Site Status Filter

Overview

Sites can extend the functionality of a blade through a class defined in TractorSiteStatusFilter.py. Customizations include:

- addition of metrics to determine a blade's readiness to request more work,
- responses to events such as command launches, launch failures, completions, and errors, and
- responses to patterns in command output, such as the suppression of verbose output or the early termination of a command due to an emitted warning.

Installation

TractorSiteStatusFilter.py must be installed in a directory visible to a blade, ideally the directory defined by SiteModulesPathin blade.config.

To start, copy lib/python2.7/site-packages/tractor/apps/blade/TractorSiteStatusFilter.py from the Tractor install directory to the SiteModulesPath directory.

Note that it will be necessary to reload blade.config for blades to reflect changes to SiteModulesPath or to TractorSiteStatusFilter.py.

Scenario: New Metrics

New metrics can be employed to determine the ability of a blade to perform work by overriding the FilterBasicState() andTestBasicState() methods. In the following example, www.example.com might represent a resource such as a license server that needs to be reachable before a blade would consider requesting work.

```
    def FilterBasicState(self, stateDict, now):
        if stateDict is not None:
            rcode = os.system("ping -q -t 1 -c 1 www.example.com >& /dev/null")
            stateDict["server_reachable"] = (rcode == 0)
        self.super.FilterBasicState(stateDict, now)

    def TestBasicState(self, stateDict, now):
        if stateDict is not None and not stateDict.get("server_reachable"):
            return False
        return self.super.TestBasicState(stateDict, now)
```

Scenario: Logging Errors

One could maintain a distinct log of errored processes by overriding SubprocessEnded().

```
    def SubprocessEnded(self, cmd):
        if cmd.exitcode:
            with open("/tmp/errors.log", "a") as f:
                f.write("process %d errored with exit code %d\n" % (cmd.pid, cmd.exitcode))
        self.super.SubprocessEnded(cmd)
```

Scenario: Early Process Termination

FilterSubprocessOutputLine() is called by the blade for each line of output from a process.

In this example, any prman process that emits a line starting with R10007 will be immediately terminated. The logic could be modified to examine the output of other applications, look for other patterns, and peform other actions. Note that non-matches should defer to FilterSubprocessOutputLine() of the superclass.

```
    def FilterSubprocessOutputLine (self, cmd, textline):
        '''
        Return a 2-tuple giving a disposition code and a value.
        The pass-through result would be:  (self.TR_LOG_TEXT_EMIT, textline)
        '''
        if "prman" == cmd.app and textline.startswith("R10007"):
            # example:  immediately fail on prman warning number "R10007"
            return (self.TR_LOG_FATAL_CODE, 10007, textline)
        else:
            # otherwise defer to the usual built-in filters
            return self.super.FilterSubprocessOutputLine( cmd, textline )
```