# Tractor URL API

The various Tractor components communicate using HTTP requests. Most responses to queries are JSON-formatted data dictionaries.

Here is a summary of the query URLs and their replies:

- *ENGINE* below is "hostname:port" of the tractor-engine.
- *BLADE* below is the "hostname:port" of the blade to which the command is directed. Each tractor-blade process is listening on a specific, possibly unique, port number for administrative control requests.
- **NOTE** - many of the **edit** requests require authentication using Tractor's current simple scheme. These request URLs (or their form data) must include a login "session identifier" of the form "&tsid=123abc-456". The actual session-specific tsid value is generated by the monitor during the initial login exchange.
- **JSON Reply Formatting** - Note that many URL directives to the engine generate JSON-formatted HTTP reply bodies from the engine. By default, the reply HTTP header contains "Content-Type: application/json" following typical JSON conventions as expected by many JavaScript agents or other requesters. However, if you are manually entering some of the queries below into the URL field of your web browser, your browser may *display* the returned JSON as a hard-to-read unstructured block of plain text. In these "manual" unscripted cases, you can add an additional parameter to the URL request that causes the reply to be "pretty-printed" in your browser window: Adding "&fmt=pre" to the request URL causes the reply Content-Type to be set to "text/html" and the entire reply to be placed in an HTML "<pre>" block, which should preserve the structured formatting for easier human readability. Note that this format is not appropriate for use in scripted situations where actual JSON text is expected.

## Tractor Dashboard - User Interface

**To start a Dashboard session, direct your web browswer to:**

http://ENGINE/tractor/dashboard/

## Queue Controls

**Login, creating an authenticated session necessary for some types of control operations:**

http://ENGINE/Tractor/monitor?q=login&user=UUUU
(note: at sites using login passwords, an additional parameter "c=CCCCC" giving the site-specified password hash or encryption is also required.)

**Retire a job, deleting it from the active assignment queue and dashboards:**

http://ENGINE/Tractor/queue?q=jretire&jid=1234&tsid=XXX
(note: the tsid value is an authentication token returned from a prior q=login request).

**Restore a previously deleted job, if the underlying job data is still present:**

http://ENGINE/Tractor/queue?q=jrestore&jid=1234&tsid=sss

**Retry all tasks with errors in a particular job:**

http://ENGINE/Tractor/queue?q=jretry&jid=1234&tsubset=error&tsid=sss

**Stop and Requeue all currently active tasks in a particular job:**

http://ENGINE/Tractor/queue?q=jretry&jid=1234&&tsubset=active&tsid=sss

**Retry a specific task, e.g. tid=22, in a job:**

http://ENGINE/Tractor/queue?q=jretry&jid=1234&tsubset=22&tsid=sss

**Stop and Requeue all currently active tasks on a particular blade:**

http://ENGINE/Tractor/queue?q=ejectall&blade=host/addr&tsid=sss

**Skip a specific task in a job, allowing its parent to continue:**

http://ENGINE/Tractor/queue?q=tskip&jid=1234&tid=22&tsid=sss
(note: use "tid=error" to skip all error tasks in that job, or "tid=active" to interrupt all active tasks and continue past them)

**Completely restart (respool) an entire job:**

http://ENGINE/Tractor/queue?q=jrestart&jid=1234&&tsid=sss

**Interrupt a running job, kill active commands and pause to prevent further dispatching:**

http://ENGINE/Tractor/queue?q=jinterrupt&jid=1234&tsid=sss

**Interrupt a specific task, leaving that task in error state while other tasks continue to run:**

http://ENGINE/Tractor/queue?q=jinterrupt&jid=1234&tid=9876&tsid=sss
(note: the tid parameter can also specify a comma-separated list of task IDs)

## Job Queue Information Queries

**List the users who have previously spooled jobs:**

http://ENGINE/Tractor/monitor?q=users

**List the jobs currently queued for a particular user:**

http://ENGINE/Tractor/monitor?q=jobs&owner=name&filter=filtername&tsid=sss
(Note: the owner and filter restrictions are optional, but unfiltered lists may be large. The tsid parameter is only needed when a filter is specified. Add "&metadata=0" to also suppress metadata blocks.)

**Get the task tree for a particular job:**

http://ENGINE/Tractor/monitor?q=jtree&jid=1234

**List the commands associated with a task:**

http://ENGINE/Tractor/monitor?q=taskdetails&jid=1234&tid=11

**Get the URL for the logs from a particular task:**

http://ENGINE/Tractor/monitor?q=tasklogs&owner=name&jid=1234&tid=11

---

## Editing Job Attributes

**General job attribute editing** - Attributes entries listed in a job's

jobinfo.json dictionary can be changed using an http URL of the form:

/Tractor/queue?q=jattr&set_ATTRNAME=VALUE&jid=NNNN&tsid=sss

Similarly, attributes at the command-level can be changed using the "q=cattr&cid=..." variant (note: only set_service is supported at this time):

/Tractor/queue?q=cattr&set_ATTRNAME=VALUE&jid=NNNN&cid=CCCC&tsid=sss

Note that crews.config can specify permissions that restrict editing of particular attributes to specific crews, i.e. Wranglers only.

**Change the job-level Service Key requirements**:

http://ENGINE/Tractor/queue?q=jattr&jid=5678&set_service=someKey&tsid=sss

**Reprioritize a job:**

http://ENGINE/Tractor/queue?q=jattr&jid=5678&set_priority=12.34&tsid=sss

**Suspend new dispatching in a job:**
http://ENGINE/Tractor/queue?q=jattr&jid=5678&set_pause=1&tsid=sss

Use "&set_pause=0" to unpause the job. This type of "pause" suspends new task dispatching - the job will be skipped during blade assignment passes until unpaused. Any already running tasks from that job will continue to run.

## Session Queries

There are several files of user specific information stored in the user area below the tractor spool folder. The following queries allow the UI to save and retrieve these files.

**List all stored filters for user of specified type:**

http://ENGINE/Tractor/monitor?q=filters&user=name&type=filtertype

**Retrieve a user's named filter:**

http://ENGINE/Tractor/monitor?q=getfilter&user=name&key=filtername

**Store a user's named filter:**

http://ENGINE/Tractor/monitor?q=putfilter&user=name&key=filtername

**Retrieve user's preference information:**

http://ENGINE/Tractor/monitor?q=getpreference&user=name&key=filename

**Store a user's preference information:**

http://ENGINE/Tractor/monitor?q=putpreference&user=name&key=filename

**Delete a user's preference information:**

http://ENGINE/Tractor/monitor?q=delpreference&user=name&key=filename

**Retrieve user's session file:**

http://ENGINE/Tractor/monitor?q=getsession&user=name&key=filename

**Store a user's session file:**

http://ENGINE/Tractor/monitor?q=putsession&user=name&key=filename

**Delete a user's session file:**

http://ENGINE/Tractor/monitor?q=delsession&user=name&key=filename

**Retrieve a list of available file rule definitions:**

http://ENGINE/Tractor/monitor?q=filterrules&type=filtertype

---

## Administrator Controls

**Reload configuration files (blade, crews, limits):**

http://ENGINE/Tractor/ctrl?q=reconfigure

**Reload a single configuration file, for example: limits.config**

http://ENGINE/Tractor/ctrl?q=reconfigure&file=limits.config

**Query basic engine state:**

http://ENGINE/Tractor/ctrl?q=status

**Query current active limit tallies:**
http://ENGINE/Tractor/queue?q=limits

NOTE: this query is relatively expensive, it locks all threads. Add &full=1 to also include zero-valued "transitory" limits not defined in limits.config:**

**Query unrolled crew definitions:**

http://ENGINE/Tractor/monitor?q=crewdetails

**Query the current Dispatching Tier definitions:**
http://ENGINE/Tractor/monitor?q=enumtierlist

**Query unrolled blade profile definitions:**

http://ENGINE/Tractor/config?q=get&file=blade.config

**List recently connected Dashboard users (session mailboxes):**

http://ENGINE/Tractor/monitor?q=mboxes

**Trace engine assignment decisions:**
http://ENGINE/Tractor/ctrl?q=tracer&fmt=plain -- next assignment pass

add &t=bbbb -- for blade named bbbb add &t=jjjj -- for job with jid jjjj

**Remove an old entry from the displayed list of active blades:**
http://ENGINE/Tractor/btrack?q=delist&id=hhhhh/aa.aa.aa.aa

where hhhhh is the blade's hostname and aa.aa.aa.aa is the blade's current tcp/ip address.

**Diagnostic: query engine internal message queue lengths:**
http://ENGINE/Tractor/ctrl?q=status&qlen=1

NOTE: this query is relatively expensive, it locks all threads.

**Engine log level:** change the logging threshold level of the engine's own diagnostic logs:
http://ENGINE/Tractor/ctrl?q=loglevel&v=debug

Recognized level names are: severe, notice, info, debug, trace

**Engine database contexts:** bounce the engine's own internal client connections to the database server:

http://ENGINE/Tractor/ctrl?q=dbreconnect

## Blade Queries and Controls

**List all of the currently connected blades, with a recent status snapshot:**

http://ENGINE/Tractor/monitor?q=blades

**Stop and Requeue all currently active tasks on a particular blade:**

http://ENGINE/Tractor/queue?q=ejectall&blade=host/addr&tsid=sss

**Query the engine for its information on a specific blade:**

http://ENGINE/Tractor/monitor?q=bdetails&b=NAME

**Query the engine for blade information, and include results from an (expensive) direct probe of the blade itself:** http://ENGINE/Tractor/monitor?q=bdetails&b=NAME&probe=1

Use this query cautiously, especially in scripts, since it can cause processing delays on the engine as it waits for the blade response -- especially when probing blades that are slow to respond. In this form of the query, the NAME parameter can be "hostname" or "hostname/address". The address portion is historical and is ignored, the engine derives blade addresses from the last entry in the blade database.

**Query the current state of a blade directly:**

http://BLADE/blade/status

**Change the NIMBY state of a blade, via the engine:**

First, send a login request to receive a session ID (tsid), then:

http://ENGINE/Tractor/ctrl?q=battribute&tsid=SSS&b=BBB&nimby=NNN

where:

SSS = the tsid from the login reply
BBB = identifies the blade, as "blade_hname/192.168.0.1"
NNN = the desired new nimby state:
nimby=1 -- blade accepts only local Cmds from jobs spooled from that host
nimby=0 -- reset nimby state to unrestricted
nimby=jobOwner1,owner2,... -- blade only accepts jobs owned by the specified users

**Change the NIMBY state of a blade via direct http connection to the blade:**
http://BLADE/blade/ctrl?nimby=NNN

Where NNN is the same as for the engine proxy case, above. Note that this direct type of connection may be disallowed by the NimbyConnectPolicy setting in blade.config.

## Engine Metrics and Statistics

**Get the most recent engine health statistics:**
http://ENGINE/Tractor/monitor?q=statistics

The query returns various tractor-engine performance and status metrics. The currently provided values are intended to give administrators a quick set of engine "vital signs" to check when looking at overall Tractor health or for engine hot spots.

The statistics report is organized as a JSON dictionary. Each key in the dictionary represents one type of metric. The value field in the dictionary for each key is an array of recent samples for that metric.

The current dictionary metrics names are listed here. The list may evolve over time.

- "dt" - the actual length of each sampling interval
- "enla" - engine normalized load avg, cpu load on the engine host
- "rtot" - total inbound engine queries in the interval
- "asn" - assignments made during the interval
- "nrun" - count of currently dispatched cmds running on the farm
- "nctot" - total backlog of unexecuted commands in all waiting jobs
- "ncrdy" - total backlog of  ready commands in all waiting jobs
- "nslot" - total slots on the farm reported by blades
- "nsin" - total slots in use on the farm
- "tecpu" - estimate of current CPU utilization due to tractor-engine
- "temem" - estimate of current RAM utilization due to tractor-engine
- "dbcpu" - estimate of current CPU utilization due to the job database
- "dbmem" - estimate of current RAM utilization due to the job database
- "dbcqn" - backlog of pending job state database commits
- "iqn" - backlog of waiting requests on the main intake queue
- "sqn" - backlog of waiting requests on the shipper pool queue
- "aqn" - backlog of waiting requests on the assigner queue
- "qqn" - backlog of waiting requests on the job archive queue
- "jqn" - backlog of waiting requests on the job submission queue
- "mqn" - backlog of waiting requests on the monitor UI queue
- "nmbox" - subscribed UI sessions

**Get a recent history of engine statistics:**
http://ENGINE/Tractor/monitor?q=statslog

This query is like q=statistics, above, but it returns several minutes worth of samples for each metric.

Note: for continuous monitoring add "&stats=1" to your recurring q=subscribe request; statistics will arrive as 's' mbox messages.

The last entry in each array is the most recent sample. The engine maintains a simple ring buffer of these samples, so on the next update the first array element is dropped and all of the other samples appear to "shift left" with a new value appearing at the end. The statslog query returns all of the samples in the buffer so that a UI can generate a complete graph from a single query.