Packages

A job in Qube consists of information used to 'route' the job to the proper host; to determine what priority the job carries; and how to execute the job.

In a simple queuing system, the information on how to execute a job is typically a single string representing the command line to be executed on the host/Worker machines. Qube generalizes this by abstracting the information required and allowing the transmission of complex abstract data structures between hosts, rather than just a single command line.

The argument for such a design is apparent when a developer tries to create a command line, which is longer than the physical limitation of the command prompt. It is even more apparent when the developer is forced to create complex character escape schemes to deal with special characters such as quotes and brackets. Qube's scripting API implementations automatically import and make use of the packaging libraries.

In the Python API, a job package is a dictionary which can contain any data that can be serialized. For those familiar with Perl, during the submission of a job, a developer is permitted to use hashes, arrays and scalars in order to define the data to be transmitted to the execution end of the job.

Example

```
my $job = {
name => "my job"

};
my $package = {

scenefile => "/my/scene/file/location.ma",

outputdir => "/location/to/my/images" };

$job->{package} = $package;

qb::submit($job);
```

In the example above, the developer chose to use a hash reference with the elements scenefile and outputdir. Once the job has been started on a host, the corresponding execution script is able to access this information by referencing the API.

Example

```
my $job = qb::jobobj();
print $job->{package}->{scenefile}, "\n"; print $job->{package}->{outputdir}, "\n";
```

This system promotes rapid development, and is a clean method of abstracting the submission methods from the execution code.
Note: In Qube this is called the front end and the back end.