Search Clauses

Tractor search clauses are used in the Dashboard's query page, the Python query API, and the tq command-line tool. They employ a high-level syntax that simplfies the search of job and blade information from the engine. The syntax uses Pythonic conventions for boolean expressions and comparison operators; however, there are additional conveniences to reduce syntactic overhead, which will be explained below. The result is a terse search syntax that is easy to write, read, and remember.

Note that when used on the command-line, some characters my need to be escaped or the expression needs to be enclosed in quotes so as to avoid interpretation by the shell.

Boolean Expressions

Standard comparison operators, boolean terms, and parentheses are used to build the search clause.

This matches jobs with active tasks:

```
numactive > 0
```

This matches active jobs with no errored tasks:

```
numactive > 0 and numerror = 0
```

This matches jobs that are active or could become active, with errored tasks. These are jobs that will use up more compute time on the farm but can't completely finish due to errors:

```
(numactive > 0 or numready > 0) and numerror > 0
```

Stand-alone Terms

Numeric attributes can be stand-alone to mean "not equal to zero". Given than numactive can't be negative, the following two clauses are equivalent

```
numactive > 0 and numerror = 0
numactive and not numerror
```

String Matching

String attributes can be matched for equality. The following matches all jobs owned by adamwg:

```
owner="adamwg"
```

String attributes can also be matched with regular expressions using like. The following matches jobs with candy in the title:

```
title like "candy"
```

The following matches jobs with RHEL followed by an integer:

```
title like "RHEL\d+"
```

Optional Quotes

Quotes are not required for single-word literals that do not conflict with attribute names or aliases.

For example, the following clauses matches jobs owned by adamwg:

```
owner="adamwg"
owner=adamwg
```

The following clause uses quotes because Smith Tower is a multi-word literal:

```
title="Smith Tower"
```

The following clause uses quotes in order to find jobs with the literal owner in its title. Without quotes, jobs with its actual owner in its title would be found because owner is an attribute of a job.

```
title like "owner"
```

Double quotes are not required in this clause because adamwg is not an attribute of a Tractor entity.

```
title like adamwg
```

List Matching

The in operator can be used to seach fields that store a list of values. The following matches jobs with prman in its tags:

```
prman in tags
```

List matching can be used to simplify certain queries. The following clauses match jobs owned by tom, dick, or harry:

```
owner=tom or owner=dick or owner=harry
owner in [tom dick harry]
```

The has operator can be used to search arrays for multiple values. The following two clauses matches jobs with prman and linux in their keys:

```
prman in tags and linux in tags
tags has [prman linux]
```

Aliases

Certain commonly typed arguments can be replaced with aliases. There are four kinds of aliases supported by tq: *attribute*, *search*,*argument*, and *commanc* aliases.

Attribute aliases are merely alternative names for an attribute. For example, file can be used in lieu of a job's spoolfile, or user can be used instead of owner. These aliases can be used in search clauses and in lists specifying column names or sort order. The following search clauses are equivalent:

```
owner in [adamwg dml]
user in [adamwg dml]
```

Search aliases represent a substitution for a boolean expression, and can be combined with other clauses and aliases. For example, when searching for jobs, active means numactive > 0. The following search clauses are equivalent:

```
numactive > 0 and error=0
active and not error
```

Clause aliases can change meaning depending on which command is used. To contrast with the above example, when searching for tasks, active means state=active.

Argument and command aliases are beyond the scope of this document concerning the search clause syntax; however it can be stated that such aliases include a search clause as a part of their definition.

All known aliases can be listed on the command line using:

```
% tq aliases
```

Affiliated Entities

Sometimes a search involves attributes of related entities. For example, the following clause matches tasks that are active:

```
state=active
```

But one may be interested in active tasks of jobs of a certain owner. The following clause matches tasks of jobs owned by adamwg:

```
state=active and owner=adamwg
```

The system will automatically determine that the owner attribute is associated with the job. However, if two entities have the same attribute name, it will be necessary to qualify attributes with their entity.

For example, the following clause matches active tasks with "render" in the task title:

```
state=active and title like render
```

While the following clause matchs active tasks with "render" in the job title:

```
state=active and Job.title like render
```

Time Comparisons

Several shortcuts exist for comparing database fields that represent time or elapsed seconds. Below is a breakdown of acceptable formats.

```
10am        - 10am on the current day
5pm         - 5pm on the current day
11:37       - 11:37am on the current day
16:21       - 4:21pm on the current day
4:21pm      - same thing
3/15        - midnight on March 15 of the year closest to the current date
3/15|4pm    - 4pm on March 15 of the year closest to the current date
3/15.4pm    - 4pm on March 15 of the year closest to the current date
3/15|17:37  - 5:37pm on March 15 of the year closest to the current date
3/15/05     - midnight on March 15, 2005
3/15/05|4am - 4am on March 15, 2005
-1s         - 1 seconds ago
-1m         - 1 minutes ago
-1h         - 1 hours ago
-1d         - 1 day ago
-1w         - 1 week ago
```

The following matches all jobs spooled after January 6, 2014:

```
spooltime > 1/6/14
```

The following matches all tasks started in the past hour:

```
starttime > -1h
```

The following matches all blades that haven't registered with the engine since 9am this morning:

```
heartbeattime < 9am
```

Units

The syntax supports different units for time and space related attributes.

For example, the following matches invocations that ran with a wall clock time of over 1 hour:

```
elapsedreal > 3600
elapsedreal > 3600s
elapsedreal > 60m
elapsedreal > 1h
```

The following matches blades with less that 1G of available memory:

```
availmemory < 1
availmemory < 1G
availmemory < 1024M
availmemory < 1048576K
```