
(index.html)

iTerm2 should require little explanation for users accustomed to terminal emulators. Even if you are an experienced user, take the time to read through the highlights section of this document. It will familiarize you with some features of iTerm2 that you may not have seen in other terminal emulators that can make a real difference in the way you work.

### Table of Contents


- Highlights for New Users
- General Usage
- Menu Items
- Preferences
- Scripting
- Shell Integration
- Smart Selection
- Triggers
- Captured Output
- Fonts
- Inline Images
- Badges
- Dynamic Profiles
- Profile Search Syntax
- Automatic Profile Switching
- Coprocesses

- Session Restoration
- Utilities
- Proprietary Escape Codes


---

# Highlights for New Users

This chapter describes features of iTerm2 that go beyond basic usage and are not generally found in other terminal emulators.

### Text Selection

There are two ways to select text to copy to the clipboard: you can use the mouse, or you can use the find feature's "mouseless copy" feature. Text selection by mouse is described later in General section.



To select text without using the mouse, press cmd-f to open the find field. Enter the beginning of the text you wish to copy and the find feature will select it in your window. Then press tab and the end of the selection will advance by a word. To move the beginning of the selection to the left, press shift-tab. At most one line of text can be selected this way.

In Prefs > Profiles > Keys you can assign keys to move the beginning or end of the selection by a single character, word, or line. No such kes are bound by default.

### Split Panes

iTerm2 allows you to divide a tab into many rectangular "panes", each of which is a different terminal session. The shortcuts cmd-d and cmd-shift-d divide an existing session vertically or horizontally, respectively. You can navigate among split panes with cmd-opt-arrow or cmd-[ and cmd-]. You can "maximize" the current pane--hiding all others in that tab--with cmd-shift-enter. Pressing the shortcut again restores the hidden panes.

### Hotkey Window

iTerm2 offers a special terminal window that is always available with a single keystroke. This window is called the "hotkey window" and is most commonly used for occasional administrative tasks. To enable this feature, go to Preferences > Keys. Enable "Show/Hide iTerm2 with a system-wide hotkey". Click in the field and enter the key combination you'd like to use. Then check "hotkey toggles a dedicated window with profile:". A new profile will be created that is optimized for the feature. Pressing the hotkey will drop a terminal window down from the top of the screen, and pressing it again (or clicking in any other window) causes it to disappear.

### Swap Cmd and Option

iTerm2 allows you to remap modifiers. You have separate control over left and right command and option keys. One common need is to exchange cmd and option. To do this, go to Preferences > Keys. Set Left option key to Left command key and Left command key to Left option key (and do the same for Right command and Right option if you please). You can add exceptions if you don't want

certain combinations to be remapped (for example, cmd-tab) by adding a new global shortcut with the action "Do Not Remap" and the keystroke of the (unremapped) key you wish to keep unaffected by modifier remapping.

### Save Mark/Jump to Mark

You can mark a location in a session with cmd-shift-M and then jump back to it with cmd-shift-J. This is useful, for instance, if you suspend your editor to compile a program and it emits errors. You can save a mark at that point and then return to your editor to fix the errors. As you work, you can jump back to the compilation errors with cmd-shift-J.

### Regular Expression Search

When you open the find field (cmd-f) there is a down-arrow on the left of the field by the magnifying glass. Clicking it opens a menu of options in which you can enable regular expression search. The ICU syntax (http://userguide.icu-project.org/strings/regexp#TOC-Regular-Expression-Metacharacters) is used.

### Autocomplete

Any text that exists in a tab or its scrollback buffer can be autocompleted in that tab. To use autocomplete, type the beginning of a word and then press cmd-;. An autocomplete window opens showing the top 20 choices for words beginning what what you have entered. The list can be filtered by typing a subsequence. The filter can be reset by pressing backspace. If you make a selection and press return, it will be entered for you. If you make a selection and press tab, your autocomplete will be extended with the selection.

### Paste History

Whenever text is copied or pasted in iTerm2 it is added to the paste history. You can access paste history with cmd-shift-H. It can be filtered by typing a subsequence, and the filter can be cleared by pressing backspace. You can choose to have your paste history saved to disk by turning that option on under Preferences > General > Save copy/paste history to disk.

### Instant Replay

Sometimes interactive programs will overwrite something of interest on the screen (for example, top(1) does this all the time). Normally, this would be lost forever. With Instant Replay, you can step back in time to see exactly what was on your screen at some point in the recent past. To enable, press cmd-opt-B. Once you are in instant replay mode, you can use the left and right arrow keys to navigate back and forward through time. Esc exits instant replay mode. By default, each session uses up to 4MB to save its instant replay history, and this can be adjusted under Preferences > General > Instant Replay uses __ MB per session.

Another benefit of Instant Replay is that it shows you the exact time that something appeared on your screen down to the second. This is useful when trying to figure out when an error occurred, for example.

### Full Screen

You can press cmd-enter and iTerm2 will take up the entire screen. If you had a transparent background configured, it will be turned off upon entering full screen mode to reduce distractions. You can re-enable it with cmd-U. Unlike most OS X apps, iTerm2 can open a fullscreen window in the same desktop with no annoying animation if you disable Preferences > General > Native full screen windows.

### High-Color Modes

iTerm2 supports 256 color mode. To enable this for csh shells, set your terminal to xterm-256color (under Preferences > Profiles > Terminal > Report Terminal Type). Some applications may need to be configured to support this mode. In vim, add this to your .vimrc:

```
set t_Co=256
```

iTerm2 also supports 24-bit color.

### Focus Follows Mouse

This option is off by default, but can be enabled under Preferences > Pointer > Focus follows mouse. It only affects iTerm2 windows.

### Middle Button Paste

If you have a three-button mouse, by default the middle button performs "paste".

### Cursor Finery

When using a block cursor, it's hard to pick a cursor color that's visible against every background color. If you enable Smart cursor color (under Preferences > Profiles > Colors) then the cursor color will be dynamically chosen to be visible against the text it is over and the adjacent cells.

If you prefer a white or black cursor, you can use the "cursor boost" feature (under Preferences > Profiles > Colors) to make all colors other than the cursor dimmer.

Do you have trouble finding your cursor? You can turn on the cursor guide by toggling the View > Show Cursor Guide menu item or turning on Preferences > Profiles > Colors > Cursor Guide. This can also be toggled by an escape sequence. For example, add this to your .vimrc:

```
let &t_ti.="\<Esc>]1337;HighlightCursorLine=true\x7"
let &t_te.="\<Esc>]1337;HighlightCursorLine=false\x7"
```

If you've lost your cursor, press Cmd-/ or select View > Find Cursor and the cursor's position on the screen will be indicated very clearly

### Minimum Contrast

Sometimes an application will display text with a color combination that is hard to read. Colorblind users in particular may find certain combinations hard to see if the colors differ only in hue and not brightness. If you enable minimum contrast (under Preferences > Profiles > Colors > Minimum contrast, then iTerm2 will guarantee a minimum level of brightness difference between the foreground and background color of every character. If you set this to its maximum value, then all text will be black or white.

### Growl and Notification Center Support

If you enable Growl (Preferences > Profiles > Terminal > Send Growl/Notification Center alerts) then you'll receive messages when a terminal beeps, has output after a period of silence, or terminates. There's also a proprietary escape sequence (documentation-escape-codes.html) to send a notification. You can adjust the kinds of notifications that get posted in Preferences > Profiles > Terminal > Filter Alerts.

### Exposé Tabs

If you have too many tabs and are unable to find the one you're looking for, you can use the Exposé Tabs feature to find it. Press cmd-opt-E and all your tabs will be shown at once. You can then perform a search over all tabs simultaneously to find what you're looking for.

### Window Arrangements

You can take a snapshot of your open windows, tabs, and panes with the menu option Window > Save Window Arrangement. You can restore this configuration with Window > Restore Window Arrangement, or you can choose to have it automatically restored when you start iTerm2 with Preferences > General > Open saved window arrangement.

### Smart Selection

Performing a quad-click does a "smart selection," which selects text under the pointer in a way appropriate to its content. For example, URLs, quoted strings, and email addresses (among many other objects) are recognized and selected in their entirety. You can also bind actions to a smart selection rule. The first action takes effect when you cmd-click on text matching the rule. All actions are added to the context menu when you right click on text matching the rule.

### Triggers

Triggers are user-configurable regular expressions with associated actions that run when text is received that matches the regex. Actions include highlighting the matching text, showing an alert, sending text back, and more.

One advanced use of a trigger is to capture output matching a regex and display just those matching lines in the toolbelt. For example, you could create a trigger that matches compiler errors. When you run Make the errors will appear on the side of your window and you can click each to jump right to it. More information is available at the Captured Output (captured_output.html) manual.

### Tmux Integration

iTerm2 is tightly integrated with tmux. The integration allows you to see tmux windows as native iTerm2 windows or tabs. The tmux prefix key is not needed, as native menu commands operate on tmux windows. For more information, please see the iTerm2-tmux Integration (https://gitlab.com/gnachman/iterm2/wikis/TmuxIntegration) document.

### Coprocesses

Coprocesses are programs that run alongside iTerm2 and are bound to a single session. All output bound for the session is also routed as input to the coprocess. The coprocess's output acts like the user typing at the keyboard. Coprocesses can be used to automate tasks. For more information, see the Coprocess (documentation-coprocesses.html) document.

### Dynamic Profiles

If you have hundreds or thousands of profiles, look in to Dynamic Profiles (dynamic-profiles.html). This feature allows you to define profiles in JSON.

### Automatic Profile Switching

You can automatically change the current session's profile using Automatic Profile Switching (automatic-profile-switching.html). For example, this would allow you to change the background color when you are on a production system.

### Inline Images

iTerm2 can display images inline, including animated GIFs. More information is available at the Inline Images (images.html) document. An extension of this feature allows you to download files from a remote host over ssh simply by running a shell script. This script can be installed for you automatically when you install Shell Integration, described below.

### Undo Close

If you accidentally close a session, you get five seconds (by default; configurable in Preferences > Profiles > Session) to undo it by pressing Cmd-Z.

### Shell Integration

Shell Integration is a feature exclusive to iTerm2 that uses knowledge about your shell prompt to help you navigate from one shell prompt to another, record your command history, suggest most used directories, helps you re-run commands, download files from remote hosts with a click, upload files to remote hosts with drag and drop, and more. See the Shell Integration (shell_integration.html) documentation for all the details.

### Password Manager

iTerm2 can save your passwords in the Keychain. Use the Window > Password Manager menu item to open the password manager and enter your passwords.

### Timestamps

Toggle View > Show Timestamps to indicate the time each line was last modified. This is useful for telling how long operations took or when a message was printed.

### Tab Bar on Left

You can position the tab bar on the left side of the window. This is useful if you have a really large number of tabs.

### Open Quickly

If you have lots of sessions you can quickly find the one you're looking for with Open Quickly. Select the View > Open Quickly menu item (cmd-shift-O) and then enter a search query. You can search by tab title, command name, host name, user name, profile name, directory name, badge label, and more. Open Quickly also lets you create new tabs, change the current session's profile, and open arrangements. If you start your query with a / then that gives you a shortcut to various commands. Enter a query of / to see them.

---

## General Usage

### Tabs

When you first start iTerm2, a window opens showing a terminal session. If you want to open more that one session at a time, you have a few options: You can create a new window (Shell > New Window), you can create a new tab (Shell > New Tab), or you can split the current session into two panes (Shell > Split Horizontally, Shell > Split Vertically), each of which is a separate session.

Tabs in iTerm2 behave like tabs in other programs, most notably web browsers like Safari, Firefox, and Google Chrome. Note that you can drag and drop tabs to reorder them within a window. You can drag tabs from one window to another, and you can drag a tab from a window into a new window by dropping it outside any iTerm2 window's tab bar.

By default, the label of each tab is the name of the job that's running in that session. Some systems are configured to augment this with additional information such as the hostname you're logged in to or your current directory (this is done by sending a special code of ESC]0;string ^G).

Tab labels have indicators that tell you their status. A blue dot means new input was received. An activity indicator means new out is being received. When the session ends, a ⃠ icon appears in the tab. You can customize these indicatorsc in Preferences > Appearance.

### Edit Current Session

The *Edit Current Session* panel lets you modify the appearance of a single session. If you customize some attribute of the session (for example, by changing the default text color) then subsequent changes to that same attribute in the profile will not affect the customized session. However, changes to other attributes of the profile will affect the customized session.

### Pointer

The primary use of the mouse in iTerm2 is to select text, and (by default) text is copied to the clipboard immediately upon being selected. You can click and drag to perform a normal selection. Double-clicking selects a whole word. Triple-clicking selects an entire line. Quadruple-clicking performs a "smart select", matching recognized strings such as URLs and email addresses. You can add custom pointer actions in Preferences > Pointer. I recommend using three-finger tap for smart selection, but you must ensure that *System Preferences* > *Trackpad* does not have any other action already assigned to three-finger tap.

If you hold shift while clicking the existing selection is extended. In fact, you can single click in one location and shift click in an other location to make a selection: no dragging needed.

If you hold cmd while dragging it will create a noncontinguous selection.

If you hold cmd and click on a URL it will be opened. If you hold cmd and click on a filename, it will be opened. There is special support for MacVim, TextMate, and BBEdit when you cmd-click on a text file's name: if it is followed by a colon and line number, the file will be opened at that line number. The current directory is tracked if you have your shell prompt set the window title, as described here (http://www.faqs.org/docs/Linux-mini/Xterm-Title.html#toc4), or if you have Shell Integration (shell_integration.html) installed.

If you hold cmd you can drag and drop selected text.

If you hold cmd and option while selecting, a rectangular selection will be made.

If mouse reporting is enabled (in Preferences > Profile > Terminal) and the currently running terminal application is using it, pressing option will temporarily disable it so you can make a selection.

If you hold the control key and click, that simulates a right click. If you'd prefer to send this combination to applications using mouse reporting, check "-Click reported to apps, does not appear in menu" in Preferences > Pointer.

Right clicking on a number shows its conversion to or from hex, or if it looks like a unix timestamp its representation in local time will be shown.

Moving the mouse's scroll wheel scrolls up and down. You can configure it to send arrow keys in interactive programs by turning on Preferences > Advanced > Scroll wheel sends arrow keys when in alternate screen mode, but it will only work if **Preferences > Profiles > Terminal > Disable save/restore alternate screen** is turned off.

A three-finger swipe left or right on a trackpad (if configured to "navigate") will select an adjacent tab.

Middle clicking on a tab (if your pointing device has a middle button) closes it.

Hold Control while resizing a window to temporarily disable snap-to-grid. You can permanently disable snap-to-grid in Prefs > Advanced > Terminal windows resize smoothly.

### Keyboard

Every aspect of the keyboard can be configured in iTerm2. These keystrokes may be useful to remember:

- Cmd+left arrow, Cmd+right arrow navigates among tabs. So does Cmd-{ and Cmd-}.
- Cmd+number navigates directly to a tab.

- Cmd+Option+Number navigates directly to a window.
- Cmd+Option+Arrow keys navigate among split panes.
- Cmd+] and Cmd+[ navigates among split panes in order of use.

You can configure any key combination to perform any action in two places: in Preferences > Keys, you can define global key shortcuts that affect all profiles. In Preferences > Profiles > Keys, you can define key shortcuts that affect only a single profile.

You can remap modifiers like Option and Cmd within iTerm2. Some users find that pressing Option frequently is uncomfortable, and configure iTerm2 to swap the function of the Option and Cmd keys. This is done in Preferences > Keys under Remap Modifier Keys. If there is some key combination that you don't want to be affected by this change (such as Cmd-tab) add a new global shortcut key with the action Do Not Remap.

iTerm2 allows you to define a global hotkey. This is a single keystroke that iTerm2 listens for even when another application has keyboard focus. When it is pressed, iTerm2 comes to the front. Press it again, and iTerm2 goes away. You can choose to bind the hotkey to a single dedicated window. This feature is similar to other programs like Visor, Guake, and Yakuake.

#### Context menus

By right-clicking in a session a context menu opens. You can use it to open a new session, perform various actions on selected text, or access frequently used features to affect the clicked-on session.

#### Profiles

Many settings are stored in profiles. A profile is a named collection of settings, and you can have as many of them as you like. Most users only have one profile, but if you find that you often connect to different servers, they may be useful for you. A key feature of a profile is that you can associate a command with it that is run when it begins. For instance, if you often ssh to a particular host, you could create a profile with the command "ssh example.com" to automate that process.

---

## Menu Items

### iTerm2 Menu

#### iTerm2 > Secure Keyboard Entry

When this is enabled, the operating system will prevent other programs running on your computer from being able to see what you are typing. If you're concerned that untrusted programs might try to steal your passwords, you can turn this on, but it may disable global hotkeys in other programs.

#### iTerm2 > Show Tip of the Day

When you start using iTerm2 it will offer to show you a daily tip describing a feature. You can show a tip immediately by selecting this item.

#### iTerm2 > Check for Updates

Checks to see if a new version of iTerm2 is available. If Preferences > General > Prompt for test-release updates is turned on then this includes beta versions; otherwise only stable versions are downloaded.

#### iTerm2 > Toggle Debug Logging

This saves helpful debugging information in memory. When it is toggled off it is saved to /tmp/debuglog.txt.

#### iTerm2 > Install Shell Integration

Runs a shell script that modifies your .bash_profile (or other startup script) and enables the Shell Integration (/shell_integration.html) features.

---

### Shell Menu

#### Shell > New Tab with Current Profile

This creates a new tab using the same profile as the current session rather than the default profile.

#### Shell > New Window/Tab

This creates a new window or tab with the default profile. If the current session is a tmux integration session, then you will be prompted for whether to create a local or tmux session.

#### Shell > Duplicate Tab

Creates another tab with the same arrangement of split panes, profile, etc.

#### Shell > Split Vertically/Horizontally

These menu items allow you to divide a tab into two or more split panes. The panes can be adjusted by dragging the line that divides them. They will use the default profile.

#### Shell > Split Vertically/Horizontally with Current Profile

These menu items allow you to divide a tab into two or more split panes. The panes can be adjusted by dragging the line that divides them. They will use the profile of the current session.

#### Shell > Restart Session

After a session ends (e.g., because the shell exits) this menu item becomes enabled. It will re-run your profile's command in the same viewport as the terminated session.

**Shell > Broadcast Input > ...**

These options allow you to send keyboard input to more than one session. Be careful.

**Shell > Run/Stop Coprocess**

Allows you to start and stop a coprocess linked to the current session. Learn more about coprocesses.

**Shell > tmux > ...**

These commands let you interact with the tmux integration. The tmux dashboard is a view that lets you see all your tmux sessions and windows at a glance, adjust which are visible, rename them, and change the current tmux session.

**Shell > Log > Start/Stop**

Logging saves all input received in a session to a file on disk.

---

## Edit Menu

**Edit > Copy (With Styles)**

Hold down Option to turn Copy into Copy With Styles, which includes fonts and color information in the copied text.

**Edit > Undo Close Session/Tab/Window**

After you close a session, tab, or window then you have five seconds to undo it. The amount of time is configurable in PReferences > Profiles > Session.

**Edit > Paste Special > Advanced Paste**

This opens the Advanced Paste window which lets you select a string from the pasteboard in recent history, select different representations of the current pasteboard, and modify the string before pasting it. You can modify it by appling a regex substitution, using various built-in modifiers (such as base-64 encoding), or edit it by hand.

**Edit > Paste Special > Paste Selection**

Pastes the currently selected text (which may differ from the text in the pasteboard).

**Edit > Paste Special > Paste File Base64-Encoded**

If there is a file on the pasteboard then this this is enabled. When invoked, it base64-encodes the file and pastes the encoded value.

**Edit > Paste Special > Paste Escaping Special Characters**

"Paste Escaping Special Characters" pastes the current string in the clipboard, but places a backslash before spaces and backslashes.

**Edit > Paste Special > Paste Slowly**

"Paste Slowly" pastes the current string in the clipboard, but it doesn't send the whole string at once. It is sent in batches of 16 bytes with a 125ms delay between batches.

**Edit > Paste Special > Paste Faster/Slower**

Adjusts the speed of pasting to be faster or slower.

**Edit > Paste Special > Paste Slowly Faster/Slower**

Adjusts the speed of slow pasting to be faster or slower. You must hold down option for this menu item to be visible.

**Edit > Paste Special > Warn Before Multi-Line Paste**

When enabled, you'll be warned before pasting more than one line.

**Edit > Selection Respects Soft Boundaries**

When enabled, vertical lines of pipe characters | will be interpreted as pane dividers (as in vim or emacs) and selection will wrap at them.

**Edit > Open Autocomplete...**

Shows the autocomplete window, which offers to finish typing a word that you've begun. Learn more about autocomplete on highlights page (documentation-highlights.html).

**Edit > Edit Current Session**

This opens a window that lets you change the settings of the current session without affecting any other sessions. Changes made in this panel will not be overridden by subsequent changes to the profile. Settings *not* cahnged in this panel will be affected by changes to the profile.

**Edit > Open Command History...**

If you use Shell Integration (shell_integration.html) then Open Command History presents a list of recently used commands to select from.

**Edit > Open Recent Directories...**

If you use Shell Integration (shell_integration.html) then Open Recent Directories presents a list of recently used directories to select from.

**Edit > Open Paste History...**

"Open Paste History" opens a window showing up to the last 20 values that were copied or pasted in iTerm2. You can search its contents by typing a (non-necessarily-consecutive) subsequence of characters that appear in the value. You can use arrow keys and enter to make a selection, or you can click on an item to choose it, and it will be pasted. If you enable the Save copy/pate history to disk preference then these values will persist across sessions of iTerm2.

**Edit > Find > Fine URLs**

Searches the current session for URLish looking strings.

**Edit > Marks and Annotations > Set Mark**

Records the current scroll position. Use Edit > Jump to Mark to restore the scroll position.

**Edit > Marks and Annotations > Add Annotation at Cursor**

Adds an annotation to the word beginning at the cursor. An annotation is a scratchpad for you to write notes about a chunk of text in your history.

**Edit > Marks and Annotations > Alerts > Alert on Next Mark**

When a mark is set (typically by Shell Integration (shell_integration.html) when the currently running shell command terminates) then show an alert.

---

## View Menu

**View > Show Tabs in Fullscreen**

If enabled, tabs are shown in fullscreen windows.

**View > Use Transparency**

This toggles transparency. It only has an effect if you have configured your session to be transparent under Preferences > Profiles > Window > Transparency. When Full Screen mode is entered, transparency is turned off by default, but you can select this menu item to re-enable it.

**View > Find Cursor**

Reveals the current cursor position.

**View > Zoom In on Selection/Zoom Out**

When a selection is present this is enabled. Zooming on a selection removes all other text from the session and lets you focus on just the zoomed-in-on text. Pressing escape will invoke Zoom Out when you are in the Zoom In state.

**View > Show Cursor Guide**

Toggles the visiblity of the cursor guide which is a horizontal rule showing the location of the cursor.

**View > Show Timestamps**

Indicate the time of last modification of each line on the screen.

**View > Show Annotations**

Toggles the visibility of annotations.

**View > Auto Command Completion**

Automatically shows a window with command completion suggestions as you type. Only usable when you have command history built up with Shell Integration (shell_integration.html).

**View > Open Quickly**

If you have lots of sessions you can quickly find the one you're looking for with Open Quickly. Select the View > Open Quickly menu item (cmd-shift-O) and then enter a search query. You can search by tab title, command name, host name, user name, profile name, directory name, badge label, and more. Queries are scored according to relevance and sorted by score. Open Quickly also lets you create new tabs, change the current session's profile, and open arrangements. If you start your query with a / then that gives you a shortcut to various commands. /a followed by an arrangement name restores the arrangement. /f restricts the query to existing sessions, excluding options to open new tabs, etc. /p restrics the query to profile names to switch the current session to. /t restricts the results to "open new tab" for matching profile names.

**View > Maximize Active Pane**

When there are split panes present, this toggles whether a given pane expands to fill the tab. When a maximized pane is present, the tab will be inscribed with a dotted outline.

**View > Start Instant Replay**

Stepping through time allows you to see what was on the screen at a previous time. This is different than going back through the scrollback buffer, as interactive programs sometimes overwrite the screen contents without having them scroll back. Once in this mode, you can use the left and right arrow keys to step back and forward, respectively. The "esc" key exits this mode, as does clicking the close button in the bar that appears on the bottom. You can adjust the amount of memory dedicated to this feature in Preferences > Instant Replay uses xx MB per session. The more memory you assign, the further back in time you can step.

**View > Tab Color**

Allows you to select a tint color for the tab, to make it easier to distinguish. You can also change the tab color in Profiles > Preferences > Colors.

## Profiles Menu

### Profiles > Open Profiles...

This opens the "Profiles Window" which allows you to create new windows, tabs, or panes from one or more profiles. You can perform a search by entering text in the search field. Profile names and tags are searched, and the listed profiles are filtered as you type. You can use the up and down arrow keys to make a selection. Pressing enter will open a new tab, while shift-enter will open a new window. You can make multiple selections by holding down shift or cmd and clicking on profiles. The "New Tabs in New Window" button is enabled only when more than one profile is selected: it will open a new window and create a new tab for each profile selected.

---

## Scripts Menu

If you have scripts located in `$HOME/Library/Application Support/iTerm/Scripts` they'll be added to this menu. The menu will not exist if there are no scripts.

---

## Toolbelt Menu

### Tolbelt > Captured Output

This toggles the visibilty of the Captured Output tool. It shows captured output located with the Capture Output trigger. See Captured Output (captured_output.html) fo r more information.

### Toolbelt > Command History

This toggles the visibility of the Command History tool. It shows recently used commands. You must install Shell Integration (shell_integration.html) for this to know your command history.

### Toolbelt > Show Toolbelt

This toggles the visibility of the Toolbelt on the right side of all windows.

### Toolbelt > Jobs

This toggles the visibility of the Jobs tool, which shows the running jobs in the current session, and allows you to send them signals.

### Toolbelt > Notes

This toggles the visibility of the Notes tool, which provides a freeform scratchpad in the toolbelt.

### Toolbelt > Paste History

This toggles the visibility of the Paste History tool, which shows recently pasted strings in the toolbelt.

#### Toolbelt > Profiles

This toggles the visibility of the Profiles tool, which lets you select profiles to open new windows, tabs, and split panes.

#### Toolbelt > Recent Directories

This toggles the visibility of the Recent Directories tool. It shows recently used directories sorted by a combination of recency and frequency of use. You must install Shell Integration (shell_integration.html) for this to know your directory history. You can right click a directory to open a context menu that allows you to "start" a directory. This keeps it pinned at the bottom of the list so it's easy to find.

---

### Window Menu

#### Window > Exposé All Tabs

All iTerm2 tabs will be shown tiled on the main screen. You can mouse over a tab to see it larger, and clicking on it will restore the windows and select that tab and bring its window to the front. You can search the contents of all tabs by typing in the search field that appears on the left. Making a selection from the results below it will highlight the tab that contains that text. This is useful when you have many tabs open and can't find the one you're looking for.

#### Window > Save/Restore Window Arrangement

The current state and positions of windows, tabs, and spit panes is recorded and saved to disk with Save Window Arrangement. Restore Window Arrangement opens a new collection of windows having the saved state. You can automatically restore the arrangement in Preferences > General > Open saved window arrangement.

#### Window > Password Manager

Opens the password manager.

---

## Preferences

### General

---

#### Startup

The first dropdown box lets you select how windows will be opened when iTerm2 is launched. Most users will want *Use Systme Window Restoration Setting* as it works best with Session Restoration (restoration.html). Users who exclusively use the Hotkey Window may prefer *Don't Open Any Windows*. If you have a default window arrangement saved then *Open Default Window Arrangement* will be available.

**Open profiles window**

If selected, the Profiles Window will automatically open when iTerm2 is started.

**Quit when all windows are closed**

If selected, iTerm2 will automatically quit when its last terminal window is closed.

**Confirm closing multiple sessions**

If selected, commands that close one session will not be confirmed, but commands that close multiple sessions (such as clicking the red button on a window with two or more tabs) will be confirmed with an alert box.

**Confirm Quit iTerm2 Command**

If selected, the Quit iTerm2 (cmd-Q) command will be confirmed if any terminal windows are open.

**Instant Replay Uses X MB per Session**

This setting specifies the maximum amount of memory allocated to instant replay for each tab or split pane. More memory means instant replay is able to go farther back into the past. You can enter instant replay with View > Step Back in Time.

**Save copy/paste and command history to disk**

If selected, every time text is copied or pasted in iTerm2 it will be saved to disk. The last 20 values are recorded. They can be accessed with Edit > Open Paste History.... If you use Shell Integration (shell_integration.html) then when this is enabled your command history, directory history, and remote hostname and usernames will also be saved to disk. Unchecking this will erase all of the saved information.

**Add Bonjour hosts to profiles**

If selected, all Bonjour hosts on the local network have a profile created for them as long as they're around.

**Check for updates automatically**

If enabled, iTerm2 will periodically check if a new version of iTerm2 exists, and if so it will prompt you to download and upgrade.

**Prompt for test-release updates**

If enabled, iTerm2 will periodically check if a new unstable version of iTerm2 exists, and if so it will prompt you to download and upgrade.

**Load preferences from a custom folder or URL:**

If enabled, iTerm2 will load its preferences from the specified folder or URL. After setting this, you'll be prompted when you quit iTerm2 if you'd like to save changes to the folder.

**Save changes to folder when iTerm2 quits**

When you've turned on *Load preferences from a custom folder* and this is on then any changes you make to your settings will be written to the custom folder.

**Copy to clipboard on selection**

If enabled, text is copied to the clipboard immediately upon selection. If not selected, you must select Edit > Copy to copy it.

**Copied text includes trailing newline**

If enabled, a terminal newline will be copied to the pasteboard when the selection includes one; otherwise, no selection will ever include a terminal newline.

**Applications in terminal may access clipboard**

If enabled, clipboard access will be granted via escape code to programs running in iTerm2. They will be able to retrieve and set the contents of the system pasteboard.

**Characters considered part of a word for selection**

When you double-click in the terminal window, a "word" is selected. A word is defined as a string delimited by characters of a different class. The classes of characters are whitespace, word characters, and non-word characters. The characters in this field define the set of non-word characters.

**Smart window placement**

If enabled, new windows will be opened where they least overlap existing windows.

**Adjust window when changing font size**

If enabled, a change to a session's font will cause the window to grow or shrink.

**Zoom button maximizes vertically only**

If enabled, the green "Zoom" button expands a terminal window vertically but does not affect its width. This can be overridden by holding down shift while clicking the zoom button.

#### Native full screen windows

If enabled, fullscreen windows will animate into a special desktop, as is typical in OS X 10.7 and later. If disabled, fullscreen windows will instantly go fullscreen without changing desktops.

#### tmux Integration

The first dropdown box in the **tmux Integration** section allows you to define how tmux windows should be mapped to native constructs. When attaching to a new tmux session with the tmux integration, tmux windows not seen by iTerm2 before will open in either new windows or tabs, as specified by this preference.

#### Open dashboard if there are more than...tmux windows

When attaching to a tmux session with the tmux integration, windows are normally opened automatically. If there are too many of them, then the tmux dashboard is opened instead.

#### Automatically hide the tmux client session after connecting

When the tmux integration is entered by running tmux -CC, the window in which that command was run will miniaturize into the dock if this option is enabled.

### Appearance

---

#### Tab Bar Location

Defines whether tabs appear at the top, bottom, or left side of your windows. Select from a list of tab looks.

#### Theme

Allows you to select the theme. Light and Dark options are available, along with high-contrast versions that are easier to read.

#### Show tab bar even when there is only one tab

If selected, the tab bar will remain visible when a window contains exactly one tab.

#### Show tab numbers

If selected, tabs will indicate their keyboard shortcut.

#### Show tab close buttons

If selected, tabs show close buttons. If not selected, the close buttons only appear when the mouse hovers over the tab.

**Show tab activity indicators**

If selected, the activity indicator in each tab will be displayed when new output is recevied and the tab is not selected.

**Flash tab bar when switching tabs in fullscreen**

If selected, the tab bar will show briefly when switching tabs in a fullscreen window. It will also show briefly when the number of tabs changes.

**Show tab bar in fullscreen**

If selected the tab bar will be visible in fullscreen windows.

**Show per-pane title bar with split panes**

When a tab has split panes, this option controls whether each split pane will have its own title bar.

**Auto-hide menu bar in non-native fullscreen**

When native fullscreen mode is disabled (in Prefs > General), this option is available. If you'd like the menu bar to remain visible when a fullscreen window is present on a screen with a menu bar, turn this on.

**Show window number**

If selected, window titles include the window number. You can navigate to a window by pressing cmd-opt-N where N is the window number.

**Show current job name**

If selected, tab and window titles will show the name of the foreground job.

**Show profile name**

If selected, tab and window titles will show profile names.

**Dimming affects only text, not background**

When a window or pane is dimmed, this option controls whether the background color is dimmed or only the text colors.

**Dimming amount**

This slider controls how much to dim inactive windows or panes.

**Dim inactive split panes**

If selected, split panes that do not have keyboard focus will be slightly dimmed.

**Dim background windows**

If enabled, windows in the background (that is, those not receiving keyboard input) are dimmed according to the above settings.

**Show border around window**

If selected, a 1-pixel border will be shown around the edges of terminal windows.

**Hide scrollbars**

If selected, scrollbars will be hidden in terminal windows.

**Disable transparency for fullscreen windows by default**

If enabled, entering fullscreen mode will automatically turn off transparency for that window.

## Profiles > General

**Name**

Gives the name of the profile which is shown in menus, preferences, and the profiles window.

**Shortcut key**

This shortcut can be used to open a new window or tab. By default, it opens a new tab, but if you hold down the option key while pressing the shortcut, a new window will be opened instead.

**Tags**

Tags are a collection of words or phrases that annotate a profile. When you search your profiles (for instance, in the profiles window), the tag names are searched in addition to the profile name. If a tag name contains a slash that defines a hierarchy of menu items in the **Profiles** menu.

**Badge**

The badge is a large label visible in the top right of a terminal session behind its text. For more information see Badges (badges.html).

**Command**

This is the command that is executed when a new session with the profile is created. If *login shell* is chosen, then `login` is invoked. You can put special terms surrounded by $$ in this field (example: $$USERNAME$$). When a new session is created, you will be prompted to enter a value for each such term. See the description of URL Schemes below for details about the special "$$" value that can go in this field.

#### Send Text at Start

This text will be sent when a session begins. If it is not empty then a newline will be sent afterwards. It does not accept any special characters or require any escaping.

#### Working directory

Normally, new sessions begin in your home directory. You can choose to open new sessions in the same directory as the current session (but only when creating a new tab), or you can specify a starting directory.

#### URL Schemes

You can configure a profile to handle a URL scheme, such as ssh. When a hyperlink is clicked on with that scheme, a new tab is opened with the selected profile. It is recommended that you set the command to "$$", in which case an ssh command line will be auto-generated. For other schemes, you can uses these variables in the Command field and they will be replaced with the appropriate part of the URL:

- $$URL$$ The complete url
- $$HOST$$ The host portion of a url like scheme://host/
- $$USER$$ The username portion of a url like scheme://user@host/
- $$PASSWORD$$ The password portion of a url like scheme://user:password@host/
- $$PORT$$ The port number of a url like scheme://host:port/
- $$PATH$$ The path portion of a url like scheme://host/path
- $$RES$$ The portion of a url following the scheme.

### Profiles > Colors

Clicking on any of the color wells opens a color picker that lets you change the setting for the selected color. iTerm2 has a custom color picker. If you don't like it you can revert to the system color picker by clicking the rectangular icon to the right of the eyedropper.

#### Smart cursor color

When selected, a block cursor will be displayed in reverse video. If this would result in confusion, then a different color is chosen that will be most visible given the surrounding cells' background colors.

#### Minimum contrast

If text is displayed against a similar background color, the minimum contrast setting will move the text color towards black or towards white to ensure some minimum level of visibility. Setting this slider all the way to maximum will make all text black and white.

#### Cursor Boost

Cursor Boost dims all colors other than the cursor colors to make the cursor stand out more.

#### Tab Color

If enabled, this color will decorate the tab control. Tabs indicate the color of their current session if there is more than one split pane.

#### Cursor Guide

The cursor guide is a horizontal rule that indicates the vertical position of the cursor. You can adjust its color, including alpha value, to make it more visible against your background color.

#### Color Presets...

iTerm2 ships with some color presets, which you may load from this popup menu. You can import and export color presets to files with the extension "itermcolors". There is an online color gallery where users may share color presets, and a link to it is provided in this menu. When importing a color preset, the name it is assigned is based on the filename imported.

### Profiles > Text

---

#### Cursor

This lets you select a cursor shape.

#### Blinking cursor

If checked, the cursor will blink slowly to improve visibility.

#### Draw bold text in bold font

If selected, bold text will be drawn in a bold version of the selected font. If the font does not have a bold version, then a bold appearance is simulated by "double striking" the text: that is, drawing it twice, shifting it one pixel horizontally the second time.

#### Draw bold text in bright colors

If selected, bold text will be drawn in a bright version of its color.

**Blinking text allowed**

If selected, text with the blink attribute set will actually blink. Oh, the humanity.

**Italic text allowed**

If selected, text with the italic attribute set will be rendered in italics. The font you select must have an italic face.

**Use thin strokes for anti-aliased text**

Anti-aliased text will be drawn with thinner strokes by default on Retina displays. The effect may be more or less visible depending on your particular hardware and OS version. Some people prefer the older, heavier strokes, so you may customize the setting here.

**Regular font**

ASCII text (latin letters, numbers, and some symbols) will be drawn using this font. Select "Anti-aliased" to draw the text with smooth edges.

**Non-ASCII font**

All non-ASCII text (many accented Latin letters, non-Latin text, less-common symbols, and thousands of miscellaneous unicode characters) will be drawn with this font. It is recommended that you use the same point size for both regular and non-ASCII fonts. Select "Anti-aliased" to draw the text with smooth edges.

**Treat ambiguous-width characters as double width**

Some characters (e.g., Chinese ideograms) are double-width, and take two cells to display. Other characters (e.g., Latin letters) are single width and take only one cell to display. There is another category of characters known as "ambiguous width". One example of ambiguous-width characters are Greek letters. Depending on your application, you may prefer to display them as double-width or single-width. If most of the text you deal with is double-width, then you should enable this setting as it will help things to line up correctly in that context.

**Use HFS+ Unicode Normalization**

Use HFS+ normalization instead of NFC. This helps preserve the fullwidth attribute of composed characters.

## Profiles > Window

---

**Transparency**

This sets the transparency of the window background. It can be temporarily disabled with View > Use Transparency.

**Keep background colors opaque**

If selected, non-default background colors wil be opaque.

**Blur**

If selected, the window background is blurred provided the background has some transparency. Selecting a large radius will blur the background more, but (especially on Retina displays) comes with a performance penalty.

**Rows/Columns**

When creating a new window with this profile, it will be created with this many rows and columns.

**Hide after opening**

If enabled, a window created with this profile will immediately miniaturize after its creation.

**Open Toolbelt**

If enabled, a window created with this profile will feature an open toolbelt.

**Background Image**

This allows you to select an image to display behind the terminal's text. If Tile image is selected, then the image will be shown at its actual size and tessellated; otherwise, it will be stretched to fit the whole pane. The blending slider determines how strongly the image dominates over the text's background color.

**Style**

This defines the window style. Bottom- and Top-of-screen windows will disregard the columns setting. Left-of-screen windows will disregard the rows setting. Full-screen windows only respect the rows and columns setting when full-screen mode is exited. The full-width/full-height styles ignore the column or rows setting, respectively, to use all the available space except what is reserved by the system.

**Screen**

If you have more than one screen connected, this lets you select the screen on which a new window should open. It is particularly useful for fullscreen and top-of-screen window styles. The *Screen with Cursor* option affects the initial screen of the window, but it won't follow your cursor from screen to screen.

**Space**

If you have enabled Spaces (or your OS uses Desktops instead of spaces) and have set Spaces/Mission Control to use Control+Number to switch spaces/desktops, then you can use this setting to select the initial space/desktop to open a new window using this profile.

#### If showing profile name in tab title, keep it when the title is changed

You can specify that profile names are shown in window and tab titles under Preferences > Appearance > Show Profile Name. If that is not set, then this option is irrelevant. When in use, a host may send an escape code that changes the window title. This setting causes the profile name to be preserved in that session-set title.

#### Force this profile to open in a new window, never in a tab

If you ask for a new tab with this profile, it will just open in a window instead. This is for people who hate tabs.

### Profiles > Terminal

---

#### Scrollback lines

The number of lines of scrollback buffer to keep above the visible part of the screen. Unlimited scrollback will allow it to grow indefinitely, possibly using all available memory.

#### Save lines to scrollback when an app status bar is present

Some programs (such as vim or tmux) keep a status bar at the bottom of the screen. For some applications (like vim) it is undesirable to save lines to the scrollback buffer when the application scrolls. For others (like tmux) you may want to save scrolled-off lines into the scrollback buffer. When this setting is enabled, lines scrolled off the top of the screen in the presence of a status bar are added to the scrollback buffer. The screen is considered to have a status bar if it has a scroll region whose top is the first line of the screen and whose bottom is above the bottom of the screen.

#### Save lines to scrollback in alternate screen mode

When in alternate screen mode, lines that scroll off the top of the screen will be saved to the scrollback buffer only if this option is enabled.

#### Character encoding

The encoding to send and receive in. For most people, "Unicode (UTF-8)" is the right choice.

#### Report terminal type

The TERM variable will be set to this value by default. If xterm-256color is selected and your system is missing the terminfo file, you will be prompted to install it when you open a new session.

**ENQ answer back**

Text to send when the ENQ sequence is received. Not normally used.

**Enable mouse reporting**

If selected, applications may choose to receive information about the mouse. This can be temporarily disabled by holding down Option.

**Terminal may report window title**

Programs running in a terminal may send an escape code to request the current window title. You may disable this feature by enabling this option. It should be disabled if you're communicating with an untrusted party, as there are possible injection attacks.

**Terminal may set tab/window title**

If enabled the terminal may set the window or tab title with an escape sequence.

**Disable session-initiated printing**

If enabled, escape codes that initiate printing will be ignored.

**Disable save/restore alternate screen**

Some programs (such as vim, tmux, and less) switch into a so-called "alternate screen". A characteristic of this behavior is that when these programs terminate the screen's contents are restored to their state from before the program was run. If this option is selected, alternate screen mode is disabled and the screen cannot be restored by an application.

**Disable session-initiated window resizing**

If the host sends an escape code to resize the window, it will be ignored if this option is selected..

**Silence bell**

If selected, the bell (control-G) will not make an audible sound.

**Send Growl/Notification Center alerts**

If selectedinstalled, iTerm2 will post a notifications when sessions receive output, become idle, ring the bell, close, or get a proprietary escape sequence to post a notification. If Growl is installed it is preferred over Notification Center.

**Filter Alerts**

This button opens a panel that lets you customize which notifications will be posted.

**Flash visual bell**

If selected, a bell graphic will be flashed when the bell character is received.

**Show bell icon in tabs**

If selected, tabs will indicate that a bell has rung by displaying a bell graphic.

**Set locale variables automatically**

If enabled, LANG and LC_CTYPE environment variables will be set based on your machine's language settings.

**Insert newline before start of command prompt if needed**

If you have Shell Integration (shell_integration.html) installed and a command's output does not end in a newline, this setting will ensure your prompt does not begin in the middle of the line.

**Show mark indicators**

If you have Shell Integration (shell_integration.html) and this setting is selected then a blue or red arrow appears next to each shell prompt. Turn this off to hide the arrow.

## Profiles > Session

**Automatically close a session when it ends**

If selected, a session's pane, tab, or window will automatically close when the session ends.

**"Undo" can revive a session that has been closed for up to X seconds**

When you close a session, window, or tab the shell is not terminated until X seconds pass. While that time period has not elapsed, *Undo* will reopen the session, tab, or window.

**Prompt before closing**

When a session will close, you can choose when to be prompted with a modal alert.

**Automatically log session input to files in:**

If enabled, every session's output will be logged to a file in the specified directory. File names are formatted as Date_Time.ProfileName.TerminalID.ProcessId.RandomNumber.log.

**When idle, send ASCII code X every Y seconds.**

If selected, the specified ASCII code "X" (a number from 0 to 255) will be transmitted every Y seconds while nothing is happening. Don't use this unless you know what you're doing as it can have unexpected consequences. Seriously, it's probably not what you want.

#### Avoid repainting while cursor is hidden to reduce flicker while scrolling

When selected, the screen will slightly delay redraws while the cursor is hidden. This improves the visual appearance of scrolling in many programs but might introduce noticeable delays for some users.

## Profiles > Keys

This panel shows key mappings. You can double-click on a mapping to edit it. When the "Keyboard Shortcut" field has focus, you should press the keystroke that you want to modify (even if it involves modifiers like Cmd). The following actions are available:

#### Ignore

The keypress will do nothing.

#### Do not remap modifiers

If modifier remapping is in effect (set under Preferences > Keys), it can be disabled for certain key combinations. When you choose this action, modifier remapping is temporarily disabled so you can press the key combination unremapped in the key field.

#### Remap modifiers in iTerm2 only

If modifier remapping is in effect (set under Preferences > Keys), it can be set to not affect other applications that may listen for global hotkeys. When you choose this action, modifier remapping is temporarily disabled so you can press the key combination unremapped in the key field.

#### Split/New Window/Tab with Profile

These actions allow you to create a new session with a specified profile when a key is pressed.

#### Start Instant Replay

This is equivalent to the menu item View > Start Instant Replay.

#### Cycle Tabs Forward/Backward

This implements tab switching the same way Cmd-Tab (or Cmd-Shift-Tab) switches windows, with the most-recently-used stack.

#### Next/Previous Tab/Window/Pane

These actions navigate among tabs, windows, and split panes.

**Move tab left/right**

Changes the tab's position in the order.

**Increase/Decrease Width/Height**

Changes the size of the current session.

**Scroll to End/Top/Up/Down**

These actions move through the scrollback buffer.

**Select Split Pane Above/Below/Left/Right**

These actions navigate split panes.

**Send ^? / ^H Backspace**

Modern systems use ^? for backspace, while some legacy systems use ^H.

**Send Escape Sequence**

This action allows you to enter some text that will be sent when the associated key is pressed. First, the ESC character is sent, and then the text you entered is sent. There are no special characters and no escaping is necessary.

**Send Hex Code**

This action allows you to enter a sequence of hex codes that will be sent. Each value should begin with "0x" followed by one or two hex digits (0-9, a-f, or A-F). Each code should be separated by a space. You can see a list of hex codes on http://asciitable.com/ in the "Hx" column.

**Send Text**

This action allows you to enter a text string that will be sent when the associated key is pressed. The following escape characters are supported: \n (newline), \e (escape), \a (bell), \t (tab).

**Send Text with "vim" Special Characters**

This action allows you to enter a text string that will be sent when the associated key is pressed. The following special sequences are supported, where the "." characters are placeholders: ... (three-digit octal number), .. (two-digit octal number; must be followed by non-digit), . (one-digit octal number; must be followed by non-digit), \x.. (two-digit hex number), \x. (one-digit hex number), \u.... (four-digit hex number), \b (backspace), \e (escape), \f (form feed), \n (newline), \r (carriage return), \t (tab), \ (backslash), \" (double quote), <C-.> (control key), <M-.> (meta key)

**Find Regular Expression**

Performs a search for a saved regular expression.

**Undo**

Invokes the Undo action. Could be used to undo closing a session/tab/window.

**Paste (from selection)**

Same as Edit > Paste and Edit > Paste Special > Paste Selection

**Toggle Fullscreen**

This action enters or exits full screen mode.

**Toggle Hotkey Hides When Focus Lost**

Toggles whether the hotkey window hides when it loses focus.

**Run Coprocess**

This action launches a Coprocess. Learn more about coprocesses (documentation-coprocesses.html).

**Move Start/End of Selection Back/Forward**

Adjusts the range of selected text.

**Select Menu Item...**

This action allows you to enter the name of an iTerm2 menu item. It must be entered exactly the same as it appears in the menu. Ellipses can be typed with option-semicolon.

---

You can add a new keymapping by pressing "+". You can remove an existing mapping by selecting it and pressing "-". Three presets are provided: "Xterm defaults" is the normal key mappings, while "Xterm defaults with numeric keypad" disables the "application keypad" in favor of the numbers and symbols that the numeric keypad typically emits. "Terminal.app Compatibility" tries to emulate the way that Terminal.app sends keys by default.

**Left/Right Option Key Acts As**

It is common to use a modifier to send so-called "meta keys". For most users, selecting "+Esc" here is the right choice. The "Meta" option sets the high bit of the input character, and is not compatible with modern systems.

**Delete sends ^H**

If you are on a legacy system that does not accept ^? for backspace, select this and it will add a key mapping for you.

## Profiles > Advanced

### Triggers

Triggers are actions that are performed when text matching a regular expression is received. Each trigger has a regular expression, which defines when it runs. It has an action, which defines what it performs, and it has an optional parameter, whose meaning depends on the action. When the parameter is textual, \0 is replaced with the entire match, and \1...\9 are replaced with match groups. Each trigger has a checkbox in the "Instant" column. Instant triggers run as soon as text matching the regular expression is matched; triggers that are not instant only match after the cursor moves off the current line (such as whena newline is received).

Full details can be found at Triggers (documentation-triggers.html).

### Semantic History

Semantic history is used to open a file when you Cmd-Click on it. The current working directory for each line in the terminal is tracked to help find files. If Semantic History is set to "Open with default app," then files are passed to the OS to be opened with whatever is associated. Alternatively, you can choose "Open URL..." to open a specific URL (with \1 replaced with the filename and \2 replaced with the line number, if applicable). If you choose "Open with editor..." then text files will be opened with the designated editor, while other files are opened with the default app for their file type. For more flexibility, choose "Run command..." and specify a command to execute. \1 will be replaced with the file name, \2 will be replaced with the line number (if applicable), \3 with text in the line prior to the click location, \4 with text in the line subsequent to the click location, and \5 for the working directory of the line clicked on. Finally, "Always run command..." is like "Run command...," but takes effect even if the object clicked on is not an existing filename.

### Automatic Profile Switching

You can specify rules that, when satisified, changes any session's profile to this one. See Automatic Profile Switching (automatic-profile-switching.html) for all the details.

## Keys

### Remap modifier keys

iTerm2 allows you to change the meanings of the modifier keys only within iTerm2. This is useful, for example, if you find it difficult to press "option" for "meta" and would prefer to use "command" for that purpose.

#### To switch tabs

Tabs are normally navigated with cmd+number, but you can change the modifier used for that function here.

#### To switch windows

Windows are normally navigated with cmd+opt+number, but you can change the modifier used for that function here.

#### Show/Hide iTerm2 with a system-wide hotkey

When enabled, you can focus the Hotkey: field and press a keystroke. From then on, pressing that keystroke (even when iTerm2 is not the front application) will cause iTerm2 to come to the front. If it is the foreground app, it will be sent to the back. This requires that you enable access for assistive devices in the Universal Access panel of System Preferences.

#### Hotkey toggles a dedicated window with profile

If enabled, the hotkey set above will toggle a single window with a specific profile. This provides an always-available terminal.

#### Hotkey window hides when focus is lost

If enabled, the hotkey window will stay open even when another window gains keyboard focus.

#### Global shortcut keys

This interface works like the keyboard shortcut system in profiles (described above) but it affects all profiles. Settings here are overridden by those in a profile's key mappings.

### Arrangements

This tab lets you view saved window arrangements. You can delete them with the minus button and select the default arrangement.

### Pointer

#### Mouse Button and Trackpad Gesture Actions

You may assign custom actions to mouse clicks and trackpad gestures. The left mouse button is not configurable because its behavior is rather complex, however. This is especially useful if you have a mouse with many buttons. Any combination of mouse button + number of clicks + modifiers may be assigned an action. For gestures, three finger taps and swipes may be configured in combination with modifiers. The following actions are available:

**Extend selection**

The text selection will grow, either from its beginning or end, to the location of the pointer.

**Move pane**

The current pane will turn green. Click in another window's tab bar or in another pane to split to move the now-green pane.

**New Horizontal/Vertical split with profile**

The pane under the pointer will be split and the new split will use the specified profile.

**New Split/Tab/Window With Profile**

A new split pane/tab/window will be opened with the specified profile.

**Next/Previous Tab/Window**

Navigates through tabs and windows.

**Open Context Menu**

Opens the menu normally opened by a right click.

**Open URL in background**

Opens the URL under the pointer in your web browser without bringing the browser to the foreground.

**Open URL/Semantic History**

Opens the URL under the pointer, bringing the web browser to the foreground. If what's under the cursor is a filename on the local machine, it will be opened with Semantic History.

**Paste from Clipboard**

This is identical to Edit > Paste.

**Paste from Selection**

Pastes the most recent selection made in iTerm2, even if it's not what's in the pasteboard.

**Quicklook**

Defines the word under the cursor or, if it's a URL, opens it in a web browser popover.

**Select Next/Previous Pane**

Navigates panes according to how recently they were used.

**Select pane Above/Below/Left/Right**

Navigates panes by their layout.

**Send Escape Sequence...**

This action allows you to enter some text that will be sent when the associated key is pressed. First, the ESC character is sent, and then the text you entered is sent.

**Send Hex Code**

This action allows you to enter a sequence of hex codes that will be sent. Each value should begin with "0x" followed by one or two hex digits (0-9, a-f, or A-F). Each code should be separated by a space. You can see a list of hex codes on http://asciitable.com/ in the "Hx" column.

**Send Text**

This action allows you to enter a text string that will be sent when the associated key is pressed. The following escape characters are supported: \n (newline), \e (escape), \a (bell), \t (tab).

**Smart Selection**

Performs smart selection on the text under the pointer.

**Smart Selection ignoring Newlines**

Performs smart selection on the text under the pointer, ignoring newlines (e.g., if a URL is split by a hard newline, it can still be selected as a single item).

---

**Cmd-Click Opens Filename/URL**

If enabled, then clicking on a filename (of an existing file on the local machine) or a URL will open it.

**Ctrl-click reported to apps, does not open menu**

If enabled, ctrl-click will be sent to applications that support Xterm mouse reporting (if mouse reporting is enabled).

**Option-Click moves cursor**

If enabled, option-click will move the cursor to where the mouse pointer is. If you install shell integration, this will be well-behaved at the shell prompt by not sending up and down arrow keys.

**Focus follows mouse**

If enabled, moving the mouse over an inactive window will cause it to receive keyboard focus.

#### Double-click performs smart selection

If enabled, double click performs smart selection instead of word selection as is standard on OS X.

#### Triple-click selects full wrapped lines

If enabled, a triple click selects a whole line, even if it was longer than one row in the terminal. If off, then triple click selects exactly one row.

#### Three-finger tap reports middle click to apps

If enabled, a three-finger tap acts like a middle click for the purposes of mouse reporting.

---

# Scripting

iTerm2 features Applescript support which allows you to automate many aspects of its behavior. Quite a bit of customization is also possible by writing shell scripts.

## Applescript

iTerm2 has sophisticated Applescript support allowing one to write stand-alone scripts to launch the application and open multiple sessions with profiles into either new tabs or new windows. You can also set some other parameters for a session such as foreground and background colors, and transparency.

These scripts can then be saved as stand-alone executable applications.

#### Autolaunching Scripts

iTerm2 also supports autolaunching of an Applescript on startup. On startup, iTerm2 looks for an Applescript file in "~/Library/Application Support/iTerm/AutoLaunch.scpt". If it is found, the "AutoLaunch.scpt" script is launched and executed.

#### User-Defined Scripts

iTerm2 also supports launching of user defined scripts from the "Scripts" menu. The scripts need to be stored under the ~/Library/Application Support/iTerm/Scripts directory. You can create this directory if it does not already exist. iTerm2 checks this directory on startup. Scripts must be named with the extension .scpt or .app.

# Syntax

The following is a comprehensive list of supported applescript commands, methods, and properties:

```
tell application iTerm2
  -- application-level commands
  -- These commands return a window.
  set newWindow to (create window with default profile)
  set newWindow to (create window with default profile command "ls -l -R /")

  select first window
  set newWindow to (create window with profile "Default")
  set newWindow to (create window with profile "Default" command "ls -l -R /")

  -- window-level commands
  repeat with aWindow in windows
    tell aWindow
      tell current session
        set newSession to (split horizontally with default profile)
        -- Optional command argument added 12/5/2015
        set newSession to (split horizontally with default profile command "ssh example.com")
      end tell
    end tell
  end repeat
  tell current window
    -- These commands return a tab
    set newTab to (create tab with default profile)
    set newTab to (create tab with profile "Projection")
  end tell
  tell current window
    tell current session
      set columns to 40
      set rows to 40
    end tell
  end tell

  -- tab-level commands
  tell current window
    tell second tab
      select
    end tell
    tell first tab
      close
    end tell
    tell current tab
      repeat with aSession in sessions
        tell aSession
          write text "Hello"
        end tell
      end repeat
       end tell
   end tell

  -- session-level commands
  tell current session of first window
    write text "cat > /dev/null"
    write text "cat > /dev/null" newline NO
    write contents of file "/etc/passwd"
    -- Get the path to the current session's tty and write it
    write text (tty)
    -- Get the content of the session and write it back
    write text (contents)
```

```
    -- Get the session's unique identifier and write it back
    write text (unique ID)

    -- These commands return a session
    set newSession to (split vertically with default profile)
    set newSession to (split vertically with profile "Default")
    set newSession to (split vertically with same profile)

    -- Optional command argument added 12/5/2015
    set newSession to (split vertically with default profile command "ssh example.com")
    set newSession to (split vertically with profile "Default" command "ssh example.com")
    set newSession to (split vertically with same profile command "ssh example.com")

    set newSession to (split horizontally with default profile)
    set newSession to (split horizontally with profile "Default")
    set newSession to (split horizontally with same profile)

    -- Optional command argument added 12/5/2015
    set newSession to (split horizontally with default profile command "ssh example.com")
    set newSession to (split horizontally with profile "Default" command "ssh example.com")
    set newSession to (split horizontally with same profile command "ssh example.com")

    set foreground color to {65535, 0, 0, 0}
    set background color to {65535, 0, 0, 0}
    set bold color to {65535, 0, 0, 0}
    set cursor color to {65535, 0, 0, 0}
    set cursor text color to {65535, 0, 0, 0}
    set selected text color to {65535, 0, 0, 0}
    set selection color to {65535, 0, 0, 0}
    set ANSI black color to {65535, 0, 0, 0}
    set ANSI red color to {65535, 0, 0, 0}
    set ANSI green color to {65535, 0, 0, 0}
    set ANSI yellow color to {65535, 0, 0, 0}
    set ANSI blue color to {65535, 0, 0, 0}
    set ANSI magenta color to {65535, 0, 0, 0}
    set ANSI cyan color to {65535, 0, 0, 0}
    set ANSI white color to {65535, 0, 0, 0}

    set ANSI bright black color to {65535, 0, 0, 0}
    set ANSI bright red color to {65535, 0, 0, 0}
    set ANSI bright green color to {65535, 0, 0, 0}
    set ANSI bright yellow color to {65535, 0, 0, 0}
    set ANSI bright blue color to {65535, 0, 0, 0}
    set ANSI bright magenta color to {65535, 0, 0, 0}
    set ANSI bright cyan color to {65535, 0, 0, 0}
    set ANSI bright white color to {65535, 0, 0, 0}

    set background image to "/usr/share/httpd/icons/small/rainbow.png"
    set name to "New Name"
    set transparency to 0.5
    -- The name of the session's profile (different from the
    -- session's name, which can be changed by editing the Session
    -- Title field in Edit Session or by an escape sequence).
    -- Added 10/6/15.
    write text (profile name)

    -- is processing means it has received output in the last two seconds.
    if (is processing) then
      set foreground color to { 65535, 65535, 65535, 65535 }
```

```
    end if

    -- This will only work if shell integration is installed.
    -- Otherwise it always returns false.
    if (is at shell prompt) then
      set background color to { 65535, 0, 65535, 65535 }
    end if

    -- New in 2.9.20160104
    set answerback string to "Hello world"

    -- New in 2.9.201601. See https://iterm2.com/badges.html for more on variables.
    variable named "session.name"
    set variable named "user.phaseOfTheMoon" to "Gibbous"
  end tell
end tell
```

## Supporting both old and new versions of iTerm2

If your application needs to support both the old and new Applescript syntax, this is the recommended technique:

```
on theSplit(theString, theDelimiter)
    set oldDelimiters to AppleScript's text item delimiters
    set AppleScript's text item delimiters to theDelimiter
    set theArray to every text item of theString
    set AppleScript's text item delimiters to oldDelimiters
    return theArray
end theSplit

on IsModernVersion(version)
    set myArray to my theSplit(version, ".")
    set major to item 1 of myArray
    set minor to item 2 of myArray
    set veryMinor to item 3 of myArray

    if major < 2 then
        return false
    end if
    if major > 2 then
        return true
    end if
    if minor < 9 then
        return false
    end if
    if minor > 9 then
        return true
    end if
    if veryMinor < 20140903 then
        return false
    end if
    return true
end IsModernVersion

on NewScript()
    -- Return the modern script as a string here; what follows is an example.
```

```
    return "create window with default profile"
end NewScript

on OldScript()
    -- Return the legacy script as a string here; what follows is an example.
    return "
    set myTerm to (make new terminal)
    tell myTerm
        launch session \"Default\"
    end tell"
end OldScript

tell application "iTerm"
    if my IsModernVersion(version) then
        set myScript to my NewScript()
    else
        set myScript to my OldScript()
    end if
end tell

set fullScript to "tell application \"iTerm\"
" & myScript & "
end tell"

run script fullScript
```

---

# Shell Integration

iTerm2 may be integrated with the unix shell so that it can keep track of your command history, current working directory, host name, and more--even over ssh. This enables several useful features.

## How To Enable Shell Integration

The easiest way to install shell integration is to select the *iTerm2>Install Shell Integration* menu item. It will download and run a shell script as described below. You should do this on every host you ssh to as well as your local machine. The following shells are supported: tcsh, zsh, bash, and fish 2.2 or later. Contributions for other shells are most welcome.

When you select the *iTerm2>Install Shell Integration* menu item, it types this for you:

```
curl -L https://iterm2.com/misc/install_shell_integration.sh | bash
```

Don't care for piping curl to bash? Do it by hand. First, download the right script for your shell:

```
curl -L https://iterm2.com/misc/`basename $SHELL`_startup.in >> \
~/.iterm2_shell_integration.`basename $SHELL`
```

Then add this to your login script (.login for tcsh, .bash_profile for bash, .zshrc for zsh, or config.fish file for fish):

```
source ~/.iterm2_shell_integration.`basename $SHELL`
```

Don't want to or can't install a login script? See the workaround at the end of this document using triggers.

## Features

Shell Integration enables numerous features:

### Marks

These are saved locations in history. They make it easy to navigate to previous shell prompts or other locations of interest.

### Alert when current command finishes running.

iTerm2 will present a modal alert when a long-running command finishes, if you ask it to.

### View information about commands.

You can see the return status code, working directory, running time, and more for shell commands entered at the prompt in the past.

### Download files from remote hosts with a click.

You can right click on a filename (e.g., in the output of *ls*) to download it.

### Drag-drop files to upload with scp.

Hold down option and drag-drop a file from Finder into iTerm2 to upload it.

### View command history.

It can be seen and searched in the toolbelt or quickly accessed in a popup window.

### Easy access to recently and frequently used directories.

iTerm2 remembers the directories you use, sorting them by "frecency" and giving you access to them in the toolbelt and in a popup window.

### Assign profiles to hostnames, usernames, or username+hostname combinations.

Sessions will automatically switch profiles as you log in and out according to rules you define.

### Ensures the command prompt always starts at the left column, even when the last command didn't end in a newline.

Each of these features are described in more detail below.

## How it works

Shell Integration works by configuring your shell on each host you log into to send special escape codes that convey the following information:

- Where the command prompt begins and ends.
- Where a command entered at the command prompt ends and its output begins.
- The return code of the last-run command.
- Your username.
- The current host name.
- The current directory.

## How to use it

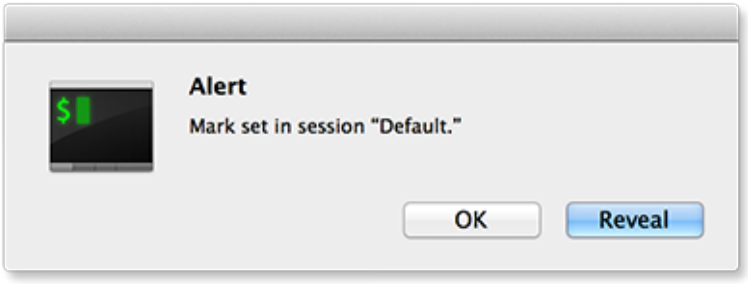
### Marks

When shell integration is enabled, iTerm2 automatically adds a mark at each command prompt. Marks are indicated visually by a small blue triangle in the left margin.


You can navigate marks with Cmd-Shift-Up and Down-arrow keys.

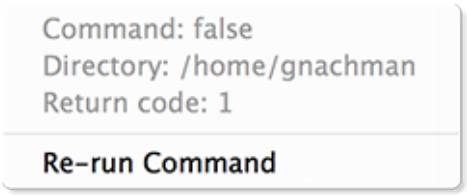
### Alert on next mark

iTerm2 can show an alert box when a mark appears. This is useful when you start a long-running command. Select *Edit>Marks and Annotations>Alert on next mark* (Cmd-Opt-A) after starting a command, and you can go do something else in another window or tab. When the command prompt returns, a modal alert will appear, calling attention to the finished job.



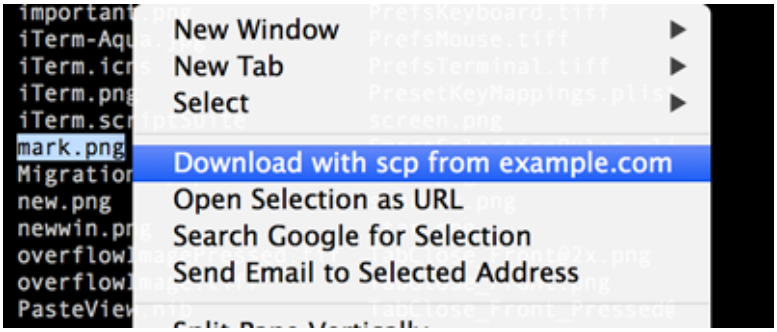

### Command status

The mark on a command line will turn red if a command fails. You can right click the mark to view its return code.

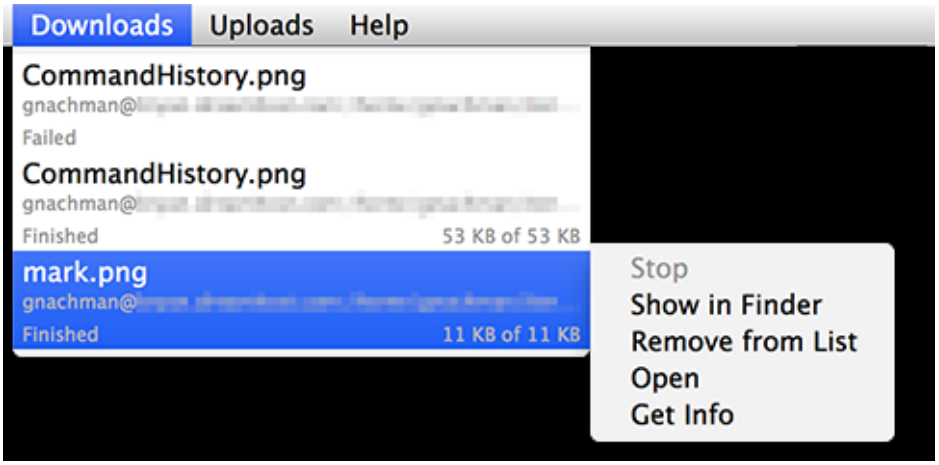


### Download with scp

You can right-click on a filename (e.g., in the output of *ls*) and select *Download with scp from* hostname**, and iTerm2 will download the file for you.




A new menu bar item will be added called *Downloads* that lets you view downloaded files and track their progress.




### Upload with scp

If you drop a file (e.g., from Finder) into iTerm2 while holding the option key, iTerm2 will offer to upload the file via scp to the remote host into the directory you were in on the line you dropped the file on. A new menu bar item will be added called *Uploads* that lets you view uploaded files and track their progress.

### Command history

With shell integration, iTerm2 can track your command history. The command history is stored separately for each username+hostname combination. There are four places where this is exposed in the UI:

#### Command history popup

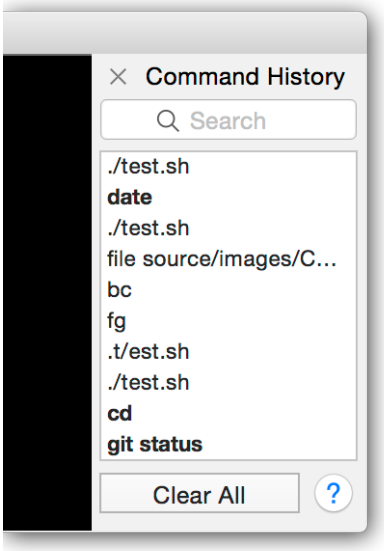
You can view and search the command history with *Edit>Open Command History...* (Shift-Cmd-;).

#### Autocomplete

Commands in command history are also added to Autocomplete (Cmd-;). If *Preferences>General>Save copy/paste history and command history to disk* is enabled, then command history will be preserved across runs of iTerm2 (up to 200 commands per user/hostname).

### Toolbelt

A command history tool may be added to the toolbelt by selecting *Toolbelt>Command History*.




Bold commands are from the current session. Clicking on one will scroll to reveal it. Double-clicking enters the command for you. Option-double-clicking will output a "cd" command to go to the directory you were in when it was last run.

### Command Completion

iTerm2 will present command completion suggestions automatically when *View>Auto Command Completion* is selected.

## Recent Directories

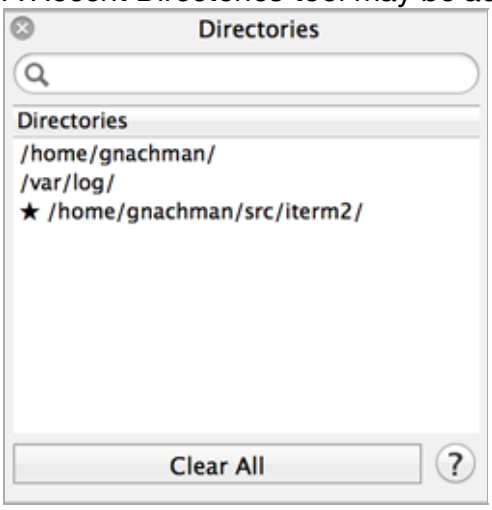
With shell integration, iTerm2 will remember which directories you have used recently. The list of preferred directories is stored separately for each username+hostname combination. It is sorted by "frencency" (frequency and recency of use). There are two places it is exposed in the UI:

### Recent Directories popup

You can view and search your recently and frequently used directories in *Edit>Open Recent Directories...* (Cmd-Opt-;).

### Toolbelt

A *Recent Directories* tool may be added to the toolbelt by selecting *Toolbelt>Recent Directories*.




Double-clicking a directory will type its path for you into the current terminal. Option-double-click will enter a "cd" command for you. You can also right-click on a directory to toggle its "starred" status. A starred directory will always appear at the bottom of the list so it is easy to find.

### Automatic Profile Switching

Please see the documentation at Automatic Profile Switching (/automatic-profile-switching.html).

### Shell Integration for root

If you'd like to be able to use shell integration as root, you have two options. The first option, presuming you use bash, is to become root with `sudo -s` (which loads your `.bashrc` as root) and add this to your `.bashrc`:

```
test $(whoami) == root && source "${HOME}/.iterm2_shell_integration.bash"
```

The alternative is to use Triggers to emulate shell integration as described in the following section.

### Triggers

For some users, installing a login script on every host they connect to is not an option. To be sure, modifying root's login script is usually a bad idea. In these cases you can get the benefits of shell integration by defining triggers. There are two relevant triggers: *Report Host & User* and *Report Directory*

Use these triggers to tell iTerm2 your current username, hostname, and directory. Suppose you have a shell prompt that looks like this:

```
george@example.com:/home/george%
```

It exposes the username, hostname, and working directory. We can harvest those with a regular expression. First, define a trigger with this regex:

```
^(\w+)@([\w.]+):.+%
```

It captures the username and hostname from the example prompt above. Set the trigger's parameter to:

```
\1@\2
```

Then create another trigger with the action *Set Directory*. This regular expression will extract the directory from the example prompt:

```
^\w+@[\w.]+:([^%]+)%
```

Set this trigger's parameter to

```
\1
```

Make sure both triggers have their *Instant* checkbox enabled so they'll take effect before a newline is received.

You may specify a user name or host name alone to *Report Host & User*. If you give just a user name then the previous host name will be preserved; if you give just a host name then the previous user name will be preserved. To change the user name only, give a parameter like `user@`. To change the host name only, give a parameter like `example.com`.

### A Note on SCP

iTerm2 can do uploads and downloads with scp as described above. There are a few things you should know.

iTerm2 links in libssh2, and does not shell out to scp. It respects /etc/known_hosts and ~/.ssh/known_hosts, and will update the latter file appropriately. Host fingerprints are verified. Password, keyboard-interactive, and public-key authentication are supported. Private keys by default come from ~/.ssh/id_rsa, id_dsa, or id_ecdsa, and may be encrypted with an optional passphrase.

iTerm2 respects ssh_config files, but only a subset of the commands are understood:

- Host
- HostName
- User
- Port
- IdentityFile

Settings pulled from ssh_config override the hostname and user name provided by shell integration. The shell integration-provided host name is used as the text against which *Host* patterns are matched.

The following files are parsed as ssh_config files, in order of priority:

- ~/Library/Application Support/iTerm/ssh_config
- ~/.ssh/ssh_config
- /etc/ssh_config

The scp code is relatively new. If you are in a high-security environment, please keep this in mind.

---

## Smart Selection

iTerm2 offers a Smart Selection feature that simplifies making selections on semantically recognizable objects.

### How do I use Smart Selection?

A quad-click (four clicks of the left mouse button in quick succession) activates Smart Selection at the mouse cursor's position. By default, the following kinds of strings are recognized:

- Words bounded by whitespace or line boundaries.
- C++-style pairs of identifiers separated by double colons, such as "namespace::identifier".
- Filesystem paths, such as "/foo/bar/baz.txt".
- Quoted strings such as "foo bar".
- Java or Python-style include paths, such as "foo.bar.baz".
- URIs with the schemes: mailto, http, https, ssh, or telnet.
- Objective-C selectors like "@selector(foo:bar:)".
- Email addresses.

### How do I Change Smart Selection Rules?

Under Preferences>Profiles>Advanced, you may edit the smart selection rules. In addition to a regular expression, each rule also has a Precision attribute, which takes a value of Very Low, Low, Normal, High, or Very High. Intuitively, it refers to how sure one can be that when a rule's regular expression finds a match that it is what the user intended. For example, the "Word" rule is low precision (it matches almost every time), while the "HTTP URL" rule is very high precision (it almost never produces false positives). This allows the "HTTP URL" rule to take precedence when both match, unless the "Word" rule matches a much longer string. That might happen, for instance, if there were a non-URL character after a URL followed by a lot more text. The precision levels have a very strong effect, so it's very rare for a lower precision rule to take precedence over a higher precision rule.

When editing rules, it is advised that you experiment with different precision levels and different kinds of strings to find one that works well. A collection of test cases may be found at smart_selection_cases.txt.

When Smart Selection is activated, iTerm2 tries each regular expression. For a given regex, various strings on the screen are tested until the longest match is found. Only matches that include the character under the cursor are of interest. The longest such match is added to a pool of "selection candidates". Each candidate is assigned a score equal to its length in characters. Among the candidates in the highest precision class (where Very High is the highest class and Very Low is the lowest) with any matches, the higheset scoring one is used as the selection.

### Actions

Actions may be associated with smart selection rules. When you right click in a terminal, smart selection is performed at the cursor's location. Any smart selection rule that matches that location will be searched for associated actions, and those actions will be added to the context menu. Actions may open a file, open a URL, run a command, or start a coprocess. A cmd-click on text matching a smart selection rule will invoke the first rule.

### Regular Expressions

Regular expressions conform to the ICU regular expressions (http://userguide.icu-project.org/strings/regexp) rules.

---

## Triggers

A trigger is an action that is performed when text matching some regular expression is received in a terminal session.

### How to Create a Trigger

To create a trigger, open the **Preferences** panel. Select the **Profiles** tab. Choose the profile to which you wish to add a trigger. Then select the **Advanced** tab. Click the **Edit** button in the **Triggers** section. A panel opens that displays any existing triggers. You can click the **+** button to add a new trigger.

Triggers have a regular expression, an action, an optional parameter, and may be marked as *Instant*.

### Regular Expression

Regular expressions conform to the ICU regular expressions (http://userguide.icu-project.org/strings/regexp) rules. Text that is written to the screen including the BEL control code are sent to the regex matcher for evaluation. Only one line at a time is matched. By default, matching is performed when a newline or cursor-moving escape code is processed. If a line is very long, then only the *last* three wrapped lines are used (that is, the last three lines as seen on the display). This is done for performance reasons. You can change this limit in Advanced Preferences > Number of screen lines to match against trigger regular expressions.

### Actions

The following actions are available:

- Bounce Dock Icon: Makes the dock icon bounce until the iTerm2 window becomes key.
- Capture Output: Save the line to the Captured Output toolbelt tool. See Captured Output (captured_output.html). The parameter is text to send (as though it had been typed) when you double-click on an entry in the Captured Output tool.
- Highlight Text: The text matching the regex in the trigger will change color. The parameter sets the color.

- Open Password Manager: Opens the password manager. You can specify which account to select by default.
- Post Notification: Posts a notification with Growl (if available) or Notification Center.
- Report Directory: Tells iTerm2 what your current directory is. You can use this to enable Shell Integration (shell_integration.html) features without installing the scripts. The parameter is your current directory.
- Report User & Host: Tells iTerm2 what your user or host name is. You can use this to enable Shell Integration (shell_integration.html) features without installing the scripts. To specify just a user name, say `user@`. For just a host, say `@host`. For both, say `user@host`.
- Ring Bell: Plays the standard system bell sound once.
- Run Command: Runs a user-defined command.
- Run Coprocess: Runs a Coprocess (coprocesses.html).
- Send Text: Sends user-defined text back to the terminal as though the user had typed it.
- Set Mark: Sets a mark. You can specify whether you'd like the display to stop scrolling after the trigger fires.
- Show Alert: Shows an alert box with user-defined text.
- Stop Processing Triggers: When this action is invoked no triggers further down the list will be invoked for the current text.

**Parameter?**

Various actions (Run Command, Run Coprocess, Send Growl Alert, Send Text, and Show Alert) require additional information. This is specified in the "Parameters" field. When the paramter is a text field with freeform entry, some special values are defined:

| Value | Meaning |
|---|---|
| \0 | The entire value matched by the regular expression. |
| \1, \2, ..., \9 | The nth value captured by the regular expression. |
| \a | A BEL character (^G). |
| \b | A backspace character ^H. |
| \e | An ESC character (ascii 27). |
| \n | A newline character. |
| \r | A linefeed character. |
| \t | A tab character. |
| \xNN | A hex value NN (for example: \x1b sends ascii code 27, an ESC). |

### Instant

When *Instant* is set, the trigger will fire once per line as soon as the match occurs, without waiting for a newline. This was added for the benefit of the *Open Password Manager* trigger, since password prompts usually are not followed by a newline. This may cause certain regular expressions (for example, ".*") to match less than they otherwise might.

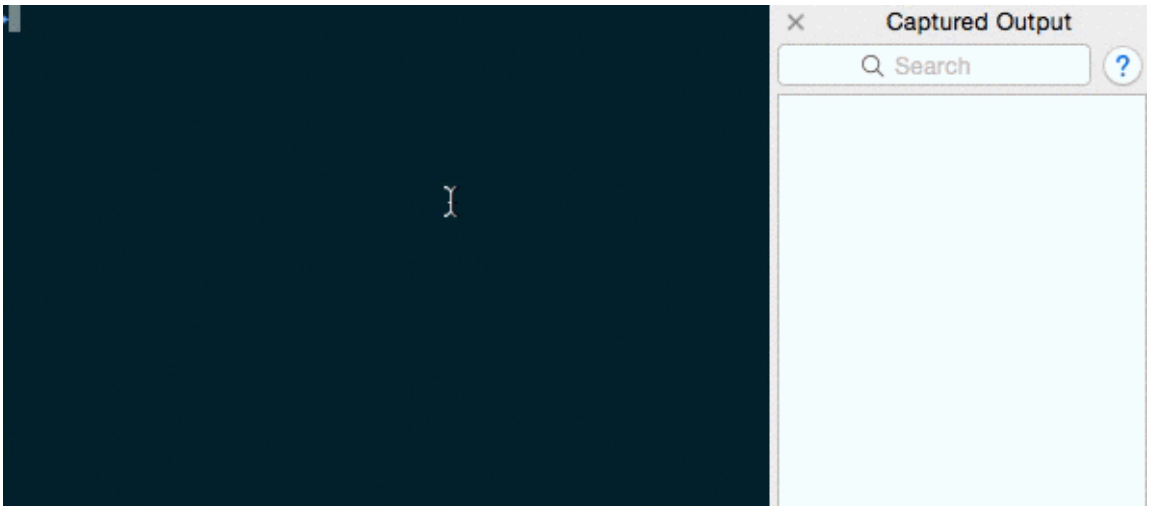
### Example

The iTerm2-zmodem (https://github.com/mmastrac/iterm2-zmodem) project demonstrates hooking up iTerm2 to zmodem upload and download.

---

## Captured Output

iTerm2 has a feature called "Captured Output" which helps you find and track important lines of output from logs, build processes, and such.




### What does it do?

Captured Output is a tool may be added to iTerm2's toolbelt (a view on the right side of terminal windows). It works in conjunction with user-defined Triggers (https://www.iterm2.com/triggers.html). A Trigger whose action is *Capture Output* looks for lines of output that match its regular expression. When one is found, the entire line is added to the Captured Output tool. When the user clicks on a line in the Captured Output tool, iTerm2 scrolls to reveal that line. Double-clicking on a line in the Captured Output tools run a user-defined Coprocess (https://www.iterm2.com/coprocesses.html).

### Shell Integration Required

Shell Integration (https://www.iterm2.com/shell_integration.html) must be installed because Captured Output ties in to command history.

### Example

One way to use Captured Output is to view compiler output. Suppose you run *make* and get thousands of lines of output, some of which may contain errors or warnings. You'd like to examine each one and take some action on it. Here's how you would use Captured Output to assist with this task:

**Step 1: Create Triggers**

Create a Trigger (/documentation-triggers.html) that matches your compiler's errors and warnings. Clang's errors look like this:

```
filename.c:54:9: error: use of undeclared identifier 'foo'
```

The following regular expression matches errors and warnings from Clang:

```
^([a-zA-Z0-9+/.-]+):([0-9]+):[0-9]+: (?:error|warning):
```

There are two capture groups defined. We'll come back to those later.

**Step 2: Open the Toolbelt**

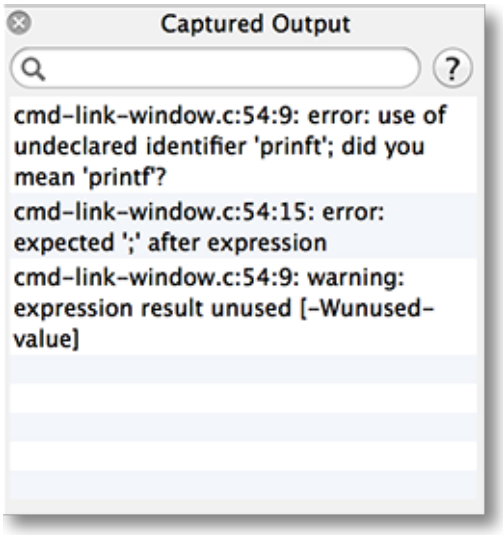
Open the Toolbelt by selecting the menu item Toolbelt > Show Toolbelt. Enable the Toolbelt > Captured Output menu item to ensure it is visible.

**Step 3: Run make**

Kick off the build by running make. It spits out thousands of lines of output.

**Step 4: Examine the Captued Output tool**

Any errors or warnings that appear in the compiler output will appear in the Captured Output tool.




Select an entry in the tool. iTerm2 scrolls to display the line and briefly highlights it in blue.

**Step 5: Open the file containing the error**

The Trigger created in step 1 takes an optional parameter. It is a command for iTerm2 to exceute as a Coprocess (https://www.iterm2.com/coprocesses.html) when you double-click an entry in the Captured Output tool. An example command is:

```
echo :e \1; sleep 0.5; echo \2G
```

This coprocess command assumes you are in vi, and types a command to open the offending file and scrolls to the offending line. This is where the capture groups in the regular expression from step 1 become useful. For example, if the filename was "filename.c" and the error was on line 20, as in this error message:

```
filename:c:20:9 error: use of undeclared identifier 'foo'
```

The coprocess would:

- Type ":e filename.c", followed by enter, as though you were typing it at the keyboard.
- Wait half a second.
- Type "20G".

#### Step 6: Check it off the list

You can right-click on an entry in Captured Output to open a menu, which contains a single item: "Toggle Checkmark". This helps you remember which entries have been dealt with as you go through errors and warnings in your compiler output.

### Navigation

Captured Output is linked to the Command History tool. If no command is selected in the Command History tool, then the most recent captured output is displayed. Otherwise, the captured output from the selected command is displayed. You can remove a selection from the Command History tool by cmd-clicking on it.

---

## Fonts

While iTerm2 does not require monospaced fonts, they look much better than proportionately spaced fonts. If you want to use Consolas, you'll need to correct its baseline offset as described at how to fix Consolas baseline (http://mbauman.net/geek/2009/03/15/minor-truetype-font-editing-on-a-mac/).

iTerm2 has the capability of rendering text with thin strokes to improve readability. You can change how this works in the **Text** panel of the **Profiles** tab of **Preferences**.

You can also specify the a "non-ASCII" font in the **Text** panel of profile preferences. This font will be used for all code points greater than or equal to 128 or for characters with combining marks.

---

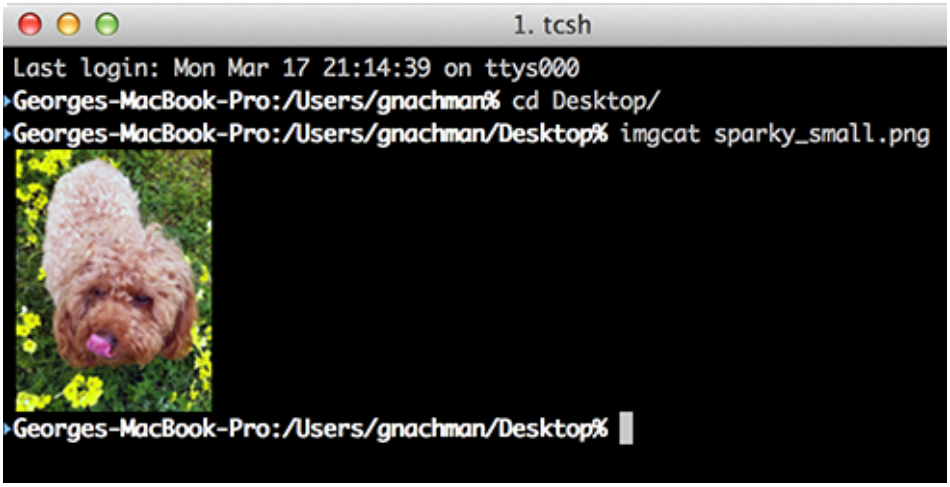
## Inline Images

iTerm2 is able to display images within the terminal. Using a similar mechanism, it can also facilitate file transfers over any transport (such as ssh or telnet), even in a non-8-bit-clean environment.

> Just want to try it out and don't care about the protocol? Use the imgcat tool. Download imgcat here (https://raw.github.com/gnachman/iTerm2/master/tests/imgcat)

### Example: imgcat

Using the imgcat (https://raw.github.com/gnachman/iTerm2/master/tests/imgcat) script, one or more images may be displayed in a terminal session. For example:




Critically, animated GIFs are supported as of version 2.9.20150512.

### Protocol

iTerm2 extends the xterm protocol with a set of proprietary escape sequences. In general, the pattern is:

```
ESC ] 1337 ; key = value ^G
```

Whitespace is shown here for ease of reading: in practice, no spaces should be used.

For file transfer and inline images, the code is:

```
ESC ] 1337 ; File = [optional arguments] : base-64 encoded file contents ^G
```

The optional arguments are formatted as `key=value` with a semicolon between each key-value pair. They are described below:

| Key | Description of value |
|---|---|
| name | base-64 encoded filename. Defaults to "Unnamed file". |
| size | File size in bytes. Optional; this is only used by the progress indicator. |
| width | Width to render. See notes below. |

| | |
|---|---|
| height | Height to render. See notes below. |
| preserveAspectRatio | If set to 0, then the image's inherent aspect ratio will not be respected; otherwise, it will fill the specified width and height as much as possible without stretching. Defaults to 1. |
| inline | If set to 1, the file will be displayed inline. Otherwise, it will be downloaded with no visual representation in the terminal session. Defaults to 0. |

The width and height are given as a number followed by a unit, or the word "auto".

- *N*: *N* character cells.
- *N*px: *N* pixels.
- *N*%: *N* percent of the session's width or height.
- auto: The image's inherent size will be used to determine an appropriate dimension.

### More on File Transfers

By omitting the `inline` argument (or setting its value to 0), files will be downloaded and saved in the *Downloads* folder instead of being displayed inline. Any kind of file may be downloaded, but only images will display inline. Any image format that OS X supports will display inline, including PDF, PICT, EPS, or any number of bitmap data formats (PNG, GIF, etc.). A new menu item titled *Downloads* will be added to the menu bar after a download begins, where progress can be monitored and the file can be located, opened, or removed.

### Sample Code

Sample code for displaying images may be found here.

**imgls (https://raw.github.com/gnachman/iTerm2/master/tests/imgls)**

Provides an augmented directory listing that includes a thumbnail of each image in a directory.

**imgcat (https://raw.github.com/gnachman/iTerm2/master/tests/imgcat)**

Displays one or more images inline at their full size.

**download.sh (https://raw.github.com/gnachman/iTerm2/master/tests/download.sh)**

Downloads a file, but does not display it inline.

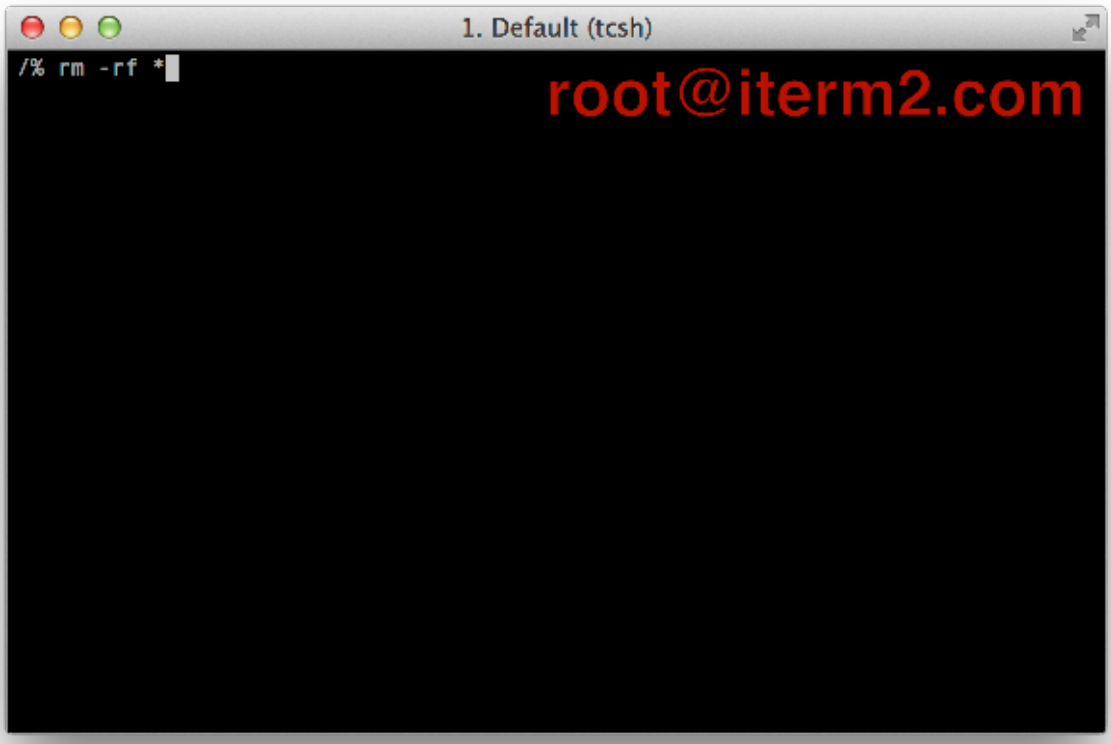
**divider (https://raw.github.com/gnachman/iTerm2/master/tests/divider)**

Draws a full-width, one line-tall graphical divider.


## Badges

A *badge* is a large text label that appears in the top right of a terminal session to provide dynamic status, such as the current host name or git branch. Its initial value is defined in **Preferences>Profiles>General>Badge** and it can be changed by an iTerm2-proprietary escape sequence. It may also reference iTerm2- and user-defined variables.

Here is an example of a session with a badge indicating the current user and host name.




### Availability

Support for badges is available in version 2.9 and later, currently distributed in the nightly builds (/nightly/latest) and is in the master branch of the GitHub repo (https://github.com/gnachman/iTerm2). It is not in version 2.1.1.

## Variables

A badge may reference variables. There are two kinds of variables: user-defined variables and iTerm2-defined variables. User-defined variables may be set by an escape sequence described below.

The syntax for referencing a variable is to place the variable name between \( and ). For example:

```
\(session.username)@\(session.hostname)
```

Undefined variables evaluate to an empty string.

## iTerm2-Defined Variables

The following variables are defined by iTerm2:

| Variable Name | Description |
|---|---|
| session.name | The current session's name. Defaults to the profile name. May be changed with the escape sequence **OSC 1 ; *name* ST**, or by editing the session title in **View>Edit Current Session**. |
| session.columns | The number of columns in the current session. |
| session.rows | The number of rows in the current session. |
| session.hostname | The current hostname. Only set if Shell Integration (shell_integration.html) is installed. |
| session.username | The current username. Only set if Shell Integration (shell_integration.html) is installed. |
| session.path | The current path. Works best if Shell Integration (shell_integration.html) is installed. |

## User-Defined Variables

You can provide additional information in the form of user-defined variables. To create a user-defined variable you must modify your shell's rc script by defining a function named iterm2_print_user_vars that calls iterm2_set_user_var one or more times. See the example below.

Here's an example that sets a user-defined variable called "gitBranch" to the git branch of the current directory. Pick your shell to see the version you need: bash | fish | tcsh | zsh

```
# bash: Place this in .bashrc.
function iterm2_print_user_vars() {
  iterm2_set_user_var gitBranch $((git branch 2> /dev/null) | grep \* | cut -c3-)
}
```

To reference a user-defined variable, refer to it as *\(user.variableName)*. For example, to use the example above, you could set your profile's badge to:

```
Current git branch on \(session.hostname) is \(user.gitBranch)
```

### Escape Sequences

User-defined variables may be set with the following escape sequence:

```
OSC 1337 ; SetUserVar=name=Base64-encoded value ST
```

This is what iterm2_set_user_var sends. Generally you should use the *iterm2_print_user_vars* mechanism described above instead of sending this escape sequence directly.

The badge itself may be set with the following escape sequence:

```
OSC 1337 ; SetBadgeFormat=Base-64 encoded badge format ST
```

Here's an example that works in bash:

```
# Set badge to show the current session name and git branch, if any is set.
printf "\e]1337;SetBadgeFormat=%s\a" \
  $(echo -n "\(session.name) \(user.gitBranch)" | base64)
```

### Color

The badge's color may be set in *Preferences>Profiles>Colors*.

---

## Dynamic Profiles

*Dynamic Profiles* is a feature that allows you to store your profiles in a file outside the usual OS X preferences database. Profiles may be changed at runtime by editing one or more plist files (formatted as JSON, XML, or in binary). Changes are picked up immediately.

### Availability

Dynamic Profiles are available in iTerm2 2.9.20140923 and later.

### Usage

When iTerm2 starts, it creates a folder:

```
~/Library/Application Support/iTerm2/DynamicProfiles
```

While iTerm2 runs, it monitors the contents of that folder. Any time the folder's contents change, all files in it are reloaded.

Files in this folder are expected to be formatted as Apple Property Lists (https://en.wikipedia.org/wiki/Property_list). No particular file extension is required. All files in the folder must be valid property lists. If any is malformed, then no changes will be processed.

## Property List Format

A property list describes a data structure consisting of arrays, dictionaries, strings, integers, and boolean values. Property lists may be written in JSON or XML. Here's an example of the skeletal structure of a JSON property list that iTerm2 expects for Dynamic Profiles:

```
{
  "Profiles": [
    {
       [attributes for the first profile go here]
    },
    {
       [attributes for the second profile go here]
    },
    [more profiles]
  ]
}
```

There are two required fields for each profile:

- Guid
- Name

The "Guid" is a globally unique identifier. It is used to track changes to the profile over time. No other profile should ever have the same guid. One easy way to generate a Guid is to use the *uuidgen* program, which comes standard with OS X.

The "Name" is the name, as seen in the Profiles window or in Preferences.

Here is a fully formed (but minimal) Dynamic Profiles plist:

```
{
  "Profiles": [
    {
      "Name": "Example",
      "Guid": "ba19744f-6af3-434d-aaa6-0a48e0969958"
    }
  ]
}
```

## Attributes

Every profile preference that iTerm2 supports may be an attribute of a Dynamic Profile. Since there are dozens of attributes, you usually won't specify them all. Any attribute not specified will inherit its value from the default profile, or a specified "parent" profile (see below).

The easiest way to find the name and legal value of a profile attribute is to copy it from a known-good reference. To get the JSON for a profile you already have, follow these steps:

1. Open Preferences > Profiles
2. Select a profile
3. Open the *Other Actions* menu beneath the list of profiles
4. Select *Copy Profile as JSON*
5. Paste the clipboard contents into your favorite text editor

If you paste a whole profile into a Dynamic Profile this way, make sure you remember to change the Guid. A Dynamic Profile with a Guid equal to an existing Guid of a regular profile will be ignored.

### Parent Profiles

Normally, a dynamic profile inherits any attributes you don't explicitly specify from the default profile. You may also specify a particular profile to inherit from using the *Dynamic Profile Parent Name* attribute. The value it takes is a profile name (that is, the name you see listed in the list of profiles in Preferences box). Profile names are not guaranteed to be unique, but they are more convenient than GUIDs. If no profile with the specified name is found, the default profile is used instead. For example:

```
{
  "Profiles": [
    {
      "Name": "Example",
      "Guid": "ba19744f-6af3-434d-aaa6-0a48e0969958",
      "Dynamic Profile Parent Name": "Light Background"
    }
  ]
}
```

## Minutiae

Dynamic profiles are loaded in alphabetical order by filename. Within a particular file, they are loaded in the order they're listed in. This only matters if one dynamic profile references another dynamic profile as its parent; the parent should be placed so it loads before any of its children. For all other purposes, the filenames don't matter.

The *Dynamic* will automatically be added to all Dynamic Profiles.

## Troubleshooting

If something goes wrong loading a Dynamic Profile, errors will be logged to /var/log/system.log, and are usually visible by running Console.app.

## Example

Here's an example for a common use case: a list of profiles for *ssh*ing to various hosts. In this example, I've used the hostname as the Guid, which makes constructing this file a little easier and works well enough.

```
{
  "Profiles": [
    {
       "Name": "foo.example.com",
       "Guid": "foo.example.com",
       "Custom Command" : "Yes",
       "Command" : "ssh foo.example.com",
    },
    {
       "Name": "bar.example.com",
       "Guid": "bar.example.com",
       "Custom Command" : "Yes",
       "Command" : "ssh bar.example.com",
    },
  ]
}
```

---

# Profile Search Syntax

When iTerm2 presents a list of profiles, it usually includes a search box. The search box uses a special syntax that letes you tailor your searches to quickly find what you're looking for.

## Searching Profiles

Each word in the search query must match at least one word in either the title or the tags of of a profile in order for that profile to be matched by the query. For a word to be a match, it must be a substring.

| Query | Profile Name | Matches? |
|---|---|---|
| Linux | **Linux** | Yes |
| x | Linu**x** | Yes |
| z | Linux | No |
| George L | **George**'s **L**inux Machine | Yes |

## Operators

You may prefix a phrase in the search query with an *operator* to narrow your query. Only two operators are defined:

- The *name:* operator only tries to match words in the profile's name.
- The *tag:* operator only tries to match words in the profile's tags.

## Quoting

You can require that two or more words occur in order by putting quotes in your query. For example:

| Query | Profile Name | Matches? |
|---|---|---|
| "Linux machine" | George's **Linux machine** | Yes |
| "machine Linux" | Linux machine | No |

## Anchoring

Normally, words in a query must match a substring of a word in the title or tags of a profile. You can require that a word in your query matches a prefix of a word in the title or tags by inserting a caret (^) before the word. You can require that a word in your query matches the suffix of a word in the title or tags by appending a dollar sign ($) after the word. For example, the query ^a* *matches only profiles with words starting with "a". The query* a$ *matches words ending in "a". The query* ^a$* matches only the word "a".

| Query | Profile Name | Matches? |
|---|---|---|
| ^root | root@example.com | Yes |
| ^root | Groot! | No |
| root$ | Groot | Yes |
| ^root$ | Groot | No |
| ^root$ | root | Yes |

## Combining Features

You may combine quoting, operators, and anchors. The operator always comes first, followed by a caret, followed by a quoted string, followed by a dollar sign. Consider the following examples:

```
name:^"George's Linux Machine"$
```

Three consecutive whole words in the profile's name must equal "George's Linux Machine".

```
name:"George's Linux Machine"$
```

Would match a profile named "XGeorge's Linux Machine", unlike the previous example.

```
name:^"George's Linux Machine"
```

Would match a profile named "George's Linux MachineX", unlike the first example.

```
name:"George's Linux Machine"
```

Would match a profile named "XGeorge's Linux MachineX", unlike the first example.

```
name:^George's
name:George's$
name:^George's$
```

A word having the prefix, suffix, or exactly matching "George's" must occur in the profile's name to match these queries, respectively.

---

# Automatic Profile Switching

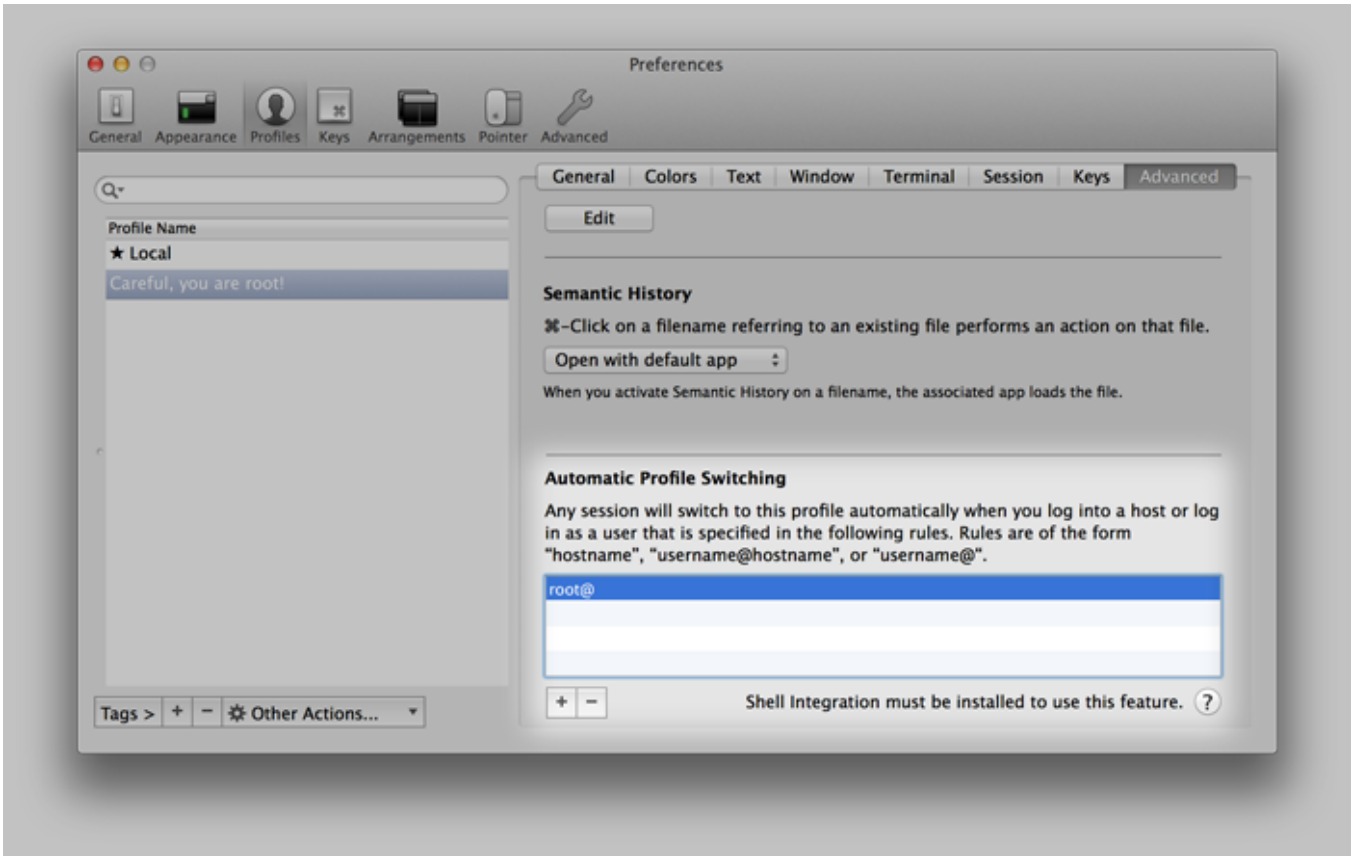
iTerm2 can use information it knows about your current path, host name, and user name to change profiles. For example, your window's background color or the terminal's character encoding could change when connecting to different hosts.

## Shell Integration Required

You must install Shell Integration (/shell_integration.html) on all machines and all user accounts where you plan to use Automatic Profile Switching (either by using the scripts or the Triggers workaround described in the Shell Integration docs).

### Rule Syntax

In *Preferences>Profiles>Advanced*, you may specify a set of rules.



When any session satisfies a rule in a given profile, it will switch to that profile. Rules consist of three optional components: the user name, the hostname, and the path. At least one component must be present, since an empty rule is not allowed. The hostname is required only when both a user name and a path are specified.

A user name is a unix accont name (e.g., *root*) followed by an `@`.

A path always begins with a `/`. Any time a hostname is followed by a path, they are separated by a `:`. For example, `iterm2.com:/users/george`. It may include `*` wildcards.

A hostname can be a DNS name, like `iterm2.com` or an IP address like `127.0.0.1`. A hostname may contain one or more `*` characters, which act as a wildcard (like globbing in Unix).

Some examples:

- george@iterm2.com:/users/george
- george@*:/users/george
- *:/users/george
- *.iterm2.com:/users/george
- dev.*.com:/users/george
- george@iterm2.com
- iterm2.com
- george@
- iterm2.com:/users/george
- /users/george
- /users/*

Because more than one rule may match at any given time, more complex rules will take priority over less complex rules. The priority order is defined like this:

- A username, hostname, and path. For example, "george@iterm2.com:/Users/george".
- A username and path, using "*" for the hostname. For example, "george@*:/Users/george".
- A combination of username and hostname. For example, "george@iterm2.com".
- A hostname and path. For example, "iterm2.com:/Users/george".
- A hostname. For example, "iterm2.com".
- A username alone. For example, "george@".
- A path alone. For example, "/Users/george".

The UI tries to prevent you from entering the same rule in two different profiles, but if that does happen then one profile is chosen arbitrarily.

### Automatic Reversion

After APS switches a session's profile, its rules may eventually cease to match (for example, the hostname changes back to "localhost" because an ssh session ends). If no profile has a matching rule, the session's original profile will be restored.

#### Implementation

Each session maintains a stack of profiles. Initially, the stack contains the profile the session was created with. When the username, hostname, or path changes, iTerm2 finds the best-matching profile. If some profile has a matching rule, one of two things happens:

- If that profile is already on the stack, profiles above that one will be removed from the stack and the session will switch to that profile.
- Failing that, the profile will be pushed on the stack and the session will switch to that profile.

If no profile has a matching rule, the stack is emptied (except for the first entry, the original profile for the session) and the session reverts to its original profile.

Rules may begin with ! to indicate "stickiness". A sticky rule causes its profile to stay even after the rule no longer applies, so long as no other rule matches.

### Triggers

Since it's impractical to install shell integration everywhere (for example, as *root*), there will be times when you need to write a trigger to detect the current username or hostname. Please see the *Triggers* section of Shell Integration (/shell_integration.html) for details.

### Troubleshooting

There are a few ways things can go wrong. Please see the Why doesn't secure copy/automatic profile switching work? (https://gitlab.com/gnachman/iterm2/wikis/scp-not-connecting) document for help diagnosing and fixing these issues.

---

## Coprocesses

iTerm2 offers support for "coprocesses". This very powerful feature will allow you to interact with your terminal session in a new way.

#### What is a Coprocess?

A coprocess is a job, such as a shell script, that has a special relationship with a particular iTerm2 session. All output in a terminal window (that is, what you see on the screen) is also input to the coprocess. All output from the coprocess acts like text that the user is typing at the keyboard.

One obvious use of this feature is to automate interaction. For instance, suppose you want to automate your presence in a chat room. The following script could be used as a coprocess:

```
#!/usr/bin/python
import sys
while True:
  line = raw_input()
  if line.strip() == "Are you there?":
    print "Yes"
  sys.stdout.flush()
```

You could disappear for years before your friends discover you're gone.

### How Do I Start a Coprocess?

There are two ways to start a coprocess.

1. Select "Run Coprocess..." from the Shell menu. Enter the name of a command to run as a coprocess.
2. Create a Trigger in Prefs>Profiles>Advanced and select Run Coprocess... as the action. Give the script to run as a parameter. Triggers also have Silent Coprocesses, which prevent any output from going to the screen. This is useful for ZModem, for example.

### Usage

A session can not have more than one coprocess at a time. When a coprocess is active, an icon will indicate that in the top right of the session.

### Technical Details

The coprocess's stdin is a byte-for-byte copy of the input from the session's pty, beginning at the time the coprocess starts. In the case of a trigger-started coprocess, the line of input that triggered it MAY be the first line of input to the coprocess, but this is not guaranteed. If a coprocess is running, triggers with a Run Coprocess action will not fire. The coprocess's stdout stream will be treated the same as keyboard input. A small amount of buffering is provided for both input and output of the coprocess. When a buffer fills, the coprocess will block.

---

## Session Restoration

Session restoration works by running your jobs within long-lived servers rather than as child processes of iTerm2. If iTerm2 crashes or upgrades, the servers keep going. When iTerm2 restarts, it searches for running servers and connects to them. The OS's window restoration feature preserves the content of your window, including scrollback history. iTerm2 marries the restored session to the appropriate server so you can pick up where you were.

tl;dr watch this: Demo Video (/misc/restoration-demo.mov)

### Notes

- You can toggle this feature with **Prefs>Advanced>Enable session restoration.**, but you *must restart iTerm2 after changing this setting*.
- If you don't have system window restoration enabled (both in System Settings and also as iTerm2's startup mode under Prefs>General) then history and screen contents will be lost.
- Force quitting iTerm2, causing it to crash, or upgrading it when prompted should restore your sessions. *NOTE: Quitting iTerm2 with Cmd-Q will terminate your jobs and they won't be restored.* There is an advanced preference to change this behavior, though.

- A session that has had only its window contents restored and not the running processes will get a reverse video *Session Restored* banner. A properly restored session will pick up right where you left it.
- If you reboot, your jobs will terminate and not be restored. The window contents should be restored.

---

# Utilities

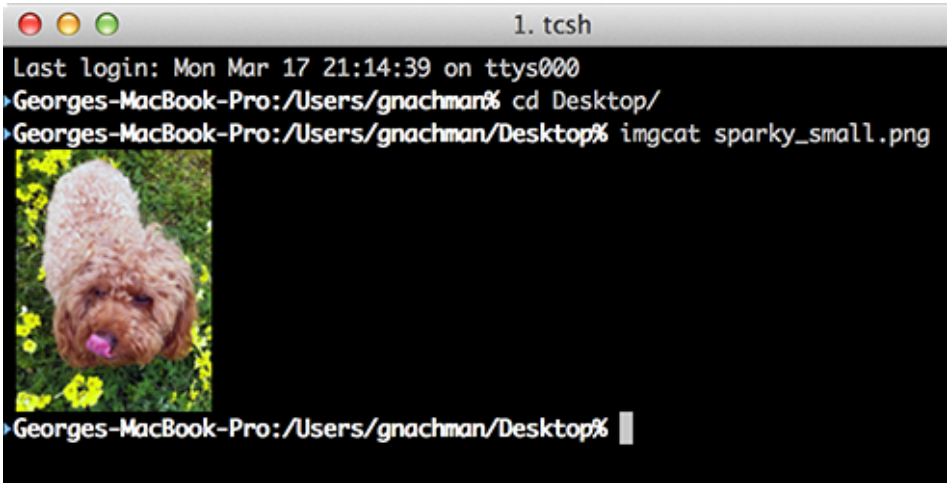
iTerm2 has a collection of shell scripts that help you take advantage of some of its unique features. When you install Shell Integration (shell_integration.html), you're asked if you'd like to install the Utilities Package as well. This page describes these utilities.

## Components

The Utilities Package contains the following programs:

### imgcat

The `imgcat` program displays images inline in your terminal.




It supports all standard image formats, including animated GIFs.

```
Usage:
imgcat filename [filename...]

or

cat image | imgcat
```

### it2dl

The `it2dl` program downloads files. This is useful when you are ssh'ed to a remote host. The downloaded files are placed in your *Downloads* folder.

```
Usage:
it2dl filename
```

### Location

The Utilities Package places shell scripts in `$HOME/.iterm2/` and creates aliases to them at the bottom of `$HOME/.iterm2_shell_integration.$SHELL`.

---

## Proprietary Escape Codes

iTerm2 supports several non-standard escape codes. These may not work properly in tmux or screen, and may have unknown effects on other terminal emulators. Proceed with caution.

A quick comment on notation: in this document, ^[ means "Escape" (hex code 0x1b) and ^G means "bel" (hex code 0x07).

The OSC command `50` used to be used but it conflicts with xterm, so it is now `1337`.

#### Set cursor shape

```
^[]1337;CursorShape=N^G
```

where N=0, 1, or 2.

- 0: Block
- 1: Vertical bar
- 2: Underline

Add this to your .vimrc to change cursor shape in insert mode:

```
let &t_SI = "\<Esc>]1337;CursorShape=1\x7"
let &t_EI = "\<Esc>]1337;CursorShape=0\x7"
```

This is derived from Konsole (https://vim.wikia.com/wiki/Change_cursor_shape_in_different_modes).

#### Set Mark

The "Set Mark" (cmd-shift-M) command allows you to record a location and then jump back to it later (with cmd-shift-J). The following escape code has the same effect as that command:

```
^[]1337;SetMark^G
```

#### Steal Focus

To bring iTerm2 to the foreground:

```
^[]1337;StealFocus^G
```

### Clear Scrollback History

To erase the scrollback history:

```
^[]1337;ClearScrollback^G
```

### Set curent directory

To inform iTerm2 of the current directory to help semantic history:

```
^[]1337;CurrentDir=/the/current/directory^G
```

### Post a Growl notification

To post a Growl notification:

```
^[]9;Message content goes here^G
```

This will have no effect if Growl is not running.

### Change profile

To change the session's profile on the fly:

```
^[]1337;SetProfile=NewProfileName^G
```

### Copy to clipboard

To place text in the pasteboard:

```
^[]1337;CopyToClipboard=name^G
```

Where name is one of "rule", "find", "font", or empty to mean the general pasteboard (which is what you normally want). After this is sent, all text received is placed in the pasteboard until this code comes in:

```
^[]1337;EndCopy^G
```

### Set window title and tab chrome background color

To set the window title and tab color use this escape sequence:

```
^[]6;1;bg;red;brightness;N^G
^[]6;1;bg;green;brightness;N^G
^[]6;1;bg;blue;brightness;N^G
```

Replace N with a decimal value in 0 to 255.

Example in bash that turns the background purple:

```
echo -e "\033]6;1;bg;red;brightness;255\a"
echo -e "\033]6;1;bg;green;brightness;0\a"
echo -e "\033]6;1;bg;blue;brightness;255\a"
```

To reset the window title and tab color, use this code:

```
^[]6;1;bg;*;default^G
```

For example:

```
echo -e "\033]6;1;bg;*;default\a"
```

### Change the color palette

```
^[]Pnrrggbb^[\
```

Replace "n" with:

- 0-f (hex) = ansi color
- g = foreground
- h = background
- i = bold color
- j = selection color
- k = selected text color
- l = cursor
- m = cursor text

rr, gg, bb are 2-digit hex value (for example, "ff"). Example in bash that changes the foreground color blue:

```
echo -e "\033]Pg4040ff\033\\"
```

### Annotations

To add an annotation use on of these sequences:

```
^[]1337;AddAnnotation=message^G
^[]1337;AddAnnotation=length|message^G
^[]1337;AddAnnotation=message|length|x-coord|y-coord^G
^[]1337;AddHiddenAnnotation=message^G
^[]1337;AddHiddenAnnotation=length|message^G
^[]1337;AddHiddenAnnotation=message|length|x-coord|y-coord^G
```

- message: The message to attach to the annotation.

- length: The number of cells to annotate. Defaults to the rest of the line beginning at the start of the annotation.
- x-coord and y-coord: The starting coordinate for the annotation. Defaults to the cursor's coordinate.

*AddHiddenAnnotation* does not reveal the annotation window at the time the escape sequence is received, while *AddAnnotation* opens it immediately.

#### Cursor Guide

```
^[]1337;HighlightCursorLine=boolean^G
```

The *boolean* should be *yes* or *no*. This shows or hides the cursor guide.

#### Attention

```
^[]1337;RequestAttention=boolean^G
```

The boolean should be *yes* to request attention by bouncing the dock icon and *no* to cancel a previous request.

#### Background Image

```
^[]1337;SetBackgroundImageFile=base64^G
```

The value of *base64* is a base64-encoded filename to display as a background image. If it is an empty string then the background image iwll be removed. User confirmation is required as a security measure.

#### Report Cell Size

```
^[]1337;ReportCellSize^G
```

The terminal responds with:

```
^[]1337;ReportCellSize=height;width^G
```

Where *height* and *width* are floating point values giving the size in points of a single character cell. For example:

```
^[]1337;ReportCellSize=17.50;8.00^[\
```

#### Badge

The badge has custom escape sequences described at . (badges.html)

**Downloads**

(badges.html)
For information on file downloads and inline images, see (badges.html). (images.html)

# Shell Integration/FinalTerm

(images.html)
iTerm2's (images.html)Shell Integration (shell_integration.html) feature is made possible by proprietary escape sequences pioneered by the FinalTerm emulator. FinalTerm is defunct, but the escape sequences are documented here.

## Definitions

- `OSC` stands for *Operating System Command*. In practice it refers to this sequence of two ASCII characters: 27, 93 (`esc ]`).
- `ST` stands for *String Terminator*. It terminates an OSC sequence and consists either of two ASCII characters 27, 92 (`esc \`) or ASCII 7 (`bel`).

OSC sequences always begin with `OSC`, are followed by a sequence of characters, and are terminated with `ST`.

Most `OSC` codes begin with a number (one or more decimal digits), which we'll call the "command" in this document. If the command takes parameters it will be followed by a semicolon and the structure of the rest of the body of the `OSC` sequence is dependent on the command. Well-behaved terminal emulators ignore `OSC` codes with unrecognized commands.

## Concepts

The goal of the FinalTerm escape sequences is to mark up a shell's output with semantic information about where the prompt begins, where the user-entered command begins, and where the command's output begins and ends.

**[PROMPT]**prompt% **[COMMAND_START]** ls -l
**[COMMAND_EXECUTED]**
-rw-r--r-- 1 user group 127 May 1 2016 filename
**[COMMAND_FINISHED]**

## Escape Sequences

FinalTerm originally defined various escape sequences in its original spec that are not supported by iTerm2 and are not described in this document. The best remaining references to these codes are in iTerm2's source code.

### FTCS_PROMPT

```
OSC 1 3 3 ; A ST
```

Sent just before start of shell prompt.

### FTCS_COMMAND_START

`OSC 1 3 3 ; B ST`

Sent just after end of shell prompt, before the user-entered command.

### FTCS_COMMAND_EXECUTED

`OSC 1 3 3 ; C ST`

Sent just before start of command output. All text between `FTCS_COMMAND_START` and `FTCS_COMMAND_EXECUTED` at the time `FTCS_COMMAND_EXECUTED` is received excluding terminal whitespace is considered the command the user entered. It is expected that user-entered commands will be edited interactively, so the screen contents are captured without regard to how they came to contain their state. If the cursor's location is before (above, or if on the same line, left of) its location when `FTCS_COMMAND_START` was received, then the command will be treated as the empty string.

### FTCS_COMMAND_FINISHED

`OSC 1 3 3 ; D ; Ps ST`

`OSC 1 3 3 ; D ST` (for cancellation only)

The interpretation of this command depends on which `FTCS` was most recently received prior to `FTCS_COMMAND_FINISHED`.

This command may be sent after `FTCS_COMMAND_START` to indicate that a command was aborted. All state associated with the preceding prompt and the command until its receipt will be deleted. Either form is accepted for an abort. If the Ps argument is provided to an abort it will be ignored.

If this command is sent after `FTCS_COMMAND_EXECUTED`, then it indicates the end of command prompt. Ps is the command's exit status, a number in the range 0-255 represented as one or more ASCII decimal digits. A status of 0 is considered "success" and nonzero indicates "failure." The terminal may choose to indicate this visually.

If neither `FTCS_COMMAND_START` nor `FTCS_COMMAND_EXECUTED` was sent prior to `FTCS_COMMAND_FINISHED` it should be ignored.

## iTerm2 Extensions

iTerm2 extends FinalTerm's suite of escape sequences.

### SetUserVar

`OSC 1 3 3 7 ; S e t U s e r V a r = Ps1 = Ps2 ST`

Sets the value of a user-defined variable. iTerm2 keeps a dictionary of key-value pairs which may be used within iTerm2 as string substitutions, such as in the Badge (/badges.html).

Ps1 is the key.

Ps2 is the base64-encoded value.

### ShellIntegrationVersion

`OSC 1 3 3 7 ; S h e l l I n t e g r a t i o n V e r s i o n = Pn ; Ps ST`

`OSC 1 3 3 7 ; S h e l l I n t e g r a t i o n V e r s i o n = Pn ST` (deprecated)

Reports the current version of the shell integration script.

Pn is the version.

Ps is the name of the shell (e.g., `bash`).

iTerm2 has a baked-in notion of the "current" version and if it sees a lower number the user will be prompted to upgrade. The version number is specific to the shell.

### RemoteHost

`O S C 1 3 3 7 ; R e m o t e H o s t = Ps1 @ Ps2 ST`

Reports the user name and hostname.

Ps1 is username. Ps2 is fully-qualified hostname.

### CurrentDir

`O S C 1 3 3 7 ; C u r r e n t D i r = Ps1 ST`

Reports the current directory.

Ps1 is the current directory.

### Table of Contents


- Highlights for New Users (documentation-highlights.html)
- General Usage (documentation-general-usage.html)
- Menu Items (documentation-menu-items.html)
- Preferences (documentation-preferences.html)
- Scripting (documentation-scripting.html)
- Shell Integration (documentation-shell-integration.html)
- Smart Selection (documentation-smart-selection.html)
- Triggers (documentation-triggers.html)
- Captured Output (documentation-captured-output.html)
- Fonts (documentation-fonts.html)
- Inline Images (documentation-images.html)
- Badges (documentation-badges.html)
- Dynamic Profiles (documentation-dynamic-profiles.html)
- Profile Search Syntax (documentation-search-syntax.html)
- Automatic Profile Switching (documentation-automatic-profile-switching.html)
- Coprocesses (documentation-coprocesses.html)

- Session Restoration (documentation-restoration.html)
- Utilities (documentation-utilities.html)
- Proprietary Escape Codes (documentation-escape-codes.html)

iTerm2 by George Nachman. Website by Matthew Freeman, George Nachman, and James A. Rosen.

Website updated and optimized by HexBrain (http://hexbrain.com)