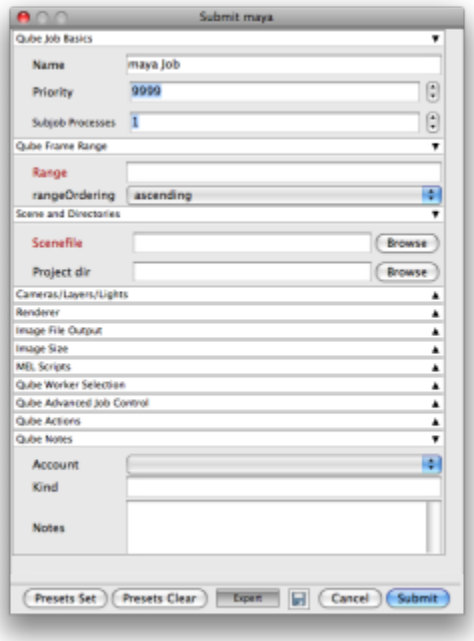
Job Submission Dialog

- Overview
- Buttons
- Common Parameters
- Basic Parameters for all Jobs
- Render Thread and Job Reservation Controls
    - Thread Control Behavior
- Parameters for Cmdline Jobs
- Parameters for Cmdrange Jobs
- Parameters for SimpleCmd Jobs
- Detailed Parameters
    - Preview Frames Submission
    - Qube Worker Selection
    - Qube Advanced Job Control
    - Qube Job Environment
    - Qube Output Parsing and Validation
    - Qube Actions
    - Shotgun Submission
    - Qube Notes

Overview

Jobs can be submitted from WranglerView using the items under the Submit menu. Selecting one will launch a modal submission dialog with the standard framework and job-specific parameters. Specify the fields and then press the "Submit" button to submit the job to the Qube! Supervisor.

Submissions for specific applications, such as Maya and Nuke, are explained in detail in their own sections. Please refer to that documentation for submissions to individual applications. This page documents a large set of parameters; individual submission targets do not necessarily expose all the parameters defined here.



Buttons

There is a common set of buttons located at the bottom of the Submission Dialog.

- **Set Defaults**: Store as defaults in the User Preferences for that Jobtype, the values listed in the current submission dialog. The interface will use those values the next time the dialog is opened. This allows you to specify common fields like Priority or Executable that should always be the same value.
- **Clear Defaults**: Remove any stored defaults for that Jobtype submission dialog
- **Expert Mode**: Toggle button to display or hide export and non-default values from the submission dialog to reduce clutter when there are

a lot of parameters. The current state of this toggle is stored in the Preferences Dialog.

- **Save [Disk Icon]**: Store the current properties of the job from the dialog as a file (by default an XML file). This can be used to submit through the WranglerView at a later time with "Submit->File…" or from the command line with the `qbsub` command, or via the API.
- **Cancel**: Cancel the job submission and close the dialog.
- **Submit**: Submit the job with the specified parameters to the Qube! Supervisor.

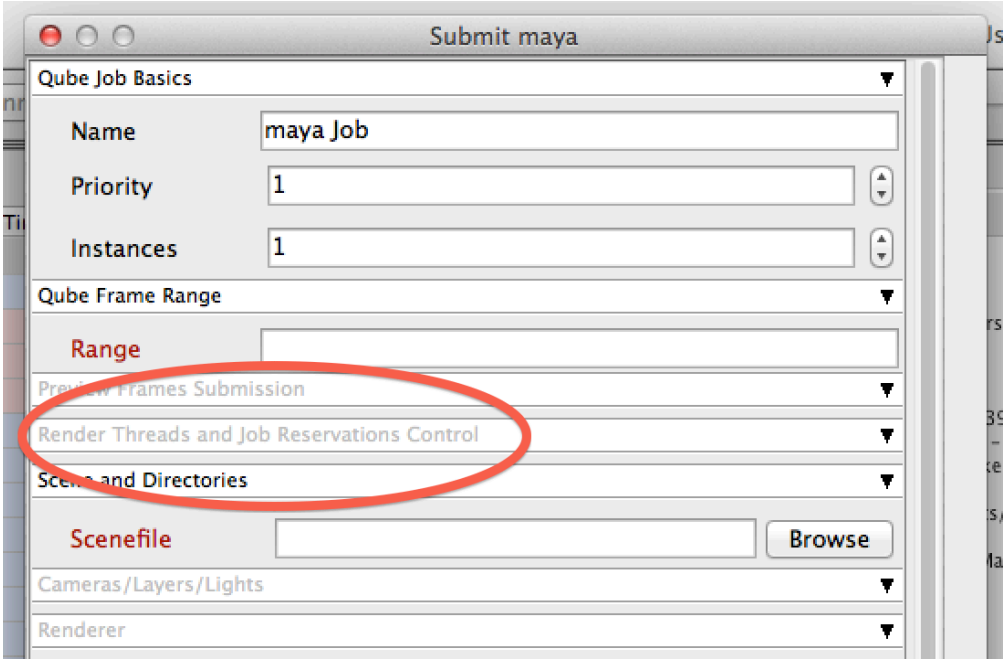
Common Parameters

Basic Parameters for all Jobs

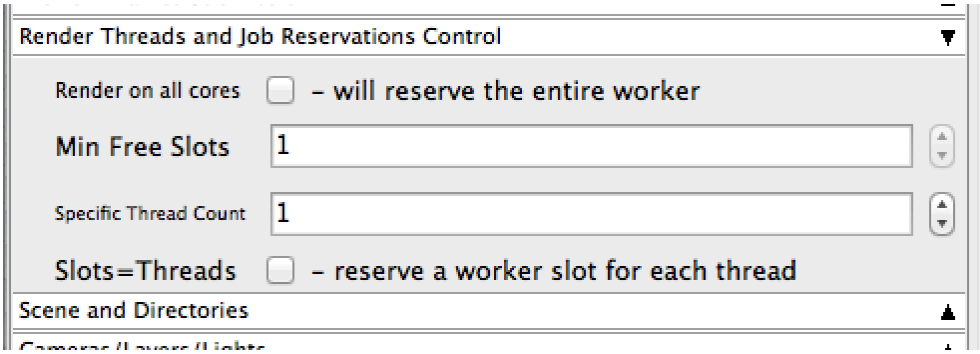
- **Name**: name used for the job, usually artist specified.
- **Priority**: priority of the job. A number between 1 and 9999. Lower numbers mean higher priority.
- **Instances**: The number of processes (or subjobs) to run the job with at the same time. When rendering, this equates to the number of frames being rendered at the same time.
- **Max Instances**: (Expert Mode only) The maximum number of additional instances to run using smartShare. The default of -1 means "unlimited"; a setting of 0 (zero) disables smartShare expansion for this job.

Render Thread and Job Reservation Controls

This section is located near the top of the submission UI, and is collapsed when 'Expert Mode' is not selected. For applications that do not support setting the number of threads, this section is not visible at all.



Enabling 'Expert Mode' at the bottom of the UI will open this field up



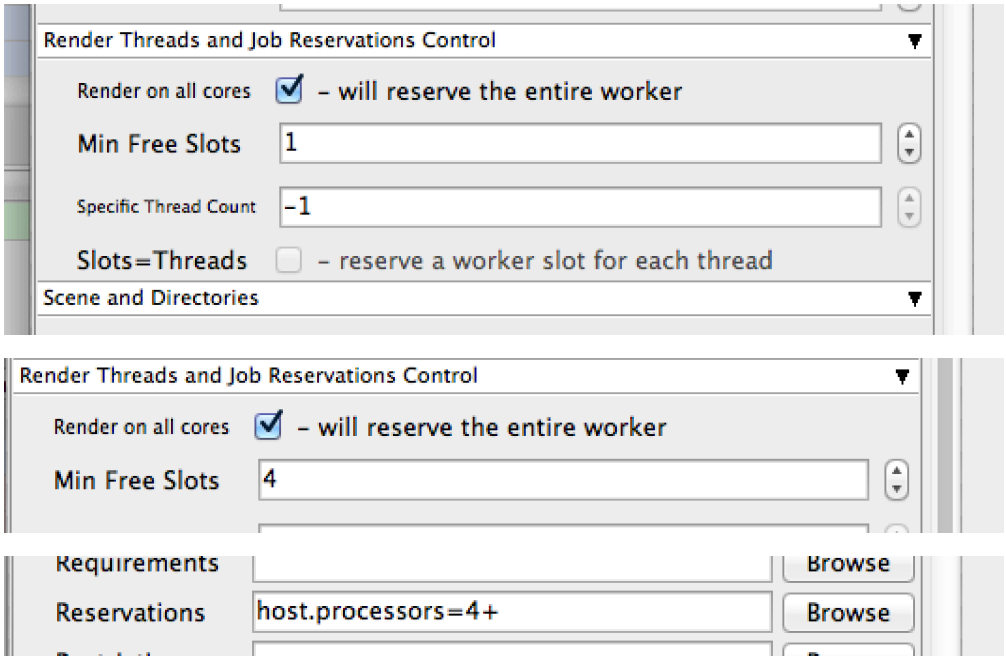
Use **Slots=Threads** only on farms where the workers are configured to have as many slots as they have cores; they will show

0/8 or 0/16 in the **Slots** when they're empty, where the 0/*N* number is the number of free slots. This is the default configuration in Qube!

The other common configuration is all the worker set to only ever accept 1 job, where they show 0/1 in the **Slots** column in the UI. **Do not** use Slots=Threads in this configuration, since none of your workers will have enough free slots to accept the job.

Thread Control Behavior

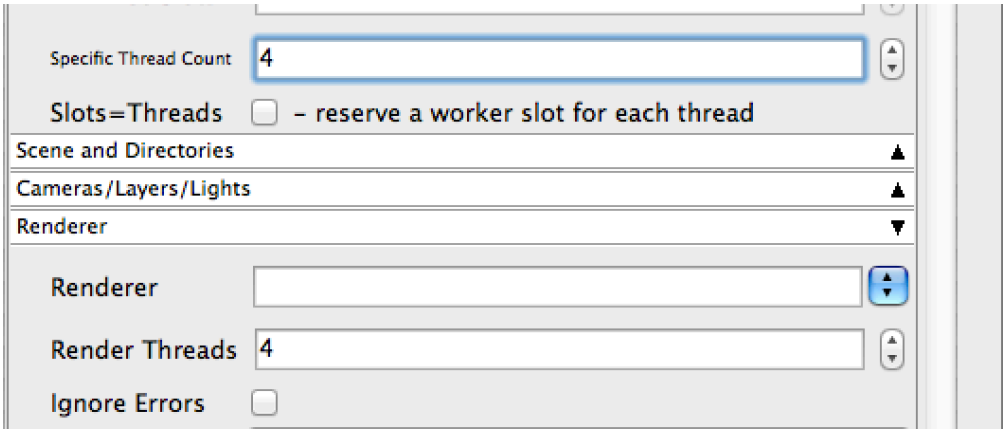
Checking **Use All Cores** for applications or renderers that support auto-detection of the number of cores installed on a worker host will set the renderer's appropriate control to enable this feature. In the case of a Maya job, it sets **renderThreads=0**; for the Mentalray renderer, it sets **autoRenderThreads=True**. It will try to "do the right thing" for each application where this control is visible.



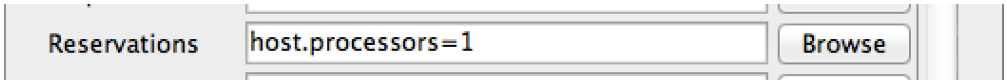
Checking **Render on all cores** will also sets the job's reservation string to reserve all cores: you will set a "**+**" at the end of the **host.processors=*N*+** . This means "only start on a worker with *N* free slots, but reserve them all, so that no other jobs will start on this worker while I'm running on it". The Min Free Slots value will affect the value of N, so setting it to 4 will say "start only on a worker with 4 free slots".

Render on all cores also disables the bottom-half of this section, **Specific Thread Count** and **Slots=Threads**, whose behavior is explained below.

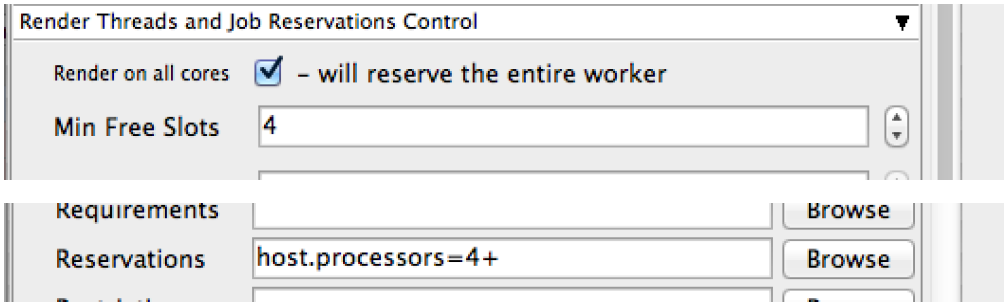
Setting Specific Thread Count, on the other hand, does not necessarily reserve all the cores on the worker, it only sets the renderer's particular number of threads control, in Maya's case it's **Render Threads**:



But you will notice that the job's reservations have not changed, and is probably still at the default value of **host.processors=1**. This is to support workers which are configured for fewer slots than they have cores.



The **Slots=Threads** control will link the job reservations to the thread count, so a 4-threaded job will reserve 4 worker slots. It also disables the **Render on all cores** and **Min Free Slots** controls.



As you change the Specific Thread Count control the reservations value will automatically update.

Parameters for Cmdline Jobs

- **Command**: The command to run on the Worker. Paths and syntax should be what the Worker's OS expects, not the submitting machine.
- **Shell (Linux/OS X):** Specify the shell to use when executing the command line on the Worker. Only visible in Expert Mode.

Parameters for Cmdrange Jobs

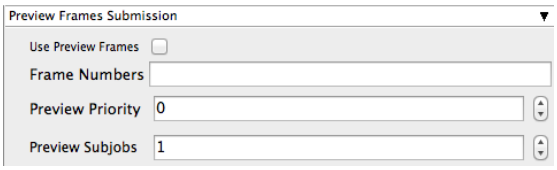
- **Command**: Same as the Cmdline Job, though it will substitute the following strings based on the frame being executed for the given task.
  - QB_FRAME_NUMBER
  - QB_FRAME_START
  - QB_FRAME_END
  - QB_FRAME_STEP
  - QB_FRAME_RANGE
- **Range**: The frame range to execute.
- **Execution**
  - Individual Frames
  - Chunks with n Frames
  - Split into n Partitions
- **Ordering**: Specify whether tasks should be executed in ascending, descending, or binary sort (first, last, middle, split the middle values, …) order.

Parameters for SimpleCmd Jobs

- *All the fields for either Cmdline or Cmdrange jobs (see above), plus*
- **Cmd Template**: String used to construct the command line along with the rest of the job parameters. Python string representations are used, e.g. %(val)s to represent the string value from "val". If listed, %(argv)s places all optional arguments at that location instead of at the end. This constructs a command line that is then used by the Cmdline or Cmdrange jobtypes on the Workers. See above for the string replace values for CmdRange.
- **Executable** – path to the renderer or executable to run. Unless it is in the path on the Worker's environment, this needs to be set to an absolute path (to where the executable is located on the Worker)

Detailed Parameters

Preview Frames Submission



Click here for details...

Use Preview Frames

Enabling preview frames will create 2 jobs:

- A primary dependent job with a higher priority that will render the selected frames first
- A secondary job with lower priority that will render the remaining frames. T his will return the selected frames faster so that you can check the accuracy of your renders.

Frame Numbers

Choose the frames that you wish to render first. If left blank the default is to render the first frame, the last frame and the middle frame in that order. You can select specific frames by adding comma separated frame numbers e.g 1,2,10,15,75, or a range with, e.g., 1-100x5 (1 to 100, every 5th frame)

Preview Priority

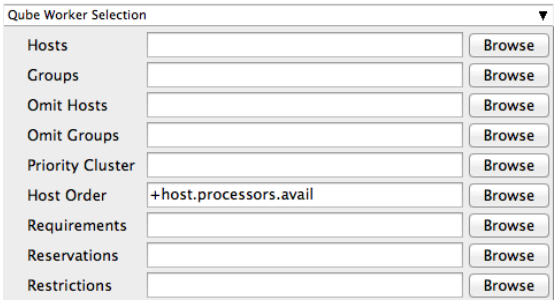
Choose the priority for the preview job. This can be set by the site admin.

Preview Subjobs

Choose the number of instances / subjobs for the preview frames. By default, this is equal to the number of preview frames - that is, it will try to do all the preview frames at the same time.

Note that when you submit a job with preview frames enabled, it will actually submit 2 jobs—one with the preview frames list at a higher priority, and another with the rest of the agenda, at the normal priority (as specified in the job's **Priority** field). You will get, consequently, 2 job IDs for the submission.

Qube Worker Selection



Click here for details...

Hosts

Explicit list of Worker hostnames that will be allowed to run the job (comma-separated).

Groups

Explicit list of Worker groups that will be allowed to run the job (comma-separated). Groups identify machines through some attribute they have, eg, a GPU, an amount of memory, a license to

run a particular application, etc. Jobs cannot migrate from one group to another. See worker_groups.

Omit Hosts

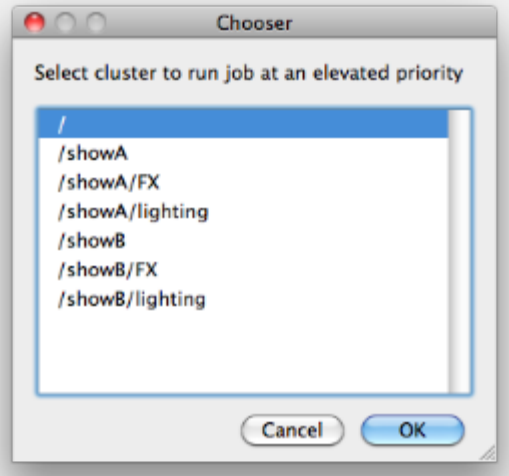
Explicit list of Worker hostnames that are ***not*** allowed run the job (comma-separated).

Omit Groups

Explicit list of Worker groups that are ***not*** allowed to run the job (comma-separated).

Priority Cluster

Clusters are non-overlapping sets of machines. Your job will run at the given priority in the given cluster. If that cluster is full, the job can run in a different cluster, but at lower priority. Clustering

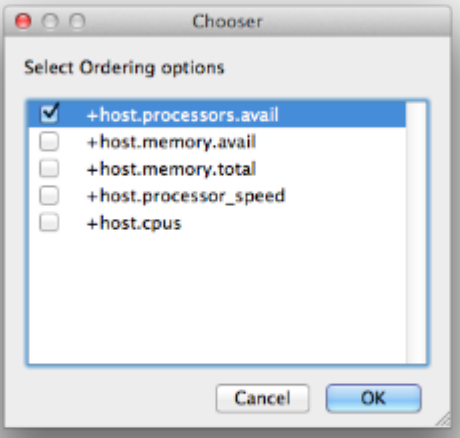


Example:

- A job submitted to /showB/lighting will run with its given priority in /showB/lighting cluster.
- If /showB/lighting is full, that job can run in /showB/FX, but at a lower priority.
- If both /showB/lighting and /showB/FX are full, the job can run in /showA/* at an even lower priority.

Host Order

Order to select Workers for running the job (comma-separated) [+ means ascending, - means descending].



Host Order is a way of telling the job how to select/order workers

- "+host.processors.avail" means prefer workers which have more slots available
- "+host.memory.avail" means prefer workers which have more memory available
- "+host.memory.total" means prefer workers which have more total memory
- "+host.processor_speed" means prefer workers with higher cpu speeds
- "+host.cpus" means prefer workers with higher total cpu slots

Requirements

Worker properties needed to be met for job to run on that Worker (comma-separated, expression-based). Click 'Browse' to choose from a list of Host Order Options.

Requirements is a way to tell the workers that this job needs specific properties to be present in order to run. The drop-down menu allows a choice of OS:

- "winnt" will fill the field with "host.os=winnt" which means only run on Windows based workers
- "linux" will fill the field with "host.os=linux" which means only run on Linux based workers
- "osx" will fill the field with "host.os=osx" which means only run on OSX based workers

You can also add any other Worker properties via plain text. Some examples:

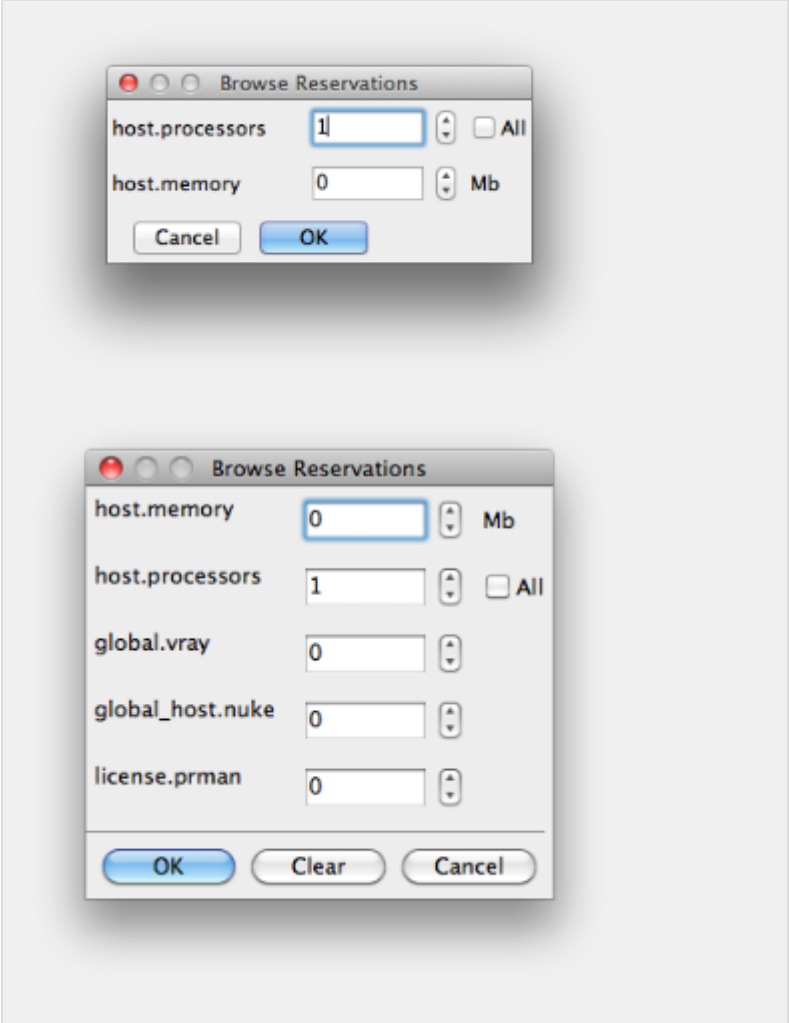
- "host.processors.avail.=4" means only run this job on workers that have 4 or more slots available
- "host.processors.used=0" means only run this job on workers with 0 slots in use
- "host.memory.avail=400" means only run this job on workers that have 400 memory available

With integer values, you can use any numerical relationships, e.g. =, <, >, <=, >=. This won't work for string values or floating point values. Multiple requirements can also be combined with AND and OR (the symbols && and || will also work).

The 'Only 1 of a "kind" of job' checkbox will restrict a Worker to running only one instance with a matching "kind" field (see below). The prime example is After Effects, which will only allow a single instance of AE on a machine. Using this checkbox and the "Kind" field, you can restrict a Worker to only one running copy of After Effects, while still leaving the Worker's other slots available for other "kinds" of jobs.

Reservations

Worker resources to reserve when running job (comma-separated, expression-based).



Reservations is a way to tell the workers that this job will reserve the specific resources for this job.

Menu items:

- "host.processors" this will fill the field with "host.processors=X" which means reserve X slots on the worker while running this job
- "host.memory" this will fill the field with "host.memory=X" which means only reserve X memory on the worker while running this job

Other options:

- "host.license.nuke=1" when a Global Resources entry has been made you can reserve any arbitrary named item. **New in 6.6:** Once you create a global resource, it will show up in this menu (eg global.vray above).
- See also Job Reservations

Restrictions

Restrict job to run only on specified clusters ("||"-separated) [+ means all below, * means at that level]. Click 'Browse' to choose from a list of Restrictions Options.

Restrictions is a way to tell the workers that this job can only run on specific clusters. You can choose more than one cluster in the list.

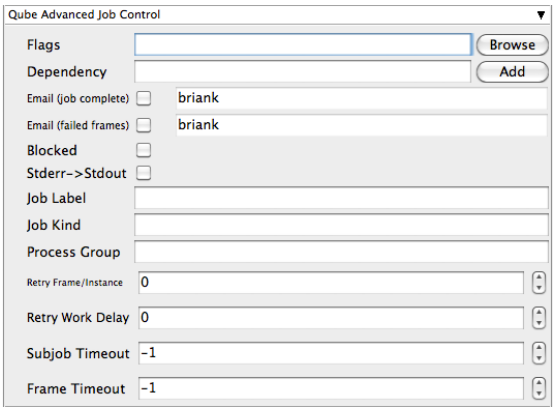
Examples:

- Choosing /showA would restrict the job to machines that are only in the /showA cluster, and no other cluster, not even those below /showA.
- Choosing /showA/* would restrict the job to the cluster(s) *below* /showA, but *not including* /showA
- Choosing /showA/+ would restrict the job to /showA and all the clusters below it.

See Also

- Controlling Host Selection
- How to use qbwrk.conf
- worker_groups
- worker_cluster
- How to use clustering for workers

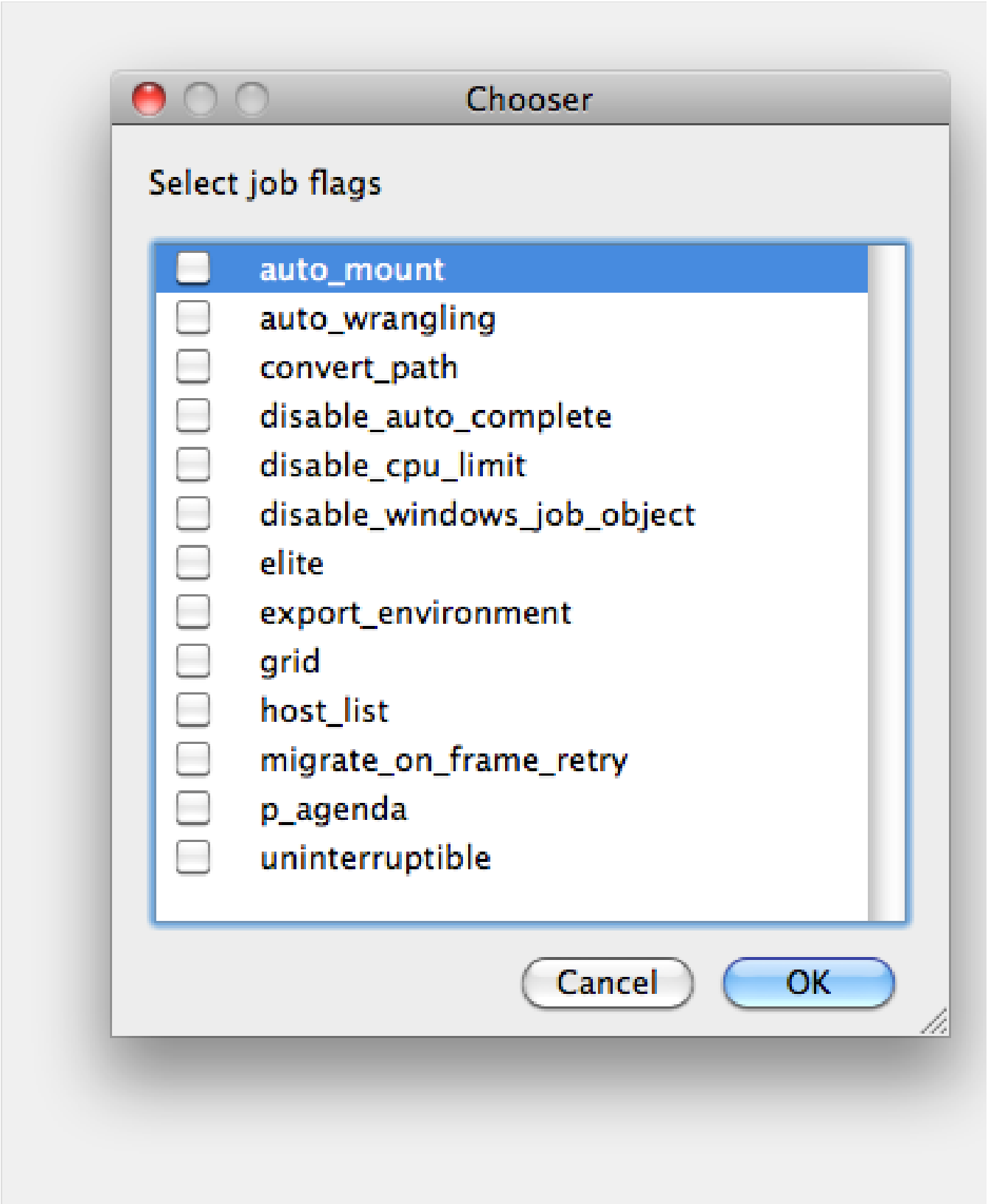
Qube Advanced Job Control



Click here for details...

Flags

List of submission flag strings (comma separated). Click 'Browse' to choose required job flags.

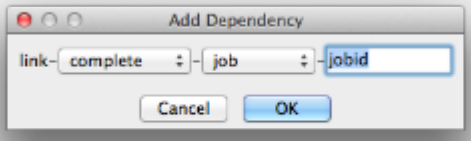


See this page for a full explanation of flag meanings

Dependency

Wait for specified jobs to complete before starting this job

(comma-separated). Click 'Add' to create dependent jobs.



You can link jobs to each other in several ways:

- "complete" means only start this job after designated job completes
- "failed" means only start this job if the designated job fails
- "killed" means only start this job if the designated job has been killed
- "done" means start this job if the designated job is killed/failed/complete

The second menu chooses between "job" (the entire set of frames) and "work" (typically a frame). So to link frame 1 of one job to frame 1 of a second, job, you would choose "work" in this menu. If you want to wait for all the frames of one job to complete before starting a second, then choose "job". The other option, "subjob", refers to the instance of a job. This is much less common, but means that, for example, the instance of Maya that was running frames has completed.

For a complete description on how to define complex dependencies between jobs or frames, please refer to the Callbacks section of the Developers Guide.

Email (job complete)

Send email on job completion (success or failure). Sends mail to the designated user.

Email (failed frames)

Sends mail to the designated user if frames fail.

Blocked

Set initial state of job to "blocked".

Stderr->Stdout

Redirect and consolidate the job stderr stream to the stdout stream. Enable this if you would like to combine your logs into one stream.

Job Label

Optional label to identify the job. Must be unique within a Job Process Group. This is most useful for submitting sets of dependent jobs, where you don't know in advance the job IDs to depend on, but you do know the labels.

Job Kind

Arbitrary typing information that can be used to identify the job. It is commonly used to make sure only one of this "kind" of job runs on a worker at the same time by setting the job's requirements to

include "not (job.kind in host.duty.kind)". See How to restrict a host to only one instance of a given kind of job, but still allow other jobs

Process Group

Job Process Group for logically organizing dependent jobs. Defaults to the jobid. Combination of "label" and "Process Group" must be unique for a job. See Process group labels

Retry Frame/Instance

Number of times to retry a failed frame/job instance. The default value of -1 means don't retry.

Retry Work Delay

Number of seconds between retries.

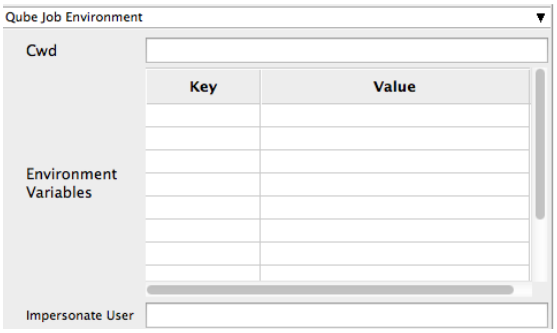
Subjob Timeout

Kill the subjob process if running for the specified time (in seconds). Value of -1 means disabled. Use this if the acceptable instance/subjob spawn time is known.

Frame Timeout

Kill the agenda/frame if running for the specified time (in seconds). Value of -1 means disabled. Use this if you know how long frames should take, so that you can automatically kill those running long.

Qube Job Environment



Click here for details...

Cwd

Current Working Directory to use when running the job.

Environment Variables

Environment variables override when running a job. You can specify key/value pairs of environment variables

This is useful when you might need different settings for your render applications based on different departments or projects.

Impersonate User

You can specify which user you would like to submit the job as. The default is the current user. The format is simply <username>. This is useful for troubleshooting a job that may fail if sent from a specific user.

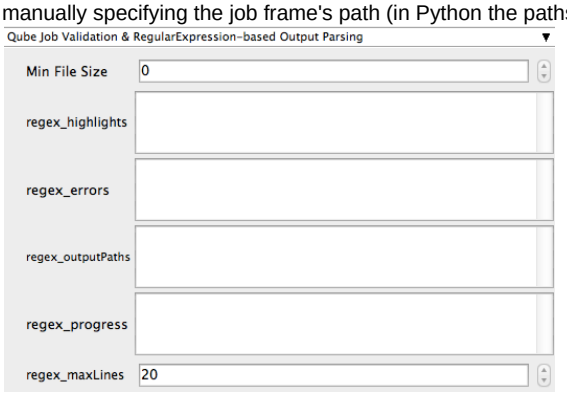
Example:
Setting "josh" would attempt to submit the job as the user "josh" regardless of your current user ID.

Note: In order to do this, the submitting user must have "impersonate user" permissions.

Qube Output Parsing and Validation

An optional basic file size check for the cmdline/cmdrange simplecmd framework jobtypes provides basic image validation if the path to the image can be determined. If an image is identified from parsing the stdout or stderr from a cmdline/cmdrange job (or any job using the simplecmd framework), it can have rudimentary validation automatically performed on it to make sure that it is valid.

For this to work, the **regex_outputPaths** submission parameter needs to be specified for it to parse the stdout/stderr of the job for images, or manually specifying the job frame's path (in Python the paths are represented by jobFrame['resultPackage']['outputPaths']).



Click here for details...

Min File Size

Used to test the created output file to ensure that it is at least the minimum size specified. Put in the minimum size for output files, in bytes. A zero (0) disables the test.

regex_highlights

Used to add highlights into logs. Enter a regular expression that, if matched, will be highlighted in the information messages from stdout/stderr.

regex_errors

Used to catch errors that show up in stdout/stderr. For example, if you list "error: 2145" here and this string is present in the logs, the job will be marked as failed. This field comes pre-populated with expressions based on the application you are submitting from. You can add more to the list, one entry per line.

regex_outputPaths

Regular expression for identifying outputPaths of images from stdout/stder. This is useful for returning information to the Qube GUI so that the "Browse Ouput" right-mouse menu works.

regex_progress

Regular expression for identifying in-frame/chunk progress from stdout/stderr.
Used to identify strings for relaying the progress of frames.

regex_maxlines

Maximum number of lines to store for regex matched patterns for stdout/stderr.
Used to truncate the size of log files.

> **Examples**
> To see examples of regular expressions for these contexts, look at the Nuke (cmdline) submission dialog - it has several already filled in.

Qube Actions

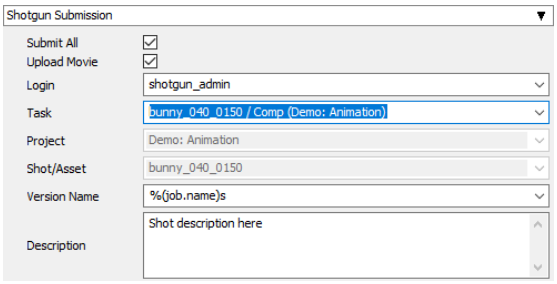
Click here for details...

GenerateMovie

Select this option to create a secondary job that will wait for the render to complete then combine the output files into a movie.

Note: For this to work correctly the "Qube (ImagesToMovie) Job..." has to be setup to use your studios transcoding application.

Shotgun Submission



Click here for details...

Shotgun Submission

- **Submit All:** Submit all entries in OutputPaths
- **Upload Movie:** Upload movie file in addition to setting "Path to Movie"
- **Login:** Login username for Shotgun
- **Task:** Shotgun task
- **Project:** Shotgun project
- **Shot/Asset:** Shotgun shot/asset
- **Version Name:** Shotgun version template. Performs variable substitution using python format %(key)s to get the value. The dict keys are prefixed with "job.", "package.", and "shotgun." respectively.
- **Description:** Shotgun description. Performs variable substitution using python format %(key)s to get the value. The dict keys are prefixed with "job.", "package.", and "shotgun." respectively.

Qube Notes

Click here for details...

Account

Arbitrary accounting or project data (user-specified). This can be used for creating tags for your job.

You can add entries by typing in the drop-down window or select already created accounts from the drop-down.

See also Qube! Job Tags

Notes

Freeform text for making notes on this job. Add text about the job for future reference. Viewable in the Qube UI.