Getting Started

This section covers the basics of installing and running Tractor.

Initial overview and problem avoidance

TRACTOR INSTALLATION LOCATION and SERVICES

Tractor is most often used at studios where the various components are running as system services, intended to restart automatically when their hosts are rebooted.

Where is the software installed? If you run the Tractor installer locally on the engine host and again on each render blade node, then on Linux it will appear in /opt/pixar/Tractor-2.x. If instead you are running Tractor from a shared fileserver install, then that must be mounted before the services automatically restart. Refer to the installation and admin docs for the typical service setup and steps.

You can use the command line command "ps -eaf | fgrep -i tractor" to confirm that Tractor processes are running on your various machines. You will need *one* instance of the "tractor-engine" executable running on the engine host, and one instance of the "tractor-blade" process running on each of the blade nodes (it will appear as "rmanpy -m tractor.apps.blade").

LICENSING

Tractor itself requires licenses to dispatch jobs, but not to start up. Licenses govern the number of concurrent tasks that tractor-engine will dispatch at once. The engine log will show the number of licenses found. You can also see this number in the Dashboard, click on the small tractor icon in the upper left, one of the text entries shown will be "# Licenses" which is the count of Tractor licenses that the engine was able to acquire. If that is zero then no dispatching will occur. If you have a valid license, check the engine log for messages about connection problems to the license server.

CONNECTIVITY TO THE ENGINE

The first thing to confirm is that you actually have a route and firewall clearance to connect to the named engine host from your various clients, including the blades.

Usually the first thing to check is to connect a web browser to the engine, type in a url pointing to the engine "web server" something like "http://tractor-engine.yoursite.com/tractor/"

The dashboard, blade hosts, and other Tractor clients need to know the name of the engine host in order to connect. So you either need to specify what name you are using in each case, or you need to set up a name server alias (e.g. in DNS) for the default name that the blades will attempt to use, namely "tractor-engine". It is easier in the long run to set up the DNS alias because so many of the Tractor utilities will default to trying that name. An alias also allows you to change the engine host to a different machine or different address without having to update all of the blades, dashboards, and other clients. The blades and other clients also need to know what port the engine is listening on, again configurable, but it is easier to go with the default when you set up the engine: port 80.

If you have the DNS entry and the engine is listening on port 80, then the dashboard test above should be working. On blade hosts another simple network sanity test is to run:

```
/bin/tq ping
```

That will attempt to connect to the tractor-engine process and perform a connectivity sanity check. You can run that from the command line on blade nodes or user desktops to confirm connectivity. It will generate various error messages for different types of failure. "Unknown host" means that "tractor-engine" is not resolvable as a host name, for example.

If you have not set up the "tractor-engine" alias, then you'll need to run your test with a parameter naming the engine host and port. You can use a hostname or address if necessary. For example: (non-working example!)

```
tq --engine=my-other-host:80 ping
tq --engine=192.168.0.2:8888 ping
```

MATCHING BLADES TO JOBS

The basic mechanism for sending Tractor work (commands) to blades is based on Service Key matching. Tractor allows different blade nodes to advertise different capabilities - not all tasks can run on all machines. For example some tasks may not run on Windows, or some tasks require some application software to be installed, like RenderMan!

Studios can develop their own conventions for keywords that represent these capabilities. Jobs are submitted into the Tractor queue with keywords representing their requirements, and blades are started on render nodes using keywords that represent their capabilities. Tractor assigns a command to a blade when there is a match. By default jobs look for a keyword named "PixarRender" and all blades advertise that service by default.

Different Service Keys can be specified at job authoring time (on a per command basis if needed), and blade capabilities are specified in the blade.config file, via the "Provides" entries. If there are no blades advertising the service that your job needs, then no dispatching will occur.

If you have jobs sitting in the queue and visible in the Dashboard, then you can right-click on a given job and select "Trace Job" from the menu. Text will appear in the lower right output pane describing the matching tests applied to the next blade that happens to ask for work. Often this diagnostic is enough to determine typical dispatching issues.