Python job submission with Python language callback that runs an external script on the supervisor

Supervisor_language_flags
supervisor_language_flags must contain "python" or all python callbacks will be ignored.

Sometimes the best way to perform complex behavior in a callback is to put that behavior into an external script which can take the job ID as an argument.

The following is an example of a callback that runs for every frame when the job completes. Admittedly, it may be more efficient to run it once and have the external script iterate over every frame, but this is for demonstration purposes.

Use Popen.subprocess in the callback
Please note the use of `Popen.subprocess()` to run the external script in the callback. This returns immediately and allows the callback to continue running, rather than blocking and waiting for the external script to complete; this ties up the supervisor process for the duration of the external script's execution.

Do **not** use os.system() to run the external script, as this call will block until the external script exits. When a large number of callbacks tie up supervisor processes at the same time, your supervisor performance will suffer.

Never use sys.exit() in a callback
Do not call `sys.exit()` at the end of the callback code, this kills the calling supervisor process.

```
#!/bin/env python

import sys,os

sys.path.append('%s/api/python' % os.environ['QBDIR'])
import qb

scriptPathOnSupervisor = '/Users/jburk/test/testCbScript.py'

########################################
# build the job
########################################
job = {
    'prototype': 'cmdrange',
    'name': 'supervisor-side script execution callback test',
    'package': {'cmdline': 'hostname'},
    'agenda': [],
    'callbacks': [ ]
}
########################################
# iterate over the frame range to build the job's agenda and callbacks
########################################
for i in range(5):

    # an agenda item can simply be a frame number
    work = {'name': i}
    job['agenda'].append(work)

    # build a callback for each item in the agenda (each frame)
    cb = {
        'language': 'python',
        'triggers': 'done-job-self',
        'code': '''
import os,subprocess

jobId = qb.jobid()
workName = %s

script = '%s'
if os.path.exists(script):
    pid =  subprocess.Popen(["python", script, str(jobId), str(workName)]).pid
''' % (i, scriptPathOnSupervisor)
    }

    # append the frame's callback to the job's callback list
    job['callbacks'].append(cb)

########################################
# submit the job
########################################
submitted = qb.submit(job)
for job in submitted:
    print 'submitted %(id)s: %(name)s' % job
```

Python path

In line 41 of the above script, using "python" assumes that python is in the supervisor user's PATH environment. If not, use the full path to python. If the python script can be invoked on its own (without calling it as an argument to the python executable itself), then you can leave out "python" all together.

Execution
Remember, callbacks are executed on the supervisor by the user who is running the supervisor (typically the local system account or "root"). Keep this in mind when reading from or writing to the shared file system.