /afs/slac/g/cd/soft/cvs/). Therefore, the cvs commit operation will have to temporarily change the permissions and/or ACLs of the cvs/ directory in order to write into it, and then change them back.
2. The cvs commit operation will additionally perform necessary cvs update operations to update the reference area, (any subdirectory of /afs/slac/g/cd/soft/ref/ which is a CVS checkout area).

5.4.1.1 IOC software file version management

Special handling of the IOC directories will be needed to handle the fact that a single CVS module corresponds to three reference directories:

Repository: `/afs/slac/g/cd/soft/cvs/ioc/`
Reference areas:

- `/afs/slac/g/cd/soft/ref/epics/R3.13.1/ioc`
  `/afs/slac/g/cd/soft/ref/epics/R3.13.2/ioc`
  `/afs/slac/g/cd/soft/ref/epics/R3.13.6/ioc`

See http://www.slac.stanford.edu/grp/cd/soft/cvs/cvs_for_ioc_development.html for description of this mechanism and the CVS update operations needed when committing to cvs/ioc/. Presently, the best thing to do whenever you commit any file to cvs/ioc/, is to cvs update all three reference areas; although one or more of these updates may be redundant. The cvs commit script should take care of performing the cvs update on all 3 directories after changes to cvs/ioc.

5.5 Requirements and Job List

5.5.1 Requirements

1. The cvs update done by cvs commit must specify the module to update, since it must not cause a cvs update of all directories in /afs/slac/package.

5.5.2 Job List

This is an incomplete list of jobs to implement this above plans:

1. Import $CD_SOFT/ref/package/artemis into CVS module cvs/package/artemis.
2. Import $CD_SOFT/ref/app/channelWatcher into CVS module cvs/app/channelWatcher.
3. Move aida to CD_SOFT/cvs/util

[3] The cvs update command would ideally have been "cvs update $CD_SOFT/ref/package", and since ref/package is a symbolic link to /afs/slac/package that would update all packages under /afs/slac/package. But some packages under /afs/slac/packages may well have their own CVS, and this command would have updated them too! So, the command has to be specific about which package under /afs/slac/package must be updated.

4. Move all items in CD_SOFT/support/package to /afs/slac/package. Delete CD_SOFT/support.
5. Modify cvs commit to automatically update ref areas. Needs to manipulate permission bits to allow the update to happen.

TO ADD: lots of other transition plan items.

6 File Protections

This chapter describes the file protections (ACLs and permission bits) of directories /afs/slac/g/cd, afs/slac/g/cd/soft and below.

6.1 cd and cd/soft

This section describes the file protections presently in place and planned for the top level /afs/slac/g/cd and /afs/slac/g/cd/soft directories. Those for cd/soft shall extend generally to directories below cd/soft. However, see below for the plans for CVS repository and reference areas specifically.

6.1.1 Present File Protections at cd and cd/soft

This section outlines file protections in $CD_SOFT in general. See below for the specific protections employed in CVS repository and reference areas.

At present, the ACLs for cd and cd/soft look like this:

Table 15: Present ACLs for cd and cd/soft

Directory	Important ACL Entries	Mode bits	Members
/afs/slac/g/cd	owner-g-cd	?	?
	g-cd	rlidwka	jjo, brooks, saa, luchini
/afs/slac/g/cd/soft	owner-g-cd:soft	rla	saa, greg, luchini
	g-cd:soft	rlidwk	Everyone
	g-cd	As above	As above

The owner organization of the principal ACL entries for these directories is the following. Note, owner-g-cd is not in the ACL of /afs/slac/g/cd/soft, so owner-g-cd:soft is owned by an entry in the ACL of its parent directory.

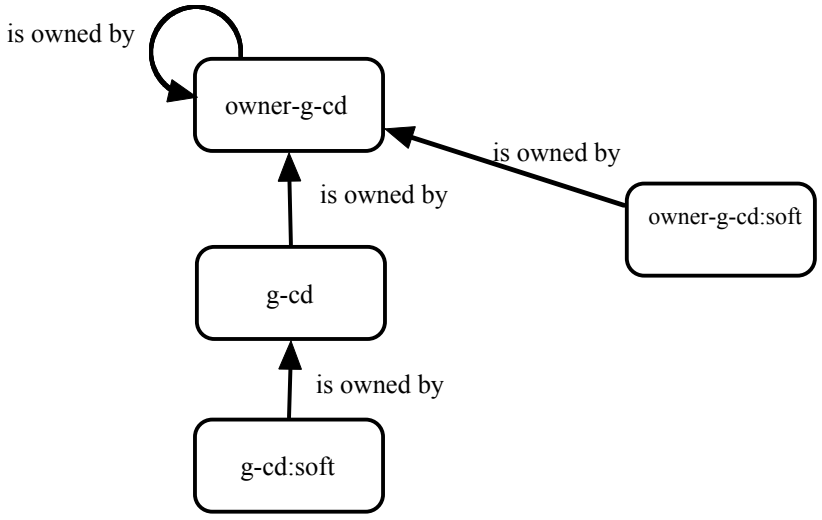

Figure 1: Present ACL entry ownership hierarchy for /afs/slac/g/cd/soft/

6.1.2 Planned File Protections of cd and cd/soft

1. The organization of the ACL in /afs/slac/g/cd/soft should be modified so that the ownership of g-cd:soft is moved from g-cd to owner-g-cd:soft.
2. The ACL entry g-cd should be removed from /afs/slac/g/cd/soft and all subordinate directories, so that protections are managed by g-cd:soft by default.

Then the ACL for our important top level directories would be set to a pattern described by the following table.

Directory	Important ACL Entries	Mode bits	Members
/afs/slac/g/cd	owner-g-cd	?	Unchanged
	g-cd	rlidwka	Remove brooks, saa
/afs/slac/g/cd/soft	owner-g-cd:soft	rla	Remove saa, add zelazny
	g-cd:soft	rlidwk	Everyone

Table 16: Planned ACL at cd and cd/soft

The ACL ownership hierarchy planned for cd and cd/soft is described below under the section on CVS, since most ACL changes have been made to protect the CVS repository, the reference directories, and the production against unauthorized writes, such as from mistaken **gmake**s.

6.2 CVS

This section describes the file protections (ACLs) of directories for file versioning.

6.2.1 Present System of file protections for CVS

Directory	Important ACL Entries	Mode bits	Members
/afs/slac/g/cd/soft/cvs (and all dirs below)	owner-g-cd:soft g-cd:soft g-cd	rla rlidwk rlidwka	As above As above As above
/afs/slac/g/cd/soft/ref (and all dirs below)	owner-g-cd:soft g-cd:soft g-cd	rla rlidwk rlidwka	As above As above As above

Table 17: Present ACLs on CVS directories

6.2.2 Planned Design References for File Protections for CVS

Each of the top level 3 software directories under cvs/ and ref/ (app, util, package, and ioc) will have the same ACL entry pattern. These entries will be replicated for each subdirectory of those directories – that is every app will have an ACL that looks like the example for app/alh's ACL described below, but with "app/alh" replaced with its own name, like util/cmdSrv, e.g. g-cd:soft:cvs:app:alh, g-cd:soft:cvs:util:cmdsrv etc.

Table 18: Planned ACLs on CVS directories

Directory	Important ACL Entries	Mode bits	Members
/afs/slac/g/cd/soft/cvs	owner-g-cd:soft g-cd:soft:cvs	rla rlidwka	Luchini, zelazny, greg
/afs/slac/g/cd/soft/ref	owner-g-cd:soft g-cd:soft:cvs	As above As above	As above As above
/afs/slac/g/cd/soft/cvs/CVSROOT	owner-g-cd:soft g-cd:soft:cvsroot	As above rliwk	As above Everyone
/afs/slac/g/cd/soft/cvs/app	owner-g-cd:soft g-cd:soft:cvs:app	As above rlidwk	As above People who create new apps
/afs/slac/g/cd/soft/cvs/app/alh (and all dirs below)	owner-g-cd:soft g-cd:soft:cvs:app:alh	As above rlidwk	As above People who develop alh

/afs/slac/g/cd/soft/ref/app	owner-g-cd:soft g-cd:soft:cvs:app	As above rlidwk	As above People who create new apps
/afs/slac/g/cd/soft/ref/app/alh (and all dirs below)	owner-g-cd:soft	As above	As above
	g-cd:soft:cvs:app:alh	rlidwk	People who develop alh

6.2.3 Planned ACL Ownership Hierarchy

The planned ownership hierarchy of ACL entries in directories under /afs/slac/g/cd/soft/ in general, including CVS repository and reference directories, is given below. See Figure 1: Present ACL entry ownership hierarchy for /afs/slac/g/cd/soft/, for the present picture of this.

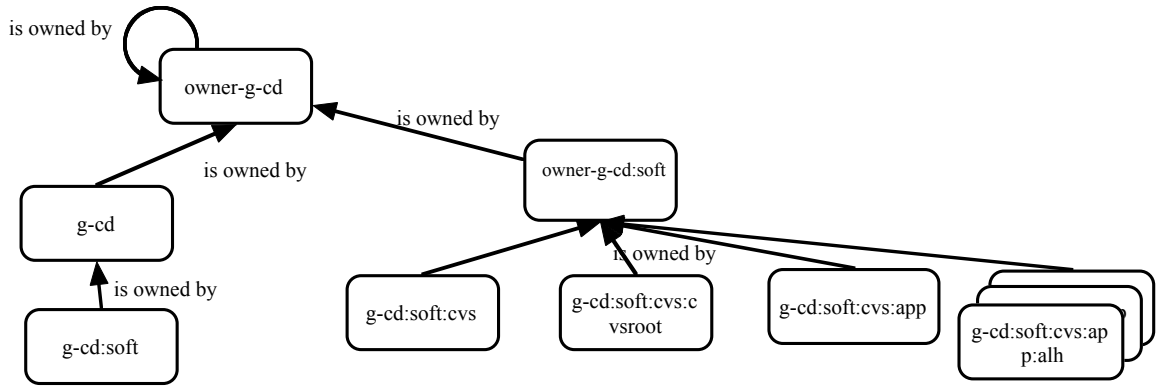

Figure 2: Planned ACL ownership hierarchy

In this picture g-cd:soft is shown as owned by g-cd, but after we have added the ACLs under owner-g-cd:soft, and created the release escalation directories using this scheme, we shall move g-cd-:soft to being owned by owner-g-cd:soft. We're just not going to do that straight away because we don't know what problems it may cause the operational software right now.

6.2.4 Planned Design References for File Versioning (CVS) File Protection

From the SLAC AFS User's Guide: "If the w mode bit is present [in the user bits of the NIS permission bits], anyone with the write and lookup rights on the ACL of the file's parent directory can modify the file; if the w bit is off, no one can modify the file, not even the owner." *This is true even if the file is owned by a different user!* So, if a file in the repository is owned by zelazny, greg can write it if the NIS user permission bit for write is on (ie, -rw-r—r--), and not otherwise. Furthermore greg can chmod u+w the file, even though he is not the owner, as long as he is a member of an ACL entry of the directory that has the w mode bit present (eg rlid**w**k). Given those factors the design for protection of the CVS repository and reference area may be:

1. Both /afs/slac/g/cd/soft/cvs and /afs/slac/g/cd/soft/ref should get a new ACL entry, g-cd:soft:cvs, with permissions rlidwk (see Figure 2: Planned ACL ownership hierarchy).
2. Every file in the repository will be assigned NIS permission bits "`-r--r--r--`". Note, some already have this, but many don't.

Given this model for making temporary permissions changes, the permissions will be manipulated by:

1. The cvs commit precommit operation will set the NIS user permission bit of the repository file "on" (chmod +w) before doing a cvs commit.
2. The cvs commit postcommit operation will set the NIS user permission bit of the repository file "off" (chmod -w) after completing the cvs commit.
3. The cvs commit postcommit operation will set the NIS user permission bit of the reference file "on" (chmod +w) before attempting the cvs update operation on the repository.
4. The cvs commit postcommit operation will set the NIS user permission bit of the reference file "off" (chmod -w) after completing the cvs update operation on the repository.

6.3 Requirements and Job List

6.3.1 Requirements

The ACLs for directories under $CD_SOFT/ref/ will be created piecemeal as they are added to CVS. Specifically, we will not create a system for managing permissions for directories now in ref/ but not in cvs.

CVS manipulation scripts will be kept in cvs/common/script/cvs. Where they are run from depends on whether they're executable or not (see 0).

6.3.2 Job List

1. Implement ACL changes for cd and cd/soft per 6.1.2.
2. Create all ACLs for cvsed dirs (Table 18: Planned ACLs on CVS directories) with ownership hierarchy described in Figure 2: Planned ACL ownership hierarchy.
    a. All repository directories
    b. All reference directories
3. Put ref/common/script in cvs and write Makefile.Host that distributes all these to dev/script or dev/@sys/bin.
4. Run ref/common/script/Makefile.Host to release all scripts from ref/common/script to the release directories.
5. Write cvs commit pre-and post action scripts, to manipulate user write permission bit.
6. Set all files in $CD-SOFT/cvs to -r—r—r—.
7. Set all files in $CD_SOFT/ref to –r—r—r—. *At this point, all callers of executable scripts in ref/common/script, will break because they won't have the x bit set.* Callers should be calling the version in the release directory /dev/@sys/bin.

7 Release Directories and Escalation

This chapter describes the directories that will be used to hold software on development is described, then the procedure for getting software from the development test directory into the place from which it will be used in the production control system is described (that is, what we call the “dev”, “new”, “prod” sequence).

7.1 Directories for Scripts

This section describes the existing and planned system for scripts.

7.1.1 Present System for Scripts

7.1.1.1 Present Directories used for Scripts

Presently production scripts reside mostly in the following directories:

Table 19: Present System of Directories for Scripts

Host Type	Directory	Purpose
Development (CD_SOFT= /afs/slac/g/cd/soft)	$CD_SOFT/script	Your basic dumping ground for scripts not associated with a specific application
	$CD_SOFT/ref/common/script	Not much here right now
	$CD_SOFT/ref/common/setup	Session setup scripts, pathSetup, setAlias etc
	$CD_SOFT/ref/app/<appname>/src	The scripts of each application
	$CD_SOFT/dev/script	The dir into which each application in ref/app, is supposed to deliver “non-executable” (sourced) scripts for either “dev” or “production on development” (see Table 2)
	$CD_SOFT/dev/@sys/bin	The dir into which application in ref/app, is supposed to deliver “executable” scripts for either “dev” or “production on development” (see Table 2)
Production (CD_SOFT= /usr/local/cd/soft)	$CD_SOFT/dev/script	The dir into which “non-executable” (sourced) scripts are copied for use on the Production system
	$CD_SOFT/dev/solaris/bin	The dir into which “executable” scripts are FTPed for use on the Production system

7.1.1.2 Present Release Procedure for Scripts

Although there is a system for “releasing” scripts into production on the Development System, it is not much used at present. This release procedure is oriented towards scripts which are part of an application (so they start out in ref/app/<appname>/src) and releasing them is composed of copying them, on development, to $CD_SOFT/dev/script or $CD_SOFT/dev/@sys/bin. Which of those two directories depends on whether the script is run by sourcing it ($CD_SOFT/dev/script) or by running it in a sub-shell ($CD_SOFT/dev/@sys/bin). The copy is effected by running “gmake” from the reference directory, such as $CD_SOFT/ref/app/<appname>/src. The Makefile.Host of the directory from which the script should be copied, for instance in $CD_SOFT/ref/app/<appname>/src/Makefile.Host, describes which directory each script in that directory should be copied to. If the script is non-executable its name should be added to the SCRIPTS macro in Makefile.Host, and it will be copied to $CD_SOFT/dev/script/ by a “gmake”; if the

script is executable then its name should be added to the SCRPTS macro (no “I”) in Makefile.Host, and it will be copied to $CD_SOFT/dev/@sys/bin by a “gmake”.

Release to production consists of manually “FTPing”[4] the files to the corresponding directory on production.

7.1.2 Design Constraints for Scripts

The plan for scripts is to:

1. Emphasize distinction between “executable” and “non-executable” scripts in order to support a release escalation procedure for executable scripts.
2. Distinguish more clearly between platform dependent and platform independent scripts. This will be needed in the future for Linux on the host side, and artems on the ioc side.

7.1.2.1 Executable/non-executable and the Release Procedure notes for scripts

Executable scripts (those that are started just be typing their pathname) can be found by the shell if they are in the PATH. So there will be a number of directories, tst, dev, new, prod, bck for executable scripts. See below, under code development sequence, to see how executable scripts shall be distributed to those directories. Non-executable scripts (those which are “sourced”) cannot be found by PATH, the source statement must include a complete pathname. For those we shall not (at least at first) have an escalation procedure – they will all be delivered into $CD_SOFT/dev/script (if platform independent) or $CD_SOFT/dev/@sys/script (if platform dependent).

7.1.2.2 Sources of scripts

The principal repository directories for scripts shall be $CD_SOFT/cvs/common/script and $CD_SOFT/cvs/common/setup, for scripts not directly associated with an application or package.

For applications and packages which have their own scripts the situation is a bit more confused: for software we write (in ref/app or ref/util) we might say the standard place for scripts is ref/{app|util}/<name>/script (the script/ directory of the suite), since that is close to being an EPICS standard, but for EPICS extensions, and for packages we get from outside, or packages we write, the scripts may be anywhere in the sub-tree of the suite. Putting these possible locations together we have:

1. $CD_SOFT/ref/app/<appname>/script/
2. $CD_SOFT/ref/util/<utilname>/script/
3. /afs/slac/package/<packagename>/…
4. /afs/slac/package/epics/<version>/extensions/<extensionname>/…

The point here is that scripts may come from all over. So, to control the number of places one has to look for a script we should make as many programs as possible deliver into $CD_SOFT/dev/script, $CD_SOFT/dev/@sys/script, or into the executable script directories, but its going to be impossible to do that for all scripts.

For 1 and 2, we can say by policy, if the program is in our reference area ($CD_SOFT/ref) then its scripts and binaries must be delivered into our release directories. However, it will be very difficult to uniformly modify makefiles of outside programs, so for 3 above, the scripts will probably just have to stay with the program, will not be delivered into our standard script directories for release, and will not go through a release procedure. For 5. (extensions) it’s less clear, do we take responsibility to modify the makefiles of all extensions so as to deliver into our release directories? See 13-2.

[4] Some secure FTP, such as scp or ssh is preferred.

7.1.2.3 Classification of scripts

Platform dependent scripts have to go in sub-directories appropriate to the platform, just so that you can have one script for each platform and find the one you want at runtime. Executable and non-executable scripts have to be distinguished because executable scripts can be easily put through a release escalation using PATH, but non-executable can't.

7.1.2.4 Hard to change release directory name $CD_SOFT/dev/script

Kristi says that very many sourced scripts are hard coded to call others in $CD_SOFT/dev/script, on both development and production, so it will be hard to change the name of the existing primary dir for sourced scripts to be anything else – preferably one that disn't have "dev" in the name. So, let's leave the primary dir for non-executable scripts as $CD_SOFT/dev/script, but additionally create a second dir for *platform dependent* non-executable scripts, $CD_SOFT/dev/@sys/script/.

The above constraints give rise to the following taxonomy and directories. The directories shown are those on Development. The Production directory names will be identical but where "@sys" is used on Development, the Production file-system will use the equivalent platform name (see Table 27: Distribution for Scripts: From/To).

Table 20: Classification of scripts and release directories

Classification	Where will go when released on Development
Platform independent non-executable scripts	$CD_SOFT/dev/script/
Platform dependent non-executable scripts	$CD_SOFT/dev/@sys/script/
Platform independent executable scripts	$CD_SOFT/<release-level>/bin/
Platform dependent executable scripts	$CD_SOFT/<release-level>/@sys/bin/

7.1.3 Design References for Scripts

The following areas must include a Makefile.Host which deliver all scripts into release procedure directories (See Table 20: Classification of scripts and release directories).

Table 21: Directories from which you must copy scripts into release directories via Makefile.Host macro targets

$CD_SOFT/ref/common/script/
$CD_SOFT/ref/common/setup/
$CD_SOFT/ref/app/<appname>/script/
$CD_SOFT/ref/util/<utilname>/script/

Scripts must not be referred to by a pathname that includes these directories (Table 21), they must instead if non-executable, be referred to by their location in $CD_SOFT/dev/script, $CD_SOFT/dev/@sys/bin, or if executable be referred to only by filename so that they can be found by PATH in one of the executable script release directories.

When you add a script to any of these directories you must edit the Makefile.Host in that directory, to add the script's name to the appropriate delivery macro. Non-executable scripts should be added to the Makefile.Host as follows:

```
SCRIPTS_SRCD += {<filename>}                These replace
SCRIPTS_SRCD_HOSTSPEC += {<filename>}       SCRPTS += {<filename>}
```

Executable scripts should be added to Makefile.Host as follows:

```
SCRIPTS += {<filename>}                     These replace
SCRIPTS_HOSTSPEC += {<filename>}            SCRIPTS += {<filename>}
```

Given the above constraints then, we shall use the following list of directories for scripts on the development system.

Table 22: Planned Directories used for scripts on Development

Purpose	Directory	Comment
To be removed	$CD_SOFT/script	Shall be deleted. All contents will be moved to /ref/common/script or ref/common/setup.
CVS	$CD_SOFT/cvs/common/script	Your basic script dumping ground. Will be initially populated from $CD_SOFT/script.
	$CD_SOFT/cvs/common/setup	The repository for ref/common/setup
	$CD_SOFT/cvs/app/<appname>/script	The repository of each application
	$CD_SOFT/cvs/util/<utilname>/script	The repository of each utility
Reference Directories	$CD_SOFT/ref/common/script	Scripts should NOT be executed from this directory[5].
	$CD_SOFT/ref/common/setup	Session setup scripts, pathSetup, setAlias etc. Scripts should NOT be executed from this directory.
	$CD_SOFT/ref/app/<appname>/script	The ref dir for scripts of each application. Scripts should NOT be executed from this directory.
	$CD_SOFT/ref/util/<utilname>/script	The ref dir for scripts of each utility. Scripts should NOT be executed from this directory.
Release Directories	$CD_SOFT/dev/script	The release dir into which scripts from items in Table 21 should deliver *platform independent* "non-executable" (sourced) scripts.
	$CD_SOFT/dev/@sys/script	The release dir into which scripts from items in Table 21 should deliver *platform dependent* "non-executable" (sourced) scripts.
	$CD_SOFT/tst/bin $CD_SOFT/dev/bin/ $CD_SOFT/new/bin/ $CD_SOFT/prod/bin/ $CD_SOFT/bck/bin/	The release directories into which scripts from items in Table 21 should deliver platform independent "executable" scripts. These shall be in the PATH.
	$CD_SOFT/tst/@sys/bin/ $CD_SOFT/dev/@sys/bin/ $CD_SOFT/new/@sys/bin/ $CD_SOFT/prod/@sys/bin/ $CD_SOFT/bck/@sys/bin/	The release directories into which scripts from items in Table 21 should deliver platform dependent "executable" scripts. These shall be in the PATH.

[5] In the phrase "executed from this directory", 'from' is taken to mean the directory containing the script being executed, not the directory from which the execute statement is initiated (the working directory). In short, don't execute scripts in $CD_SOFT/ref/common/script, execute scripts in a CD_SOFT/dev directory.

7.1.4 Questions regarding Directories for Scripts

1. How does Makefile.Host know which directory to put a script in depending on the platform? It must just be putting all scripts in /afs/slac/g/cd/soft/dev/O.solaris/bin if it is run from Solaris. But that won't work if scripts in the same source directory are really platform specific because there is no way to tell Makefile.Host the actual platform to which each script should be installed?

7.1.5 Job List for Scripts

1. Create directories per Table 20: Classification of scripts and release directories.
2. Change Rules.Host so as to replace SCRPTS with 2 different target macros for 2 the non-executable scripts dirs.
3. Change Rules.Host so as to replace SCRIPTS with 2 different target macros for 2 the executable scripts dirs.
4. Add executable scripts dirs per Table 20: Classification of scripts and release directories, to default PATH in pathSetup.csh for both development and production.

7.2 Directories for Binaries and Supporting Files of IOC Software

The present system of directories is described giving both development and production directories; the planned system for directories is described only for the development system in this section, see Ch 8 Distribution, for the plans for where distributed IOC software is located on production.

7.2.1 Present Directories for Binaries and Supporting files for IOC software

The following table summarises the directories recently put in place for IOC software.

Table 23: Present Directories for IOC software on Development

Host Type	Directory	Purpose
Development (CD_SOFT= /afs/slac/g/cd/soft)	$CD_SOFT/cvs/ioc/...	The source of each application and supporting files (such as db).
	$CD_SOFT/ref/epics/<version>/ioc/{bin,db,etc}/	The binaries and supporting files of each application when built. All applications install in to these directories.
Production (CD_SOFT= /usr/local/cd/soft)	$CD_SOFT/ioc/<version>/{bin,db,etc}	The dirs into which binaries and other files are FTPed for use on the Production system

7.2.2 Planned Directories for Binaries and Supporting files for IOC Software on Development

Our plans for IOC software release will be in two stages:

1. In the first stage we will not create any new directories to support an escalating software release procedure. We will just improve software distribution from the existing directories on development to the existing directories on production as outlined in 7.2.1. Specifically /afs/slac/g/cd/soft/ref/epics/<version>/ioc/{bin,db,etc}/ to the corresponding production directories $CD_SOFT/ioc/<version>/{bin,db,etc}. IOC software will continue to be built under ref/. So the directory structure for this stage will remain as present.
2. In the next stage we will support an escalation procedure for IOC software, with the four escalation directories on the development side, but still only one on the production side. See 8 below for how these directories on development correspond to directories on production when distributed.

Table 24: Planned Directories to be used for IOC Software on Development Stage 2

Purpose	Directory	Comment

Purpose	Directory	Comment
CVS	$CD_SOFT/cvs/ioc/...	The source of each application and supporting files (such as db).
Reference Directories	$CD_SOFT/ref/epics/<version>/ioc/{<app>,db,dbd}/	Reference directory.
Release Directories on Development	$CD_SOFT/tst/epics/<version>/ioc/{bin,db,etc}/ $CD_SOFT/dev/epics/<version>/ioc/{bin,db,etc}/ $CD_SOFT/new/epics/<version>/ioc/{bin,db,etc}/ $CD_SOFT/prod/epics/<version>/ioc/{bin,db,etc}/ $CD_SOFT/bck/epics/<version>/ioc/{bin,db,etc}/	The release directories on Development.

7.3 Directories for Binaries and Supporting Files of Host Software

7.3.1 Present Directories used for Binaries for Host Software

Presently production binaries reside mostly in the following directories.

Table 25: Present Directories for Binaries of Host Software

Host Type	Directory	Purpose
Development (CD_SOFT= /afs/slac/g/cd/soft)	$CD_SOFT/ref/app/<appname>/src/O.solaris	The binaries of each application when built.
	$CD_SOFT/ref/package/<pkgname>/bin/	Bindir of packages.
	$CD_SOFT/dev/@sys/bin/	The dir into which application in ref/app, is supposed to deliver binaries on the Development system, for either “dev” or “production on development” (see Table 2)
	$CD_SOFT/dev/javalib/	Where java packages are supposed to be installed
Production (CD_SOFT= /usr/local/cd/soft)	$CD_SOFT/dev/solaris/bin/	The dir into which binaries are FTPed for use on the Production system

Q: Is there a release procedure in existence now for releasing on development (that is, for moving files from $CD_SOFT/ref/app/<appname> to $CD_SOFT/dev/@sys/bin?

7.3.2 Planned Directories for Binaries and Supporting files of Host Software on Development

See Release Procedure (8 below) for how these directories on development correspond to directories on production when distributed.

7.3.2.1 Locations of Host Binary Software on Development

1. $CD_SOFT/ref/app/<appname>/O.solaris
2. $CD_SOFT/ref/util/<utilname>/
3. /afs/slac/package/<packagename>/…
4. /afs/slac/package/epics/<version>/extensions/<extensionname>/…

Recall for 1 and 2, we will say by policy, if the program is in either of these areas (app or util) then its scripts and binaries must be delivered into our release directories. However, since it will be very difficult to uniformly modify makefiles of outside programs, for 3 above, the executable lib and images will probably just have to stay with the program, will not be delivered into our standard release directories, and will not go through a release procedure. For 4. (extensions) it's less clear, do we take responsibility to modify the makefiles of all extensions so as to deliver into our release directories? This table of release directories assumes then only app and util software goes through our release procedure. See Questions (13-2).

Table 26: Planned Directories for Host Software on Development

Purpose	Directory		Comment
CVS	$CD_SOFT/cvs/app/...		Repository for host applications.
	$CD_SOFT/cvs/util/...		Repository for host "packages" that will install into our release directories.
	$CD_SOFT/cvs/package/...		Repository for host "packages" that will **not** install into our release directories.
Reference Directories	$CD_SOFT/ref/app		Reference directory only, building is done in ref.
	$CD_SOFT/ref/util		Reference directory only, building is done in ref.
	$CD_SOFT/ref/package@ -> symlink to /afs/slac/package/ (!)		Reference directories for packages in cvs/package are really in /afs/slac/package/. Building is done in /afs/slac/package/.
Lib Directories	$CD_SOFT/dev/@sys/lib/		Archive libs (.a)
Release Directories on Development	$CD_SOFT/tst/ $CD_SOFT/dev/ $CD_SOFT/new/ $CD_SOFT/prod/	lib/ bin/ pbin/ matlab/ disp/ python/ ora/ include/	The release directories on Development for software from ref/app and ref/util.

7.4 Release Procedure

This section outlines the procedure a programmer will go through to move their software from a development, working, directory to operational use in the accelerator control system.

Note that some software which runs the control system, is run from a Taylored, AFS machine. For instance, 8-pack, NLCDEV and Aida, all run software in this "production on development" way. The production directory in those cases shall simply be the "prod" directory system in $CD_SOFT on development (there will be no distribution stage for that software).

7.4.1 Summary

All software will be built in $CD_SOFT/tst/. It will stay in tst/ during the "dev", "new" and "prod" operations. Tst/ will be cleaned after the "prod" operation. Gateway software will be copied from tst/ to the dev, new and prod directories on the gateway.

7.4.2 CVS commit and update the reference and build directories

A programmer will issue a cvs checkout into their home directory. After they made changes they will issue a cvs commit. The cvs commit will "automatically" cvs update the appropriate $CD_SOFT/ref/ and $CD_SOFT/tst/ subdirectories, depending on the CVS module committed. The ref/ area will normally be read-only, so the cvs commit command must be able to turn off the read-only attribute of the ACLs and/or the Unix permission bits in order to perform its update.

```
cd ~/work
cvs co app/channelWatcher
```

make changes in local directory, then

```
cvs commit                      cvs/ ref/ and tst/ updated by commit
```

7.4.3 "tst"

The "tst" level of release is for building and testing software in isolation. tst software will be in (some subdir of) $CD_SOFT/tst/. That is, a programmer will only be able to use software in "tst" if they proactively add $CD_SOFT/tst/util/bin or $CD_SOFT/tst/app/bin to their own PATH with caddpath.

After doing the cvs commit, the programmer would cd to the appropriate directory in tst/ and issue "gmake tst". gmake tst will cp –R from the current subdirectory of $CD_SOFT/ref/ to $CD_SOFT/tst/, and then issue a gmake from the corresponding directory just copied into in $CD_SOFT/tst/. So gmake tst copies the software from ref to tst and makes it in tst. Note that the subdirectories (such as O.solaris) and files created by the gmake will be created locally under tst/ only (not under ref/ as they are now). The gmake will change permissions in tst/ to permit the build, perform the build, and then set the permissions so that user can no longer write into tst/.

```
cd $CD_SOFT/ref/app/channelWatcher
gmake tst
[gmake tstdist - if production host software ]
caddpath $CD_SOFT/tst/@sys/bin
…
test software
```

7.4.3.1 tst Distribution

Production host (Gateway) software in tst/ may be distributed to the appropriate gateway machine. To do that, the programmer will issue "gmake tstdist" from the tst/ subdirectory (such as $CD_SOFT/tst/app/channelWatcher/ on the development machine). The tstdist target will compile an RDIST command file of items in that were built in tst/ by "gmake tst", and RDIST those from tst/ on development to tst/ on production.

7.4.4 "dev"

The "dev" level of release is for testing software for a longer term than tst, and to test its performance in collaboration with other control system software. Software in "dev" will be in (some subdir of) $CD_SOFT/dev/. The "dev" directories will be on both development and production hosts. However, they will only be in the default PATH on development hosts (as defined in pathSetup.csh), not in the PATH on production hosts. On a production host, a programmer will be able to use software in "dev" if they proactively add some $CD_SOFT/dev/ directory (see above) to their own login session's PATH, on production, with caddpath.

To move to the "dev" level of general release, the programmer will, on development, cd to some directory under $CD_SOFT/tst/, and issue "gmake dev". This will move all "install" files from tst/@sys/bin, tst/@sys/lib/ and the tst db/, etc/ and javalib/ directories to their corresponding directories under dev/. By

"install" files is meant all those files now moved by the "install" target of ref/common/build/RULES.Host (see that file for the list of installed files). The dev/ directories will normally be read-only, and be set writable only for the gmake dev process.

```
> cd $CD_SOFT/tst/app/channelWatcher
> gmake dev
[> gmake devdist  if PROD host software]
… test software
```

7.4.4.1 dev Distribution

Production host (Gateway) software in dev/ must be distributed to the appropriate gateway machine. To do that, the programmer will issue "gmake devdist" from the tst/ subdirectory (such as $CD_SOFT/tst/app/ChannelWatcher/ on the development machine). The devdist target will compile an RDIST command file of items in that were moved from tst/ to dev/ by "gmake dev", and RDIST those from dev/ on development to dev/ on production.

7.4.5 "new"

The "new" level of release is for software which has been fully tested and is now in a probationary period in production. Software in "new" will be in (some subdir of) $CD_SOFT/new/ on both development and production. The relevant directories under $CD_SOFT/new will be in the default PATH on both development and production. On development or production therefore, a programmer will be able to use software in "new" if they simply restart the process making use of that software.

To move to "new", the programmer will cd to the same directory in tst/, and issue "gmake new". This will copy all "install" files from tst/@sys/bin, tst/@sys/lib/ and the tst db/, etc/ and javalib/ directories to their corresponding directories under new/, and then delete those files in dev/. By "install" files is meant all those files now moved by the "install" target of ref/common/build/RULES.Host (see that file for the list of installed files). The new/ directories will normally be read-only, and be set writable only for the gmake new process.

```
> cd $CD_SOFT/tst/app/channelWatcher
> gmake new
[> gmake newdist  if PROD host software]
… restart software
```

7.4.5.1 new Distribution

Production host (Gateway) software in new/ must be distributed to the appropriate gateway machine. To do that, the programmer will issue "gmake newdist" from the tst/ subdirectory (such as $CD_SOFT/tst/app/ChannelWatcher/ on the development machine). The newdist target will compile an RDIST command file of items in that were moved from tst/ to dev/ by "gmake dev", and RDIST those from new/ on development to new/ on production.

7.4.6 "prod" (or "Sweep") and "bck"

Periodically we shall move everything in prod/ to bck/, and then everything in new/ to prod/.

The "prod" level of release is for software which has completed the probationary period in production, and should now be used in the regular operation of the control system. Software in "prod" will be in (some subdir of) $CD_SOFT/prod/ on both development and production. The relevant directories under $CD_SOFT/prod will be in the default PATH on both development and production. On development or

production therefore, a programmer will be able to use software in “new” if they simply restart the process making use of that software.

Periodically one of us will cd to $CD_SOFT/tst/, and issue “gmake prod”. This will copy all “install” files from tst/@sys/bin, tst/@sys/lib/ and the tst db/, etc/ and javalib/ directories to their corresponding directories under prod/, and then delete those files in new/. By “install” files is meant all those files now moved by the “install” target of ref/common/build/RULES.Host (see that file for the list of installed files). The new/ directories will normally be read-only, and be set writable only for the gmake new process.

The last action (if all goes well) of gmake prod will be to do a gmake clean in the tst/ subdirectory. This will clear out the tst/ directory which was built (since tst/ remains populated, through gmake dev and gmake new). Note that this behaviour will not “block” new code from coming in from cvs, since the cvs update command will just put “more recent” files on top of older files. *There is no “block” in this scheme and everything in test at the time of the Sweep is moved to production!*

```
cd $CD_SOFT/tst
gmake prod  (moves software from tst/ to prod/)
gmake proddist (Distributes stuff in /prod to production)
gamke clean (cleans tst/)
```

7.4.6.1 “bck”

“gmake prod” copies everything in prod/ to bck/ before moving files from tst/ to prod/.

7.4.7 “Backshr”

In the existing VMS environment there is a script called Backshr which is able to “de-escalate” given, named, software. This functionality will be hard to reproduce in the unix environment because the escalation is being done through gmake. gmake will have made the determination about what to escalate, and to which directories, based on file revision times and the contents of its install targets.

7.4.8 “Newsoftware.dat”

In the existing VMS environment there is a text file called “newsoftware.dat”, which a programmer is prompted to edit when they release to the “new” level. A similar system can be implemented in gmake. The new and newdist targets would include an instruction like “emacs NEWSOFTWARE”, where NEWSOFTWARE is an environment variable pointing to a file like $CD_SOFT/shr/newsoftware.dat. On VMS this file is quite extensively managed, copying it out to a backup directory and clearing it’s contents once a week and so on. Q: Not sure that is worth-while?

7.5 Job List for Release Procedure

Add parameters to caddpath for common or complex commands; “tst”, “dev”, “new”, “deassign”.

8 Distribution

Production host programs, such as those running on gateways, must be moved to the production host as part of their release. This moving operation is called “distribution”. Since AFS is not available on the non-Taylored production hosts, distribution from development to production may not be as easy as a file copy. In addition to binary distribution may well include such things as EPICS db files and scripts, and the list of such files in a distribution can be quite large. Additionally the distribution has to be done at each “dist” stage described above, devdist, newdist and proddist. These are gmake targets, whose action will be to compile the list of files to be distributed, and move them to the non-Taylored production host.

8.1 Requirements

The Distribution mechanism must satisfy these requirements:

1. Determine which files need to be moved from development to production host (we’re assuming gmake can be made to determine this).
2. Determine whether the given user has permissions to move files to production (knowing the cddev password may be enough to satisfy this requirement).

8.2 What to do about @sys on development hosts?

There is no equivalent of @sys on production (NFS) hosts. The following assumes the distribution program will copy the files from the directory in which they were found, through “pwd”, to a same named directory on production. So, just as exists now, symlinks should be used identically on development and production to translate the fully qualified name of the platform into the logical name, for instance “sun4-solaris2” into “solaris”.

dev directories on Development (afs)	dev directories on Production (nfs)
solaris -> sun4-solaris2 sun4-solaris2 sun4x_57 -> sun4-solaris2 sun4x_58 -> sun4-solaris2	solaris -> sun4-solaris2/ sun4-solaris2/ sun4x_57 -> sun4-solaris2/ sun4x_58 -> sun4-solaris2/

On a Development (AFS) machine, a “pwd” shows the un-aliased name, and that’s the name of the equivalent directory on Production.

[flora05]/afs/slac/g/cd/soft/dev> cd @sys
[flora05]/afs/slac/g/cd/soft/dev/@sys> pwd
/afs/slac.stanford.edu/g/cd/soft/dev/sun4-solaris2

8.3 Distribution of Scripts

Scripts which must be run on a production host should be distributed with the following directory mapping:

Table 27: Distribution for Scripts: From/To

From Development Directory	To Production Directory
$CD_SOFT/dev/script/	$CD_SOFT/dev/script/
$CD_SOFT/dev/@sys/script/	$CD_SOFT/dev/<same-name>/script/
$CD_SOFT/tst/bin/ $CD_SOFT/dev/bin/ $CD_SOFT/new/bin/ $CD_SOFT/prod/bin/ $CD_SOFT/bck/bin/	$CD_SOFT/tst/bin/ $CD_SOFT/dev/bin/ $CD_SOFT/new/bin/ $CD_SOFT/prod/bin/ $CD_SOFT/bck/bin/

From Development Directory	To Production Directory
$CD_SOFT/tst/@sys/bin/	$CD_SOFT/tst/<same-name>/bin/
$CD_SOFT/dev/@sys/bin/	$CD_SOFT/dev/<same-name>/bin/
$CD_SOFT/new/@sys/bin/	$CD_SOFT/new/<same-name>/bin/
$CD_SOFT/prod/@sys/bin/	$CD_SOFT/prod/<same-name>/bin/
$CD_SOFT/bck/@sys/bin/	$CD_SOFT/bck/<same-name>/bin/

8.4 Distribution of Binaries and Supporting files for Production Host Software

Host software which must be run on a production host should be distributed with the following directory mapping. On production, the name of the host specific directory to which the distribution system must deliver (which is designated in the table as "<same-name>") shall be the long name of the host architecture as it is found on the development system with "pwd". That is, the distributing program will literally do a pwd on the development side to find the @sys translation, and create that same named directory on the production side (see 8.1 above).

Table 28: Distribution of Binaries to Production Hosts: From/To

(From) Development	(To) Production
$CD_SOFT/tst/@sys/lib	$CD_SOFT/tst/<same-name>/bin/
$CD_SOFT/tst/@sys/bin/	$CD_SOFT/tst/<same-name>/lib/
$CD_SOFT/tst/@sys/pbin/	$CD_SOFT/tst/<same-name>/pbin/
$CD_SOFT/tst/javalib/	$CD_SOFT/tst/javalib/
$CD_SOFT/tst/disp/	$CD_SOFT/tst/disp/
$CD_SOFT/tst/matlab/	$CD_SOFT/tst/matlab/
$CD_SOFT/tst/python/	$CD_SOFT/tst/python/
$CD_SOFT/tst/ora/	$CD_SOFT/tst/ora/
$CD_SOFT/tst/include/	$CD_SOFT/tst/include/
$CD_SOFT/dev/<as above>	$CD_SOFT/dev/<as above>
$CD_SOFT/new/<as above>	$CD_SOFT/new/<as above>
$CD_SOFT/prod/<as above>	$CD_SOFT/prod/<as above>

8.4.1 Production Host Software - Questions

1. Note that on production the dev/include directory (/usr/local/cd/soft/dev/include/) will now become a distributed release directory, so changes made on production will be overwritten by the next dev release. However, we can choose to say that the distribution system will should not overwrite files which are newer on production than on development – should we?

2. On gtw00opi00 the dir /usr/local/cd/soft/dev/include has include files in it. Why are include files ftped to production?

8.5 Distribution of IOC Software

The release procedure for IOC software will be done in two stages, each with the following mappings for distribution. We will only do the 1st stage for now, since Kristi says we're not ready for a release procedure for ioc software yet.

Note that, since there can only be one instance of some software running in an ioc at any particular time, then there need not be discrete release directories on production.

Table 29: Distribution for IOC Software Stage 1: From/To

(From) Development	(To) Production
$CD_SOFT/ref/epics/<version>/ioc/bin	$CD_SOFT/ioc/<version>/bin
$CD_SOFT/ref/epics/<version>/ioc/db	$CD_SOFT/ioc/<version>/db
$CD_SOFT/ref/epics/<version>/ioc/dbd	$CD_SOFT/ioc/<version>/dbd

Table 30: Distribution for IOC Software Stage 2: From/To

(From) Development	(To) Production
$CD_SOFT/ioc/dev/epics/<version>/ioc/bin/	$CD_SOFT/ioc/<version>/bin
$CD_SOFT/ioc/new/epics/<version>/ioc/bin/	
$CD_SOFT/ioc/prod/epics/<version>/ioc/bin/	
$CD_SOFT/ioc/dev/epics/<version>/ioc/db/	$CD_SOFT/ioc/<version>/db
$CD_SOFT/ioc/new/epics/<version>/ioc/db/	
$CD_SOFT/ioc/prod/epics/<version>/ioc/db/	
$CD_SOFT/ioc/dev/epics/<version>/ioc/dbd/	$CD_SOFT/ioc/<version>/dbd
$CD_SOFT/ioc/new/epics/<version>/ioc/dbd/	
$CD_SOFT/ioc/prod/epics/<version>/ioc/dbd/	
$CD_SOFT/ioc/dev/epics/<version>/ioc/dbd/	Not copied to Production.
$CD_SOFT/ioc/new/epics/<version>/ioc/dbd/	
$CD_SOFT/ioc/prod/epics/<version>/ioc/dbd/	

Note that this outline does not include a /bck directory. $CD_SOFT/ioc/bck/epics/<version>/ioc/… will exist on the development host, and a procedure to move software to /prod from /bck will be created, so that a gmake proddist can then be effected from a development host to distribute that software to the IOC.

8.5.1 Distribution of IOC Software - Questions

1. Is it really true that we shouldn't have more than one directory on Production hosts for IOC software? Surely, just because an IOC can't boot more than one system doesn't mean we shouldn't have a system of release directories on production from which you can boot an IOC.

8.6 Future Source Software Distribution

In the future we expect that the "tst", "dev" and "new" directories on AFS will be mounted on the production systems! Only the "prod" directory will be on NFS. Remote distribution will not be needed as such then for the development directories, and will be done as a "move" operation from new to prod. However, it's expected that it'll take a while to set this up, so we're going to do distribution by RDIST first.

Another possibility is to do software distribution with Taylor. Again, this is a way off.

9 Session environment definition

This chapter describes how the session environment will be defined for the various session (or, loosely speaking, “login”) environments needed in the Unix control system. The unix session environment must be defined for:

1. Developer’s interactive login into a development host
2. Interactive login to cddev on a development host
3. Interactive login to cddev on a production host
4. Non-interactive login processes for startup items on a production host.

9.1.1 Requirements of Session Environment Definition Generally

The four session logins, should to the extent that we can make them, be the same, so that there is a consistency and predictability between login types.

9.2 Developer’s Interactive Login into a Development Host

9.2.1 Present Developer’s Interactive Login Environment Definition

At present some of us use a system built on the HEPiX framework to define our login environment; some source one or more of the scripts written by Kristi for defining the environment necessary for EPICS on the Development System, ENVS.csh (athough Kristi’s recommendation is not to source that from one’s .cshrc), and some use the old SLAC wide login session definition mechanism, and then may or may not be using ENVS.csh to define their EPICS environment.

9.2.1.1 Problems of the Present Developers’ Environment Definition

Basically the problem is inconsistency between HEPiX and the EPICS session environment definition being maintained through ENVS.csh. In particular:

1. The HEPiX framework presently sources $CD_SOFT/dev/script/setAlias.csh, but not the other scripts that ENVS.csh sources. So, some things are defined the same in both systems, and others are defined differently.
2. The HEPiX scripts, as they are written presently, and ENVS.csh, take different views on how the environment should be set up to ensure maintainability: HEPiX’s group_conf.sys.csh takes the view that process global definitions, such as PATH, path, LD_LIBRARY_PATH, and CLASSPATH, should be defined once, and in a single place, so that their definition is obvious. PathSetup.csh (which is called by ENVS.csh to define the process globals), takes the view instead that it should be easy to add or remove elements of the definition of these variables, so that they can be defined and redefined repeatedly.

9.2.1.2 Requirements for Developers’ Interactive Login Environment Definition

1. The basic developer’s interactive login session should be defined for all developers, without them having to take significant steps to define it themselves, so that it can be centrally managed and changes distributed to all developers
2. For the Developer’s interactive login into a development host, the process global definitions, such as PATH, path, LD_LIBRARY_PATH, and CLASSPATH, should be defined in a single place and so be the same for all developers.

9.2.2 Planned Developer’s Interactive Login Environment Definition

1. Make our HEPiX scripts call ENVS.csh, so that EPICS environment definition is done consistently.

2. Move non-EPICS specific definitions out of ENVS.csh and into the HEPiX scripts.

9.2.3 Needed Decisions

1. Should global variables such as PATH, LD_LIBRARY_PATH and so on, be defined once centrally, even on all machines, or should we make it easy to add items to the path of a login? If not centrally defined, then how do we ensure a predictable stable environment on all machines?

9.3 Interactive login to cddev on a development host

TO ADD: Describe present situation, problems and plan. (Kristi)

9.4 Interactive login to cddev on a production host

TO ADD: Describe present situation, problems and plan. (Kristi)

9.5 Non-interactive login processes for startup items on a production host

TO ADD: Describe present situation, problems and plan. (JingChen)

9.6 Accounts

Vxworks on production hosts, should be removed. TO ADD: why?

10 Documentation and Web Support

10.1 Present System of Documentation

Documentation for unix based software is now primarily centered in /afs/slac/www/grp/cd, to which there is a softlink from /afs/slac/g/cd/soft/html/.

10.1.1 Problems of the Present System for Web based Documentation

1. The URL for /afs/slac/www/grp/cd is http://www.slac.stanford.edu/grp/cd/soft/, however, there is no index page at http://www.slac.stanford.edu/grp/cd/soft/, so all documentation in that system of directories has to be found by either knowing the exact URL beforehand, or navigating directories.
2. There is no organization of the directories or web pages under http://www.slac.stanford.edu/grp/cd/soft/.
3. That is, there are no directories such as “requirements”, “user guides” “design” and so on, from which you can search for a document or place a new one. If you are looking for a requirements document for, say cmlog, you don’t have a well defined place to look other than under cmlog.
4. Node names and other sensitive information are in publicly accessible folders.
5. Operationally critical documentation is kept on the Unix web servers, which are hosted on AFS. So that documentation is vulnerable to AFS and network connection failure.
6. It’s difficult to publish web pages. In particular, there are no good web page writing tools on Unix, so web pages hosted on Unix tend to be written using Windows editing tools like FrontPage or Netscape Composer. Saving the html file from Windows onto the AFS file system is fiddly. The easiest way requires that the Windows machine has an AFS client, but that won’t be available soon.
7. There is no user guide to help developers write web documentation and publish it.
8. Most of our web pages don’t have standard headers.
9. The result of these problems is that we don’t, as a group, produce that much documentation, and that hits our productivity.

10.1.2 Proposed Design References for Web based Documentation

1. An index page shall be created at http://www.slac.stanford.edu/grp/cd/soft/. The index page shall create a logical organization of the web pages under /afs/slac/www/grp/cd.
2. We shall look at the files and directories under /afs/slac/www/grp/cd/ to see if there is an organizational scheme we can create to better organize them.
3. We shall create additional subdirectories of /afs/slac/www/grp/cd/ for some functional document types, ie $CD_SOFT/html/req/, $CD_SOFT/html/design, $CD_SOFT/html/ug.
4. Files with sensitive information shall be moved to /afs/slac/www/grp/cd/slaconly/
5. A new IIS web server shall be acquired for operationally critical items, and hosted locally (in the MCC machine room, but administered by SCS).
6. We shall implement WebDAV technology, and related Windows IIS technologies, to support remote publishing, on the Windows IIS server (which is presently available). We shall also get SCS to pursue WebDAV on the Unix web servers.
7. We shall write a user guide to help documentation writers create web pages in a standard way, and guide them through the publishing process.

10.1.3 Transition plan

The specific requirements and job list for our Unix Web Based Documentation are at http://www.slac.stanford.edu/grp/cd/soft/req/unixWebRequirements.html.

11 Filesystem Cleanup and Reorganization.

Items marked "-a" shall be moved to afs/slac/package OR removed if already there:

```
CD_SOFT/
      support/
            package/
                  X11R6   -a
                  chimera -a
                  image_lib       -a
                  lesstif-0.93.18 -a
                  netbeans        -a
                  openMotif       -a
                  openMotif-2.2.1 -a
                  rdist -a
                  regex-0.12      -a
                  screen  -a
                  xpm3.4k -a
      package -> support/package
```

X11R6 : is in afs/slac/package.
      Make sure final location is in LD_LIBRARY_PATH as specified in
      hepix and pathSetup.csh, and check EPICS setup scripts.
chimera: remove link
image_lib : remove, Judy not using anymore
lesstiff-0.93.18 : If we still need this move to afs/slac/package/lesstiff.
netbeans: remove. Email group to use afs/slac/package/netbeans or ffj4, as set up by HEPiX.
openMotif : Q: Do we need both openMotif and openMotif-2.2.1 dirs? If so
            create dev/ and prod/ subdirectories of afs/slac/package/openMotif and
            include a VERSION card in each.
openMotif-2.2.1. See above
rdist   : move to afs/slac/package
regex-0.12 : move to /afs/slac/package/regex
screen : move to /afs/slac/package/screen
xpm3.4k : move to /afs/slac/package/xpm.

12 Glossary

Development Control System (syn. Development, Development System). The machines on which fred software is written and tested.
Development environment. The user's login unix environment, such as the variables and aliases defined as part of their login process, which help them develop software of the control system. The development environment shall be principally defined by scripts run by the HEPiX framework. To define the development control system environment that a developer needs, HEPiX should call ENVS.csh.
Production Control System (syn. Production, Production System). For the software engineering group, this is the operational software of the control system. It is composed of the software which as has been "released" into production. Much of this software is on the "Production Hosts", such as gateways, but also much of the Production Control System is resident on Development Hosts (Taylored, AFS machines).
Production environment. The unix session environment used to run operational control system software. This will be defined differently for different production software: 1. The login environment of cddev on gateway machines/afs/slac/g/cd/soft/dev/script/ENVS.csh. 2. The session environment of startup processes of production hosts. This should be defined by ENVS.csh.

TO ADD:
Gateway
OPI

13 Major Buy-offs

1. Production on Development. Are we prepared to run production software on Taylored/AFS hosts? If so, then the release escalation procedure has to support that, and it will surely become heavily used.
2. Should the makefiles of extensions be modified so as to follow the release procedure for scripts and binaries? Eg, Must makefiles of extensions copy scripts into /afs/slac/g/cd/dev, or are extensions permitted to leave their scripts in ref?
3. Can we possibly get away with not having platform dependent scripts? So rather than having one platform independent directory for each of executable and non-executable scripts, plus n platform specific directories for each of executable and non-executable directories, we just have the two directories, one for each kind:

   afs/slac/g/cd/soft/dev/script
   afs/slac/g/cd/soft/dev/O.solaris/script

4. We don't right now address whether EPICS site and base should be in our CVS repository or its own.
5. We don't right now address whether EPICS extensions should be in our CVS repository or EPICS' own.
6. We don't right now address whether extensions should conform to our makefiles and release mechanism.

13.1 Proposals not included in this document

1. The "app" directory should be split into two, one for "application configuration files", where in fact the application itself may be an EPICS extension and the directory only contains configuration files, like alh, and "applications that we write", where the directory contains source code and makefiles and all that stuff software engineers do. Remember that stuff!

14 Follow-up

The following is a list of items that we might pursue to make the Unix Environment more effective:

1. Write a Developers' Guide. Describes the Unix Development System and how to write code for it.
2. Get AFS on the Production machines on the pep-ii subnet. Then there would be no "Distribution" stage as such, software would be run out of the same directory as it was copied to on development by the release escalation procedure.