DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.
# Table of Contents

## CHAPTER 1

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Observations</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Definitions</td>
<td>2</td>
</tr>
<tr>
<td>1.3.1 Add On Directory Structure</td>
<td>2</td>
</tr>
<tr>
<td>1.3.2 CVSROOT</td>
<td>3</td>
</tr>
<tr>
<td>1.3.3 EPICS_BASE</td>
<td>3</td>
</tr>
<tr>
<td>1.3.4 EPICS_ADD_ON</td>
<td>3</td>
</tr>
</tbody>
</table>

## CHAPTER 2

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool Developer</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Creating an Application Development Area</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Checking Out an Existing Product</td>
<td>4</td>
</tr>
<tr>
<td>2.3 Creating a New Product</td>
<td>5</td>
</tr>
<tr>
<td>2.4 Importing an existing product</td>
<td>5</td>
</tr>
<tr>
<td>2.5 Preparing a new release</td>
<td>5</td>
</tr>
</tbody>
</table>

## CHAPTER 3

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add On CONFIG Tools</td>
<td>7</td>
</tr>
<tr>
<td>3.1 Script Files</td>
<td>7</td>
</tr>
<tr>
<td>3.1.1 get_new_version</td>
<td>7</td>
</tr>
<tr>
<td>3.1.2 set_op_version</td>
<td>7</td>
</tr>
<tr>
<td>3.2 Skeleton Makefiles</td>
<td>7</td>
</tr>
<tr>
<td>3.2.1 Leaf Node</td>
<td>7</td>
</tr>
<tr>
<td>3.2.2 Non Leaf Node</td>
<td>8</td>
</tr>
<tr>
<td>3.3 CONFIG</td>
<td>8</td>
</tr>
<tr>
<td>3.4 CONFIG_SITE_&lt;arch&gt;</td>
<td>8</td>
</tr>
<tr>
<td>3.5 CONFIG_COMMON_&lt;arch&gt;</td>
<td>8</td>
</tr>
<tr>
<td>3.6 CONFIG_COMMON</td>
<td>9</td>
</tr>
<tr>
<td>3.7 HRULES</td>
<td>9</td>
</tr>
<tr>
<td>3.8 RULES</td>
<td>9</td>
</tr>
</tbody>
</table>

## CHAPTER 4

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPICS System Manager</td>
<td>10</td>
</tr>
<tr>
<td>4.1 Creating The Epics Add On Environment</td>
<td>10</td>
</tr>
<tr>
<td>4.2 Adding an Add On Product</td>
<td>10</td>
</tr>
<tr>
<td>4.3 Rebuilding an existing tool</td>
<td>10</td>
</tr>
<tr>
<td>4.4 Replacing An Existing Tool</td>
<td>11</td>
</tr>
<tr>
<td>4.5 WHAT ABOUT MULTIPLE EPICS SITES?</td>
<td>11</td>
</tr>
</tbody>
</table>
CHAPTER 1

Introduction

This document describes source/release control for EPICS add on tools.

1.1 Observations:

1. At APS we have had problems managing changes to individual tools. Locally developed tools have been evolving at a rather rapid rate. Operators have experienced problems because of using the wrong version of a tool.

2. Up to release 3.11 all EPICS supplied tools were part of a release. As the number of tools expands this becomes hard to manage. Starting with release 3.11.6 we are unbundling tools, starting with all Channel Access client tools.

3. We are using cvs rather than sccs.

4. Developers have complained about imake.

1.2 Overview

USER

The user automatically executes the latest version of each add on tool that has been installed by the epics system manager. The user is also able to selectively choose other versions of add on tools.

These features are implemented via two environment variables: PVT_ADD_ON and EPICS_ADD_ON. The following is done: The user has to add a command to his/her start up script that executes a script supplied by the epics system manager. This script does the following:

1. If environment variable EPICS_ADD_ON is not defined, it is defined.

2. The users path is modified as follows:
   a. If PVT_ADD_ON is defined then $PVT_ADD_ON/bin is searched first.
   b. EPICS_ADD_ON/bin is searched second.
   c. The rest of the previous path is searched.

The script is stored in /usr/local/epics/cshrc or /usr/local/epics/profile.
Tool Developer

The following support is provided:

cvs        Cvs is used to manage all source files
add_on     A developer can develop in a private add_on directory.
make       Makefiles are used to build tools. By following a few simple rules
            makefiles that work on multiple Unix architectures and for arbitrary
            epics releases can be created. In addition it is possible to easily use
            other epics add on tools.

Epics System Manager

The epics system manager is responsible for maintaining a few configuration files that
contain site specific definitions. Other configuration files are provided that are main-
tained by the epics community.

1.3 Definitions

1.3.1 Add On Directory Structure

    add_on
    src
    config
    <tool>
    ...
    bin
    include
    lib
    man
    doc
    ...

A system wide add on area (EPICS_ADD_ON) is provided by the epics system man-
ager. In addition each tool developer can create one or more private add on areas.

bin

This directory contains the executables generated from various tools. The makefile for
each tool provides the option of “make install”. The installation is done by placing the
new version into bin and setting a soft link to reference the latest version. Assume the
tool is named <tool> and the last installed version is <tool>.00019. The result of make
install is to create the files

    <tool>.00020
    <tool>

where <tool> is a soft link to <tool>.00020
NOTE: when make install is performed in the EPICS_ADD_ON area users will see the newly installed tools immediately.

NOTE: We are still deciding if we can find a nice way to base the release numbers on the release tag.

**include**
This directory contains any include files generated by tools.

**lib**
This directory contains any libraries generated by tools.

**man**
This directory contains Unix man pages.

**doc**
This directory contains tool release notes and other tool related documentation.

**src/config**
This contains a set of configuration files that are included directly or indirectly in the makefiles which build tools. The various files are described later in this document.

**src/<tool>**
Each tool has its own directory, which can be an entire tree. Developers should be wary of constructing large trees. This means that many tools are being bundled into one which is normally a bad idea.

### 1.3.2 CVROOT
This is the cvs repository for epics. The cvs documentation should be consulted for a description of cvs.

Version Management with CVS, Per Cederqvist

A postscript version of this manual is stored in $EPICS_ADD_ON/doc/CVS_man.ps.

NOTE: The individual tools are located at $CVSROOT/add_on.

### 1.3.3 EPICS_BASE
A release of epics unbundled software. This document will not discuss epics releases.

### 1.3.4 EPICS_ADD_ON
The epics system manager will build all tools that users expect to find in EPICS_ADD_ON
CHAPTER 2  Tool Developer

2.1 Creating an Application Development Area

1. Define the environment variable CVSROOT.
2. Create the directories:
   
   add_on
   src
   bin
   include
   lib
   man
   doc
   ...

   You will probably want your PVT_ADD_ON environment variable to point to here.

   The directories include, lib, man, etc. are created as necessary.
3. cd to src and execute the command:
   
   cvs checkout -d config add_on/config

   NOTE: The only config file you should ever need to edit is CONFIG. This file allows you to override definitions obtained from other config files. Any changes you make to this file should be considered temporary, i.e., when you finally prepare a new release build with the original CONFIG file. NEVER commit any config file without consulting with your epics system manager.

2.2 Checking Out an Existing Product

1. cd <add_on>/src
2. cvs checkout -d <prod> add_on/<prod>
3. cd <prod>
4. touch .DEPENDS
5. make depend

   At this point you are ready to make changes to the product.
2.3 Creating a New Product

1. cd <add_on>/src
2. mkdir <prod>
3. cd <prod>
4. touch .DEPENDS
5. cvs import -m "Creating" add_on/<prod> <prod> R0 <prod> R0
6. cvs add <filename>
7. Add the cvs keyword $Log$ someplace in each source file.

The add command is issued for each file to be placed under cvs control. Don’t forget the Makefile.

2.4 Importing an existing product

In the source directory where the product currently exists execute:

    cvs import -m "Importing" -d <prod> add_on/<prod> <prod> R0 <prod> R0

WARNING: Make sure you have a clean directory structure before invoking import because it recursively descends all directories from where it is invoked.

If the source files were previously maintained via sccs, it is best to import a version obtained via the sccs get command. If the sccs keywords such as %W% and %G% were used then the file will contain a record of the sccs version that was converted to cvs.

Perform the following steps:
1. cd <add_on>
2. cvs checkout <prod>
3. cd <prod>
4. touch .DEPENDS
5. Create a Makefile
6. cvs add Makefile
8. Add the cvs keyword $Log$ to each source file. If a Modification log was manually maintained previously, just add the $Log$ after the last log entry.
9. make depend
10. make
11. make install

2.5 Preparing a new release

1. In your application area prepare and test the new version.
2. Execute the commands:
   cvs update
   cvs commit
   cvs tag <prod>_R<x>

3. Notify the epics system manager that he/she can generate a new release.
CHAPTER 3  Add On CONFIG Tools

The files residing in <add_on>/src/config make it easy for tool developers to create makefiles that work on multiple Unix architectures.

3.1 Script Files

3.1.1 get_new_version
This command file is used by make install to find the latest version of an installed tool in $EPICS_ADD_ON/bin.

3.1.2 set_op_version
This is used to select the specific executable version of an EPICS add_on program that will be used on future command invocations. It is meant to be run in the FVT_ADD_ON directory.

3.2 Skeleton Makefiles

3.2.1 Leaf Node
The following is a skeleton makefile for a leaf add-on product directory:

ADD_ON = ../..
include $(ADD_ON)/src/config/CONFIG
#define USR_xxx definitions as necessary
SRCS = <define all sources>
OBJS = <define all objects>
#Give the name of one or more products to build
PROD = <products>

include $(ADD_ON)/src/config/RULES
#define a build rule for each product
3.2.2 Non Leaf Node

The following is a skeleton makefile for a non-leaf add-on directory:

```
ADD_ON = ../..
include $(ADD_ON)/src/config/CONFIG
SUB_DIRS = <sub directories containing makefiles>
include $(ADD_ON)/src/config/HRULES
```

3.3 CONFIG

This file contains the following:

```
ARCH = sun4
include $(ADD_ON)/src/config/CONFIG_SITE_$ARCH
# override CONFIG_SITE_$ARCH definitions here
include $(ADD_ON)/src/config/CONFIG_COMMON
# override CONFIG_COMMON definitions here
include $(ADD_ON)/src/config/CONFIG_COMMON_$ARCH
# override CONFIG_COMMON_$ARCH definitions here
```

This file is the only config file a tool developer should ever have to modify. Remember that the epics system manager is responsible for this file. Tool developers should never commit changes to this file.

A tool developer can change the ARCH in case he/she want to build for another architecture. Definitions coming from the other include files can be overridden by adding definitions after the appropriate include.

3.4 CONFIG_SITE_<arch>

This file contains site specific definitions for things like X11_LIB, etc. Two epics related definitions that must be assigned values are:

- **EPICS_BASE** This is the location of the release of the epics bases system used to build the various tools. For example this is where the channel access client library is located.

- **EPICS_ADD_ON** This defines the location of the epics add on area. This is defined so that a tool developer can have access to other tools that have been built in the epics add on area.

This file is the responsibility of the epics system manager.

3.5 CONFIG_COMMON_<arch>

This file contains architecture dependent definitions that are common to all epics sites. Thus this file is maintained by the epics community.
3.6 CONFIG_COMMON

This file contains definitions, common to all epics sites, that are architecture independent. It is maintained by the epics community.

3.7 HRULES

If an add on tool is composed of a source subtree rather than a single source directory then each non leaf directory must contain a makefile that defines the symbol SUB_DIRS, which is a list of the immediate subdirectories of the node.

3.8 RULES

all::

Builds $(PROD) and $(LIBNAME)

depend::

Creates a file .DEPEND containing dependency rules.

install::

Installs the current version on the tool in $(ADD_ON)/bin. For each file specified in $(PROD), it does the following (in bin):

1. Locates the latest version already in bin. It adds 1 to the version number.
2. Copies the file to <file>.<version>.
3. Creates a soft link from <file> to <file>.<version>

clean::

Removes all .o and .a files in addition to all $(PROD) files.

Note that of the above rules are followed by :: meaning that it is possible to add to each rule in each makefile.
CHAPTER 4  

EPICS System Manager

This chapter explains the duties of the epics system manager related to managing the add_on products.

4.1 Creating The Epics Add On Environment

1. Create the file CVSROOT. If you are supporting multiple architectures (sun4, hp, etc), there should be only one cvs repository. Thus CVSROOT can be a soft link. It is advisable to have a cvs repository devoted exclusively to EPICS and it’s add on products. The cvsinit command must be issued to initialize the repository.

2. Install the epics base system.

3. Install the epics add on products as follows:
   - Create the directory EPICS-ADD_ON.
   - cd to $EPICS-ADD_ON.
   - Untar the add_on products file.
   - cd to src.
   - perform a cvs import on config and on each product desired.
   - rm -fr *
   - perform cvs checkout on each product.
   - Modify config/CONFIG and config/CONFIG_SITE_<arch>
   - perform a make and make install on each product.

4. Provide a script for users to include in their .cshrc or profile. It adjusts their path to include PVT_ADD_ON and EPICS_ADD_ON.

4.2 Adding an Add On Product

For add_ons that already exist but are not under cvs control use the cvs import command. After importing the product, follow the rules for rebuilding an existing product (including checkout).

4.3 Rebuilding an existing tool

Normally if you are rebuilding an existing tool, it is because a local developer has made changes. Assume that the changes have been checked and the developer has issued a cvs commit command.

1. cd EPICS_ADD_ON/src
2. `cvs checkout <tool>`. This will normally include the `-r` flag.
3. `cd <tool>`
4. `make`
5. `make install`

### 4.4 Replacing An Existing Tool

Explain how to update tool obtained from elsewhere

### 4.5 WHAT ABOUT MULTIPLE EPICS SITES??