Universal Callbacks

New in Qube 6.6

Overview

Universal callbacks are operations which are automatically attached to every job that is submitted. These are installed by a site administrator either on the Supervisor's local disk or on a file system which is accessible by the supervisor service.

Setting Up Universal Callbacks

To set up Universal Callbacks, the site administrator needs to make a directory (default $QBDIR/callback) and create a text-based configuration file, called "callbacks.conf" in that directory. Further, in the same directory, files containing the implementation (aka "code") of each Universal Callback must also be installed.

The callbacks.conf file serves as a map that tells the system which callback code should be triggered to run on what events.

The "callbacks.conf" file

The callbacks.conf file is a text file, much like qb.conf, containing one or more lines with a "key = value" pair, associating each implementation file to a trigger event. The syntax is:

```
# lines starting with a hash mark are comments
filename = trigger
```

The **filename** points to a file in the same directory that implements the callback code. Note that Universal Callbacks support Python, Perl, and Qube callbacks, and the filename must have the extension .py, .pl, or .qcb, respectively.

The **trigger** specifies the triggering event that activates the callback, described in details at Triggers

Here's an example:

```
#
# callbacks.conf
#
# syntax of this file is :
# filename = triggers
#
logFailuresToDB.py = failed-job-self
mail-status.qcb = done-job-self
checkWork.pl = done-work-self-*
submitted.py = submit-job-self
```

In this example, there are presumably 4 implementation files in the callback directory, logFailuresToDB.py, mail-status.qcb, submitted.py, and checkWork.pl, that have the implementation code in them.

Use subprocess.Popen in callbacks
If you ever need to run an external script in a callback, we recommend the use of `subprocess.Popen()` to run the external script inside the callback. This returns immediately and allows the callback to continue running, rather than blocking and waiting for the external script to complete; otherwise the supervisor process is tied up 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.

submitted.py

```
#!/usr/bin/env python

import sys
import qb
import traceback

fh = open('/tmp/univeral_callback_test', 'a')
try:
    # ==================================================
    #  === NOTE: ===
    #  the qb.jobinfo() in callbacks is not the
    #  same as the one in the external python API
    # ==================================================
    job = qb.jobinfo("-id", qb.jobid())[0]
    fh.write('submitted %(id)s: %(name)s\n' % job)
except:
    fh.write(traceback.format_exc())
fh.close()
```

Universal Callbacks vs. FlightChecks

At first glance, Universal Callbacks and the Job Pre- and Post-FlightChecks appear similar, but they have an important difference:

- **Universal callbacks** are run by and on the **supervisor** host.
- **Flight checks** are only run on the **worker** hosts.

qb.conf Parameters

supervisor_universal_callback_path

- Path to the directory where Universal Callbacks (the callbacks.conf file and the implementation files) are found
- May be a comma-separated list to specify multiple locations
- default: $QBDIR/callback