

# Clarisse Job Submission

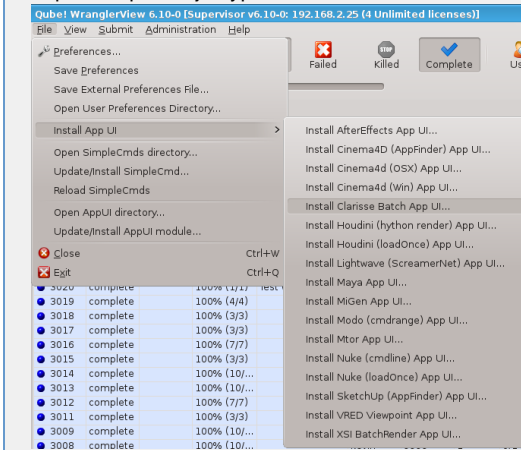
Step by step instructions for submitting Clarisse jobs with Qube!

## Step 1 (First Time Only)

### Install the Qube! Submission Menu

Open WranglerView on the machine you are going to submit from, and install the submission UI by locating it in the File > Install App UI > Install Clarisse Batch App UI.. line, as shown here.

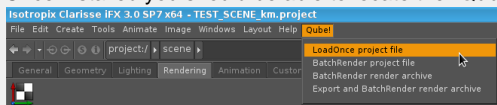
The Clarisse Jobtype comes with the standard install of Qube! and does not require a specific jobtype to be installed.



## Step 2

### Open the Submission Menu

Once installed you should be able to locate the "Qube!" menu in Clarisse.



### Step 3

This will present a pre filled submission UI. Ensure sections marked in red have the correct details.

#### Optional Settings

While not strictly required, the following settings are useful for getting a better result. You will need to turn on "Expert Mode" (check box at the bottom of the submission UI) in order to get access to them.

- By default, Qube! will not retry failed frames. So set the "Retry frame/instance" value to 3 or 5, to get 3 or 5 retries before it gives up. Related to that, the default wait between retries is zero seconds, which is typically not useful. Set the "Retry Work Delay" to a value like 5 or 10 to allow machines time to recover from temporary problems such as network errors.
- Make sure the correct version and location of the Clarisse installation is set, especially if you are submitting from one version but expecting to pick up jobs on another.

Submit Clarisse (loadOnce)

Qube Job Basics

Name: TEST\_SCENE\_km\_shot01

Priority: 5000

Instances: 1

Qube Frame Range

Range: 0-50x1

Render App Auto-threads & Uses the Entire Worker

Render on all cores:  (Reserves the entire worker)

Min Free Slots: 7

Clarisse Paths

Executable: /home/kevin/Downloads/clarisse/cnode

Render File: /home/kevin/Dropbox/PipelineFX/TEST\_SCENE\_... (Browse)

Output File Path:

Image Prefix:

Image/Layer: project://scene/shot01

Format: Open EXR (16bit)

Clarisse Options

Qube Worker Selection

Qube Advanced Job Control

Email (job complete):  kevin

Email (failed frames):  kevin

Set Defaults Clear Defaults  Expert Mode Cancel Submit

### Step 4

Click "Submit".

For further details on the submission UI see below.

Set Defaults Clear Defaults  Expert Mode Cancel Submit

#### Job Submission Details



Not all sections need to be filled in in order to render only the fields marked in red are required

Qube Job Basics	
Name	<input type="text"/>
Priority	9999
Instances	1
Max Instances	-1

Click here for details...

### Name

This is the name of the job of the job so it can be easily identified in the Qube! UI.

### Priority

Every job in Qube is assigned a numeric priority. Priority 1 is higher than priority 100. This is similar to 1st place, 2nd place, 3rd place, etc. The default priority assigned to a job is 9999.

### Instances

This is the number of copies of the application that will run at the same time across the network. The combination of "Instances=1" and "Max Instances=-1" means that this job will take as much of the farm as it can, and all jobs will share evenly across the farm.

Examples:

On a 12 slot(core) machine running Maya if you set "Instances" to 4

"Reservations" to "host.processors=3"

Qube! will open 4 sessions of Maya on the Worker(s) simultaneously, which may consume all slots/cores on a given Worker.

if you set

"Instances" to 1

"Reservations" to "host.processors=1+"

Qube will open 1 session of Maya on a Worker, consuming all slots/cores ("host.processors=1+" is used for all slots/cores).

### Max Instances

If resources are available, Qube! will spawn more than 'Instances' copies of the application, but no more than 'Max Instances'. The default of -1 means there is no maximum. If this is set to 0, then it won't spawn more than 'Instances' copies.

More on [Instances & Reservations & SmartShare Studio Defaults](#)

Qube Frame Range	
Range	1-100
Execution	Individual frames 1
rangeOrdering	ascending

Click here for details...

### Range

Frame range for the job (e.g 1-100, or 1-100x3, or 1,3,7,10)

Most jobs require a frame range to execute on the workers. You can set this range in a few different ways :

- "1-100" will just render the range between 1 and 100
- "1-100x3" will render the range 1 to 100, every third frame, so 1, 4, 7, etc.
- "1,3,7,10" will only render the selected frames 1,3,7,10

### Execution

How to break up frame range to be executed. Use QB\_START\_FRAME, QB\_END\_FRAME and QB\_FRAME\_NUMBER

When submitting a job to the farm it may be more efficient to "chunk" your job. This means that when the job is sent to the worker it tells the worker to render N consecutive frames before requesting more work. You would do this to keep from reopening the scene file for each frame. Large scene files can take substantial time to open, which is wasteful across dozens or hundreds of frames.

The drop down options are below:

- "Individual frames" this tells the worker to render 1 frame at a time.
- "Chunks with n frames" this tells the worker to render consecutively the number of frames specified in the field.
- "Split into n partitions" this tells the worker to render consecutively the total

frames in the range divided by the number in the field.

Examples:

- range 1-100 with "individual frames" set will render 1 frame at a time
- range 1-100 with "Chunks with n frames" and the field set to 5 will send 20 frames to each instance
- range 1-100 with "Split into n partitions" and the field set to 4 will send 25 frames to each instance

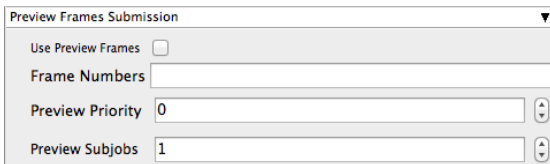
## rangeOrdering

Order to render the items.

(Ascending=1,2,3,4,5...,Descending=10,9,8...,Binary=first,middle,last...)

You can set the order in which your frames are rendered. The drop down options are:

- "Ascending" - this will render the frames counting upwards from your start frame
- "Decending" - this will render the frames counting backwards from your end frame
- "Binary" - This will render the first, last, and middle frames of the range, then the middle frame of the first half and the middle frame of the second half, and so on. This is useful for sampling the frames in the sequence to make sure it is rendering correctly.



Preview Frames Submission

Use Preview Frames

Frame Numbers

Preview Priority

Preview Subjobs

[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. This 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

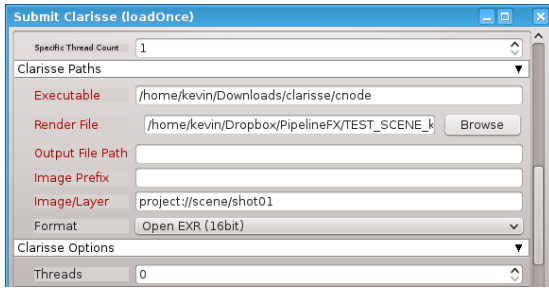
Choose the priority for the preview job. This can be set by the site admin.

## 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.

Parameters Specific to Clarisse



[Click here for details...](#)

## Clarisse Paths

### Executable

Explicit path to the Clarisse CNode executable.

### Render File

Browse or enter manually the location of the scene file to be rendered. This is a required field for submission.

Important: Best practice is to ensure the scene file and all of its dependent files such as textures are on network storage accessible by the workers.

### Output File Path

Specify the Directory for output.

### Image Prefix

Specify the padding on frame number with #.

### Image/Layer

Clarisse Image or layer to be rendered.

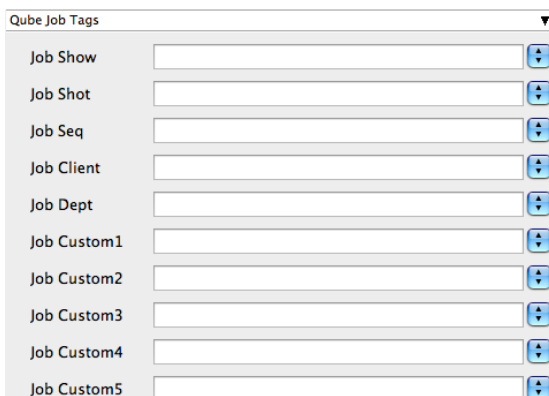
### Format

Specify the image format from the dropdown.

## Clarisse Options

### Threads

Specify the number of threads used by CNODE.  
Set 0 to use all available threads.



[Click here for details...](#)

## Qube Job Tags

### New in Qube 6.5

**Note:** The Job Tags section of the submission UI will not be visible unless they are turned on in the [Preferences](#) in the Wrangler View UI. Job Tags are explained in detail on the [Job Tags](#) page.

Qube Worker Selection		
Hosts	<input type="text"/>	<input type="button" value="Browse"/>
Groups	<input type="text"/>	<input type="button" value="Browse"/>
Omit Hosts	<input type="text"/>	<input type="button" value="Browse"/>
Omit Groups	<input type="text"/>	<input type="button" value="Browse"/>
Priority Cluster	<input type="text"/>	<input type="button" value="Browse"/>
Host Order	+host.processors.avail	<input type="button" value="Browse"/>
Requirements	<input type="text"/>	<input type="button" value="Browse"/>
Reservations	<input type="text"/>	<input type="button" value="Browse"/>
Restrictions	<input type="text"/>	<input type="button" value="Browse"/>

✓ [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

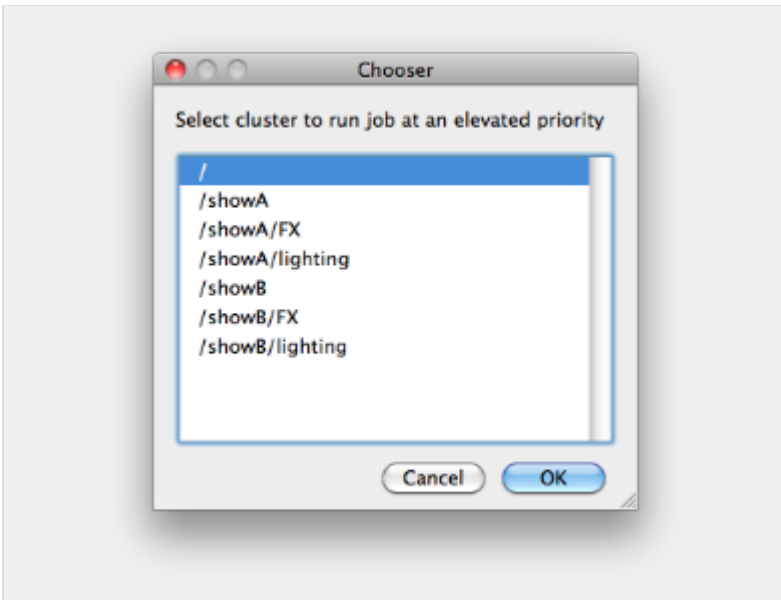
Explicit list of Worker hostnames that are **not** allowed run the job (comma-separated).

## 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](#)

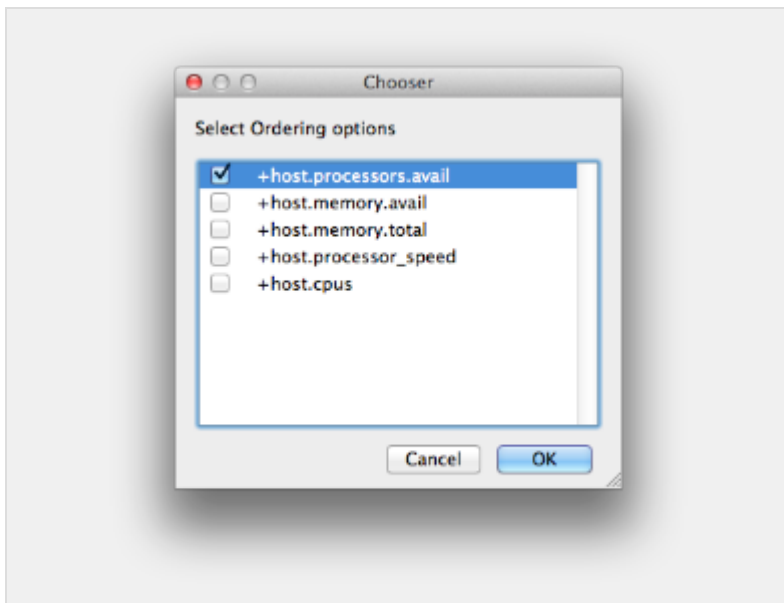


Example:

- 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

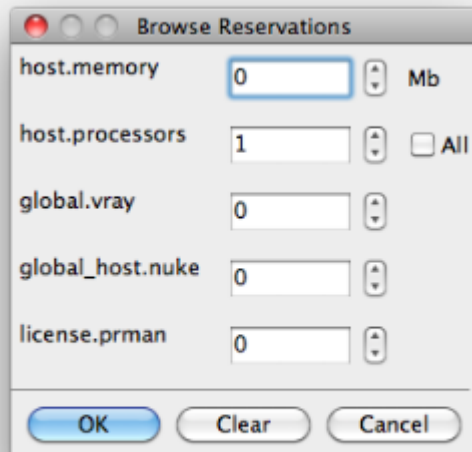
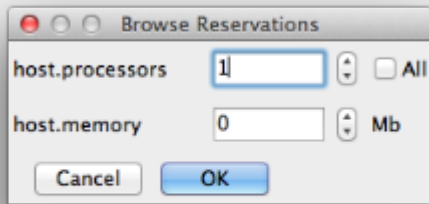
With integer values, you can use any numerical relationships, e.g. =, <, >, <=, >=. This won't work for string values or floating point values. Multiple requirements can also be combined with AND and OR (the symbols && and || will also work).

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.

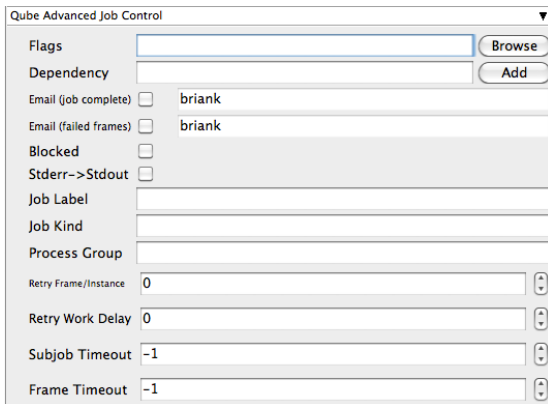
Restrictions is a way to tell the workers that this job can only run on specific clusters. You can choose more than one cluster in the list.

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](#)

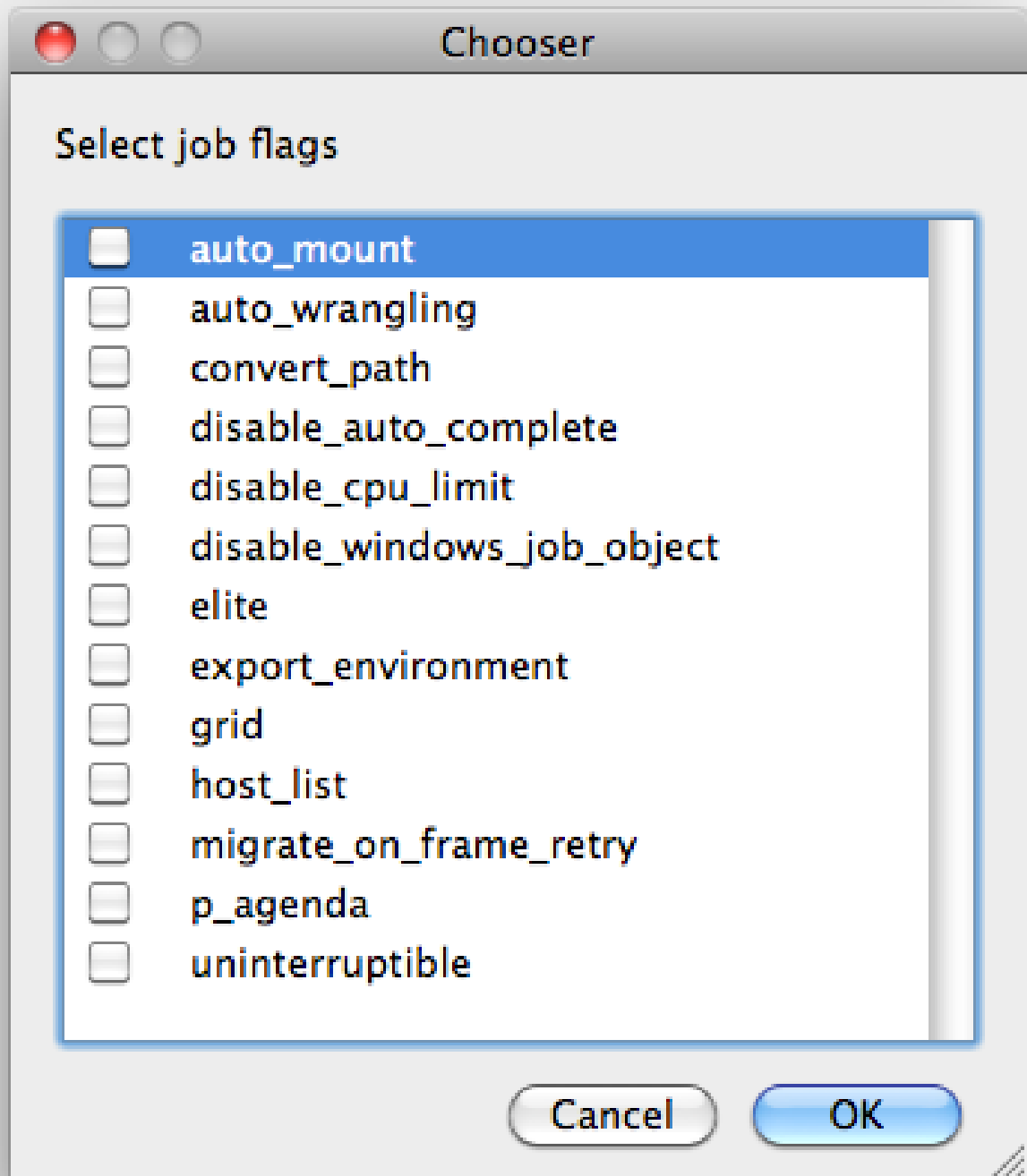


The screenshot shows a web form titled "Qube Advanced Job Control". It contains several input fields and checkboxes. The "Flags" field is empty and has a "Browse" button next to it. The "Dependency" field is empty and has an "Add" button next to it. There are two "Email" fields, both with the value "briank" and a checkbox to their left. Below these are "Blocked" and "Stderr->Stdout" checkboxes, both unchecked. There are three empty text input fields for "Job Label", "Job Kind", and "Process Group". At the bottom, there are four numeric input fields with spinners: "Retry Frame/Instance" (0), "Retry Work Delay" (0), "Subjob Timeout" (-1), and "Frame Timeout" (-1).

[Click here for details...](#)

## Flags

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

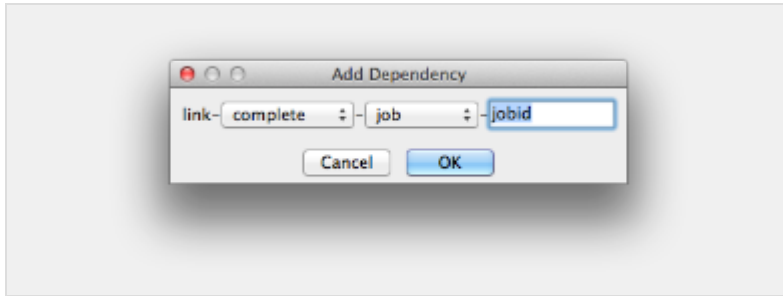


See [this page](#) for a full explanation of flag meanings

### 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.

FlightCheck scripts		
Job Pre-flight	<input type="text"/>	Browse
Job Post-flight	<input type="text"/>	Browse
Work Pre-flight	<input type="text"/>	Browse
Work Post-flight	<input type="text"/>	Browse

[Click here for details...](#)

### Job Pre-flight

Some executable (e.g. a script) that will run *on the Worker* before the job runs. If the pre-flight fails, the job will not run on this Worker, but will be moved to another. Useful for setting up environments, copying required images, checking the existence of fonts on the Worker, etc.

### Job Post-flight

An executable that will run *on the Worker* after the job has finished on that Worker. If this fails, the instance is failed.

### Work Pre-flight

Some executable (e.g. a script) that will run *on the Worker* before the frame/work runs. If this fails, the frame/work will be marked as failed and rendering will be skipped. If there are retries configured, the frame/work will be treated according to those settings.

### Work Post-flight

An executable that will run *on the Worker* after the frame/work runs. If this fails, the frame/work will be marked as failed, even if the render, itself, was successful. If there are retries configured, the frame/work will be treated according to those settings.

Qube Job Delayed Start		
hh:mm M/D/Y	<input type="text"/>	Time and Date

[Click here for details...](#)

### hh:mm M/D/Y

The job is blocked on the Supervisor until this day and time. Click on "Time and Date" to get a calendar and time UI.

The screenshot shows a window titled "Qube Job Environment". It contains three main sections: "Cwd" with a text input field, "Environment Variables" with a table, and "Impersonate User" with a text input field. The table has two columns: "Key" and "Value".

Key	Value

✓ [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 Job Validation & RegularExpression-based Output Parsing

Min File Size

regex\_highlights

regex\_errors

regex\_outputPaths

regex\_progress

regex\_maxLines

Qube Job Validation & RegularExpression-based Output Parsing

Min File Size

regex\_highlights

regex\_errors

regex\_outputPaths

regex\_progress

regex\_maxLines

[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/stderr. This is useful for returning information to the Qube GUI so that the "Browse Output" 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

generateMovie

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.

Qube Notes

Account

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.