Crews

The crews.config file defines user access to Tractor, defining who can login and their level of control over aspects of the system. The crews.config file is typically copied in its entirety to the [site override directory](#) and then customized there.

Definition

A crew is simply a named list of users:

```
"the_crew_name":  ["john", "paul", "george", "ringo"],
```

Reserved Crew Names

In addition to any site-defined crews, there are four reserved crews names used for special purposes by Tractor: ValidLogins, BannedLogins, Wranglers, Administrators.

The ValidLogins crew enumerates all user's who are allowed to login to Tractor. This crew *must* exist. See the note regarding the meta-names @syslogins and @externlogins below.

The BannedLogins crew lists user names that are explicitly denied login. It is mostly only useful for providing exceptions whenValidLogins contains either of the meta-names @syslogins or @externlogins.

Meta Crew Names

The user name list for a given crew can include either of these special names. @syslogins stands for all user login names that are valid locally on the engine host (i.e. resolvable by the function getpwnam(3)), which frequently includes those resolvable by the site nameservers, such as LDAP or NIS.

```
"ValidLogins": ["@syslogins"],
```

The name @externlogins causes the engine to skip its internal validation of a potential Dashboard user names, and instead the engine passes all login names directly to the site's external trSiteLoginValidator.py script (or whichever validation app is named as the SitePasswordValidator below). The external validator can accept or deny a given name+password pair.

```
"ValidLogins": ["@externlogins"],
```

Note that wildcard characters in user names (e.g. *) are NOT expanded or treated specially when searching for user access. However, the list of user names can also contain OTHER CREW names, which will be expanded recursively to include the users in those crews. Crew name references that form "reference loops" will be truncated, but may cause unexpected memberships.

Any user name in a crew's membership list whose first character is a minus-sign will be removed from that crew definition, similarly for sub-crew names.

```
"ourRestrictedCrew": ["ourBigCrew", "-knownMiscreants"],
```

Crews that are referenced in name lists can be denoted with a leading $ in order to make them clearly distinguished as crew names rather than user names. This approach also allows tractor-engine to issue better diagnostic messages when a referenced crew is unknown -- otherwise than name is just treated as another user name.

```
"theLuncheon": ["terry", "vandana", "$theGuysFromShipping"],
```

A particular user name may appear in more than one crew.

```
"Crews": {
    "ValidLogins": ["@syslogins"],
    "BannedLogins": [],

    "Wranglers": [],
    "Administrators": ["root"]
},
```

Job Edit Permissions

By default, everyone with a valid login is given "standard" permissions, meaning the basic ability to run and modify their own jobs and to view the state of the queue and the blades. Users listed in the Wranglers crew are also allowed to alter the general job queue, including the state of other user's jobs. Users listed in the Administrators crew are allowed to change site-wide policies through the URL interface. The permission levels are cumulative: Wranglers have "standard" access plus the additional wrangler permissions. Administrators have standard, wrangler, and admin access.

The optional subdictionary JobEditAccessPolicies can be used to further restrict or extend edit access to job attributes, such as job priority. A different access list can be specified for each editable attribute, otherwise the settings for the entry named default will apply. The special meta-crew name @owner in these edit lists grants each job owner the permission edit the given attribute in their own jobs.

Edit permissions are organized into named *policies* with the one named defaultPolicy applied by default. Job scripts can specify the policy that applies to them using the Job -editpolicy {NAME} option. Typically this is used by a studio's job creation pipeline to apply additional restrictions on jobs from particular shows or administrative jobs. In cases where the job's specifically requested policy is missing here, then the engine applies the settings from the defaultPolicy entry, if it exists, rather than denying all edits until the missing policy becomes defined. Users in the Administrators crew will always have edit control over any job.

The recognized fine-grained edit permission keywords include: argv, afterjids, aftertime, bnotes, jnotes, comment, crews, cwd, delete, envkey, interrupt, jrestart, jshuffle, lock, metadata, pause, priority, retract, retry, runsecsbounds, service, skip, slotbounds, tags, title

```
    "JobEditAccessPolicies": {
Note:  don't forget commas after each entry!

    "defaultPolicy": {
    "priority": ["Wranglers", "Administrators"],
    "tier":     ["Wranglers", "Administrators"],
        "default": ["@owner", "Wranglers", "Administrators"]
    },

    "SomeCustomPolicyName": {
        "priority": ["Wranglers", "Administrators"],
        "tier":     ["Wranglers", "Administrators"],
        "default": ["jedi", "Administrators"]
    }
},
```

SitePasswordValidator

SitePasswordValidator specifies a validation scheme for passwords sent to the engine from user interfaces and scripts. Use an empty string "" to disable password checking, allowing any valid user name to proceed as a client (the names authorized in ValidLogins).

Any non-empty value for SitePasswordValidator will cause passwords to be required, and specifies the type of validation that will be applied. All user+password validation occurs on the tractor-engine host, so values here are relative to the engine's environment, filesystem, operating system, etc.

Tractor engine has built-in support for the "PAM" authentication system provided by Linux and Mac OS. Only the engine needs this support, clients can be on any platform. Use the string value internal:PAM below to enable password validation through the PAM modules on your engine host.

The PAM scheme itself requires you to choose an appropriate access policy module (see /etc/pam.d or /etc/pam.conf). Tractor will attempt to load a module named "tractor" by default, and you must define that module first, using appropriate policies for your site. Alternatively, you can specify a different PAM module name here, one that may already exist and support your site's requirements. For example to get the su authentication rules, use internal:PAM: su below.

Tractor-engine sends a session cookie to the dashboard web browser after you authenticate with a password. The saved session information can be used later for automatic login when a fresh dashboard page is reloaded. This auto-login behavior can be disabled, forcing users to reenter their passwords when the dashboard page is reloaded, by changing the prefix in the SitePasswordValidator setting; change "internal:" to "internal_nocookie:", or add the prefix "external_nocookie:" for external validation scripts.

NOTE: The PAM approach requires tractor-engine to deliver a clear-text password to the PAM interface locally on the engine host. Thus it must receive a recoverable password encoding from the Dashboard and other client scripts. The built-in transfer encoding provides modest protection from snooping, but your principal password transport protection is expected to come from the security of your site's private LAN and VPN connections into it.

Tractor also supports a more involved custom alternative solution in which you provide your own client-side one way password hashing plug-in (javascript or python, as appropriate), and a matching engine-side hash validator that you name here. See the comments in the template examples, and the Tractor administrative docs for more details.

```
    "SitePasswordValidator": ""
    "SitePasswordValidator": "internal:PAM:tractor"
    "SitePasswordValidator": "internal:PAM:su"
    "SitePasswordValidator": "internal_nocookie:PAM:su"
    "SitePasswordValidator": "python ${TractorConfigDirectory}/trSiteLoginValidator.py",
    "SitePasswordValidator": "python ${TractorConfigDirectory}/trSiteLdapLoginValidator.py"
```