Engine

The tractor.config file contains configuration parameters that control both the basic start-up behavior of the central tractor-engine process itself, as well as many of the basic scheduling policy settings. Typically very few override changes are necessary to this file, and only those few changes need to be copied to the site override directory as a small tractor.config file there.

EngineOwner

EngineOwner is the login name that will be the owner of the tractor-engine process, its data files, and its child processes such as the job database server. If the parameter is missing or set to the empty string, then tractor-engine will run as whichever user-id starts it. This inherited user mode can be useful for testing, but is sensible in cases where the engine is run as a system service and login identity is expected to be set externally through the service definition. When a non-empty value is given here, then either that user must start the engine or it must be started by root and the engine will setuid to the given owner name after acquiring listener ports. For security reasons, the value given here cannot be root nor will the engine continue to run as root if the setting is empty (a policy also enforced separately by the database server). Studios are encouraged to create a login account specifically for use as the owner ID of the engine process, largely to encapsulate whatever permissions model is desired. The use of pre-existing "service" accounts or nobody is also possible in some cases, but should be discouraged to avoid conflicts and confusion.

```
    "EngineOwner": "tractor",
```

ListenerPort

ListenerPort is the TCP/UDP service port that the engine uses to accept client requests from web-browsers, blades, and spoolers. The specified port value must not be in use by another application (such as a webserver), it must be accessible to other hosts on the site network, and the owner of the engine process must have permissions to use it.

80 is the built-in default value that clients that use to connect to the engine. If you choose a different value here then you *must* also supply that value in the TRACTOR_ENGINE environment variable or --engine parameter to blades and spoolers, tq sessions, and in every URL connection from Dashboard browsers and admin scripts. Please also see the comments regarding EngineDiscovery below for an alternative on small networks; and the ListenBacklog setting for large networks.

Typically, ListenerPort is a simple integer port number, like 80.

Alternatively, a full "interface and port" pair can be specified instead, causing the engine to listen only on the given network interface rather than all of them. Use a quoted string such as localhost:8181 or 10.0.0.1:80 to specify the interface+port pair. The interface portion can be given in dotted-quad notation or as a resolvable hostname.

```
    "ListenerPort": 80,
```

EngineDiscovery

EngineDiscovery controls the engine's advertisement of its listener address on the LAN, in lieu of a working DNS or other nameserver. Tractor components can find each other using a lightweight multicast discovery scheme (a subset of UPnP SSDP functionality). Possibe settings include:

- An empty json string "" tells the engine disable multicast announcements, which may be desirable at large sites seeking to reduce network chatter.
- @ tells the engine to always multicast, advertising the engine's own IP address and port as obtained from command-line parameters and introspection; this setting is suitable for small studios aiming to minimize configuration efforts, and wanting to ensure that discovery services are always on.
- tractor-engine or other hostname enables "multicast if unknown" mode. The engine will ONLY send announcements if the given hostname is UNKNOWN. That is: these multicast discovery packets will not be used if your nameserver (DNS, AD, LDAP, /etc/hosts) is already able to resolve the given name. This setting will avoid multicast traffic for those studios that already have a hostname alias for "tractor-engine", but will offer discovery for small sites that make no other configuration changes.

The default value is:

```
    "EngineDiscovery": "tractor-engine",
```

LicenseLocation

Tractor looks for licensing information by examining the same license file locations as other Pixar applications. It expects to find the filepixar.license in the root of the main Pixar application directory, assumed to be the parent directory of the app's specific install location. If a site needs special behavior beyond that, then administrators can specify the license information with the environment variablePIXAR_LICENSE_FILE, or by explicitly naming the file or license server with LicenseLocation. Only use one of the settings here when the shared locations are not workable.

```
    "LicenseLocation":         "${TractorInstallDirectory}/../pixar.license",
    "LicenseLocation":         "${PIXAR_LICENSE_FILE}",
    "LicenseLocation":         "9010@pixarserver",
```

MaxConcurrentDispatch

By default, the engine will consume all available Tractor license seats and use that value to determine the number of concurrent processes that can be launched on the render farm. You can restrict the number of concurrent dispatches (and hence license seats) used by a given tractor-engine by specifying the maximum here. This is typically only useful during testing when a test engine should limit its license usage relative to the main production engine. Use the number zero to indicate "use all available" seats.

```
"MaxConcurrentDispatch": 255,
```

EngineWorkerThreads

EngineWorkerThreads sets the total number of threads to be divided among several roles within the engine process. These are typically devoted to high-latency i/o operations like writing to client transaction sockets. Thread parallelism allows dispatching and UI operations to proceed while replies to prior requests are delivered. Since the threads are mostly used for latency-hiding they typically do not consume a lot of CPU. Note that the engine also allocates several other required subsystem threads as well, in addition to the "worker" pool count given here. There are several main internal worker thread pools pre-allocated from this count, and a typical operating value here is 10 + (number_of_blades / 100). Use the value zero to let the engine pick a default based only on engine core count.

```
"EngineWorkerThreads": 32,
```

TractorDataDirectory

The Tractor data directory contains at least two distinct types of subdirectories. The psql/ subdirectory holds the inbound spooled jobs from users, as well as checkpointed information about their progress. The user/ subdirectory holds files related to each Tractor user, primarily regarding their dashboard configuration settings.

All of the files in the data directory should be considered to be important assets. They should be part of regular back-ups. The tractor engine performs best when it has high-bandwidth, low-latency access to the files in this directory.

The default value is /var/spool/tractor.

```
"TractorDataDirectory": "/ssd/tractor",
```

EngineLogFile

EngineLogFile``specifies the log filename to be used for the engine's own diagnostic logs.  use the empty string "" to cause logging to be printed to the terminal.  The default is to log to ``engine.login the data directory. ${TractorDataDirectory} is used to specify the path to the data directory, and will adjust accordingly should the data directory be changed.

```
"EngineLogFile": "${TractorDataDirectory}/engine.log",
"EngineLogFile": "/tmp/tractor.log",
"EngineLogFile": "",
```

EngineLogVerbosity

EngineLogVerbosity specifies the log severity level threshold. Recognized values, in increasing level of verbosity, are: SEVERE, NOTICE, INFO, DEBUG, and TRACE. The default value is NOTICE.

```
"EngineLogVerbosity": "DEBUG",
```

Command Logging

Some of the following parameters are used for storing and retrieving output from commands launched by tractor. Certain values may specify a *template string*, which can contain substitution patterns as follows:

> - *%u the job owner's login (userid, string)*
> - *%j the job id (integer)*
> - *%t the task id (integer)*

SiteCmdLogServerStartup

SiteCmdLogServerStartup specifies how the python logging server should be started automatically when the engine is started. Use the empty string to indicate that the engine should not start a log server. The default value is the empty string.

```
    "SiteCmdLogServerStartup": "'${TractorInstallDirectory}/bin/tractor-cmdlogger' --
    filetemplate='${TractorDataDirectory}/cmd-logs/%u/J%j/T%t.log' --port=9180",
```

SiteCmdLogRetrievalURL

SiteCmdLogRetrievalURL is sent to UIs giving them a URL at which to retrieve command output logs.

Some site configurations will allow UIs to fetch these logs directly from the fileserver where they are stored, such as when another webserver (like Apache) has direct access to the log files. By default, the tractor-engine itself acts as the webserver to deliver the log text to the UI browser, assuming it has access to the actual output logs. The blades are told where to put their logs in blade.config. By default the engine starts a simple python log-receiver subprocess (see SiteCmdLogServerStartup, above), and the default blade.config directs the blades to connect to it for logging. In that scenario, if the engine is serving up the logs, then the *relative* URL given below is sufficient; the URL does not mention a host. A UI that is already connected to the engine for UI service will get that URL and use the same host that it queried to fetch the URL.

You can specify a different host by including the http://HOST prefix explicitly. The default value is /tractor/cmd-logs/%u/J%j/T%t.log.

```
    "SiteCmdLogRetrievalURL": "http://HOST:PORT/tractor/cmd-logs/%u/J%j/T%t.log",
    "SiteCmdLogRetrievalURL": "/tractor/cmd-logs/%u/J%j/T%t.log",
```

SiteURLMap

SiteURLMap is a mapping of inbound URL resource prefixes to on-disk assets. This is a simple json list of string pairs, the first item in each pair is the prefix and the second item is the location. *Order is important* in this list since prefixes are tested in the order given and the first matching one is used. Assets with prefixes that are NOT listed are assumed to lie below /tractor/.

```
    "SiteURLMap": [
       "/tractor/cmd-logs/",   "${TractorDataDirectory}/cmd-logs/",
       "/tractor/dashboard/",  "${TractorDashboardDirectory}/tv/",
       "/tractor/tv/",         "${TractorDashboardDirectory}/tv/",
    ],
```

SiteContentTypeMap

SiteContentTypeMap defines the HTTP Content-Type header that is sent from tractor to web browsers when non-tractor files are loaded, such as image previews or log files. Some sites may have special browswer plug-ins or registered helper applications that will display a particular filetype automatically if the Content-Type header is set appropriately. Conversely, setting the type to something that is not recognized by the browswer will often allow the user to choose between downloading the file or launching an application of their own choice. Tractor also has an internal table of mappings that it uses for common types, if the file extension is not listed here explicitly.

```
    "SiteContentTypeMap": {
       ".tif":  "image/tiff",
       ".exr":  "image/x-exr",
       ".png":  "image/png",
       ".jpg":  "image/jpeg",
       ".log":  "text/plain"
    },
```

SiteMaxListReplyCount

SiteMaxListReplyCount truncates the number of jobs or blades sent to the dashboard in order to limit the processing time spent by the engine to create the replies as well as to prevent possible javascript crashes in some web browsers when parsing very large reply lists. The default value is 2500.

```
    "SiteMaxListReplyCount": 1000,
```

JobScriptQuotingMode

Tractor can accept job descriptions in the Alfred job script format. Alfred required certain quoting and escaping conventions for nested strings within a job script (requiring an extra level of escape characters). Tractor can either follow the same conventions for use with legacy job generators, or it can accept a simpler quoting style for newer jobs. Use the value alfred for alfred compatibility, or the value tractor for the simpler scheme. The default value is tractor.

```
    "JobScriptQuotingMode": "alfred",
```

JobSchedulingMode

JobSchedulingMode determines the order in which tasks are considered when assigning to a blade.

P+FIFO is the default scheme for assigning available blades to jobs, it always picks the job with the highest priority value, and if there are several jobs with the same priority then it selects the one that was spooled first. If there are more available blades than runnable tasks in the first job, then the next job is considered.

P+RR is a simple "Round-Robin" scheme used when considering jobs of the same priority. Several jobs with many ready tasks can be running concurrently, however it tends to favor jobs with long-running tasks, since they can "collect" additional blades on each subsequent turn. The "ATCL" modes are usually preferable in production settings.

P+ATCL, *Active Task Count Leveling*, first sorts jobs by strict priority, then within a block of jobs with the same priority it prefers to assign available blades to jobs with the fewest active tasks. This mode tends to allocate roughly the same number of blades to each job, while favoring older jobs over newly spooled ones. Given that it maintains roughly equal blades counts, jobs with short-running tasks will finish sooner than jobs with the same number of long-running tasks.

P+ATCL+RR is similar to *P+ATCL* above, but differs in how it handles the "even" distribution of blades across jobs when there are more waiting tasks than blades. It adds a kind of round-robin component, tending to equalize blade count over time across *all* jobs in the priority block. In general, P+ATCL favors the oldest jobs, P+ATCL+RR favors jobs that have been waiting longest for blades.

NOTE: Any scheduling mode other than P+FIFO will incur extra computational expense during blade assignment, and will therefore decrease system throughput by some amount. The magnitude of the observed effect will vary depending on overall job load, farm size, task mix, and engine host. The default mode is P+FIFO.

```
    "JobSchedulingMode": "P+ACTL",
```

DispatchTiers

Each job on the queue is associated with a a dispatching tier, typically just the default tier. Tiers provide an additional layer of control over the order in which jobs are dispatched to the farm, beyond the usual job priorities and the current scheduling policy mode. Specifically, jobs from high-valued tiers are considered before those from low-valued tiers. This kind of abstract categorization and ordering may be useful in several circumstances, but tiers will usually be most helpful during a wrangling "crisis" of some sort, large or small. For example, it may be useful to move several important jobs to a higher tier while leaving their job priorities unchanged, thus preserving their relative sorting. Each tier can also be *paused* individually in the Dashboard, so dispatching across entire groups of jobs can be temporarily suspended or enabled -- for example, general dispatching can be paused while a series of administrative jobs are allowed to proceed. Currently the configurable tier attributes are very simple, just a group name and it's priority relative to other tiers. Additional attributes and behaviors may become available in future releases. The Dashboard also provides some controls for filtering and sorting jobs based on tiers. The following shows the default tier configuration.

```
    "DispatchTiers": {
       "admin":   {"priority": 111.0},
       "rush":    {"priority":  99.0},
       "default": {"priority":  50.0},
       "batch":   {"priority":  10.0}
    },
```

SiteHttpOriginAccess

SiteHttpOriginAccess controls the outbound HTTP protocol reply value for the Access-Control-Allow-Origin header line. This setting allows the engine-as-webserver to work within a site's CORS policy restrictions. The default value is * (asterisk) for ease of use. Use the empty string to disable CORS-style sharing. This setting will primarily affect the ability of the Tractor Dashboard to fetch command log files when the specified log server hostname is different from the hostname in the URL used to load the Dashboard itself. The default is *.

```
    "SiteHttpOriginAccess":  "*",
```

CmdAutoRetryAttempts and CmdRetryrcRetryAttempts

CmdAutoRetryAttempts and CmdRetryrcRetryAttempts control the number of automatic attempts made by tractor-engine to retry commands that exit with a non-zero exit status. These automatic retries *may* be a useful workaround at sites that frequently experience command failures due to temporary transient conditions.

In some job scripts, certain RemoteCmds are annotated with the -retryc {5 17 84} option, where the actual numbers given represent known exit codes from the launched app representing conditions that warrant a retry. The number of retries attempted in this situation is controlled by CmdRetryrcRetryAttempts below. Set it to zero to disable retryrc retries, or set it to a small integer to allows these retry attempts.

CmdAutoRetryAttempts is used to control automatic retries on exit codes not listed in a Cmd's -retryrc option, or for any non-zero exit when -retryrc is not specified. Set it to zero to disable these automatic retries, or set it to a small integer to force these retries globablly.

Commands that exit due to SIGINT, SIGTERM, or SIGKILL on unix-style platforms are *not* retried since those signals are almost always due to deliberate external actions intended to actually stop the given task from proceeding. On Windows, requests to interrupt running subprocesses are handled with the TerminateProcess call; the Windows blade will report these exits using code "-15" which is analogous to SIGTERM in the sense that it will not trigger an auto-retry in the engine.

Default settings are shown below.

```
    "CmdAutoRetryAttempts": 0,
    "CmdRetryrcRetryAttempts": 1,
```

InitialJobLoadProgressLog

On initial start-up, the engine loads previously spooled jobs from the job database to continue dispatching the unfinished ones. While loading these jobs, the engine can print a DEBUG-level diagnostic to its log after every N jobs have been loaded.InitialJobLoadProgressLog controls the number of jobs on which this diagnostic is reported. Set this value to 0 to suppress these log entries. The default value is 1000.

```
    "InitialJobLoadProgressLog": 5,
```

LogLongRunningRequests

This setting controls some additional engine debug-level diagnostics related to long-waiting and long-running http requests. The two parameter values are thresholds [WAIT, RUN] that trigger diagnostic entries in the engine log whenever an inbound (or internal) http request exceeds the given wait time or run time, in seconds. Use zero to disable either check.

```
    "LogLongRunningRequests": [0.0, 5.0],
```

ListenBacklog

This setting, a connection count, controls the amount of storage devoted to holding unprocessed inbound "connect" requests from TCP clients. These are typically HTTP requests from tractor-blades and Dashboards. The traditional SOMAXCONN compile-time value is too small for modern web servers under load, so a run-time control is necessary. Use the value 0 (zero) to cause tractor-engine to query the operating system for the currently configured maximum listen backlog.

```
    "ListenBacklog": 0,
```

These operating system values are typically configured by system administrators during a careful tuning of the engine host for the usage case of a particular studio. For example,

```
    Linux:    /proc/sys/net/core/somaxconn
    Mac OSX:  sysctl kern.ipc.somaxconn
```

Note that an engine restart is required to pick up changes to either the value in this file, or changed system config values. Also, other coordinated operating system tuning changes are usually required, for example on Linux in /etc/sysctl.conf:

```
    net.ipv4.tcp_max_syn_backlog = 2048
    net.core.somaxconn = 1024
    net.ipv4.tcp_syncookies = 1

    then apply these changes with: sysctl -f /etc/sysctl.conf
```

SiteSMTPServer and SiteDefaultMainDomain

The site SMTP server may be used by Tractor to deliver important messages to administrators at your site, or in some cases to complete user account creation. It will connect to the host specified by SiteSMPTServer via smtp protocol to deliver messages. It will use the SiteDefaultMainDomain as a suffix for email addresses that are simple usernames. For example, given a userid of eve and the default empty string for the domain, then mail delivery would simply attempt to use eve on the smtp server; if the domain is specified as buyandlarge.com then the recipient will be specified as eve@buyandlarge.com.

The default values are as follows:

```
    "SiteSMTPServer":          "smtp",
    "SiteDefaultMailDomain":   ""
```