AppFinder Jobs

AppFinder jobs require that Python be installed on each executing worker.

What are they, and what do they do?

These jobs differ from the standard SimpleCmd cmdline- and cmdrange-based jobs in that they will "find" a particular version of a 3rd-party application on the worker when the job runs. This has several benefits:

- The submitting user is not required to know that application's installation path on the remote worker.
- Since the application does not include a hard-coded path to the 3rd-party application, the job is able to run across machines of differing operating systems at the same time.

The appFinder jobs do not perform a "best match" for version number. For example, if AfterEffects CS5.5 is specified, but not installed, it will not look for CS6. Instead, a warning message will appear in the job logs, and the job instance will be restarted on another worker.

Applications currently supported as AppFinder jobs in WranglerView:

1. AfterEffects (all versions)
2. Cinema4D (R14 and later)
3. Maya BatchRender

How do they work?

When the job is submitted, the command line contains a "application template" that looks like __XYZ__ The XYZ portion between the double-underscores specifies which application to run. Currently, the following application templates are supported, as specified in the API library file `$QBDIR/api/python/qb/backend/appDefaultPaths.py`

- AE
- C4D
- NUKE
- MAYA
- XSI

The job's `package` dictionary will contain an `appVersion` python tuple which specifies what version of that application to search for. So if the command-line contains an __AE__ application template, and the package's appVersion is the tuple (6,), then AfterEffects CS6 will be used on the worker. To specify CS5.5, the appVersion would be set to (5,5).

The jobtype backend code on the worker will examine the job's command-line and appVersion values, and check and see if a suitable application can be found on the worker. Only the 3rd-party software developer's default installation paths for a given application are scanned. See the section Supporting non-default installation paths with AppFinder if your studio installs applications in non-standard locations.

If a suitable application installation is found on the worker, the application template in the command-line is replaced with the path to the application's executable. Something like the following should appear in the jobs' `STDERR` logs:

On an OS X worker:

```
INFO:CmdRangeBackEnd: attempting auto-pathing
INFO:CmdRangeBackEnd: Paths in the command have been translated as per this worker's worker_path_map
INFO:CmdRangeBackEnd: __C4D__ -nogui -frame 67 67 1 -render "/Users/jburk/Documents/C4D/test.c4d"
INFO:CmdRangeBackEnd: -> "/Applications/MAXON/CINEMA 4D R14/CINEMA 4D.app/Contents/MacOS/CINEMA 4D"
-nogui -frame 67 67 1 -render "/Users/jburk/Documents/C4D/test.c4d"
```

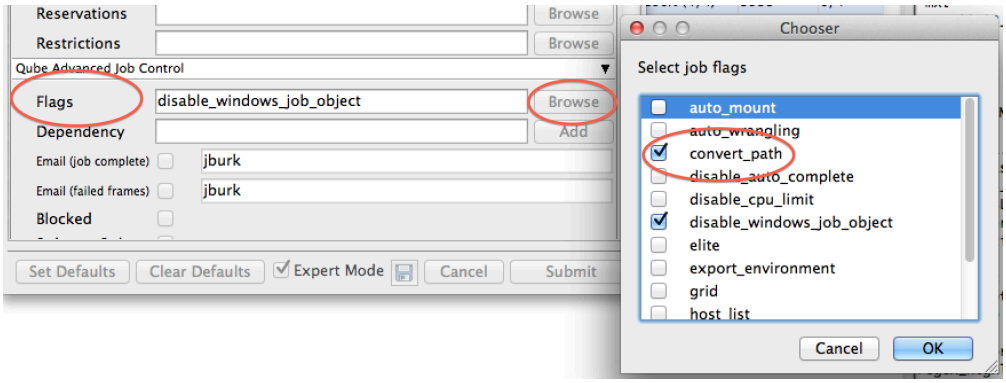
The same job on a Windows worker:

```
INFO:CmdRangeBackEnd: attempting auto-pathing
INFO:CmdRangeBackEnd: Paths in the command have been translated as per this worker's worker_path_map
INFO:CmdRangeBackEnd: __C4D__ -nogui -frame 3 3 1 -oimage "/Users/jburk/Documents/C4D/renders/test"
-render "/Users/jburk/Documents/C4D/test_R14.c4d"
INFO:CmdRangeBackEnd: -> "C:\Program Files\MAXON\CINEMA 4D R14\CINEMA 4D 64 Bit.exe" -nogui -frame 3 3
1 -oimage "Z:/Documents/C4D/renders/test" -render "Z:/Documents/C4D/test_R14.c4d"
```

Supporting non-default installation paths with AppFinder

If your studio does not install 3rd-party applications into standard locations, there are 2 approaches you can take:

1. Replace the application template __XYZ__ with the full path to the executable on the worker (not recommended)
2. Define your own application template in the worker's worker_path_map, and use that application template in the "executable" field in the job submission UI. NOTE**:** this approach also requires that you set the 'convert_path' job flag in the submission UI's "Advanced Job Control" section.



For example, to implement approach 2 for a non-default installation of AfterEffects CS6, you could decide to use an application template of __AE6__. Then, define this value in the worker's worker_path_map value (which can be managed via the central worker configuration file qbwork.conf).

Windows example:

```
worker_path_map = {
    "/Users/jburk" = "Z:"
    "/Users/jburk/test" = "Y:"
    "/tmp" = "C:/temp",
    "__AE6__" = "D:/Programs/AfterEffects/CS6/aerender.exe"
}
```

OS X worker example:

```
worker_path_map = {
    "Z:" = "/Users/jburk"
    "Y:" = "/Users/jburk/test"
    "C:/temp" = "/tmp"
    "__AE6__" = "/Volumes/HD2/3rd Party Apps/Adobe/AfterEffects/CS6/aerender.exe"
}
```