GIT TOOLBOX VERSION 2.0 : MAIN NEW FEATURES

1) Introduction of a cache sytem :

A cache system has been set up on machines merou, beaufix (login nodes), prolix (login nodes), alose, orphie, pagre, rason, in order to minimize the accesses to the metadatas Git/GCO's server. Those caches are updated as soon as ...

- ...a *git_post* command is executed ;
- ...a branch is created using *git_branch* ;
- ...branch status (private/public) has changed by the use of *git_public* .

Note that this mechanism is already working with current server version.

2) The end of *git_login* ?

Authentification using *git_login* is now useless. It will be based on your system user name, all the possible aliases (ex : mrpm602/khatib/elkhatibr) are – in theory... - taking into account (if it is not the case, ask GCO for help!). However, if you are already authentified as a Git user on a machine, this authentification will prevail until you disconnect using *git_logout* .

The *git_login* command can still be used to connect with any user name, provided you know this user's password.

3) Faster... :

Direct consequences of point (1) :

- the *git_view* command is now very fast ;
- commands *git_diff* and *git_history* are faster (around 10%) is some cases, and much faster (around 70%) in other cases, it depends on concerned file..

4) ...but asynchronous :

Use of commands *git_post* & *git_branch* will work (more or less) asynchronously.

- With *git_post*, everything will be done asynchronously. The Git « fetch » operations, the update of metadatas basis, and update of caches, will work in background. It is highly recommended to wait for the report of *git_post* execution (by mail) before doing anything else !
  However, it's optionally possible to use *git_post* synchronously with the option ***-sync*** .

- With *git_branch*, the creation of a branch if local Git repository will work synchronously. Only the update of the base of metadatas and caches will be performed in an asynchronous way.

5) Let's break everything and start over !

You create a Git branch, you do modifications inside, you post them... and you realize too late that there are mistakes in your posted modifications, even such an important number of mistakes you just want to remove everything and start over ! Now it will be possible, and there is more than one way to do it !

- If the entirety of your 150 modified routines is OK, except for a few routines, you can use *git_commit* with the new option ***–amend***, equivalent to the option ***–amend*** of native Git command *commit*. This option allows to amend the previous commit, even if it has already been posted. You

will have to post once again your modifications with *git_post*, using the new option ***–force***, otherwise it will not work.

- If you wish to ...
  * throw away your whole branch ;
  * return to a previous version of your branch ;
  * return to a previous version of your branch, but just for some modified files ;
  ...it will be possible, even if you have already posted your modifications ! You will have to use the new command *git_reset*, which replace *git_rmbranch*. In a case of a files' reset, it will be necessary to perform *git_commit* and *git_post* .

- If you performed a reset of your branch with the native Git command *reset*, don't panic ! You just have to post again your modifications with *git_post*, using the option ***–force*** .

- It wil be possible to join consecutive commits performed on a same branch with git_commit, using the new option ***–join*** .

WARNING : if you really would like to perform such a massacre whereas your branch has already been merged by GCO, please discuss with GCO before doing it...

6) Revision of the *-k* option of *git_edit* :

The ***-k*** option of *git_edit* didn't have a satisfying way of working... Now, using this option will have the effect of saving current of edited file, outside of the repository. In addition, the option ***-r*** has been implemented in *git_edit* : it allows to restore a previous version of a file, edited with ***-k*** option in the current branch.

7) Disappearance of *git_rmbranch* :

As it was evoked in the point (5), the command *git_rmbranch* disappears. To remove the content of a branch, you will have to use *git_reset*. To remove the branch itself, you will have to use *git_branch* with the new option ***-d*** .

8) New commands

- ***git_cache_show :***

This command returns the cache status for a branch, a commit, a version, a tag, or a git user.

- ***git_editor :***

This new command allows to set user's editors for file edition, « diff », and merge of files, among the list of defined editors returned by the command *git_editor –list –long* . If an editor is available on current machine, but does not appear in this list, it will be possible to add it (and then to set it as default) with the option ***–add***.

If you wish to use an editor without setting it as default, it is possible to use environment variables GIT_EDITOR, GIT_DIFF_EDITOR, GIT_MERGE_EDITOR.

For information, default editors are the following :
- on merou : edit=gvim, diff=xcleardiff, merge=xcleardiff ;
- on beaufix & prolix : edit=gvim, diff=meld, merge=meld ;
- on alose/orphie/pagre/rason : edit=gvim, diff=gvimdiff, merge=gvimdiff.

- ***git_restore* :**

The new command *git_restore* allows to restore another version of a file (taken from its history), or the previous version of this file. Not to be mistaken for the use of *git_edit* with ***-r*** option !

- ***git_sync :***

The new command *git_sync* replaces *git_fetch* . It allows to synchronise current branch (or another one) with the central Gir repository . A native Git *fetch* command is performed each time you execute *git_sync* . For example, when you execute *git_sync* without any options, you will immediately knowif current branch...

- ...is up to date with respect to central repository (status : *is up-to-date*) ;
- ...is behind central repository branch (status : *update needed*) ;
- ...is ahead central repository branch (status:*post needed*) ;
- ...is desynchronized with cnetral repository branch (status:*forced update needed*).

- ***git_fmerge :***

Despite its name, this new command has nothing to do with the old ClearCase script *cc_fmerge* !! This command will help to overcome two problems inherent to the way of working of native Git command *merge* :

- when you perform the merge of a branch in current one, you're going to merge all the commits « seen » by branch to merge which have not been merged in current branch, not only commits performed on branch to merge ;
- it is not possible to merge a selection of files from a branch, you have to merge the whole branch.

The command *git_fmerge* allows to overcome those two problems : it only takes into accounf commits performed in branch to merge, and it's possible to merge a selection of files from this branch. It does not use native git command *merge*, so this is not a real Git merge...

It is important to say that in large majority of cases the use of command *git_merge* is recommended !

- ***git_export :***

The new command git_export allow to export...

- ...the content of current branch ;
- ...the content of git repository ;
- ...a source version : complete (i.e. : CY43), or incremental (i.e.:CY41T1..CY41T1_op1.15).

If you use option ***–tgz***, you will get an archive file, not a files tree. Besides, it's possible to export sources on a distant machine.