tq: Tractor Query tool

tq is a powerful command line tool to search for and operate on jobs, tasks, commands, invocations, and blades. It permits extensive inspection of the current state of the queue and render farm, as well as historical analysis of commands that have run.

tq has extensive built-in help so that you don't have to leave the command-line to get assistance. To access this help, simply type tq help.

For more in-depth examples, see the tq Cookbook.

A Simple Example

tq takes a subcommand name as its first argument. In this simple example, tq jobs lists all of the jobs in the system.

```
% tq jobs
    jid user   title                      pri wait actv err done spooled
======= ====== ======================== ==== ==== ==== === ==== ===========
1305301 joe    RenderMan Monkey Left     400       840      160 05/30|14:26
1305303 joe    RenderMan Monkey Right    400  925   72   1    3 05/30|14:48
1305311 eve    Comp MV-CA 8_25            50         1          05/31|12:02
1305312 pablo  Interactive 4              20    5              3 05/31|12:03
1305313 btr    Interactive 5              20         1         7 05/31|12:07
1306042 mcgr   Interactive 6              20    8                05/04|16:47
...
```

In a similar fashion, tasks, commands, invocations, and blades can be listed.

Typically, a user wants to see only a subset of the items. The remaining non-option arguments on the command line form a "search clause" that is used to narrow the search. For example, this command lists all of the jobs owned by user mango that were spooled over 24 hours ago:

```
% tq jobs "owner=mango and spooltime < -1d"
```

In short, the search clause is a boolean expression (here with and) composed of comparisons (=, <) of the entity's attributes (owner, spooltime). Accommodations are made to reduce the amount of typing on the command line. For example, the literal mango is not enclosed in quotes, and the string -1 d is a short form meaning "one day ago".

More details on the search clause syntax can be found here.

Columns

tq has a default list of attributes it displays for each of its commands. A different set of attributes can be displayed using the --columns or -c option.

For example, to display the job id, owner, title, and metadata columns:

```
% tq jobs --columns jid,owner,title,metadata
```

Also, columns can be added or removed relative to the default set of columns using + or - with the column name. For example, the following command removes the title column and adds the spoolhost and spoolfile columns:

```
% tq jobs -c -title,+spoolhost,+spoolfile
```

Some columns have a default fixed width; wider values will have charaters replaced with ... to fit in the column. Other columns have a dynamic width, set to the widest value to be displayed. Column width can be specified using the = operator, where 0 is used to specifiy a dynamically sized column. In this example, the metadata column has a width of the widest displayed value, and the existing title column is fixed at 30 characters:

```
% tq jobs -c +metadata=0,=title=30
```

Note that in some shells like zsh, option arguments must be enclosed in quotes if they start with an equal sign.

```
% tq jobs -c "=title=30,+metadata=0"
```

Sorting

Results can be ordered using the --sortby or -s option. Multiple attributes can be specified to indicate secondary sorting. Items are ordered in ascending order, unless the attribute is prepended with - to indicate descending order.

For example, to show the order in which similar jobs would be processed in a FIFO queue, we can list them in priority descending, spooltime ascending order as follows:

```
% tq jobs --sortby -priority,spooltime
```

Limiting

Result sets can be shortened, making them useful for "top" lists. For example, this command will list the ten jobs with the most number of active tasks on the farm:

```
% tq jobs active -s -numactive --limit 10
```

This command lists the twenty hosts with the least amount of disk space.

```
% tq blades -s availdisk --limit 20
```

Affiliation

In the above examples, attribute of items are used in search clauses, column specifications, and sorting. Attributes of affiliated entities can also be specified in all of these cases by prefixing the attribute with the entity name. For example, the following command displays the active tasks owned by walt.

```
% tq tasks state=active and Job.owner=walt
```

Similarly, the following command displays all error tasks from jobs spooled within the last 24 hours, including their jobs' titles and spooltimes, and sorted by job priority.

```
% tq tasks "state=error and Job.spooltime > -24h" -c +Job.spooltime,Job.title -s Job.priority
```

Because spooltime and priority are attributes of no other entity than a job, the entity prefix is not required. The following is equivalent:

```
% tq tasks "state=error and spooltime > -24h" -c +spooltime,Job.title -s priority
```

Operations

Not only can tq list items, it can operate on them. This command shows all of my active jobs with errors:

```
% tq jobs mine and active and error
```

They can all be paused as follows:

```
% tq pause mine and active and error
```

This command nimbies the ten blades with the least amount of disk space:

```
% tq nimby --sortby availdisk --limit 10
```

All of the possible commands can be listed with tq help commands.

Some operations have additional options. For example, when changing the priority of a job, the --prioriy or -p option is used to specify the priority. The following command changes the priority of all of my jobs to 500.

```
% tq chpri -p 500 mine
```

Before an operation is executed on an item, the item will be displayed and the user will be prompted.

```
% tq delete mine
       jid user   title                pri blkd redy actv err done spooled
========== ====== ================== ==== ==== ==== ==== === ==== ===========
      1234 adamwg starting sleep test    1                      26 01/31|16:53
Delete this job? yes/no [y]:
```

This is helpful to avoid wide-spread destruction with a single mistyped command. If you **absolutely** certain that the targeted items have been properly identified, the prompt can be bypassed with the --yes option, which appears *before* the command. This is helpful when applying an operation to thousands of items, or when tq is being called from another script or by a cron job.

For example, this will kiss all of your jobs good-bye.

```
% tq --yes delete mine
```

Global Options

There are some options that are considered global to all commands, and as such appear **before** the command. One such option is --yesas shown above.

tq help usage lists all of the global options.

--user, password, --login and --logout are used when authentication is required.

--engine is used to point tq do a different engine from the default one, or the one defined by the TRACTOR_ENGINE environment variable.

--force is required to operate on jobs owned by a different user to help prevent accidental damage to other users' jobs. Note that a user would still need the appropriate permissions by the engine to successfully complete the operation.

Archived Jobs

By default, only entities of non-deleted jobs are searched when running tq. However, those of deleted jobs can be *included* using the --archives or -a global option. This makes it useful for looking at historical invocation data.

The following command will find all invocations that ran on blade c2001 in the past week, including those of deleted jobs:

```
% tq --archives invocations "blade=c2001 and starttime > -1w"
```

If you want results for *only* deleted jobs, you can add the deleted alias to the search clause.

```
% tq --archives invocations "blade=c2001 and starttime > -1w and deleted"
```

The --archives flag is necessary in order to use the undelete subcommand, since only deleted jobs can be undeleted.

```
% tq -a undelete jid=1234
```

More Examples

Many helpful examples can be found in the tq Cookbook.

Built-in Help

tq has detailed built-in help with useful examples so that you don't have to leave the command line to get more information.

```
% tq help
```

will show a top-level view of help topics.

```
% tq help subcommands
```

will show the full list of possible tq subcommands.

To get help on a specific command, such run tq help with the command name. For example:

```
% tq help jobs
```

will display the usage statement, in addition to examples and a detailed description of each available option.