Control Mode

Jump to bottom

Nicholas Marriott edited this page on Sep 1, 2020 · 6 revisions

Control mode

Control mode is a special mode that allows a tmux client to be used to talk to tmux using a simple text-only protocol. It was designed and written by George Nachman and allows his iTerm2 terminal to interface with tmux and show tmux panes using the iTerm2 UI.

A control mode client is just like a normal tmux client except that instead of drawing the terminal, tmux communicates using text. Because control mode is text only, it can easily be parsed and used over *ssh(1)*.

Control mode clients accept standard tmux commands and return their output, and additionally sends control mode only information (mostly asynchronous notifications) prefixed by `%`. The idea is that users of control mode use tmux commands (`new-window`, `list-sessions`, `show-options`, and so on) to control tmux rather than duplicating a separate command set just for control mode.

Entering control mode

The `-C` flag to tmux starts a client in control mode. This is only really useful with the `attach-session` or `new-session` commands that attach the client.

`-C` has two forms. A single `-C` doesn't change any terminal attributes - leaving the terminal in canonical mode, so for example echo is still enabled. This is intended for testing: typing will appear, control characters like delete and kill still work.

Two `-C` (so `-CC`) disables canonical mode and most other terminal features and is intended for applications (that, for example, don't need echo). In addition, the `-CC` form sends a `\033P1000p` DSC sequence (similar to ReGIS) that a listening terminal can use to detect control mode has been entered and sends a `%exit` line and a corresponding `ST` (`\033\`) sequence when the client exits.

With either form, entering an empty line (just pressing `Enter`) will detach the client.

For example, this shows output from starting a new tmux server on a socket called `test` with a new session (it runs `new-session`) and attaching a client in control mode, then killing the server:

```
$ tmux -Ltest -C new
%begin 1578920019 258 0
%end 1578920019 258 0
%window-add @1
%sessions-changed
%session-changed $1 1
%window-renamed @1 tmux
%output %1 nicholas@yelena:~$
%window-renamed @1 ksh
kill-server
%begin 1578920028 265 1
%end 1578920028 265 1
```

In this example, `kill-server` is a command entered by the user and the remaining lines starting with `%` are sent by the tmux server.

Commands

tmux commands or command sequences may be sent to the control mode client, for example creating a new window:

```
new -n mywindow
%begin 1578920529 257 1
%end 1578920529 257 1
```

Every command produces one block of output. This is wrapped in two guard lines: either `%begin` and `%end` if the command succeeded, or `%begin` and `%error` if it failed.

Every `%begin`, `%end` or `%error` has three arguments:

1. the time as seconds from epoch;
2. a unique command number;
3. flags, at the moment this is always one.

The time and command number for `%begin` will always match the corresponding `%end` or `%error`, although tmux will never mix output for different commands so there is no requirement to use these.

Output from commands is sent as it would be if the command was used from inside tmux or from a shell prompt, for example:

```
lsp -a
%begin 1578922740 269 1
0:0.0: [80x24] [history 0/2000, 0 bytes] %0 (active)
1:0.0: [80x24] [history 0/2000, 0 bytes] %1 (active)
1:1.0: [80x24] [history 0/2000, 0 bytes] %2 (active)
%end 1578922740 269 1
```

Or:

```
abcdef
%begin 1578923149 270 1
parse error: unknown command: abcdef
%error 1578923149 270 1
```

Getting information

Most control mode users will want to get information from the tmux server. The most useful commands to do this are `list-sessions`, `list-windows`, `list-panes` and `show-options`.

The `-F` flag should be used where possible for output in a known format rather than relying on the default. The `q` format modifier is useful for escaping.

For example listing all sessions with their ID and name:

```
ls -F '#{session_id} "#{q:session_name}"'
%begin 1578925957 337 1
$4 "\"quoted\""
$2 "abc\ def"
$0 "bar"
$1 "foo"
$3 "😀"
%end 1578925957 337 1
```

Pane output

Like a normal tmux client, a control mode client may be attached to a single session (which can be changed using commands like `switch-client`, `attach-session` or `kill-session`). Any output in any pane in any window in the attached session is sent to the control client. This takes the form of a `%output` notification with two arguments:

1. The pane ID (*not* the pane number).
2. The output.

The output has any characters less than ASCII 32 and the `\` character replaced with their octal equivalent, so `\` becomes `\134`. Otherwise, it is exactly what the application running in the pane sent to tmux. It may not be valid UTF-8 and may contain escape sequences which will be as expected by tmux (so for `TERM=screen` or `TERM=tmux`).

For example, creating a new window and sending the *ls(1)* command:

```
neww
%begin 1578923903 256 1
%end 1578923903 256 1
%output %1 nicholas@yelena:~$
send 'ls /' Enter
%begin 1578923910 261 1
%end 1578923910 261 1
%output %1 ls /\015\015\012
%output %1 altroot/     bsd.booted   dev/         obsd*        sys@\015\012bin/
bsd.rd       etc/         reboot*      tmp/\015\012
%output %1 boot         bsd.sp       home/        root/        usr/\015\012bsd
cvs@         mnt/         sbin/        var/\015\012
%output %1 nicholas@yelena:~$
```

Note that output generated by tmux itself (for example in copy or choose mode) is not sent to control mode clients.

Notifications

Notifications are sent to control mode clients when a change is made, either by another client or by the tmux server itself.

The following notifications are supported:

Notification	Description
`%pane-mode-changed %pane`	A pane's mode was changed.

Notification	Description
`%window-pane-changed @window %pane`	A window's active pane changed.
`%window-close @window`	A window was closed in the attached session.
`%unlinked-window-close @window`	A window was closed in another session.
`%window-add @window`	A window was added to the attached session.
`%unlinked-window-add @window`	A window was added to another session.
`%window-renamed @window new-name`	A window was renamed in the attached session.
`%unlinked-window-renamed @window new-name`	A window was renamed in another session.
`%session-changed $session session-name`	The attached session was changed.
`%client-session-changed client $session session-name`	Another client's attached session was changed.
`%session-renamed $session new-name`	A session was renamed.
`%sessions-changed`	A session was created or destroyed.
`%session-window-changed $session @window`	A session's current window was changed.

`$session`, `@window` and `%pane` are session, window and pane IDs.

Special commands

tmux provides two special arguments to the `refresh-client` command for control mode clients to perform actions not needed by normal clients. These are:

- `refresh-client -C` sets the size of a control mode client. If this is not used, control mode clients do not affect the size of other clients no matter the value of the `window-size` option. If this is used, then they are treated as any other client with the given size and may set the window size.

- `refresh-client -f` (`-F` is also accepted for backwards compatibility) sets flags for the control mode client. Some of the flags are general but several are only for control mode clients:
  - `no-output` does not send any `%output` notifications;
  - `wait-exit` waits for an empty line after `%exit` before actually exiting;
  - `pause-after` is used for flow control.
- `refresh-client -A` is used for flow control and `-B` for format subscriptions.

In addition, `send-keys` has a `-H` flag allowing Unicode keys to be entered in a hexadecimal form.

A few commands like `suspend-client` have no effect when used with a control mode client.

Flow control

tmux provides a mechanism for flow control of control mode clients. Flow control works by allowing tmux to pause output from a pane to the client once it becomes too far behind. Once a pane is paused, the client can ask tmux to continue sending output once it is ready. It is up to the client to update the content of the pane if necessary, for example using `capture-pane`.

Flow control is enabled by setting the `pause-after` flag using `refresh-client -f`. This takes a single argument which is the length of time in seconds before a pane should be paused:

```
refresh-client -f pause-after=30
```

When a pane is paused, the `%pause` notification will be sent with the pane ID. The pane can be continued with `refresh-client -A`:

```
refresh-client -A '%0:continue'
```

Once continued, the `%continue` notification is sent.

When flow control is enabled, the `%output` notification is not sent; instead the `%extended-output` notification is used. This has additional arguments terminated by a single `:` argument. Currently there are two arguments: the pane ID and the number of milliseconds by which the pane is behind. For example:

```
%extended-output %0 1234 : abcdef
```

`refresh-client -A` can also be used to manually pause a pane (`-A '%0:pause'`) or to turn it on or off. Turning a pane off tells tmux that the client does not require output from the pane to be sent and allows tmux to choose to stop reading from the pane if possible.

Format subscriptions

Control mode clients may subscribe to a format and be informed every time its expanded value changes. A subscription is added or removed with the `refresh-client -B` command. This takes a single argument which has three pieces separated by colons:

1. A subscription name.
2. The type of item to subscribe to, one of:

Type	Description
Empty	The attached session.
`%n`	A single pane ID.
`%*`	All panes in the attached session.
`@n`	A single window ID.
`@*`	All windows in the attached session.

3. The format.

If the type and format are omitted and only the subscription name is given, the subscription is removed.

tmux expands the format once for each matching item for the given type and if the resulting value has changed sends a `%subscription-changed` notification. This happens at most once a second.

General notes

A few other notes:

- Using session, window and pane IDs rather than names or indexes is strongly recommended because they are unambiguous.
- User options can be used to store and retrieve custom options in the tmux server (they can be set to server, session, window or pane). `show-options -v @foo` shows only the option value for user option `@foo` .

- Formats are the primary method of inspecting properties of a session, window or pane or the tmux server itself. The `display-message -p` command is useful for this as well as the `-F` flag to the list commands.

Clone this wiki locally

`https://github.com/tmux/tmux.wiki.git`