SCons Command Line Tag Collector

Status

Script tagCollector.py has been committed to CVS (grits-tools/python/tagCollector.py). This page has been updated for version **1.27** for which the bulk of the work is done in batch under certain circumstances and which has new options, --submit_batch and --log.

The most recent version of the tagCollector.py script should be used. On SLAC linux, the most recent version is maintained at

/afs/slac.stanford.edu/g/glast/ground/scripts/tagCollector.py

Warning

Since tag collecting is not an atomic process, a poor internet connection can cause failure. As of version 1.22, in order to avoid such problems and to preserve output from actual tagging (--real option supplied), commands including the --real option submit a batch job. Therefore, such commands must be issued from a SLAC machine for which batch submission is enabled. There is no such restriction for --fake.

Interface changes from previous versions

Version 1.27 changes default output to email. If --log option is used, output will go to specified file
Version 1.22 adds new options --log and --submit_batch. Also, the --real option now implies --verbose. See details below.
Version 1.19 adds a new option, --branch, making it possible to create new tags along a branch.
Version 1.13 accepts the value NONE for the --parent option. In this case, --new=HEAD is not allowed, but you can specify a positive integer, e.g. --new=123 in order to create a tag for HEAD 123.
Version 1.11 has the additional flag --fake. The options --real and --verbose are now also true flags, taking no argument.

Functionality

Currently handles these activities:

1. Make a new release tag which is identical to current HEAD build
2. Make a new HEAD, not based on any previous HEAD.
3. Make a new HEAD by changing some package tags, adding new packages or removing packages as compared to current HEAD
4. Make a new release tag along a branch. The only files which need to be on the branch are the top-level files packageList.txt, ChangeLog and, if it exists, containerNotes.txt.

There must be a file packageList.txt, listing all packages and versions belonging to the build at the top level (e.g. ScienceTools-scons/packageList.txt). It is used to determine what gets the new tag in cases 1., 2. and 4. above. For 3., tagCollector.py will update the packageList.txt belonging to the old HEAD appropriately .

At some point would like to add ability to make a new tag along a branch by updating, adding or removing packages relative to an existing tag on that branch.

Call Interface, Usage

Here is the help output:

```
jrb@noric11 $ python tagCollector.py --help
Usage:  tagCollector.py [options] container, e.g.
python tagCollector.py --new=HEAD ScienceTools
python tagCollector.py --new=HEAD --real ScienceTools
python tagCollector.py --parent=NONE --new=35 GlastRelease
python tagCollector.py --new=ScienceTools-09-24-00 ScienceTools

CVS Version: $Id: tagCollector.py,v 1.27 2011/09/01 16:14:36  jrb Exp $
See also https://confluence.slac.stanford.edu/x/GgFyBQ

Options:
  -h, --help            show this help message and exit
  -p PARENT, --parent=PARENT
                        HEAD or NONE. Someday also (branched) release tag to
                        act as base [default: HEAD]
  -n NEWTAG, --new=NEWTAG
                        Specify tag to be created: HEAD (interpreted as next
                        available HEAD), positive integer (number of HEAD), or
                        literal tag name. Required option
```

```
 -b BRANCH, --branch=BRANCH
                       Bracnh along with to make tag. packageList.txt,
                       containerNotes.txt and ChangeLog will come from this
                       branch. [default: MAIN]
 -u UPGRADEFILE, --upgrade_list=UPGRADEFILE
                       File listing packages to upgrade, add or remove; used
                       only when creating a new HEAD [default: upgrade.txt]
 -r, --real             Really do it. Job is submitted to batch; email will be sent to
                       relmanlist. Implies --verbose [default: not set]
 -f, --fake             Just print what we *would* do [default: yes]
 -v, --verbose          If set show all cvs command output [default: not set]
 -s, --submit_batch    Forces submit to batch even if --fake
 -l LOGFILE, --log=LOGFILE
                       Batch log file path [default: no log file; send email instead]
 --internal
```

In this version PARENT can only be HEAD on NONE.

Constraints on arguments and options

- The single argument is required and must be equal to one of the known container names, currently ScienceTools, GlastRelease, CHS, ASP, TMineExt, GRBAnalysis, and Toy.
- When --branch has value other than "MAIN", must also specify --parent=NONE
- When --branch has value other than "MAIN", --new cannot have value HEAD nor can it be an integer; it must be the complete literal tagname to be applied.
- The only acceptable literal tagnames if --branch is unspecified or has value "MAIN" have the form ***containerName*** -dd-dd-dd or ***containerName*** -dd-dd-dd-wwdd where containerName is one of the known container names, each d is a digit and each w is a letter, digit, or the underscore character. Examples are TMineExt-01-02-03 or GlastRelease-17-35-23-gr03
- If --branch has been specified and does not have value "MAIN" the requirements on tagnames are slightly looser: in the above templates one may insert any alphanumeric string immediately after ***containerName*** (before the first hyphen). Example: GlastReleasePass7-17-35-04-gr01

Production tagging

As of version **1.22**, if --real has been specified

- the --verbose flag will automatically be set
- the command will be submitted to batch
- output will be emailed to the caller and to relmanlist
- output will also be written to a file. By default this file is $HOME/tagCollector.log. An alternate path may be specified with the --log option.

If --fake is in effect, the command normally will be run interactively, as for earlier versions. However it is possible to submit to batch for --fake as well by specifying the --submit_batch option. In this case the job will be submitted to batch as above and the output will be written to a file and emailed, however it will only be emailed to the caller, not to relmanlist.

Use of upgrade.txt

This file is only used in the case where you are making a new HEAD from the previous version. Otherwise, the list of packages to be tagged will be taken from the appropriate already-existing version of packageList.txt.

Here is a sample upgrade file:

```
# Upgrade for ScienceTools
// Comments can start with a '#' as above or with a slash
# The second column should be the full <cvs-path-to-package>

 u  dataSubselector dataSubselector 08-02-01
 u  tip             tip-02-17-01

#  One can also add or remove packages from HEAD, e.g.
#  a  myNewPackage   myNewPackage-00-00-01
#  r  obsoletePackage
```

and here is part of the output when it was used as the value for upgrade_list.

```
jrb@noric11 $ python tagCollector.py --new=HEAD --fake ScienceTools
tagCollector.py  was invoked with argument  ScienceTools  and options
parent= HEAD
new= HEAD
upgrade_list= upgrade.txt
branch= MAIN
for_real= False
verbose= False

CVS version:  $Id: tagCollector.py,v 1.20 2011/08/01 17:42:02 jrb Exp $

about to issue command
cvs -d /nfs/slac/g/glast/ground/cvs  rlog  -h ScienceTools-scons/SConstruct
lastHeadTag is:  ScienceTools-HEAD-1-896

about to issue command
cvs -d /nfs/slac/g/glast/ground/cvs  rlog -h ScienceTools-scons/packageList.txt

about to issue command
cvs -d /nfs/slac/g/glast/ground/cvs  rlog -h ScienceTools-scons/packageList.txt

about to issue command
cvs -d /nfs/slac/g/glast/ground/cvs  co -r ScienceTools-HEAD-1-896 ScienceTools-scons/packageList.txt
Found changed package:  tip
Found changed package:  dataSubselector

about to issue command
cvs -d /nfs/slac/g/glast/ground/cvs update -A ScienceTools-scons/packageList.txt

about to issue command
cvs -d /nfs/slac/g/glast/ground/cvs commit -m "tagCollector.py updating pkg list"
        ScienceTools-scons/packageList.txt
Just fooling; it wasn't for real

about to issue command
cvs -d /nfs/slac/g/glast/ground/cvs  rlog -h ScienceTools-scons/containerNotes.txt
No containerNotes.txt file found for  ScienceTools-scons

about to issue command
cvs -d /nfs/slac/g/glast/ground/cvs rtag ScienceTools-HEAD-1-897 ScienceTools-scons/ChangeLog
Just fooling; it wasn't for real

about to issue command
cvs -d /nfs/slac/g/glast/ground/cvs rtag ScienceTools-HEAD-1-897 ScienceTools-scons/packageList.txt
Just fooling; it wasn't for real

about to issue command
cvs -d /nfs/slac/g/glast/ground/cvs rtag -r xmlBase-05-05-10 ScienceTools-HEAD-1-897 xmlBase
Just fooling; it wasn't for real

about to issue command
cvs -d /nfs/slac/g/glast/ground/cvs rtag -r astro-03-11-11 ScienceTools-HEAD-1-897 astro
Just fooling; it wasn't for real

?(followed by many similar lines tagging all the individual packages comprising ScienceTools.
The last one is always SConsFiles:)

about to issue command
cvs -d /nfs/slac/g/glast/ground/cvs rtag -r SConsFiles-00-12-01 ScienceTools-HEAD-1-897 SConsFiles
Just fooling; it wasn't for real
```

If the option --real had been used, the cvs commands appearing in the output all would have been executed. Since it was False for this run, only the read-only cvs commands, like the initial cvs rlog, were executed.

The overall strategy in the make-new-HEAD-from-previous case shown above is

- check out version of packageList.txt belonging to parent tag
- merge changes requested in upgrade file into packageList.txt; commit new packageList.txt

- tag top-level files (packageList.txt, ChangeLog, containerNotes.txt) belonging to no subpackage with new HEAD tag
- tag each package listed in packageList.txt with new HEAD tag, saving SConsFiles for last

The function to make a release tag from HEAD is similar except that no new packageList.txt file needs to be generated. Just use the one from the parent tag as is. For the case --parent=NONE, use the most recent packageList.txt (along the specified branch if the --branch option has been used).

packageList.txt

This file lists all packages comprising the container, somewhat analogous to the CMT requirements file for the container package. It is not normallly edited by hand.

> **Interface change**
>
> As of version 1.10 of tagCollector.py, this file is a **required** part of every container package.

<cvs-path-to-package> <tag>

e.g.,

flux flux-08-40-07

The file may also contain blank lines or comments (any line beginning with #), which are ignored.

The file is not used as input to the tagging operation, but it is updated to reflect the changes specified in the user's upgrade file, thereby providing a record of exactly which tags are in each head build or release tag.

containerNotes.txt

If the container has a file containerNotes.txt at the top level, as of version 1.9 (committed to CVS March 10, 2011) the tagCollector script will give it some special handling for HEAD tags: the most recently committed version of containerNotes.txt will get the new HEAD tag. (The file isn't required; if it doesn't exist, tagCollector will put out an informational message but will keep going.) The intent is to give the owner of the container a place to document the contents of the new HEAD. There are no constraints on the file, but it is probably most useful as a log (new information is appended or inserted; old entries are left alone) containing at a mininum the name or user id of whoever is making the new tag and a timestamp. When creating release tags, tagCollector treats containerNotes.txt just like everything else: everything with the most recent HEAD tag will also get the new release tag.

Submitting to Batch

After logging onto a central SLAC machine, issue a command of this form:

bsub -qlong -o *myTagCollectorOutput.log* **python /afs/slac/g/glast/ground/scripts/tagCollector.py** *options containerName*

for example

bsub -qlong -o newHeadOutput.log **python /afs/slac/g/glast/ground/scripts/tagCollector.py** --new=HEAD --real --verbose ScienceTools

It's a good idea to include the --verbose flag, as in the example above, to save as much information as possible.

Sticky Points

Certain assumptions are hard-coded in the script when, ideally, they should come from the RM database: for example, exact format of tags. These things don't change very fast, if at all, but if they do, someone will have to remember to update the script.

~~The way CVS tagging is currently set up (in particular, the script $CVSROOT/CVSROOT/tagger), you would have to be logged in as glast or glastrm to run the script in "for real" mode.~~

~~**Proposal:** Modify tagger script to allow creation and deletion of tmp tags with names satisfying a pattern. Should be something like HEAD pattern except replace HEAD with string HEADtmp, possibly HEADtmp<stuff> where <stuff> is an arbitrary and optional alphanumeric string if, e.g., we want to distinguish different attempts of creating HEAD of a particular version.~~ ~~**DONE**~~ ~~but, as of 1.11, temp tags are no longer used.~~

Still To Do

- ~~test adequately Creating next HEAD tag has been tested for Toy scons~~ **DONE** Testing deemed adequate as of March 10, 2011.
- ~~add as output a file which lists all the packages and their tags belonging to the new tag just created. The file probably should go at the top level (e.g., in the ScienceTools scons directory). One possibility in the new HEAD case is to create it after the temp. tag has been made, submit to CVS, move temp tag to the new version, then make the final new HEAD tag. However, if we plan to also support tag collecting from RMViewer, it would have to go through a similar procedure.~~ **DONE** The file is called packageList.txt.
- ~~add ability to make release tags along a branch~~ **DONE** As of version 1.19.