

What are we Trying to Achieve

We need a software release policy.

Please keep in mind that whatever that policy will be, we can make it work with SRT or auto-tools, with static or dynamic libraries.

How Software Release Tools (SRT) work

- It is a system meant to maintain software releases for a project with a large number of developers
 - Written at FNAL and judging by Google search results, seems to be used exclusively by HEP collaborations (ATLAS, BaBar, MINOS, CDF, probably others)
- A number of releases can be installed (and potentially worked on) on one system
 - Our software development policy may define “static” releases, i.e. releases that are compiled once and are never changed or updated
- An SRT installation expects a public (base) release
 - Default libraries and executables live there
 - Development release would be regularly updated to the latest-greatest source
- A developer would create a private (test) release and check out only the packages that (s)he is working on, so a typical “session” to modify a package would look something like
 1. Do necessary modifications, compile the package
 2. `addpkg MippIo`
 3. Test that the software still works after modifications
 4. `cvsc commit MippIo`
 5. `rmpkg MippIo`
- e907daq and PPC's would use tagged releases for production running, that is `daq_start` script would do something like

```
srt_setup SRT_BASE_RELEASE=<release name>
```

before starting all the executables
- Developers could run the DAQ with development release, where any one computer could either run production or development release
 - Example: Mike is trying to debug TPC code and he is not sure whether it's the hardware or the software that causes the crash, he could run one quadrant in development release and the rest in production release.
- Developers could also run DAQ with custom release
 - Example: I like the present stable release, but I don't like development version of MippIo library, and I want to work on RICHReadout. I would branch off the stable release, get development version of RICHReadout (in my private area), and run that way. Note that the stable release remains unchanged.

Comparison of SRT and auto-tools

<i>Feature</i>	<i>SRT</i>	<i>Auto-tools</i>
Tagging	CVS	CVS
How to select a new release	Environment variables (through srt_setup)	Soft links OR environment variables (our choice)
Installation procedure	<ol style="list-style-type: none"> 1. Tag new release 2. Import and srt_setup the release 3. Compile 	<ol style="list-style-type: none"> 1. Tag new release (no more -version-info flag) 2. Import and ./configure the release into new directory 3. Compile and install
Advantages	<ul style="list-style-type: none"> ✓ Offline software uses SRT 	<ul style="list-style-type: none"> ✓ All code lives in one directory (no chance of confusion due to incorrect srt_setup) ✓ More natural automatic configuration of build (VME ppc vs. others)
Disadvantages	<ul style="list-style-type: none"> ✗ Introduces lots of environment variables (changes behavior of make, etc) 	<ul style="list-style-type: none"> ✗ Painful to link to packages from offline (currently MippXML and Config)