Table of Contents

Introduction

The savannah cli interface enables to see/change entries in the savannah tracking systems as well as submit new items or clone a copy of existing ones. It can be used for all savannah trackers independently of the project as long as you have a user account that allows you to perform these operations.

Other use cases for the CLI is to use Savannah as a state machine if you have an automatic workflow.

For the moment the tool has only been tried out on the gLite patch tracker and there might be issues for other trackers. Be careful when trying it out as the result will become immediately live!

Mailing List / Support

If you have comments/additions/suggestions please feel free to contact the current maintainer, Andrew Elwell

There is a low volume mailing list 'savannah-cli-users' which all users of the script are strongly urged to join. See https://groups.cern.ch/group/savannah-cli-users/default.aspx. It is used for announcements, feature requests etc. All bugs should be reported against the *Savannah CLI* category in the glite integration group on, yes, savannah.

How to install

The savannah cli uses some packages internally and is based on python. To use the CLI tool, you need to have python & python-devel (to install from the tarfiles below) installed on your machine and you need to install the internal packages.

Those packages are:

- twill-0.9.tar.gz
- PyXML-0.8.4.tar.gz
- BeautifulSoup.tar.gz
- ClientForm-0.2.9.tar.gz

Depending on your distribution, some of these packages may be available as binaries: ubuntu/debian users should

```
sudo apt-get install python-twill python-beautifulsoup python-clientform
```

The easiest way is to do the manual installation using the script `install_deps.sh` in the CVS which does everything automatically. It is not needed to be root to run this script.

An alternative way is to download the packages manually and after that, untar them, build each one of them and install the package. The procedure is as follows (As a root user):

- `tar xvfz downloaded_file.tar.gz`
- `cd  new_directory`
- `python setup.py build`
- `python setup.py install`

The script is still under development, see the #Development section for SVN (and legacy CVS) checkout details

Usage

Please be **VERY CAREFUL** with this tool: if items are wrongly updated, they will have to be fixed manually; the DB cannot be restored for a *single* project (backups are for dealing with hardware problems).

The tool presently supports the following operations

- Read a parameter of a given savannah item
- Set a parameter of a given savannah item
- Get available options for a listbox item of a given savannah item
- List all fields of a savannah item
- Clone an item **Attention the priority is not conserved via this option as this is not part of the submit fields**
- Submit an item
- Add a comment to a bug/patch

The options to be specified depend on the actual operation (see below)

Example of execution:

Clone an existing patch

This is most commonly used by developers who need to provide similar patches for more than one platform. You start by creating a standard savannah entry for one node type / middleware / OS filling out all the necessary fields, Save, then clone using the CLI: `./savannah -t patch -o clone -i 2256 -g jra1mdw` and edit the new resulting patch to reflect the changes (such as OS or other component thats different from the original)

Bulk update bugs to new status

If you want to move multiple bugs from one state to another, the simplest way to do this is generate a list of bugs to work on and call the CLI for each of those. The following examples show changing the 'Status' field to *Accepted*

- small no of bugs:

```
$ for b in 50000 50034 47245 ; do ./savannah -t bugs -o set -i $b -n Status -v Accepted -g jra1md
```

- Large list of bugs in a file

for instance where you have done a search for a particulat category / state and wish to update them all - copy the resulting list into an editor so that you just have a list of bug numbers, one per line

```
$ while read b ; do ./savannah -t bugs -o set -i $b -n Status -v Accepted -g jra1mdw ; done < bug
```

Submit a new item

To submit a new item you need to prepare an xml file with the corresponding information. The syntax is as follows:

```
<submit>
   <item1>xxx</item1>
   <item2>yyy</item2>
</submit>
```

where item1 is for example assigned_to.

An empty example file for the gLite patch tracker can be found here: glite-patch-submit-example.xml

Options

```
./savannah -h

Script to interface to savannah tracking system

Savannah CLI v. Revision: 1.7

Usage:  savannah

    -h --help       Show this help
    -l --login      Allows the user to explicitly log in Savannah
    -V --verbose    Activate progress messages

  Mandatory parameter:
    -t  --tracker        Savannah tracker [patch,bugs]
    -o  --operate        Operations (see below for supported operations)

  Optional parameter (depending on operation):
    -i --item       Savannah item number
    -n --name       Savannah parameter name
    -v --value      New value/comment
    -g --group      Savannah group (for gLite this is jra1mdw)
    -f --file       file containing submit information in xml format

          Supported operations are:

                 Operation                                   | required options
                 ----------------------------------------------------------------
                 comment         Add a comment               | [i,v]
                 set             Set value                   | [i,n,v]
                 list            List available values       | [i]
                 get             Get a parameter             | [i,n]
                 get_available   Get available values        | [i,n]
                                 for selectbox
                 clone           clone (priority not cloned)  | [i,g]
                 submit          submit a new item           | [g,f]
```

Known issues

These issues might be treated in future versions:

- Full support of other groups besides glite (clone and submit are partly tracker specific)

'Group' specific information from savannah:

gLite (jra1mdw)

Available fields (patch tracker)

Savannah field name	Description	Submission
resolution_id	Status	n.a.
status_id	Open/Closed	n.a.
priority	Priority	n.a.
assigned_to	Assigned to	available
discussion_lock	Discussion lock	n.a.
summary	Summary	mandatory
custom_ta1	gLite subsystem tag / ETICS configuration	available
custom_sb2	gLite release	mandatory
custom_sb3	Architecture	mandatory (New from Aug09)
custom_ta2	RPM name(s)	mandatory
custom_ta8	Affected Metapackages	mandatory
custom_ta10	Metapackage changes	mandatory
custom_ta3	External packages	available
custom_ta4	Metapackages to be reconfigured	mandatory
custom_ta5	Metapackages to be restarted	mandatory
custom_ta6	Documentation location	available
custom_ta7	Configuration changes	mandatory
custom_ta9	Release notes	mandatory
custom_tf1	Certification release	n.a.
custom_tf2	PPS release	n.a.
platform_version_id	Operating System	mandatory (New from Aug09)
release	Release	n.a.
comment	Comment	n.a.

Decoding for select box fields for the gLite patch tracker

glite release

value	Description
100	None
101	gLite 3.0
102	gLite 3.1
103	LCG 2.7.0
104	gLite 3.0 WMS
107	TEST

Status (resolution_id)

value	Description
101	TBD
102	Ready for Certification
103	In certification
104	In Pre Production
105	In Production
106	Obsolete
107	Rejected
108	Certified
109	

Status (status_id)

value	Description
1	Open
3	Closed

Priority (priority)

value	Description
1	On hold
3	Low
5	Normal
7	High

OS (platform_version_id)

value	Description
100	None
101	SL4
102	SL5
103	Debian 4
104	Debian 5
105	Mac OS X

Archi (custo

value
100
101
102

	In Configuration
110	Patch incomplete
111	Configured
112	In PPS deployment test
113	With Provider
114	Ready for Integration

Development

Active development is done in Subversion - see http://svnweb.cern.ch/world/wsvn/aelwell/savannah_cli/ for browsing.

"take me to the source": Guest access:

```
svn co http://svnweb.cern.ch/guest/aelwell/savannah_cli/trunk
```

Revision History (From version 1.7)

See also the CVS / SVN Changelogs

New features in version 1.7:

#	Requirement	Description
1	Improve the identification process	The identification process has been improved by the use of savannah cookies. The user only needs to identify once, and from that moment on, the application will use the created cookies.
2	Create an explicit identification	A new specific operation has been added (-l or --login) just for this.
3	Improve the output	Warning messages and extra information has been eliminated. They can be shown with the -V or --verbose options.
4	Clone also the CC-ers list	The list of CC-ers is being cloned in this new version.
5	Command line way of adding a new comment	This option has been added as a new operation. Example of how to use it: `./savannah -t patch -o comment -i XXXX -v "This is a new comment"`

New features in version 1.9:

#	Requirement	Description
1	Get the list of dependencies	Create a new operation with the list of the items that the selected item depends on and their status.
1	Clean the output	Translate the name of the fields to the names that appear on the web and translate also the values of those fields

Requirements to be added:

#	Bug #	Requirement	
1	58161	Attach bugs to a patch.	Example of use: `/savannah -t patch -o attach -i patch_number -s "Integration Candidate" -c` Where -s is the status of the bugs you want to move to the patch and -c is the category of th look for. These options would help you to query for the bugs.
2	58163	Change the status of patches for a certain PPS or prod release.	Example of use: `/savannah -o release -t pps -r "3.1.0 PPS Update 40"` Where -t tells you that it's a pps/prod release and therefore the state of the patches should be Preproduction"/"In production" and you have to query using the information provided in th

			a Savannah patch is the Release field (for production) and Pre-production release field (for
3	58165	Update status of all dependent bugs when a patch changes to 'In PPS'	For example, patch #2766 moved from *Certified* to *In Production* but the bugs are still in *fi* *for test*
4	58166	Check correlation between patches and tasks	Check that all patches ready for certification also have a task associated with them. Will me needs to be able to work with 2 groups at once (jra1mdw[patches] and egee-sa3[tasks]) - ite 'ready for cert' or 'in cert' and check that there's at least one task associated with it

Recent Changelogs

- glite-patch-submit-example.xml: Example submit for gLite patch tracker

This topic: EGEE > SavannahCommandLineInterface
Topic revision: r34 - 2010-05-03 - MaartenLitmaath