Blade Environment Configuration: Keys and Handlers

Overview

When tractor-blade launches a new command, such as a render, it sets up a unique set of environment variables for that subprocess. There are several mechanisms for defining these environment variables and their values.

The default approach is to simply allow each command to clone the incoming environment inherited by tractor-blade when it was started. However, this is often a specialized "daemon" environment and thus not very useful.

The simplest approach to custom environment configuration is to add name-value pairs to an environment dictionary in the blade.config file (see below). This is often completely sufficient, and is very straightforward. There is a "default" dictionary, and there can be separate ones corresponding to incoming EnvKey values from commands in the job. Environment variables specified in this manner are additions and overrides to the incoming inherited environment.

Tractor *envhandlers* are Python objects that can apply dynamic, site-specific settings to each command launched by a tractor-blade. They are most often used to configure a specific set of environment variables based on the inbound EnvKey or the job's owner. These handlers can also rewrite the actual command that is being invoked as well, which can help to deal with particular machines that have a non-standard application installation location, or other quirks.

Environment Variable Dictionaries

Additions and overrides to the inherited external environment can be added to one or more Python dictionaries that will be loaded by the blade. They can be added to the "default" EnvKey definition to affect all commands, or added to the settings for a particular EnvKeyvalue. Similarly, definitions can be added to ProfileDefaults or specifically to a single profile.

```
{
    "ProfileDefaults":
    {   "ProfileName": "default",
        ...
        "EnvKeys": [
            {
                "keys": ["default"],
                "environment": {
                    "SOME_VAR": "xyzzy",
                    "CURRENT_SHOW": "(not_set)"
                },
                "envhandler": "default"
            },
        ]
    },

    "BladeProfiles":
    [...]
}
```

These environment dictionaries are either defined in the blade.config file, or can be contained in an external file that is referenced with the @merge directive in the blade.config file. The default installation includes three external files which contain the default environment dictionary setup. These files are shared.linux.envkeys, shared.macosx.envkeys and shared.windows.envkeys.

Environment Handlers

Default Environment Handlers

Tractor comes with a default set of environment handlers. The handlers are invoked when there is a match between the incoming envkey and the **keys** defi nition in the environment key dictionary. The envkey parameter in a tractor script can actually be a list of several envkey entries. In that case each handler will be called in sequence to modify the launch environment.

Each of the default handlers has a specific purpose.

- **default** - The handler default is run on every command. It is run first, and the environment key dictionary should be setup with an intended default environment, rather than relying upon the environment of the launched blade.

  As provided the default handler simply sets the spooling host into the launch environment as **REMOTEHOST**. Default environment variables and paths should be setup here.
- **rmshandler** - The rmshandler responds to an incoming key of format rms-*-maya-*. The * indicates a globbing will be applied. This handler matches incoming keys like: rms-3.0.0-maya-2011.

  This handler extracts the RMS version, and the MAYA version and sets a number of environment variables including **RMSTREE**, **MAYA_LOCATI ON**, **MAYA_MODULE_PATH**. It also adds **RMSTREE/bin** and **MAYA_LOCATION/bin** into the path. These locations are determined by knowing the default install locations for these products on the specific platform. The handler will attempt to verify that RMS and Maya are installed before setting these variables.
- **mayahandler** - The mayahandler responds to an incoming key of format maya*. The * indicates a globbing will be applied. This handler matches incoming keys like: maya2011.

The mayahandler provides similar functionality to the rmshandler, but is designed for Maya users who do not have RMS installed, but do Maya batch rendering. This handler extracts MAYA version and sets a number of environment variables including **MAYA_LOCATION**, **MAYA_MODULE_PATH**. It also adds **MAYA_LOCATION/bin** into the path. The Maya location is determined by knowing the default install location for Maya on the specific platform. The handler will attempt to verify Maya is installed before setting these variables.

- **rmanhandler** - The rmanhandler responds to an incoming key of format prman-*. This key is used to define the version of prman to be used, and is quite often used in conjunction with the rmshandler key above.

  This handler would match an incoming key like: prman-15.2. On a match the handler sets the environment variable **RMANTREE**, and adds **RMANTREE/bin** into the path. The location of RMANTREE is determined by knowing the default install location on the specific platform. The handler will actually attempt to determine whether the prman version is installed before setting the **RMANTREE** environment variable.
- **rmantreehandler** - The rmantreehandler is a special handler, which will generally not be used by customers. This handler responds to a key of format: rmantree=*. This handler is designed to work like the rmanhandler above, however, the key provides the entire path to a non-standard RMANTREE installation. For example, the incoming key might look like:rmantree=/opt/software/pixar/RenderManProServer-15.2. The entire path as defined in the key is set as the **RMANTREE** variable, and **RMANTREE/bin** is added into the path.
- **rmstreehandler** - The rmstreehandler is similar to the rmantreehandler above, however it is designed to point to a non-standard RMSTREE location. The handler responds to an incoming key of format: rmstree=*. A typical usage might bermstree=/opt/software/pixar/RenderManStudio-3.0.0-Maya-2011. This handler sets the **RMSTREE** environment variable, and adds **RMSTREE/bin** into the path.

The default handler source code is available and is located in the TrEnvHandlers.py file located in the blade-modules directory of the blade installation.

Adding A Custom Environment Handler

- **Pick a name for your handler** - it should be vaguely descriptive of the type of EnvKeys it handles. For the examples below, let's assume there is an inbound EnvKey called "ProjectX" and so we will call our handler "projxhandler".
- **Pick a directory to hold your Python modules** - typically this will be a location on a central fileserver that is accessible to all blade hosts.
- **Add the selected directory to SiteModulesPath in** blade.config - if all hosts mount the shared location using the same path, then it can go in the ProfileDefaults section, otherwise add customized variants to each profile as needed (shown below).
- **Also add an** EnvKeys **entry for your new key** - leave the key named "default" in the dictionary as well. Then add your module name as the envhandler value.

```
{
    "ProfileDefaults":
    {   "ProfileName": "default",
        ...
        "SiteModulesPath": "/net/yourserver/tractor_extensions",
        "EnvKeys": [
            {
                "keys": ["default"],
                "environment": {
                    "SOME_VAR": "xyzzy",
                    "CURRENT_SHOW": "(not_set)"
                },
                "envhandler": "default"
            },
            {
                "keys: ["ProjectX", "SHOT=*", "SCENE=*"],
                "environment": {
                    "CURRENT_SHOW": "Project X, Venture of Mystery",
                    "CURRENT_SHOT": "$SHOT",
                    "CURRENT_SCENE": "$SCENE",
                    "RMANTREE": "/opt/pixar/RenderManProServer-15.2",
                    "PATH": "$RMANTREE:/bin;$PATH"
                },
                "envhandler": "projxhandler"
            },
        ]
    },

    "BladeProfiles":
    [...]
}
```

- **Also see the discussion on** creating custom environment handlers