Table of Contents

Code Archaeology, What changed between different versions?

When looking to see what's changed, it is often a very difficult task. It's not really possible for the release manager to know precisely what modifications future users will consider important for their use case, but for sure we try and make good guesses.

We seem to have to do this archaeology quite often often, also, so core software have provided several tools to help with this. Of all the provided tools, the svn interaction tools are the most powerful and reliable.

Table of Contents

Tool summary

Tool	Explanation	Example
release webpage	Every project has a website which amongst other things has some pretty release notes formatting.	Latest Brunel release
doc/release.notes	Every package has a doc/release.notes file, which if it has been filled correctly should give you a good idea what has changed	Tf/TrackSys head
Tag Collector	The tool intended for use by release managers to pick up changes for releasing.	Rec v13r1
svnpath	get the svn path to a given package version, very useful for svn diff statements	`svnpath Hlt/Moore v14r8`
svn diff	SVN low-level command, will really diff two svn locations, for example, two tagged packages.	`svn diff `svnpath GaudiConf v11r0` `svnpath GaudiConf v10r0``
svnDiffTags	Compare the contents of two tagged versions of a package or project and its dependent packages	`svnDiffTags Rec v11r0 v12r0`
svnProjectDeps	Get a list of all package or project and versions used by a given release. Use "-P" to see only the projects, use "-f Something" to show only matching regex.	`svnProjectDeps Rec v11r0`
SVN web	Our code repository is browsable online, and the Trac interface is very powerful at performing diffs etc.	TRAC

Recommended process:

Process:

- Contact your release manager, to see if they have any input to your query.
- Use `svnProjectDeps -P` to evaluate which stack your two software versions are on, or `svnProjectDeps -f "myPackage"` to concentrate on a specific package.
- Use `svnDiffTags AProjectOrPackage v1 v2` to see all the modifications, and hone in on interesting modifications. This will also display the release notes for you to read through.
- Use the tag collector to see whether the release manager had any other input into changes made.
- Use the SVN web browser to dig even further, for example starting specifically at your tagged version. One useful trick is to click on the revision number at the highest granularity that you want to compare: e.g. if you click on the current revision next to Rec project, you will get a history of all changes in the Rec project. Then in the next page select the two revisions between which you want to see differences.

Release Manager

Your release manager is there to help you. You should very much be able to ask your release manager to help you out with the archaeology, since we recognise that developers and users alike should not need to know everything about the inner workings of the code or svn. See SVNUsageModel.

Release Notes

Access via:

- Webpage (for projects): LHCb Computing -> Rec v14r4
- SVN (for packages): Tf/TrackSys head

Comment: The release notes are a good start, however, they have serious limitations.

Limitiations:

- **Human error**: These are **Human-written** summaries which might not reflect the code changes. Developers and release managers guess which changes are minor or major, but often make understandable errors in judgement. Sometimes the release notes are not complete enough, "made some changes" or "bugfix" are common, without explanation.
- **Read-only**: Notes are fixed at release time. A small modification may introduce a bug, which is not found until later and therefore is invisible in old release notes.
- **Branch-specific**: Only head developments make the head of the release notes. Modifications occurring on a branch are never seen in the head release notes, unless they are merged in, and even then the release notes might be a further summary of only some changes.

Tag Collector

Access via:

- Display page (for projects or packages): display -> Rec v13r1

Comment: Searching the tag collector is a step up from release notes. **If** the release manager uses the tag collector for tagging, then all modified packages within a given project are recorded (none can be accidentally missed, because then they do not make the release). The tag collector is editable such that bugs noticed later can be linked to the versions which are buggy. In the tag collector you get linked to the actual changesets, so you can make your own judgement as to whether a change is major or not. Since the tag collector is also a website, you don't have to have a local installation of software to start digging.

Limitations:

- **Uptake:** The tag collector is not used for everything by all release managers. Often releases, especially old releases will just not appear there at all.
- **Lost intermediate changes:** Sometimes only the final commit before a tag is added, and then you don't have the full list of changesets.
- **Comparability:** You can only see the additional entries used for a given version, which doesn't allow you you cross-correlate releases separated by a long distance in time.
- **Dependencies:** It's not always clear where the change you are looking for is going to be, is it in your project, or a lower-down project?

svn directly

Access via:

- Command line, on lxplus or on your local machine. Our repository is world-readable.
- Eclipse, our chosen IDE.

Comment: The only way to be 100% sure that you can see absolutely every change, without the possibility of human error or omission is to look in the code repository and compare versions directly using svn diff. Unfortunately cmt (and in future, cmake) makes this difficult, since we really version packages, rather than whole projects, and then these can be included "higher up" or can be migrated around the repository.

Limitations:

- **Expertese:** svn is not a very simple system. It requires expert knowledge, and our own repository requires another level of expert knowledge about how the releases are structured which is commonly only known to the release manager.
- **Packaging:** Which package is used in which project at which version is not something that it is simple to find out directly through navigation on SVN.
- **Time consuming:** every handshake with the repository takes some time, navigating through the repository directly is tough. You can get lost quite easily.
- **Read-only**: Notes are fixed at release time. A small modification may introduce a bug, which is not found until later and therefore is invisible in old release notes.

We provide a few wrappers for svn diff to make this a little easier.

svn via the web

Acces Via:

- LHCb:
  - Trac: https://svnweb.cern.ch/trac/lhcb/browser
  - WebSVN: http://svnweb.cern.ch/world/wsvn/lhcb
  - Statistics: https://svnweb.cern.ch/stats/lhcb/
- Gaudi:
  - Trac: https://svnweb.cern.ch/trac/gaudi/browser
  - WebSVN: http://svnweb.cern.ch/world/wsvn/gaudi
  - Statistics: https://svnweb.cern.ch/stats/gaudi
- Dirac:
  - Trac: https://svnweb.cern.ch/trac/dirac/browser
  - WebSVN: http://svnweb.cern.ch/world/wsvn/dirac
  - Statistics: https://svnweb.cern.ch/stats/dirac

Comment: The lowest level and most powerful tool is probably the svn web browser. Once you have a list of packages and modifications you care about, you can navigate to the tags you're interested in, and see which changesets were actually made. Since this is a website, you don't have to have a local installation of software to start digging.

Limitations:

- **Expertese:** svn is not a very simple system. It requires expert knowledge, and our own repository requires another level of expert knowledge about how the releases are structured which is commonly only known to the release manager. (Combine with the svn tools to remove this limitation).

- **Time consuming:** every handshake with the repository takes some time, navigating through the repository directly is tough. You can get lost quite easily.

Recommended: svn via tools

Access via:

- Command line, anywhere where lbscripts is installed.

Comment: A few wrapper tools to work with our svn repository. The tools are designed for the purposes of archaeology, and are the best of all the systems here to start digging. They are the most powerful, require the least expertese, and are the least prone to error.

If you're looking to see which stack a code release was based on, try `svnProjectDeps -P SomeProject vXrY` Which you can then compare to any version you like.

If you're only interested then in the changes in a specific package you can do: `svnProjectDeps SomeProject vXrY -f "SomePackage"`

Which will show you precisely where and which version a specific package was picked up. Removing the -f you will see the whole dependency tree of packages, which is very long, so probably you should start switching to `svnDiffTags`.

svnDiffTags is a little more powerful , and will summarize all the visible changes between two project versions (or two package versions), in terms of the full release notes, packages which were moved around, and the list of modified/added files.

```
svnDiffTags SomeProjectOrPackage vXrN vXrM
```

Limitations:

- **Command line**: Not everyone has continuous access to a machine with the LHCb environment.
- **Verbosity:** Depending on your purpose, these custom tools will either be too verbose, or not verbose enough.
- **Read-only**: Notes are fixed at release time. A small modification may introduce a bug, which is not found until later and therefore is invisible in old release notes.
