jira-cli Documentation

Release 2.2-dirty

Ali-Akber Saifee

Jul 31, 2017

Contents

Command line interface to jira.

Important: The documentation on this page only applies to `jira-cli` versions 2.x. Please see the section on the *Rewrite* for more details.

CHAPTER 1

Setup

Installation

- with easy_install or pip:

```
sudo easy_install jira-cli
sudo pip install jira-cli
```

- from source:

```
git clone http://github.com/alisaifee/jira-cli
cd jira-cli
python setup.py build
sudo python setup.py install
```

Configuration

After installation, a few configuration steps will be needed before you can start interacting with jira. You can either do this manually by populating the `~/.jira-cli/config.cfg` file or interactively by issuing the command:

```
jira-cli configure
Base url for the jira instance: http://my.atlassian.net
username: johndoe
password: *******
would you like to persist the credentials to ~/.jira_cli/config.cfg?
[WARNING: this will store credentials in plaintext [y/n]:y
```

Sample `~/.jira-cli/config.cfg` file:

```
[jira]
base_url = http://my.atlassian.net
```

```
username = johndoe
protocol = rest    # either rest or soap
```

For subsequent invocations, you can always override the configuration values by passing in the appropriate value on the command line. For example

the user:

```
jira-cli view TP-01 -u janedoe -p hersekret
```

the jira installation:

```
jira-cli view TP-01 --jira-url=http://her.atlassian.net
```

the protocol:

```
jira-cli view TP-01 --protocol=soap
```

You can additionally add aliases for frequently used sub commands:

```
[alias]
myissues = view --search-jql='assignee=me'
...
```

And then use as:

```
jira-cli myissues
jira-cli myissues --oneline
...
```

CHAPTER 2

Usage

Interacting with issues

create an issue with only a title in project TP with default priority and type Bug:

```
ali@home ~ $ jira-cli new --type=bug --priority=Major --project TP 'Test Bug'
link                 : http://jira.yourdomain.com/browse/TP-24
assignee             :
summary              : Test Bug
issue                : TP-24
reporter             : ali
```

create an issue with priority Major and a description:

```
ali@home ~ $ jira-cli new --type Bug "Test Bug" --priority=Major --project TP --
↪description='the description'
link                 : http://jira.yourdomain.com/browse/TP-25
assignee             :
summary              : Test Bug
issue                : TP-25
reporter             : ali
```

list the issue TP-25:

```
ali@home ~ $ jira-cli view TP-25
link                 : http://jira.yourdomain.com/browse/TP-25
assignee             :
summary              : Test Bug
issue                : TP-25
reporter             : ali
```

list the issues TP-20 & TP-21:

```
ali@home ~ $ jira-cli view TP-20 TP-21
link                 : http://jira.yourdomain.com/browse/TP-20
```

```
assignee             : ali
summary              : test
issue                : TP-20
reporter             : ali

link                 : http://jira.yourdomain.com/browse/TP-21
assignee             :
summary              : Test Bug
issue                : TP-21
reporter             : ali
```

list the issues in short form:

```
ali@home ~ $ jira-cli view TP-20 TP-21 TP-22 --oneline
TP-20 test < http://jira.yourdomain.com/browse/TP-20 >
TP-21 Test Bug < http://jira.yourdomain.com/browse/TP-21 >
TP-22 Test Bug < http://jira.yourdomain.com/browse/TP-22 >
```

add a comment to an existing issue:

```
ali@home ~ $ jira-cli update TP-20 --comment # opens up the editor
this is a new comment added to TP-20
```

Update the assignee of an issue:

```
ali@home ~ $ jira-cli update TP-20 --assign ali
ali assigned to TP-20
```

Add a label to an issue:

```
ali@home ~ $ jira-cli update TP-20 --label moo
TP-20 labelled with moo
```

Add an affected version to the issue:

```
ali@home ~ $ jira-cli update TP-20 --affects-version=1.0
Added affected version(s) 1.0 to TP-20
```

Add a fix version to the issue:

```
ali@home ~ $ jira-cli update TP-20 --fix-version=1.0
Added fixed version(s) 1.0 to TP-20
```

Remove versions from issues:

```
ali@home ~ $ jira-cli update TP-20 --remove-fix-version=1.0 --remove-affects-
↪version=1.0
Removed fixed version(s) 1.0 from TP-20
Removed affected version(s) 1.0 from TP-20
```

transition the issue to a new state:

```
ali@home ~ $  jira-cli update TP-20 --transition='Done'
TP-20 transitioned to "Done"
```

transition the issue and set a resolution:

```
ali@home ~ $  jira-cli update TP-20 --transition='Done' --resolution='Fixed'
TP-20 transitioned to "Done"
```

provide your own formatting:

```
ali@home ~ $ jira-cli view TP-20 --format="%reporter, %summary, %status"
```

free text search for issues:

```
ali@home ~ $ jira-cli view --search='some random words'
```

jql search for issues:

```
ali@home ~ $ jira-cli view --search-jql 'reporter=ali and type=bug'
```

list only the comments for an issue:

```
ali@home ~ $ jira-cli view TP-20 --comments-only
Thu Nov 10 08:42:55 UTC 2011 ali : this is a new comment
Fri Dec 02 00:19:40 UTC 2011 ali : another comment
Sat Mar 10 11:08:34 UTC 2012 ali : test comment
Sat Mar 10 11:08:51 UTC 2012 ali : another test comment
```

Listing types

Often you have to use certain jira specific values for specifying things such as issue type, priority, status, resolution etc. The sub-command `list` can be used to list the acceptable values.

available projects:

```
jira-cli list projects
```

available filters:

```
jira-cli list filters
```

acceptable issue types:

```
jira-cli list issue_types
```

acceptable sub task types:

```
jira-cli list subtask_types
```

issue priorities:

```
jira-cli list priorities
```

issue statuses:

```
jira-cli list statuses
```

issue resolutions:

```
jira-cli list resolutions
```

project components:

```
jira-cli list components --project=MYPROJ
```

Possible transitions for an issue:

```
jira-cli list transitions --issue=TP-20
```

your own configured aliases:

```
jira-cli list aliases
```

CHAPTER 3

Development

Project resources

- Source : Github
- Continuous Integration: Travis-CI
- Test coverage: Coveralls
- PyPi: pypi

Note: jira-cli is tested on python version 2.7

Rewrite

The project was originally written against a `3.x` version of jira which only required support for the original `soap rpc` interface. With subsequent releases of jira the `json rest` api become the recommended method of communicating with a jira installation and the need to rewrite jira-cli became almost necessary given that the original implementation did not cater for a multi-protocol approach.

The original implementation was also not at all testable and provided a very flat command approach which led to numerous options and arguments being presented in a very haphazard manner.

As of version `2.0.0-pre` the command line interface is implemented using `argparse` which allows for a cleaner separation of commands. Furthermore, the interaction with the jira installation has been re-written so that a factory can be used to load the appropriate bridge - thus supporting both the legacy `soap rpc` and the new `json rest` interfaces.