City of Heroes / City of Villains Technical Reference Guide

V3.20a – December 2021 (Homecoming Issue 27-3)

Compiled by
The Legendary Death-Defying
Shenanigunner

@Shenanigunner or **Gunner@Shenanigunner.com**

Now always found at the ***Heroica!*** website — **www.Shenanigunner.com**!

Version Notes

» **Completely updated and reformatted (in a more powerful editor) for Christmas 2020.** Sweeping update of most sections, including slash commands, emotes and window names, info and changes through Issue ~~27-2~~ 27-3.

» This is a major update to a guide I wrote over three or four years, back in the Live era. Originally the "Bind and Macro Guide," I've renamed it since it now covers so much ground regarding the technical aspects of the game.

» There are many, many, many references out on the web, as docs, forum posts, wiki entries and the like. I don't know of any that are as comprehensive, organized and thoroughly checked out as this one… but feel free to let me know of any competitors so I can challenge them to an Arena fight.

» I took the information in here almost entirely from the game itself, long before most guides and lists were widely available. The various "command dump" commands were very useful in finding and listing things. When I found new information in other lists, I followed up and verified it myself. Very few of these lists had author or creator names attached, or I would have gladly credited them. As it is, most of the game information is drawn from the same resources I used; I don't think there's one word in here that is someone else's unique creation.

» I simply never thought this material would be useful again, or that I'd spend what has become so much time happily updating and checking it as the Homecoming team has advanced the game. It's been a pleasure!

» This has been something of a lonely effort, since I never spent much time in the various online communities. It would mean a lot if every user who finds this guide helpful could drop me a note, send along a contribution or correction or game story, and most of all pass the word… repost and cross-list this Guide all to hell out there in the new Cityverse.

» Or you can throw some Inf at me, `@Shenanigunner`, on Excelsior. I wouldn't mind.

» While I retain rights to the specific effort used to create this guide, all of its contents are released (or just left in) the public domain and may be copied anywhere by anyone… but credit and a link to the whole thing would be really appreciated.

» **See the website for the new GABB updated bindfile set** …and a whole lot more!

HOW-TO

Version Notes — 1

0. INTRODUCTION — 4
- 0.1 WE'RE BACK! — 4
- 0.2 What are Keybinds & Macros? — 4
- 0.3 Why Do I Need a Technical Guide? — 5
- 0.4 Why This Guide? — 5
- 0.5 How This Guide is Organized — 6
- 0.6 Updates — 6
- 0.7 Contact — 6
- 0.8 Acknowledgements — 6
- 0.9 City of Heroes vs. City of Villains — 7
- 0.10 Things To Come—Future Plans — 7

1. BASICS — 8
- 1.1 Overview — 8
- 1.2 Terminology — 8
- 1.3 Entering Keybinds & Macros — 8
- 1.4 Starting Fresh — 9
- 1.5 Basic Syntax — 10
- 1.6 Variables — 10
- 1.7 Useful References — 11
- 1.8 Default Keybinds — 11
- 1.9 File Locations — 11
- 1.10 Editing Keybind Files — 12
- 1.11 Extending Keybind Files — 13
- 1.12 The GABB — 13

2. KEYBINDS — 15
- 2.1 Keybind Overview — 15
- 2.2 Key Names — 15
- 2.3 Shift Keys — 15
- 2.4 Basic Command Usage & Command Modifiers — 16
- 2.5 Command Separators — 17
- 2.6 Multiple Command Limitations — 18
- 2.7 Toggles and Forced Toggles — 19
- 2.8 Rollover Binds — 20
- 2.9 Pseudo-Autofire Binds — 21
- 2.10 Press-Release Binds — 23

3. MACROS — 26
- 3.1 Macro Overview — 26
- 3.2 Directed Macro Creation — 26
- 3.3 Macros Using Tray Rollover — 27
- 3.4 Macros with Custom Icons — 28
- 3.5 Macros vs. Keybinds — 28

4. Saving/Loading Interface Settings — 30
- 4.1 Common Settings vs. Individual — 30
- 4.2 Chat Configuration Save and Load — 31
- 4.3 Window Configuration Save and Load — 31
- 4.4 UI Scaling vs. Window Scale — 32
- 4.5 Option Configuration Save and Load — 33
- 4.6 Setting Options — 33
- 4.7 Option Keywords — 34
- 4.8 BuffSettings Argument List — 36
- 4.9 Character Display Options & Argument List — 37

5. Power, Tray & Inspiration Control — 38
- 5.1 Basic Powers Terminology — 38
- 5.2 Tray Management — 38
- 5.3 Tray Swapping & Rollover — 39
- 5.4 Power Execution — 40
- 5.5 Using the `/powexec_location` Command — 41
- 5.6 Inspiration Tray Management — 42
- 5.7 Pet Inspirations — 43

6. Pet Control — 44
- 6.1 Naming Pets — 44
- 6.2 Spawning & Buffing Pets — 45
- 6.3 Controlling Pets — 45
- 6.4 Ve Haff Vays to Make You Talk, Lassie — 47
- 6.5 Disposing of Pets — 48

7. Popmenus — 49
- 7.1 Popmenu Fundamentals — 49
- 7.2 Basic Popmenu Structure — 50
- 7.3 Popmenu Command Strings — 51
- 7.4 Useful Popmenu Examples — 52
- 7.5 Advanced Popmenu Features — 53

8. Gunner's Targeting Secrets 55

8.1 Basic Targeting Commands 55
8.2 Custom Targeting Commands 56
8.3 Advanced Targeting 59

9. Gunner's Way-Cool Binds List 60

9.1 General Binds 60
9.2 Combat Binds 64
9.3 User Interface Binds 66
9.4 Healer/Buffer Binds 67

REFERENCE A: Slash Commands 69

A.1 Slash Command Listing 69
A.2 Base Editing Commands 100
A.3 Mission Architect (AE) Commands 104
A.4 Group List of Slash Commands 105

REFERENCE B: Key & Mouse Button Names 111

B.1 Bindable Keyboard Key Names 111
B.2 Bindable Mouse Button/Action Names 114
B.3 Joystick/Controller Button Names 115
B.4 Virtual Mouse Mode 116

REFERENCE C: Window & Menu Names 117

C.1 Window Names 117
C.2 Using the Combat Monitor Window & **`/monitorattribute`** Command 120

REFERENCE D: Emote Codes 123

D.1 Emote Basics 123
D.2 Emote Keywords 124
D.3 Advanced Sit Emotes 137
D.4 Costume Changes & Emotes 138
D.5 Hybrid Emotes 140

REFERENCE E: Chat Bubble Color Codes 141

E.1 Option Settings for Chat Bubbles 142

Revision History 143

0. INTRODUCTION

0.1 WE'RE BACK!

Almost ten years later, our Cities and our community are back. Welcome to this never-expected continuing update of what I hope the new community will find useful.

The guide has been renamed from its Live-era version because it's become a more complete technical reference guide to the game's background features and options. And now (with v3.00) it's been updated to a completely new platform and organization.

0.2 What are Keybinds & Macros?

Keybinds and macros are ways to remap the keys, mouse buttons and game commands into a control configuration that better suits a given player's play style and preferences. Instead of being locked into a fixed set of control keys and commands, as some older games do, or providing a simple reassignment feature as most newer games do, sophisticated games like City of Heroes/City of Villians permit you to remap, change and combine the game commands and controls in an almost unlimited fashion.

For example, instead of simply letting you change your "run" command from the default R key to another, keybinds allow you to bundle two or more commands onto one key, so that you initiate running and go into Super Speed at the same time. Another example is the very useful "engage" keybind, which targets the nearest foe and locks you onto him in "follow" mode. For a melee player (scrapper or tanker) in the middle of a multi-foe fight, being able to whack one key and lock onto a foe for focused attacks can change your play style and success rate.

Or, on the fun and silly side, you can combine chat strings with actions—a local string "C'mere, you ugly SOB!" with a taunt, or "Roast in hell!" combined with a AoE (multi-foe area of effect, that is) Burn or Scorch power. Or "Let's get 'em!" combined with a suitable emote (character animation), to tell your teammates it's time to get down to business. (There are several of those predefined in the QuickChat emote menu.)

A macro is exactly the same as a keybind, except that the string of commands is bound to a power-tray button and has to be activated by a click or an associated power-activation command (by default, the associated keyboard-top number or alt-number). Generally, you should use keybinds for commands and command sequences you need to activate quickly and often, while macros can be used for actions for which you'll have time to find and click a power button.

Some players might be happy with a few reassignments from the default command keys. Others might want to do the crazy thing and create a completely custom mapping of everything. However, all players can benefit from a few keybinds that make key powers speedier and easier to use in the heat of battle.

0.3 Why Do I Need a Technical Guide?

Creating effective keybinds (or binds, for short) and macros takes some knowledge and experience, which the basic game manuals don't really cover. At a minimum, you need to know the basic syntax for writing a bind or macro, and have a list of all the console or "slash" commands (so called because they begin with a slash that identifies them as commands when you type them into the chat window).

Since the developers of CoH don't provide a comprehensive guide, and are a little loose about providing consistent information about the list of available slash commands with each update, it's fallen to the player community to keep track of the commands and teach each other how to use them. A guide of some sort is essential to help you master this complicated and flexible set of commands.

0.4 Why This Guide?

Yep, City of Heroes is gifted by an almost endless array of references, how-tos and compilations of info. Any time documentation is written by a community of users, though, you're going to see some common limitations. Those who are real hotdogs with the tools may not be very good at writing about them; those who can write well may not know enough about the process to get all the details right; even those who can do both might not have time to gather all the details or keep things updated.

I set out to write this guide because, as a player new to CoH (as was everyone at one time) and a player new to multiplayer online games, I couldn't find a good, complete, up-to-date guide that was written in language a non-MMOG maven could understand. The guides that were any good in the information department seemed to be written in poorly-translated Martian, assuming far too much previous knowledge on the part of the reader. And the guides that were incomplete, sloppily compiled and out of date were even more frustrating for a newcomer, because it was hard to identify what was right and what was useless.

Many players prefer online info, but the (presently two) game wikis are not frequently updated and old information is even less frequently removed. Maybe it's just old-guy, old-school thinking, but I'd rather have a single comprehensive source that combines info and proven how-to than a massive wiki (or, gawferbid, pinned forum posts) that have to be found and absorbed piecemeal and are often outdated, cryptic or just plain wrong.

So here's this guide—which is fully searchable in online and PDF form—added to the pile, and I hope an improvement and of value to both new and existing players. My aim was to combine many years of experience writing software and programming manuals (most for novice and nonexpert users) with my fascination of City of Heroes and all the reliable information I could lay hands on. It's my aim to keep it updated, both with new info gleaned from the forums and other users, and from reader feedback.

0.5 How This Guide is Organized

The organization of this guide is simple. Section 0 is the Introduction, which you're reading. Sections 1 through 7 are comprehensive tutorial and how-to. References A through E are just that, concise references to each complex game element like slash commands and emotes.

☞ Use the numbered sections to learn how (and advanced how)

☞ Use the References for, well, reference.

☞ Read TIPS like these, too. I haven't put them in just because I like the little hand icon.

0.6 Updates

For many reasons, this will probably be the final update of this guide. The last update was around Issue 8 in late 2006 and not very much grew out of date; even with massive changes to the game the command and emote set hasn't changed as much as in prior Issues. I am also nearing my end of involvement with the game, at least to the point where I am inspired to maintain this guide.

> **(Wasn't that cute? I almost deleted it but good humor is timeless.) And then came the rebirth of the game in 2019… and now the huge Issue 27 revision… so saddle up, buckaroos, we're off on another ride!**

Updates and corrections, especially to the command and emote lists, are encouraged. Comments on everything are welcomed. And pestering when I let the guide fall out of currency is solicited—I have a tendency to move on and not maintain efforts like this, especially when there's no feedback. Participating in my infrequent threads on Homecoming Forum helps, too.

0.7 Contact

Send email to **`Gunner@Shenanigunner.com`**, or to whatever maintenance email address is listed on the web site at **`www.shenanigunner.com`**, is the best way. You can also send the quirky in-game mail to **`@Shenanigunner`**, or tap me whenever I'm online. Use **`/getlocalname shenanigunner`** to find my current alt, if need be.

0.8 Acknowledgements

Only the general presentation of this guide, along with much direct verification of the commands, is solely mine. All of the information came from other sources—mainly, the game itself, its user manual, and the now ancient and hilarious Prima game guide.

The original list of slash commands was provided by Xocyll, in the Usenet forum **`alt.games.coh`**, back when it existed, copied from the Binds forum on the official CoH web site. Xocyll also posted a number of discoveries of his own on the Usenet group, which are included here, and provided continual feedback on the guide. Neil Cerruti provided some useful info and feedback as well for the I6/I7 updates.

A lot of the basics came from other guides and helpful people in the forums. As nearly all of it traces back to information from the game developers, and because I didn't keep track of who told me what, all I can do is offer a general, generous and heartfelt thanks to everyone who helped increase my understanding of how this is done. I make no claim at all that I could have done it without all that help. (And this has continued with the very involved GMs and developers of Homecoming!)

Thanks to Korbian on Titan Network for some very useful pointers to command and emote information.

The material in this guide is expressly placed in the public domain, but with the firm expectation that any copying or usage will be credited. (Thanks.) The net result, format, concept etc., however, are copyright and should not be reused without permission.

0.9 City of Heroes vs. City of Villains

As far as I know, both games are identical in their use of binds and macros, and almost all commands are interchangeable between the games. There are a very few commands that are peculiar to one or the other (mostly, to CoV alone). The Homecoming updates have mostly eliminated pointless duplication (such as the old difference that you could "lackey" someone in CoV but "sidekick" them in CoH. No really significant differences remain except in a few emotes and badge-linked rewards.

Ditto, in general, for Praetoria/Gold Side.

0.10 Things To Come—Future Plans

This guide will likely never be "complete" since there are always hidden or unknown commands or tweaks, and changes with every Issue and running fix by the Developers. Some of the things in the ongoing agenda include:

» Thorough testing of commands I was not able to test—mostly, those to do with groups, leagues, AE, crafting/inventions etc. Anyone who has experience in the areas I don't is invited to test and correct the commands listed here, and pass along nifty things they find.

» Adding more "cool bind" info to Section 8.

» Adding more detail on groups of slash commands and how to put them to good use. Some of this is found in this new (v3.00) update.

» For more info and the newest cool things, keep an eye on the ***Heroica!*** website at **`www.shenanigunner.com`**, which echoes a number of things in here, and vice versa, while providing files and resources that don't fit in guide.

1. BASICS

1.1 Overview

I'm going to put all the special terminology in this section—so if you run into an unfamiliar or cryptic term in the later sections, it's either because you didn't read this one or because I slipped and forgot to include it. (Let me know, in that case.) I'm also going to put most of the general, basic information that applies to both keybinds and macros in here, with a few repeats of key items in other sections.

☞ Read this section even if you think you're ahead of the curve—it will help you get going much faster and with fewer problems than if you just jump to the how-to sections!

1.2 Terminology

- » **Keybind—**A string of game commands "bound" to a single key or mouse button, which will be executed when that key or button is pressed. Also called just a "bind."
- » **Macro—**A string of game commands assigned to a Powers tray button, which will be executed when that Power button is clicked or activated via a keypress.
- » **Syntax**—The precise rules by which a keybind or macro string is constructed. If a string is constructed wrong—has faulty syntax, that is—it probably won't work, or at least won't do what you want it to do.
- » **Toggle**—To turn a power on or off, whichever state it isn't in, with a single command. Most shields and buffs are toggle powers, which you activate with a click of the button and then deactivate with a click of the same button. There are ways to force toggles to the on and off states, no matter what state they are in to begin with.
- » **Window—**Any of the individual dialogs, menus, and separate windows that are part of the user interface.
- » //—any time you see a red double slash in this guide, it means "line break" (in a long bind or macro string) and should **not** be entered. Copy or retype the string without any break. Bind/macro strings should **never** be broken across editor lines or hyphenated.
- » **Curly quotes**—Always be cautious when cutting and pasting command information from any on-screen source to the chat window. Many editors will convert single and double quotes to "curly" typographical quotes... which may be invisible in the editing but will choke the command parser. Always use/convert to "straight quotes"! I try to keep curlies out of the examples here but shift happens.

1.3 Entering Keybinds & Macros

Keybinds and macros are entered from within the game, by typing strings into the chat window's entry line. The current chat channel selected does not matter; you're going to override the chat function and direct the command to the "console" (the game's command input window) by typing a foreslash (/) as the first character. You're not going to forget that slash. Trust me, after the first time you send a bind string out to the entire server because you forgot the slash, you're not going to forget the slash.

There are some good rules for entering binds and macros. The first is to park your character in a safe place, so you won't have to deal with unexpected foes while you're tinkering. In a supergroup base is probably best. Inside a tram station or any plaza guarded by police drones is a good place. Face your character to the wall, a universal multiplayer game announcement that you're busy with some internal task and don't want to be interrupted. If you're going to be at it a while, you might type the command **`/hide`** into the console to start. This will make you invisible to everyone else in the chat and search windows, so they won't bother you. (Remember to **`/unhide`** when you're done!) Finally, select a safe chat channel, so that if you do screw up, no one will be privy to your bobble. Using the Team or Supergroup channel is good—if you accidentally send chat message, either nothing will happen or you'll just get a warning that you're not on a team.

You can also enter keybinds by editing a text file and then loading it, but that's an advanced step we'll cover separately. For now, the easiest way to start entering binds and macros is directly, in the game.

1.4 Starting Fresh

I strongly recommend that you start with a clean, new set of default keybinds. Clean out any mess you might have accumulated by going to **Menu | Options | Keymapping** and clicking the **Reset Keybinds** button. (Warning: instant effect, no confirmation!) You can also type **`/keybind_reset`** at the Chat window. Then slowly enter your new binds and test them.

The best start you can make, though, is to replace the creaky default binds with the updated, modern, consistent set in the GABB bindfile... see section 1.9 below.

If you're put in some time adding and changing binds already, you might want to save your current binds with the following command:

```
/bindsavefile MyOldBinds.txt
```

You should also save your keybinds to a local text file every time you are about to make changes, so that you can quickly reload a working set if you mess up something. Use this command until you get to the more advanced techniques:

```
/bindsavefile MyNewBinds.txt
```

and restore things to your last save point by entering:

```
/bindloadfile MyNewBinds.txt
```

1.5 Basic Syntax

The basic syntax for a keybind, which is typed into the chat window's message-entry window, is:

```
/bind key command_string
```

This will "bind" the specified command string to the specified key. You can bind commands to almost all of the keys on the keyboard, with some limitations. Once a valid command string is successfully bound to a key, any prior assignment to that key is erased and pressing that key will execute the command string.

The basic syntax for a macro, typed in exactly the same way, is:

```
/macro macro_name command_string
```

This will "bind" the specified command string to a gray power-tray button with the identifying name specified. (Macro names can be one to three letters or numbers, and some punctuation. (There is no limit to the length of a macro name, but only three characters will fit on a macro button and the whole string will appear as a "tooltip" on hover.) Macros can, confusingly, be given identical names, which is not recommended. Once a valid command string is successfully bound to a macro icon, clicking that icon will execute the command string.

The slash at the beginning of those commands is very important: if you don't include it, you'll simply send the string out to whatever chat channel you have selected, provoking much humor and wrath from whoever sees it. (Sending a bind string out into a zone- or server-wide channel is one of the top not-quite-a-newbie tricks. You are allowed to avoid it. See the suggested rules in 1.3.)

1.6 Variables

Binds and macros are a lot more useful if you can insert variables, such as player or foe names, your own name, level and archetype, etc. City of Heroes includes such variables, which may be inserted into any command string in place of fixed text. It is the dollar sign ($) first character that identifies the label as a variable, which is why you can't use a dollar sign in most macro and bind text strings.

CoX (short for City of Heroes/City of Villains... saves a lot of space) has six variables:

- **`$archetype`**
  Your player's archetype—Blaster, Tanker, etc.
- **`$battlecry`**
  The string you've entered in your ID as your battle cry. Limited to 32 characters.
- **`$level`**
  Your player's level—2, 10, 35, etc.
- **`$name`**
  Your player's name—Shenanigunner, Wolf Moon, etc.
- **`$origin`**
  Your player's origin—Natural, Magic, Science, etc.
- **`$loc`**
  Your character's current location in X-Z-Y coordinates.

» **`$target`**
The name of your currently selected target, which can be a foe, another player, or an object.

☞ It’s been suggested that **`$battlecry`** could be used as a universal variable string, since (unlike the others) it can be set by the user. It has no effect on any aspect of gameplay otherwise.

1.7 Useful References

There are several useful references for creating binds and macros. The best one around, if I may say so, is this guide and especially Reference A, which lists all the currently known slash commands. The other Reference sections contain other, more sophisticated control elements.

More current lists, and many tips and tricks, can be found on CoH-related web sites and in the official CoH forum devoted to binds. Look these resources up for help, ideas, and information I haven’t included here.

1.8 Default Keybinds

Perhaps the most useful reference you can have is a copy of the complete default keybinds, which I haven’t included here because it’s bulky. You can get your own copy right from the game:

» Save your existing keybinds as above, if you want to keep them around.

» Type **`/keybind_reset`**

☞ or **`unbindall`**, if you like that synonym better… there are lots of slash commands with multiple forms.

» Type **`/bindsavefile COXdefaultbinds.txt`**.

Boom. That file contains the complete default bind set. You can substitute any path and filename you like (but see the next section about that). Open the file with a text editor (preferably **not** a word processors like Word!) and you’ll find a complete list of the default binds and command strings.

The slightly more sophisticated way to get the default binds, by itself or as an added step, is to go download the GABB bindfile set and companion manual from the ***Heroica!*** website. The guide has complete table of the default binds, along with notes about why many of them stink and deserve to be replaced. See Section 1.9 below for more info.

It will be assumed that you have this file, in one form or another, as a companion to this guide.

1.9 File Locations

It’s a point of considerable importance that you keep your bindfiles in a convenient location, with convenience defined by in-game needs, not necessarily computer-user needs. For many years, I kept all my (editable) game files in **`c:\bindfiles`**, which made sense in the second way but not in the first... because it meant every reference to file loading, rollover bind strings, etc. had to include that full path (such as **`c:\bindfiles\MyBinds.txt`**).

I now strongly recommend that you use the game's default folder for bindfiles.

This will vary depending on which game launcher you are using.

- If you are using the "Tequila" launcher, which follows the original game model, the default folder for bind files will will be **`\data`** under the City of Heroes game installation folder. (If the game is in **`c:\games\CoH`**, the default bindfile location will be **`c:\games\CoH\data`**.)
  - With this older launcher, generic save files from **`/chatsave`**, **`/optionsave`** etc. will be stored in the game's root folder.
- If you are using the "Homecoming" launcher, which tidies up and secures the game file set better, bind files will be in **`\settings\live`** under the CoH installation folder.
  - With this launcher, generic game save files will be stored in this same **live** folder.

If you keep your files in this default path, you never need to use a path to load or save files within the game, and rollover bind files can omit that path info as well. In other words, you can use

```
/bindloadfile NewBinds.txt
```

instead of

```
/bindloadfile c:\bindfiles\NewBinds.txt
```

☞ Keep all your bindfiles in [GAME PATH]\data or [GAME PATH]\settings\live, and create a short-cut to point to this folder for desktop editing convenience.

☞ Use file names without any drive or path information in commands and binds.

1.10 Editing Keybind Files

Once you start messing with binds, you'll probably want to move on to making wholesale edits rather than laboriously typing in strings in the game. It's pretty simple; you can even do it while you're in the game, subject to some cautionary notes.

First, save your current keybinds as just described above (or better yet, in the more sophisticated per-alt mode described a little later).

Now switch to the Windows desktop and open this file in your favorite plain-text, ASCII editor. (If Notepad isn't good enough, I highly recommend the freeware enhancement Notepad++.)

Edit away, even while your alt enjoys the scenery. Don't forget to save.

When you're ready to try the commands, switch back to the game and load the update file.

Test away.

☞ Park your alt in a safe place while you're doing this. It's bad to come back dead.

☞ Be sure to use a plain-text editor, not a word processor, even if it has a save-to-text mode. You will screw up the save and create chaos for your keybinds. I highly recommend the freeware **Notepad++**; it preserves your open files when closed and reopened.

☞ **BE SURE TO USE STRAIGHT QUOTES!** If you cut and paste keybind or macro strings from this reference or any other source, make sure that all quotes (" and ') are STRAIGHT quotes in ASCII form, and not "curly quotes" as used by most word processors.

Note: it appears that **bindfiles are limited to 242 binds or lines**. For the most part, this will only affect alts that make extensive use of the numpad keys, such as healers or upper-level masterminds. If you load a bindfile and the System chat window shows an error, you'll have to clean out your binds and/or shorten the bind file.

1.11 Extending Keybind Files

One neat thing about the way the keybinds work is that you can selectively overwrite them in the game. That is, if you enter a new bind in the console, it is added to the set, or overwrites only that specific bind. To go further, you can load a keybind file with only selected entries, and those will become part of the total set, and only overwrite any specific existing binds.

This is useful when you want to, for example, load in a bunch of emote binds tailored to a specific alt. If you maintain a bindfile with nothing bound to the NumPad keys, you can write a file (for, say, pet or healer binds) and load it to overwrite only those keys.

There are a few guidelines to do this effectively:

- » If you're adding a keybind set to an existing bind set, check to make sure it won't overwrite any existing binds you want.
- » The new binds will not become part of your standard keybind file for that alt automatically. If, for example, you want to add a Mastermind pet control bind set to an alt, you have to load the specialized binds, and then
  - » SAVE the updated bind set to the alt's bind-save file; and then
  - » SAVE your prior bind-load file for archival purposes (optional) or delete it, then copy the newly saved version to the load-file name.

That will provide a reverse path should you want to undo the addition, or use a load file for a new alt without that specific of emote, combat, heal, pet or whatever binds included.

1.12 The GABB

This guide is only half of my efforts on behalf of the CoX community. The other half is a wholly optimized basic keybind set that is updated and streamlined from the (ca. 2005) original.

Some experienced gamers don't like the changes in this new set. But if you're a newcomer, either to games or CoX, or more flexible/adaptable, I strongly recommend that you start your experience or update it with the **GABB** bindfile... that's **Gunner's Advanced Basic Bindfile**. It implements many of the suggestions in this guide, extends the basic set to include many generally useful commands, and in general forms a better basis for gameplay and customization than the original.

You can get the GABB file set and the accompanying guide on the ***Heroica!*** website.

2. KEYBINDS

2.1 Keybind Overview

To recap things you should have read above, and add some useful points:

» A keybind binds one or more slash commands to a single key. When that key is pressed, the command string will be executed.

» You enter keybinds by typing them into the chat entry window, prefaced by a foreslash (/), in the form:

```
/bind keyname command_string
```

» The command string should normally be enclosed in one set of double quotes, although they can be omitted for single-word commands.

» Any binds you enter will overwrite any existing bind on that key.

» You can erase a keybind, either one you've entered or a default one, and make the key "dead" in the game, by using the **`nop`** (no operation) keyword:

```
/bind keyname nop
```

» Finally, you can retrieve the current bind for any key using:

```
/showbind keyname
```

2.2 Key Names

Nearly every key on the keyboard can be used for binds, but, like magical spells, you have to know each key's "true name"—which might not be obvious. For example, to bind something to the equals key, you can't use **`=`** —it won't work. You have to use **`equals`** instead. Many keys have similarly odd, but sensible once you understand them, names.

The list of allowable key (and mouse-button) names can be found in Reference B.

2.3 Shift Keys

All, or nearly all bindable keys can be combined with the Shift, Alt and Ctrl keys to allow additional combinations. That is, **`K`**, **`SHIFT+K`**, **`CTRL+K`** and **`ALT+K`** represent four different binds. More precisely, these combos can represent four different binds… but there's a slight catch.

Also, while left-right shift key names such as **`LSHIFT`** and **`RCTRL`** are allowed, my experience is that all three in each group are synonyms as shifting keys for other commands. That is, **`ALT`**, **`RALT`** and **`LALT`** can all be used as shift modifers but always represent just "**`ALT+`**".

It is possible to bind each of these (**`LALT`**, **`RCTRL`**, etc.) to a single tap command bind, so that **`LALT`** and **`RALT`** represent two different command keys, but in general I have found that to be needlessly confusing and complicate use of the shift keys as, well, shift keys. Avoid using shift keys as tap keys.

The alphabetic keys A-Z are case-insensitive in bindfiles; binding to **`R`** and **`r`** is exactly the same. However, **`Shift+[alphakey]`** is a different bind—that is, **`R`** and **`SHIFT+R`** are different.

The catch mentioned above is a glitch in shift key combinations that comes in when a shift combination is left undefined. That is, if you have F bound to a function, say:

```
/bind F "follow"
```

`F` will execute **`follow`**… but so will **`CTRL+F`**, **`ALT+F`** and **`SHIFT+F`**… the undefined combination is ignored and the key is interpreted as its own. If you want to prevent this unwanted behavior, you have to define the combinations as "no operation" or **nop**:

```
/bind F "follow"

/bind CTRL+F "nop"

/bind ALT+F "nop"
```

This will make **`F`** execute **`follow`**, **`CTRL-F`** and **`ALT-F`** do nothing… and because I omitted it, **`SHIFT+F`** will still work as F alone. You can usually skip padding out the bind file with exhaustive **nop**-ping, but selective use may make the keys more reliable for your style of gameplay.

2.4 Basic Command Usage & Command Modifiers

In some cases, all that needs to be done to use a slash command in a keybind is to type the name of the command:

```
/bind F "follow"
```

Note that the command string is in quotes; although you can sometimes get away without the quotes, you should make it a practice to always use them, even when the command is a single keyword, as here. This command, which mimics the default bind for the **`F`** key, will cause your character to follow the selected target. However, the following example:

```
/bind A "left"
```

won't do quite what you think (what the default bind for the **A** key does). Since hardware and operating system key repeats are disabled within City of Heroes (actually, they are discarded everywhere except in the chat text entry window), pressing **A** with this bind will cause your character to move the default amount in a strafe-left manner… and stop. Since what you probably want is for the character to keep strafing left as long as you hold the key, you have to add a modifier:

```
/bind A "+left"
```

It's that + that makes the key repeat the action as long as it's held down.

Now suppose you want to toggle on a power or state—like autorun (`R` in the default key mapping). If you use

```
/bind R "autorun"
```

what you'll get is a status response: you'll see "autorun 0" in the chat window, since the above command is treated as an inquiry into the state of the autorun command. If you try:

```
/bind R "+autorun"
```

you'll get autorun as long as the key is held down... or the same as holding down the W key, not very useful. To make autorun toggle on and off the way the default is mapped, you have to use:

```
/bind R "++autorun"
```

...and there's the trick. **The ++ tells the game that it's a toggle command**: each press will toggle the state of that power on or off. If you were to be silly and use:

```
/bind Q "++turn_left"
```

what you would get is your character spinning in left circles when you pressed Q, until you pressed Q again to stop it. Silly, but not very useful.

Commands that toggle can usually also accept a numeric toggle code. For instance:

```
/bind R "autorun 1"
```

would force autorun on, no matter how many times it was pressed. You could then bind another key:

```
/bind V "autorun 0"
```

to turn autorun off unambiguously. This isn't a very useful example, since toggling autorun on and off with one key is quite enough for most players, but there are many situations where you want a firm "on" command and a firm "off" command, with no possibility of, say, dropping your shields during a battle, or turning off Hover or Fly in a sticky situation.

There are also (newer) commands that only allow toggle-on and toggle-off. We'll go into more detail about toggles later.

Note in all these examples that there is a slash at the front of the bind string. This is needed only when directly entering commands in the Chat window. No slash is used for lines in the keybind file.

2.5 Command Separators

The real power of binds and macros isn't in binding a single command to a key or macro button: it's in the ability to string multiple commands together in that bind. There are some limitations in how you can combine actions, but generally any reasonable combination of actions can be made. If there is a limit to the length of a bind command string, it's long enough that it will rarely be a problem. (I think it's 255 characters, but few binds will be anywhere near that long.)

Here is perhaps the single most useful custom bind for melee types:

```
/bind G "target_enemy_near$$follow"
```

This extremely useful bind causes your character to target the nearest foe and follow (lock onto) them. By binding it to my **G** key (as in the GABB), I have the option of tapping **F** to follow a selected foe (useful when I want to home in on a selected boss surrounded by minions who might be closer to me), or **G** to just pounce on the closest foe. In the middle of a fight, surrounded by foes, it is a huge timesaver (and occasionally a butt-saver) to be able to whack **G** and retarget the nearest foe.

The trick here is the **$$** characters, which act as a separator between commands. If you were to simply type a list of commands separated by spaces, the console would be unable to parse the line and while it might do something, it's not likely to be what you wanted. So each command needs to be separated from the next by a **$$** pair, with no spaces around it.

Chat-text and emote commands can be added to almost any chained bind. Here's a simple mod to the above bind that can be helpful in a team situation:

```
/bind G "target_enemy_near$$g I've got the $target!$$follow"
```

This bind will target the nearest enemy, announce in the Team channel "I've got the Bone Daddy!" (or whichever foe was targeted, by name), and then follow him. Since the chat text is only in the Team channel and simply won't show up when you're not teamed, it won't bother non-team players. And in this case, all the commands will execute more or less at once, so order is not important.

☞ But... *ahem*. A word about that. It's an annoying newbie trick to put a chat message on your power activations; no one you're not teamed with cares that you've activated Fly, hurled a Zapp, turned on your Plasma Shield, etc. Most newbies who discover the joys of chat-binding do it… once. And get howled out of the zone, most likely. Don't be a clueless jerk; don't bind chat messages to your powers except very selectively in the Team channel, when the message will be helpful—every single time!—to your mates.

(And not just newbs. A 13-year charter player I know, who shall remain nameless, just did this stupid thing with a new command macro, until the game auto-muted him. Fortunately, I… I mean **he** was in a mission and only annoyed his teammate. Take the warning.)

☞ See also section 3.0 for an advanced use of this separator.

2.6 Multiple Command Limitations

You can string multiple commands together using the **$$** separator, but there are often limitations on which commands will work in certain cases and sequences. These limits have changed somewhat across the history of the game, and sometimes updates create "parsing" bugs that are later corrected. In general, though, you have to work around the following limitations when putting multiple commands in a bind or macro:

» Attack powers cannot be chained in single bind. At all. Only the last power in such a chain will execute, no matter how many times the bind key is tapped.

- » Toggle powers can be chained in a single bind, but only one will activate on each tap of the key. If you put, say, three shield powers on one key, three taps will activate all three, but there is no way to make one tap trigger more than one power.
- » Chained powers must use the “toggle on” command and not the more general “execute power” command. Using the latter (poweexecname) would make powers toggle on and off erratically. Chaining can only work for force-on and force-off commands.
- » If a power has an activation time, it will block all subsequent powers in a bind string. This behavior is erratic but in general trying to chain a power with a significant activation time won’t work.
- » And, most important of all: **Chained powers are executed in reverse order!** For some chained bind commands, all elements will execute at more or less the same time. However, if you want to chain shield powers, for example, the last power in the bind will execute first, then the second-from-last, then the third-from-last, and finally the first. This order can be tricky to manage for some binds.

You will probably have to experiment with each new combination to find one that works the way you want it to. It may take a few tries and looking up a few quirks.

2.7 Toggles and Forced Toggles

One of the problems with keybinds is that most are, by default, a toggle—the bind will simply flip the power to whichever state it’s not in. Sometimes, as with the **`autorun`** key, that’s exactly what you want. Other times, you want an absolute, guaranteed "power on" or "power off," even if you hit the key by mistake.

Easy enough. There are several "power activation" commands that operate in different ways, and it’s esy to select the one you want.

You can toggle a power by specifying its name (preferred) or which tray slot it resides in:

```
/bind P "powexec_slot 3"
```

```
/bind P "powexec_name Fire Shield"
```

Assuming Fire Shield was in slot 3 of the main tray, these binds would work exactly the same—pressing **P** would toggle Fire Shield on and off. (I can think of some uses for the slot-number method, but in general, you should stay with the power-name method.)

But if you want Fire Shield to go on, and on only, when you whack a specific key, so that you never inadvertently drop the shield during a battle, you would use:

```
/bind P "powexec_toggleon Fire Shield"
```

Which would always force Fire Shield on, even if it was already on. (That is, if the power is on, the command would have no effect.) You could turn the power off by clicking its tray button, or by adding a forced off bind:

```
/bind CTRL+P "powexec_toggleoff Fire Shield"
```

If you want to bind one key to activate multiple powers, it is essential to use **`_toggleon`** as the command, so that serial keypresses don’t toggle powers on and off unpredictably.

Because few powers have activation delays when turned off, most bind strings combining a series of **`_toggleoff`** commands will work with a single key press.

2.8 Rollover Binds

There are many reasons to have a command change with each execution—to alternate forms of a power, or an emote, or whatever. There are two basic ways to create this "rollover" effect. One is to call a macro defined in a tray slot, and make the macro change the tray to another one that has an alternate form of the macro defined in the same slot. (See the following Section 3.2 for more details on macro/tray rollover.)

An alternate approach, which is far more flexible, sturdy and unlimited, is to use keybinds that load overwriting binds.

To use rollover binds for a command, start with a bind like this in your master load file:

```
/bind ALT+G "g Gratz!"
```

Now every time a teammate levels, you can gratz them with a keyflick. And probably get tired of saying the same thing over and over, as will your team.

So do this instead:

```
/bind ALT+G "g Gratz!$$bindloadfilesilent gratz02.txt"
```

What's in the file **`gratz02.txt`**? This one line:

```
/bind ALT+G "g Congratz!$$bindloadfilesilent gratz03.txt"
```

And in **`gratz03.txt`**?

```
/bind ALT+G "g Gratz-a-roonie!$$bindloadfilesilent gratz04.txt"
```

And in **`gratz04.txt`**?

```
/bind ALT+G "g Gratz!$$bindloadfilesilent gratz02.txt"
```

…which brings it all back around in a circle. Every time you tap Alt-G, your alt comes up with a (reasonably) fresh comment. The trick here is to make the chain of files a closed loop, no matter how many there are.

The GABB makes extensive use of rollover binds for the eight or so basic communication strings, albeit using only a single update file for all.

There is effectively no limit to the number of iterations for this process, and it could be used for other fun stuff like random dances or emotes, or for serious rotation of powers and attacks.

2.9 Pseudo-Autofire Binds

Multiple commands and rollover binds can be combined to make certain powers and groups of powers all but automatic. Later note: if you don't use these, you're leaving money on the table!

The built-in auto-attack function, which will queue and fire any click power, is limited to a single power and won't work with toggle powers. It seems to have been created to allow melee archetypes to always have their "first, best punch" ready to go and fire as soon as a foe is within range—hence the official name, "auto attack"—but most players who use it do so to keep click buffs like Hasten and Practiced Brawler active.

If Hasten, for example, is set to auto-fire (by Ctrl-clicking on its icon in the power tray; a green ring indicates it's the selected auto power), it will fire every time it expires and recharges. Since regular slotting cannot reduce Hasten's recharge time to match its active time, there will be a recharge gap before auto fires it again. (With advanced slotting and recharge boosts, it can be made continuous or "perma.") Practiced Brawler and Rage are better examples, since with three SO (Single Origin) enhancements, their recharge times are slightly less than its activation time, meaning that if set to auto-fire, it will be up all but a few moments each cycle, or "perma"nently.

But there is one and only one auto-fire power at a time, and it only works on click powers. So a clever bind strategy can be used to multiply the number of automatically fired powers… as well as make it easy to keep an array of toggled defense or buff powers up with little attention.

Here's a simple version, intended to autofire two click powers:

```
[af-01.txt] W +forward$$powexec_auto Hasten$$bind_load_file_silent .\af-02.txt

[af-02.txt] W +forward$$powexec_auto Practiced Brawler$$bind_load_file_silent .\af-01.txt
```

You should be able to follow what's happening here. With these two files in your bindfile folder and one of them added to your main binds in place of the regular `W` bind (simply `+forward`), each time you press `W` to move forward, which should be almost continually in gameplay, three things will happen:

1. The other rolling bind file will be loaded.
2. The specified power will be set as the auto-fire power, and if it's recycled, it will fire.
3. You'll move forward as long as you hold `W` down, just as before.

Steps 1 and 2 will happen instantly and invisibly; you'll just move forward. But because you are continually tapping and pressing this key, the other steps will keep setting and firing each of the two click powers, making both effectively set to auto-fire.

Builds with several click buff or defense powers can have all of them similarly "auto"—just extend the ring of rollover bind files with one (and only one!) power command in each.

☞ As a backup, you can create a parallel set of rollover files and assign them to a key with no movement action (such as `CTRL+W`) so you can fire up all your click powers without having to move.

Auto-Toggle Binds

But wait, there's more! Builds often have several toggle powers that should be active in combat and, effectively, all the time. Even though they can all be assigned to a single key so that, say, tapping **U** several times brings them all up, it's easy for attacks and loss of endurance to make them drop… and you may not notice the loss of a shield or buff until you're in trouble.

The same "auto" activation can be added using the same rollover bind process. In fact, click powers and toggle powers can be combined and activated in the same loop, making the job of keeping your alt fully buffed, shielded and ready for combat completely automatic. Consider the following chain of rollover bind files:

```
[1.txt] W "+forward$$powexecauto Build Up$$bindloadfilesilent 2.txt"

[2.txt] W "+forward$$powexecauto Hasten$$bindloadfilesilent 3.txt"

[3.txt] W "+forward$$powexecauto Clarion Core Epiphany$$bindloadfilesilent 4.txt"

[4.txt] W "+forward$$powexectoggleon Assault Core Embodiment$$bindloadfilesilent 5.txt"

[5.txt] W "+forward$$powexecauto Aim$$bindloadfilesilent 1.txt"
```

…which fire four click powers and ensure a toggle power remains up. This loop can be extended almost indefinitely.

☞ By the way, any power command can be used with the click powers (**powexecname**, **powexec-toggleon**) but using **powexecauto** keeps the UI from given the annoying *"hoom"* sound when the power is already activated or not yet recharged. It cannot be used for toggle powers, though.

The Need for "Blank" Files

Because the command parser does odd things when a bind loads a new bind file, there is a need to include a "blank" or dummy file in between most active files when using this rollover technique. The rotation of files should be one "active" file containing a power command, and one that does not, to absorb the unwanted key processing. Such as:

```
[01.txt] W "+forward$$powexecauto Clarion Core Epiphany$$bindloadfilesilent B1.txt"

[B1.txt] W "+forward$$bindloadfilesilent 02.txt"

[02.txt] W "+forward$$powexectoggleon Assault Core Embodiment$$bindloadfilesilent B2.txt"

[B2.txt] W "+forward$$bindloadfilesilent 01.txt"
```

"Blank" files like B1 and B2 should be repeated through the entire rotation. The result will be more reliable power activation. The 01/B1 numbering sequence can help keep things straight as you edit and change the file set.

Bypassing Power Activation

A refinement I found useful after using movement binds on **W** (forward) for a while is to bind **E** for forward movement only, without activating powers. Just **E +forward**. This lets you move without bringing up your buff and def powers. I found it most useful on alts with invisibility; if you have to lead a hostage out, you can't do it if you keep going un-visible.

Combine with **CTRL+U powerstogglealloff** to cancel all active powers for smoothest control.

Big thanks to Placta for the above example and for raising the issue in a forum discussion. This is a technique I have explored in the past only to find that command parsing limited its usefulness (for instance, there was a time when setting a second power to auto-fire would fail until the first auto power had recycled). Being advised that it worked, so I could explore it to document here, was a very useful poke!

And huge props to Sovera, who worked all this out a while ago and both shared his notes in the forum and walked me through some of the subtle points (such as the blank files).

2.10 Press-Release Binds

An extraordinary feature is buried in the bind parsing: one keypress can be used to fire the command string on both press and release. As anyone who's mucked around with keyboard macros and keyboard interfaces and the like knows, PCs interpret a key press and release as separate actions. In all but a very tiny number of instances, this disappears into just "a key press."

But with one simple code addition, CoX binds can interpret the press and release separately. It's a very niche function but offers some rich and clever options.

Huge props to Linea, who brought this to my attention in late 2021, assuming it was a trick some number of long-time players knew but was part of the game's "lost over time" info. I have to admit I've never heard of it — which, modesty aside, is saying something — and I can find no reference to it in any game wiki or resource. So Linea's revelation is of huge value to me and all players who are looking for the most clever bind tricks.

And note this works on binds only. It doesn't work on macros.

The trick is quite simple. If you start the bind string with **+$$**, the whole command string will be executed once on press and again on release, no matter how long the key is held down in between.

Basic Examples

Here's a simple, largely useless example that you can use as a model for your own experimenting, which you will want to do, because I haven't fully determined the limits or possibilities of this trick.

```
/bind EQUALS "+$$l ONE$$l TWO"
```

When the **Equals** key is pressed, local chat will say **ONE | TWO** (on separate lines). When the key is released, it will say **ONE | TWO** again. Yay. But that clearly demonstrates the function. If you remove that leading +$$, the press (and release) will just say the words once.

Do this in your base, or way out in a zone where you won't annoy the Local crowd.

So what use is that? Two good uses were outlined by Linea's samples. The first is for the (mostly) new two-stage powers, such as Mystic Flight / Translocation. You can't teleport unless you have flight enabled, which presents all kinds of bind and macro hassles. For example, I usually bind **Y** as the “get the hell out of here” key, or **`teleport up:max`** on my base TP alts. On alts that have taken the Sorcery pool instead, it works, but only with two taps:

```
/bind Y "powexeclocation up:max Translocation$$powexectoggleon Mystic Flight"
```

One tap turns on flight, the second does the emergency TP. If flight is already on, it will TP on the first tap. That works, but it’s clumsy and it means my standard control interface varies between alts.

With the press-release trick, it gets incredibly elegant:

```
/bind Y "+$$powexecname Mystic Flight$$powexeclocation up:max Translocation"
```

Press-release, and you TP up high. No hassles.

Now, there’s a second feature there: that bind also turns off Mystic Flight again… which is good and bad. For this bind (mine), it means you start falling immediately. So I actually use this bind:

```
/bind Y "+$$powexectoggleon Mystic Flight$$powexeclocation up:max Translocation"
```

…so that flight remains on afterwards. The command is execute both times, but it’s “force on” and not toggled. So why would you want to toggle flight? To use Linea’s original bind, a nifty substitute for **Combat Teleport**. Target your evil foe and:

```
/bind Y "+$$powexecname Mystic Flight$$powexeclocation target Translocation"
```

…and flight is toggled on, you TP right to your target, and flight is toggled off. Now “defeat” the bad guy and we’ll continue.

Rolling Bind Examples

There are very few situations other than the two-stage powers where executing the whole string twice is useful. So, as Jean-Baptiste Emanuel Zorg put it, “Let’s… change… the tune.”

Executive summary: if you combine this press-release trick with rolling bindfiles, you can execute two different command strings on each press.

Want to dream the impossible dream and fire two combat powers with one keypress? No problem:

```
[2Attack-1.txt] EQUALS "+$bindloadfilesilent 2Attack-2.txt$$powexecname Lightning Bolt"

[2Attack-2.txt] EQUALS "+$bindloadfilesilent 2Attack-1.txt$$powexecname Charged Bolts"

/bind EQUALS "+$bindloadfilesilent 2Attack-2.txt$$powexecname Lightning Bolt"
```

Press **`Equals`**, and you fire one ranged attack; release, and fire the other. This can be extended for an entire attack chain using multiple rollover bind files. So yes, the command string does change in between presses, and is not ‘preloaded’ for the release activation. Cool. Another use would be to fire a buff before an attack:

```
[Boost-1.txt] EQUALS "+$bindloadfilesilent Boost-2.txt$$powexecname Build Up"

[Boost-2.txt] EQUALS "+$bindloadfilesilent Boost-1.txt$$powexecname Zapp"

/bind EQUALS "+$bindloadfilesilent Boost-2.txt$$powexecname Build Up"
```

Note that this one could get out of sequence and fire **Zapp**, then **Build Up** (or pass on the second action). A pre-reset bind or macro (to load the first bindfile) might be useful for this use, and should not be a problem to tap before firing a primary attack.

☞ Huge caveat/tip: Nearly all binds using this trick will work better and more reliably if you pause, even briefly, between press and release. A fast tap might work — always for some strings, and often for others — but a slower press/release will get past all kinds of brief delay/wait issues.

Linea provided a number of examples for both combat, defense powers, emote/costume changes etc. using this trick. Some are (to me) just variants of doing the same thing with either a regular bind string or simpler rolling binds, but many get past the limitations of the two-stage powers and stacking combat powers. Go forth and experiment!

Other Sample Binds

This little gem activates 'fly forward' as long as the button is pressed, sort of a combo of fly, jump, jump pack and combat teleport. Substitute **Fly** if your alt uses that instead:

```
U "+$$powexecname Mystic Flight$$++autorun"
```

👍 Props to Aracknight for the idea.

More to come, from my own experiments and ideas of others on the forum.

3. MACROS

3.1 Macro Overview

If you've read this far, macros are simple: they are exactly like keybinds in every way, except that they are bound to a Power tray button instead of a keyboard key. The only difference is the first two elements of the string:

```
/bind H "g I'm assisting $target!$$follow"

/macro AST "g I'm assisting $target!$$follow"
```

The second example will create a tray icon labeled AST in the first open power tray slot. Clicking this button, or activating it with an associated keypress, will be exactly the same as pressing H in the keybind example.

Macros are one of the reasons you have 90 power tray slots. Besides being able to create a couple of alternate power configurations, you can create any number of macro trays—one for soloing, one for team work, one to primarily control or defend, one for melee or ranged attack work, etc.

Macros may be named with any combination of letters and numbers from one character to... many. However, only the first three characters will fit on the button, so you would be wise to keep your macro names to three characters or less. Any longer string will be shown as a "tooltip" when the pointer hovers over the icon, so a 'macro name' of **`AST Assist Teammate`** would show **`AST`** on the icon, but the whole string on hover.

In general, macros and binds function identically and have identical limitations. However, a macro called by a bind can permit some action combinations that binds alone do not. A good example is pet emotes, which work from macros but not from binds (a "temporary" problem that shows no signs of being fixed; see Section 6.1 for details). Experiment with macro versions of a command if you run into bind limitations.

3.2 Directed Macro Creation

The only other useful thing to say about macros is that there is a second macro creation method that allows you to place a macro icon in any tray slot. The base **`macro`** command places the new icon in the first empty slot it can find. Using **`macroslot`** instead allows you to create the macro icon in any of the 90 tray slots. For example, this command assigns the newly created macro to slot 6 of Tray 1:

```
/macroslot 5 AST "g I'm assisting $target!$$follow"
```

And this would put the newly created macro in slot 2 of Tray 3:

```
/macroslot 21 AST "g I'm assisting $target!$$follow"
```

This works because the command treats the tray slots as sequentially numbered from 0 (first slot of Tray 1) to 89 (the last slot of Tray 9). Yes, the numbering for slots using this command is zero-based, so 0-9 correspond to slots otherwise numbered 1-10. This variation between “one-based” and “zero-based” numbering is a subtle trap in many of the slash commands and will be noted wherever it’s potentially confusing. See the macroslot command in Reference A for more details.

This rather complex command is of limited use unless you get into writing very complex macro-based command schemes where creating macros on the fly and having them land in a specific slot is a requirement. Most users, including all novices, can just ignore this option.But there you go.

3.3 Macros Using Tray Rollover

An advanced trick that can be used for both binds and macros is to use "tray rollover"—swapping power trays to change the macro or bind action each time.

The basic process is this:

» Create two macros that do the desired pair of actions and end by swapping trays:

```
/macro A1 "emote drumdance$$gototray2"

/macro A2 "emote victory$$gototray1"
```

» Put the first macro in tray 1, slot 1. Put the second macro in tray 2, slot 1.

» Now create a bind that fires the power in slot 1 of the primary tray:

```
/bind Q "powexec_slot 1"
```

» Now, each time **Q** is pressed, it will execute the macro in slot 1 of the primary tray, then swap trays. The alt will alternately “drumdance” and “victory-wave” using the same bind key. (Obviously, you can use more elaborate and useful commands here.)

This can be taken to very complex and extreme levels, swapping among all available trays, but requires very careful management of tray contents and other power positioning. You don’t want to swap in a primary tray that lacks a combat power or rearranges them, not just as you engage that Archvillain.

More sophisticated ‘rollover’ systems can be implemented by advanced tray swapping; see Section 5 for details.

3.4 Macros with Custom Icons

A very cool option for advanced macro users is to be able to use a power-tray icon for macros instead of the limited, gray+letters icon. Everything about using this command is the same as above—that is, actually creating macros is the same—but with one additional argument, you can assign any power-tray icon used in the game to your new macro.

The command is:

```
macro_image TEXTUREFILE "NAME" "COMMAND_STRING"
```

NAME is required, as for other macro commands (where they show up as 1-3 letters on the macro icon) but here will only show up on hover or when the Info panel is opened. They're thus similar to (and called) tooltips, and strings don't have to begin with the two or three characters used on plain icons for clarity. Use quotes around multiple words in each argument, as shown.

COMMAND_STRING is the same as for any bind or macro definition: all the commands you want execute when this macro is called. Use quotes around the string.

TEXTUREFILE is where it gets interesting. **This must be a string that points to an existing power tray icon bitmap or texture within the game's PIGG files**, and is composed of the powerset name and the power name, all run together as one word. Underscores are probably optional but should be used for clarity. The most important thing is that no spaces are allowed in this texture-name string.

All you need now is to choose a power icon texture file… which is easy, there are only thousands of them. Fortunately, a complete listing has been extracted and is readily available on the web; one copy can be found on the ***Heroica!*** website. (It's much too large to include here.) The list is as useful for figuring out naming anomalies as it is for finding any one power set or power name, because you can usually guess many names with reasonable accuracy:

```
SuperStrength_Rage
DualPistols_SwapAmmo
Flight_Fly
```

…but not all of them. Using the pictorial list to find an icon you like and learn its correct name is recommended. You can use any powerset and power icon, regardless of your alt's power sets.

Thanks to Yuro on the Titan forums and **many** thanks to Kala in the game for this info!

3.5 Macros vs. Keybinds

There are a lot of arguments in the community about macros versus keybinds for game control. They both work and are equally easy to create. **It's simple: use whichever mode, wherever it best suits you, your alt's powers and your gameplay style.**

I'm very much a bind guy, especially for commands that are used often or under frantic conditions like combat. Finding and clicking macros takes time and focus you might need in a team battle. When you get to things like several trays of macros, or rolling trays, it seems like it almost becomes a mini-game to find and click the right ones. Macros have the advantage of being self-evident and requiring little “relearning” and memory, especially if kept well organized.

On the other hand, binds are very easy to edit, update, save and reload. Macros can be a bear to manage and change around, and have annoying quirks like being ignored when you try to clear your power trays. There is also no systematic way to save, reload and transfer them between alts.

So my general suggestion is this:

» Use basic keybinds (the ones easy to remember, find and hit) for all essential gameplay, especially basic combat commands.

» Use “trickier” keybinds (pretty much anything that’s not a base key or Ctrl-key, or on the Function keyset) for things you have time to think about a moment and hit correctly.

» Generally, use keybinds and a file save/load system to enable rapid updates and transfer between alts and players.

» Use macros selectively for configuration swaps and powers you can execute at leisure, like base teleport, swapping between combat and heal modes, between solo and team modes, etc.

But hey… it’s your alt, your game and your style. If it works, there is no wrong way!

4. Saving/Loading Interface Settings

With Issue 11 or 12, City of Heroes/Villains finally resolved one of the most annoying oversights in its design. Each new alt that you designed started with a generic user interface setup, and there was no way to duplicate a favorite layout and setup without laboriously configuring each element, every time. Now, however, there are not one but three separate "save/load" functions to save an aspect of a customized user interface and reload it into another character's interface. (Four if you count keybinds.)

There are three sets of customization commands, for chat, window layout, and the grab-bag "options." All work much like the process for saving and loading binds, so any user who has mastered those basics should have no trouble with these facilities.

4.1 Common Settings vs. Individual

CoX has a fundamentally dual process for saving and loading user settings. The first is to use a simple command to load and save from a default filename in a default location. The second is commands that permit specifying a file path and name. For keybinds, the set is:

```
/bindsave
/bindload
```

…which write and read a file named **`KEYBINDS.TXT`** in the game directory. The more sophisticated commands:

```
/bindsavefile filename
/bindloadfile filename
```

…save and load from a designated filename and path, which can be somewhere more convenient for editing and named to distinguish each alt's bind set, or even variant bind sets for the same alt (in team or solo, healer or combat, etc. modes).

☞ Again, I recommend that you use the default [GAME PATH]\data location for all bindfiles, since it requires no path string for the game and commands to locate them there.

All of the following save/load options have a similar dual process. In general, you will want each alt to have its own settings you can tweak and refine for each archetype and power set. But there's one cool use for the default file operations: you can keep all of your alts' interfaces identical. For chat in particular, sharing a single saved setup is probably a good choice. You can also share an options setup in most cases. And even window setup… while you might want variations for melee alts, blasters and controllers, saving a basic optimized window layout you can load any time for any new or old alt is a nice convenience.

But in general you will want individual files for each alt's keybinds and window setup, if not chat.

4.2 Chat Configuration Save and Load

Saving a carefully designed chat window setup is now trivial.

1. Set up your chat windows as you like them, down to the last detail, on any of your characters:
   - » You can create up to four chat windows that are attached to the stack or free.
   - » You can name each window anything you like (appears on the little select tab).
   - » You can split each chat window in half and adjust the relative size of each.
   - » You can configure each chat window to hold whatever channels you want to monitor.
   - » Each chat window has a default channel. When you select that window, chat sends will go to the default channel. (This makes it easy to switch channels by just clicking a chat window.)
2. Save the chat configuration using **`/chat_save`**, which will save the configuration in the default game folder, in the file **`chat.txt`**.
3. Load the new configuration into each character's interface with the **`/chat_load`** command.

As you might tweak the chat settings with any one alt, save the file and remember to reload it for each other alt to share the exact same settings.

From there, you can save tweaked versions of the chat layout for each alt, if you like (e.g., you might want different channels for a soloing scrapper and a teaming tank), with:

```
/chat_save_file MyAltName-chat.txt
/chat_load_file MyAltName-chat.txt
```

☞ It may be possible to directly edit the **`chat.txt`** file, but one look at it showed some cryptic components (like numeric strings that likely reference specific channels). All but the most advanced users are recommended to leave the file contents alone and do all chat configuration from within the user interface.

4.3 Window Configuration Save and Load

Saving your individual preference for window layout and arrangement is now trivial, and will save tons of time and frustration as the game, your mistakes and temporary preferences mess things up.

Window and User Interface scaling has gotten a bit more complicated with the recent addition of the UI Scale menu slider. This gives you two UI/window scale settings that both interact and are reset by other actions.

Set up your user interface windows as you like them, down to the last detail, on any of your characters:

- » Start with the Window Scale setting in **Menu | Options | Windows**. Set it for a window scale of 100%.
- » Now scale UI Scale until the health/status bars are the size you want them. This window seems to override the Window Scale setting when its size is changed, so making its desired size the nominal "100%" makes everything else simpler.

   - Use the **`/windowscale`** command (and the list of window names in Reference C) to tweak each window to exactly the size you want. Position all windows where you want them and continue to adjust sizes until they are all the way you like. (Don't rescale the status bars at this point... go back to the step above if you want to change them.)
   - Use the **`/windowcolor`** command to set the window frame color and overall transparency to something you generally like; you can adjust this for each alt later.
   - **Advanced setup:** open secondary windows like contacts, missions, salvage, recipes, auction house etc. and size and position them as well, for convenient use even if you don't want them up during regular gameplay.
4. Save the window configuration using **`/wdw_save`**, which will save the window configuration in the default game folder, in the file **`wdw.txt`**.
   - **Advanced setup part 2:** Now close all the secondary windows and save the file again.
5. Load the new configuration into each character's interface with the **`/wdw_load`** command.
   - **Advanced etc.:** As far as I know, the secondary windows should load their position and size info even if they are not displayed.

While the shared configuration will make a nice base for loading your general preferences to each alt (including new ones), it is strongly recommended that you use the individual config method for each alt. As with the chat configuration, use:

```
/wdw_save_file MyAltName-windows.txt
/wdw_load_file MyAltName-windows.txt
```

And for maximum convenience as you tweak each setup (and then need to recover from changes and screwups), keybind these (**`Alt-F5`** and **`Alt-F6`** are the GABB choices):

```
/bind ALT+F5 "wdw_save_file MyAltName-windows.txt"
/bind ALT+F6 "wdw_load_file MyAltName-windows.txt"
```

- Finally, on each alt, use **`/windowcolor`** to adjust the window frame color to something pleasing and compatible with the costume, powerset, theme, etc.

☞ As with the chat files, the window configuration files are pretty cryptic. It may be possible for advanced users to directly edit the **`wdw.txt`** file, but all but the most advanced users are recommended to leave the file contents alone and do all window configuration from within the user interface.

4.4 UI Scaling vs. Window Scale

A recent addition to the menu options is the "UI Scaling" slider. This is a huge boon in the days of vastly varying display resolutions, but its use can be confusing because it interacts with the existing Window Scale setting. For one thing, it acts instantly while the Window Scale waits for the Apply Changes button. Furthermore, changing the size of the health/status bars changes the Window Scale setting, which is why the above process to make that window 100% is recommended.

There is also an automatic UI scaling option, but I don't care for how it works. You may feel differently… experiment!

4.5 Option Configuration Save and Load

Ah. Now the good stuff—the feature that lets you set any of several dozen game parameters, either individually or by loading a saved file. The Devs decided to call this grab bag "options."

Saving and loading option configuration files is the same as saving and loading bind, chat and window configurations:

1. Set all of your options in the configuration menu, down to the last detail, on any of your characters. See the rest of this section for the many details.
2. Save the option configuration using **`/option_save`**, which will save the option configuration in the default game folder, in the file **`options.txt`**.
3. Load the new option configuration into each character's interface with the **`/option_load`** command.

Options are far more generic than the other settings and it seems unlikely that most players will want to customize any options for individual alts. However, the **`/optionsavefile`** and /**`option-loadfil`**e commands stand ready to use as in the above sections, should you find a reason to customize your settings by alt.

☞ Unlike the above config choices, the **`option.txt`** file seems to be readily editable, as the contents are merely the option keywords and the status or values. Recommended for advanced/experienced users only, though.

4.6 Setting Options

You can set options in the extensive panels of the **Menu | Options** menu. You can also set individual options via the slash command **`/option_set`**, which takes two arguments: the option keyword and the new value. For example, you can toggle on dirty word bleeping with the following command:

```
\option_set allowprofanity 0
```

And return to seeing every word your angry tank teammate wants to type by using:

```
\option_set allowprofanity 1
```

Even simpler, most options can be toggled from one state to the other using optiontoggle:

```
\option_toggle allowprofanity
```

will simply flip the setting from one state to the other.

There are a number of options, and option values, that can *only* be set using this process—they are either not in the menu list, or do not offer the full range of option settings. So learning the option keywords and values can open doors to some subtle game optimization.

4.7 Option Keywords

Ah, but you ask, what *are* the available option keywords? There are two simple ways to determine them, besides looking at the handy table below. The command **`/option_list`** will dump out a list of option keywords in the System chat tab. You can use **`logchat`** or **`copychat`** to capture the stream for offline examination. I discovered, though, that this list is incomplete. So the more thorough way to learn all the option options is to save an option file, as above, and open it for examination. This will tell the savvy player many things about his or her UI setup.

In the list below, nearly every keyword is self-explanatory and can be mapped to items in the Option menu. If not, ask in the forums or experiment!

All of the **keywords in blue** are believed to be 0/1 toggles and can be set using **`/option_toggle`** or **`/option_set {0|1}`**.

The **keywords in orange** use more complex numerical arguments, some in decimal values, some in float values and some in hexadecimal. Only a few of these have been sorted out, so there is much experimentation, archive searching and code digging to do for a complete explanation. (The two keywords to set the chat bubble color have proven particularly frustrating, as even putting the numbers from the saved file back in as arguments produce erratic results.)

Complete argument sets for two groups of commands follows the table, to give savvy users a head start.

Further Information

Between this start and examination of your own options files, it should be possible to sort out all the other numeric options—maybe even the chat bubble colors, someday. Try setting different values in the Options menu, saving the file and examining the value changes… and once you have all those mapped, try using other values to see what happens.

Note that some options do indeed write their values out in hex, and may want hex arguments.

Reports on any complete, verified findings and interesting results solicited, with credit!

Option Keywords		
AdvancedPetControls	AllowProfanity	ArchitectAutoSave
ArchitectBlockComment	ArchitectNav	ArchitectToolTips
AutoAcceptTeamLevelAbove	AutoAcceptTeamLevelBelow	AutoDeclineSuperGroupInvite
AutoDeclineTradeInvite	AutoFlipSuperPackCards	BlinkCertifications
BuffSettings *[unsigned int]* *	CamFree	Chat1Fade
Chat2Fade	Chat3Fade	Chat4Fade
ChatBubbleColor1 *[signed int]* **	ChatBubbleColor2 *[signed int]* **	ChatDisablePetSay
ChatEnablePetTeamSay	ChatFade	CompassFade
ContactSort	CursorScale [0-2.0]	DeclineGifts
DeclineGiftsFromTeammates	DefaultChatFontSize *pixelheight* ***	DisableCameraShake
DisableDrag	DisableEmail	DisableLoadingTips
DisableMouseScroll	DoNotSeeEnemyLocal	EnableChatLog
EnableClickToMove	EnableJoystick	FadeExtraTrays
FriendSGEmailOnly	GmailFriendOnly	gShowPetBuffs
HideButtons	HideContactIcons	HideConvertConfirmPrompt
HideEnhancementFullMsg	HideFee	HideHeader
HideInspirationFullMsg	HidePetNames	HidePromptCoop
HidePromptDeleteEnhancement	HidePromptDeleteRecipe	HidePromptDeleteSalvage
HidePromptPlaceEnhancement	HidePromptUnslotEnhancement	HideRecipeFullMsg
HideSalvageFullMsg	HideSalvageWarning	HideStorePiecesState
HideUnclaimableCert	LogPrivateMessages	LoyaltyTreeAccessButton
MapOptionRevision *n*	MapOptions *hex*	MapOptions2 *hex*
MouseButtonReverse	MouseInvert	MousePitchSetting
MouseScrollSpeed *float*	MouseSpeed *float*	NewCertPrompt
NoXP	NoXPExemplar	OpenSalvageWarning
PreventPetIconDrag	PromptTeleportFromTeammates	RecipeHideMissingParts
RecipeHideMissingPartsBench	RecipeHideUnowned	RecipeHideUnownedBench
SeeEnemyBroadcast	ShowAllStoreBoosts	ShowArchetype *n*
ShowAssistReticles *n*	ShowBallons	ShowEnemyTells
ShowOwnerName	ShowPetControls	ShowPets
ShowPlayerBars *n*	ShowPlayerName	ShowPlayerRating *n*
ShowPlayerReticles *n*	ShowSupergroup *n*	ShowVillainBars *n*
ShowVillainName *n*	ShowVillainReticles *n*	SpeedTurn *float*
StaticColorsPerName	StoreAccessButton	TeamComplete
ToolTipDelaySec *float*	UseOldTeamUI	UseToolTips
VoucherPrompt	WebHideBadges	WebHideFriends
WindowFade		

* Needs unsigned integer value. See section 4.8.

** For reasons unknown, these two commands need signed integer arguments. See Reference E.

*** Warning: it is possible to make chat and other text unreadable by using values below 10!

4.8 BuffSettings Argument List

The complex stacking arguments for the **`BuffSettings`** option, which controls what buff icons are shown on the player, pet and group buff display and how, was worked out back in the Live era and distributed on the various wikis and discussion groups.

👍 Kudos to whomever worked them out and took the time to present them to the community; thanks to Adeon Hawkwood for bringing this list to my attention in the game.

One of the frustrating things about direct manipulation of the option values is that they are evidently based on hexadecimal numbering within the game code, but may be written to the options file in either hex or decimal. The **`BuffSettings`** control values are decimal equivalents of hex, but hex numbers cannot be used to set the option despite how simple that would be. So users who want to carefully control their buff settings, including some choices not allowed in the Options menu, have to use really big numbers to do it, and carefully add together these values to get their desired result, an **unsigned integer**. A programmer's calculator (hardware or applet) will help a lot here.

A **`BuffSettings`** setting of 0 means all default buff icon display settings will be used.

There are three groups of buff-display settings, which work identically with different ranges of numbers. All of the numbers have to be combined into one value to set the display as the user wishes.

BuffSettings argument values

Buff Icon Status	Player Display (Status Bars)	Teammate Display	Pet Display
`Hide Auto Buffs`	1	256	65536
`Hide Toggle Buffs`	2	512	131072
`Do Not Blink Icons`	4	1024	262144
`Do Not Stack Icons`	8	2048	524288
`Enable Numeric Stacking`	16	4096	1048576
`Hide Buff Numbers`	32	8192	2097152
`Stop Sending Buffs`	64	16384	4194304
`NOT USED`	128	32768	

To disable buff-icon stacking for the player, an argument of 4 would be used. To hide auto power buffs and toggle power buffs, an argument of 3 (1+2) would be used. The same method would be used for the Team set of values and the Pet set of values.

So, to hide auto and toggle powers for all three groups, you would add together the relevant values:

```
1+2+256+512+65536+131072 = 197379
```

And *carefully* enter that totaled value as the argument. Anyone with some programming experience can see how much easier this would be in hex!

And again, a value of 0 means default settings will be used.

4.9 Character Display Options & Argument List

The Options menu has a number of settings that allow you to control how you see other characters—players, villains and NPCs—and under what conditions. By default, some information is shown all the time, some is shown when the mouse pointer hovers on a character and some is shown when the character is selected (targeted). All of these combinations are configurable… and some configurations not allowed in the menu can be set by **`/setoption`** commands, mainly Villain element to "show always."

Note that all available information is shown in the Target window when any character is selected, so these options control only the heads-up UI display.

The essential settings for how you see other characters are:

Setting / Menu Item	Default	Menu Option Range	Gunner's Choice
ShowArchetype	2	All	2
ShowSupergroup	2	All	0
ShowPlayerName	1	All	1
ShowPlayerBars	6	All	6
ShowVillainName	6	0 2 4 6	1
ShowVillainBars	6	0 2 4 6	6
ShowPlayerReticles	6	0 2 4 6	4
ShowVillainReticles	6	0 2 4 6	4
Reticles cannot be set to 1, "show always"			

Each of these can be *independently* set to the following options:

Status	Menu Option	/setoption Value
Off / Not Displayed	Hidden	0
Always Displayed	Show Always	1
Display on Mouse-Over	Show on Mouse-Over	2
Display when Selected	Show when Selected	4
Display when Mouse-Over or Selected	Mouse-Over or Selected	6

The simple 3-digit binary nature of the values should be obvious; any value that sets the 1 bit will produce an "Always" display, and so forth.

My personal choice for the cleanest, most informative character display is given in the red numbers in the list above. Try them yourself, and edit as you like. There are more display options related to pets, as well.

5. Power, Tray & Inspiration Control

Nearly everything that makes up CoX gameplay comes down to powers and how they are organized in the default (primary, secondary and tertiary) or other six (floating) power trays. Even many binds and macros make use of specific slots in the trays, and flipping trays around to change active commands. This section combines several areas of game control to give you a good handle on just how powerful and flexible the power trays (and the Inspiration tray as well) can be, and how much careful, sophisticated setup can improve your gameplay, pleasure and survivability.

5.1 Basic Powers Terminology

- **Power—**Any power an alt has, represented by an icon in the Powers list window.
  - All except "auto" powers that neither require nor allow activation (generally buffs and shields) can be dragged to a power tray or otherwise activated by binds or macros even if not in a tray.
- **Tray—**The rack of power icons that can be clicked and monitored.
  - The default physical tray can be one to three trays tall, called the primary (bottom), secondary (middle) and tertiary (top) trays. Each tray has specific slash commands as well as the shared ones.
  - Up to six "tear off" or floating trays can be called up and positioned independently on the screen.
  - There are a total of nine **virtual** trays, whether displayed or not, and each tray (default or floating) can be set to show the contents of any virtual tray, including duplicates.
- **Slot—**A power icon location in a tray.
  - Each tray has ten slots, numbered 1 to 0 on the UI but referred to as 0–9 by most commands.
  - There are a total of 90 slots, referred to by various commands as "Tray n, Slot n" or linearly as slot 0-89 (Tray 2, Slot 1 is number 10; Tray 4, Slot 6 is number 35, and so on up to Tray 9, Slot 10 which is number 89).
- **Macro—**A custom command string, bound into a tray icon, that can be activated by clicking.
- **Insp(iration)—**The colorful buffs and power-ups in their own tray. Although powers and Insps are completely separate elements, many of the tools and techniques used for powers can also be used for Insps, which is why they're included here.

5.2 Tray Management

The power trays can be controlled with a few simple commands. The most basic is the window toggle, which will make the default tray set appear and disappear, while preserving its configuration (which of the three base trays are shown):

```
/tray
```

You can change which of nine virtual trays is displayed in each physical tray by clicking the arrows to each side of the tray number. This setting is 'sticky' and will remain through all other tray manipulations.

You can cycle through display of the primary, secondary and tertiary trays by clicking the arrow icon on top of the tray window.

You can open up to six more ‘floating’ trays by clicking the **+** icon on top of the default trays. (And close and configure these trays by right-clicking on them.)

There is a host of slash commands that can be used to execute these changes with binds or macros. Review the following commands in Reference A:

```
/next_tray…
/prev_tray…
/goto_tray…
/alttraysticky…
/shownewtray
```

☞ Avoid using the quirky **/altttray** and **/alttray2** commands until you carefully read their entries, and maybe not even then.

The vast number of players will only open the secondary and tertiary trays as their alt levels up and gains more powers to display; few players will ever change the default virtual tray numbers from 1-2-3. But advanced players can use tray changes to keep optimized powers at hand for, say, solo or teaming, combat or healing, etc.

The crucial thing to remember is that most binds, including both the default set and GABB, refer to tray powers by the **physical** tray, not the virtual numbered trays. If whacking **1** fires your lowest combat power in the primary tray/Tray 1 (the standard assignment)… changing the primary tray to another tray means the first power slot in that tray will be fired, not Tray1/Slot1. The same is true for binds that fire secondary and tertiary tray slots. Whatever virtual tray is assigned to that tray is what will be used.

Which leads to a very powerful concept, tray swapping.

5.3 Tray Swapping & Rollover

Let’s assume you have an alt that has two different gameplay modes, perhaps solo-optimized and team-optimized. You will want to have solo-friendly powers on tap for the first, and team-friendly and -supporting powers for the second. One way to do this easily is by swapping the primary power tray.

Set up virtual Tray 1 for your solo adventuring, probably with fewer things like holds and directed buffs and heals you can’t use yourself.

Now set up Tray 9 for teaming, with some of the combat powers replaced with buffs, holds, heals and the like.

You can manually swap the primary tray with one click of the tray-assignment arrows, or write the two following macros:

```
/macro TM "goto_tray 9"
/macro SLO "goto_tray 1"
```

and poke these up in your tertiary tray for convenient access. Click the first, and your tray is set up for teaming; the second, and you're back to being a one-alt army. You can of course swap both the primary and secondary trays, or even all three, and make other changes using the same macros.

A more sophisticated form of this tray swapping can be used with macro sets. Good examples are hard to create, but when you're a more advanced player or read other gameplay guides, this might make more sense.

Set up trays with power and gameplay macros, and when appropriate, make certain macros swap or "roll over" the same tray to another. You could set up two or three "rolling" trays so that, for example, 1 fires a low-level combat power and then rolls to another tray where 1 will fire a different, recharged, ready-to-go power. For alts with many attack powers, this is a way to stack all those powers on three or four useful buttons and not have to play the whole number line in every attack.

To use a fairly stupid example, assume these two macros:

```
/macro 1A "goto_tray 9$$powexecname Brawl"
/macro 1B "goto_tray 1$$powexecname Sands of Mu"
```

If you put the first in Slot 1 of virtual tray 1, and the second in Slot 1 of virtual tray 9, pressing **1** will fire the defined power, then roll the tray… at which point pressing **1** will fire the similar-level but recharged power (and roll the tray back to the first). Savvy users could have three-way rollover to put ten or twelve powers on three or four keys—and remember that you can duplicate power and macro icons between trays, so only a few icons might appear to change each time.

I'll leave it there. Use your imagination and search out other examples of well-crafted rolling tray setups.

5.4 Power Execution

Powers can be executed or 'fired' in several ways. The most straightforward, which many players use almost exclusively, is to just click their icons in the power trays. It works, but it makes the simple aspect of gameplay something of a speed and dexterity gauntlet… which maybe everyone doesn't find to be fun.

So there are a host of slash commands that can fire powers, which work best (frankly, work at all… if you don't have time to click an icon, you sure don't have time to type in a command!) in binds and macros.

☞ The executive summary is to go read the entries for the **/powexec…** family of commands in Reference A. Lots of them, lots of nuances, lots of options.

There are two basic commands for firing powers. The first calls a specific power by name:

```
/bind H "powexec_name Healing Flames"
```

The second, which is integral to most basic keybind sets (including the default and the GABB), is to call a power by what slot it's in:

```
/bind 1 "powexec_slot 1"
```

which fires the power in the primary tray, slot 1. (Tray slot numbers are one-based, 1-10, for these commands.)

Firing powers in the secondary and tertiary trays uses another command, which can also be used for the primary tray:

```
/bind CTRL+1 "powexec_tray 1 2"
```

fires the power in the first slot (1) of the secondary (2) tray. Note that (in another confusing quirk) the arguments are slot-tray, not the other, sensible way around, and that trays are numbered 1-9 as well. This is one of the few commands that works with tear-off trays.

☞ The variations between zero-based and one-based numbering for tray and slot commands will drive you mad. Get used to it and check the Reference A entry as needed.

These commands fire the specified power no matter what type it is, click or toggle. Click powers will be triggered (if recharged and available) or queued, with a red ring, (for recharge or target selection). Toggle powers will be… toggled, but with no specific control of the state.

☞ There is also a fourth tray that pops up when a pool travel power is activated and the short-range teleport feature is available, the "server tray." Powers in this tray can be activated, when it is visible, using the **`powexec_server_slot`** command.

This pop-up can be eliminated by talking to Null the Gull in Pocket D!

When you want to have more control over a toggle power (that is, turn it on when you want it on and off when you want it off), there are two more specific commands that can be used in binds or macros:

```
/bind U "powexec_toggleon Fire Shield"

/bind CTRL+U "powexec_toggleoff Fire Shield"
```

Pressing **`U`** any number of times will turn Fire Shield on. Pressing **`Ctrl-U`** any number of times will turn it off. This simple pairing of commands (or macro buttons) makes it foolproof to activate and deactivate powers, with no uncertainty about the starting state.

Finally, there's a fairly new power execution command that's so cool (and complex) it needs its own section.

5.5 Using the /powexec_location Command

This incredibly cool command was added to the I25 release. It allows the single-key/click execution of a power that typically requires a mouse click from a keybind or macro.

The **`powexec_location`** command takes two arguments:

```
/powexec_location target power_name
```

The **`target`** parameter substitutes for a mouse point-and-click and may be defined in several ways:

- To focus the power on yourself or the spot where you're standing, use **`me`** or **`self`**.
- To focus the power on a selected target (which may be anything targetable—friend, foe, pet or object—use **`target`**.
- To execute a power in a specific direction at a specific distance, use a **`direction:distance`** compound element:
  - **`direction`** may be any of six keywords: **`up`**, **`down`**, **`left`**, **`right`**, **`forward`** or **`back`**.
  - or, **`direction`** can be a numeric value in degrees, with **`0`** directly ahead, **`90`** to the right, etc.
  - or, **`direction`** can be specified as **`camera`**—the direction the view is currently pointing.
  - **`distance`** is numerically specified in world-feet. The keyword **`max`** may be used to specify the maximum range of the power.
- To execute a targeted power without a click, use **`cursor`**. The power will be executed at or towards the position of the cursor arrow.

The value for **`power`** is any valid power name that otherwise requires a mouse point and click to fire a targeting bulls-eye. It does not need to be enclosed in quotes unless you prefer to for clarity.

Some examples:

- **`powexec_location me Fire Imps`** will summon your Fire Imps at your location. (A cool variation for pets is to use **`0:max`**, which will cast them a bit further than your normal range and make them come scampering back, which is exactly where you want them in combat.)
- **`powexect_location target Tar Patch`** will enmesh the targeted enemy in a tar patch.
- **`powexec_location up:100 Teleport`** will teleport you up 100 feet.
  (This, used with either a numeric value or **`max`**, could be a great escape power when you're about to be overwhelmed by foes. For teleporters, anyway.)
- **`powexec_location 45:20 Recall Friend`** will teleport any selected teammate to a spot 50 feet in front of you and 45 degrees to the right. (A rolling bind could be used to vary the spot for serial use.)
- **`powexec_location camera:max`** Teleport will teleport you your maximum teleport distance in the exact direction the view is looking, including elevation. (More or less the default teleport action.)
- **`powexec_location cursor Teleport`** will teleport you to where the mouse arrow cursor is located, without any need for a click.

There are a number of limitations with this command, mostly obvious ones related to the power and target being specified. If the combination won't work as a normal command, it won't work in this way, either. If you don't have the specified power, the command will do nothing. If you have no target selected for a target location, the command will do nothing.

More and better examples and suggestions solicited! Think creatively, experiment and give a good evil laugh over all the possibilities this new power method offers.

Thanks to Korbian of Titan Network for providing me with the information from the I25 Release Notes!

5.6 Inspiration Tray Management

Powers and Inspirations ("skittles") are completely different things, but both live in trays and both have a complete set of slash commands to activate them.

Inspirations get a little complicated in that there are three levels of each one, with increasing power and effect. So if you want to fire a blue Endurance booster, you have to accommodate three possible names for it. It's easier than it sounds.

As for power trays and slots, review the following commands in Reference A:

```
/inspexec_name inspname
/inspexec_slot column
/inspexec_tray row column
```

The first command simply fires an Inspiration by name—say, Respite for a tier-one health boost:

```
/bind F1 inspexec_name Respite
```

Pressing **F1** will fire a Respite no matter where in the Insp tray one might be. It's hard to imagine any need to use any other command, but the other two allow you to fire an Inspiration from a particular column of the Insp tray (column 0-5, depending on your level and how many columns you have), and then from a specific slot in the whole tray by column and row. Since this requires you to keep your Insps sorted out according to a specific plan, I recommend not using these location-based commands at all.

☞ If you insist, Insp tray numbering is 1-based, with "1 1" being the lower left slot.

☞ For a complete set of binds to fire all types of Insps, including all tiers of each (and thus a list of those names), see the "QuickInsp" bind in Section 7. This system is implemented in the GABB.

5.7 Pet Inspirations

If you have an alt with pets, it's possible to drop Insps on pets who need them. For a more sophisticated approach, there are two slash commands that will do it for you:

```
/inspexec_pet_target inspname
/inspexec_pet_name inspname petname
```

The first will drop an Insp of ***inspname*** on the pet you have targeted.

The second will drop an Insp of ***inspname*** on the pet whose name is specified.

A little more about this in Section 6.

6. Pet Control

If you are a Controller of level 32 or more, or a Mastermind, you have the joy and pleasure of commanding your own team of minions or pets. This can get extremely complicated as the number of pets increases, their powers expand and combat gets crazy.

Alhough there is a configurable pet window that lets you do most pet management and command using clicks, this can take far too much time and attention while your pack is getting mauled by a mob of foes.

Slash commands and binds to the rescue!

From the many plaintive requests on the Help channel, novice pet owners seem to lose their Pet window a lot. The quick way to get it back is to type **`/pet`**. A bind like Alt-P can make this easier.

Note that there are two modes for the Pet window, selectable by right-clicking on it.

6.1 Naming Pets

Unless you're a really despicable pet owner, you will probably want to name your pets both as part of RP and theme, and for some convenience in controlling them.

You can rename pets by right-clicking on their line in the Pet window and entering a name. The name will persist until changed; you don't have to rename pets each session or spawn.

Pet names are subject to the same copyright and profanity rules as usernames, so if you can't name your robot Iron Man, that's why.

You can also use the **`/petrename`** ***`newname`*** command on any selected pet.

I have found it useful to name pets in their window order: **`1-Big Guy`**, **`2-Middle Guy`**, etc. This helps with quick command and buff binds, as you'll see. It also **ahem** can get past naming restrictions, but you didn't read that here.

You can reset all pet names to default with **`/clear_petnames`**.

6.2 Spawning & Buffing Pets

Pets are spawned (or respawned) by calling their associated power, all of which do have a notable End drain and recharge time. You can simply click the pet power icon, but pet spawning is also a location power, so you then have to click a spot where you want them to rise. You can use binds to speed up the process:

```
/bind P "powexec_location 350:5 Zombie Horde"
/bind CTRL+P "powexec_location 10:5 Grave Knight"
/bind ALT+P "powexec_location 0:5 Lich"
```

Three quick taps, not too quickly to allow for activation delays, and your army appears before you in a combat-ready arc.

Each time you spawn a new or replacement pet, you will need to throw or update the basic pet buff:

```
/bind O "powexecname Enchant Undead"
```

Tap twice to target the nearest pet and fire. (This buff in each set is applied to all living pets.)

The second buff in most pet power sets is directed to one and only one pet at any time, so you will usually want to cast it on your most powerful miniou:

```
/bind CTRL+O "powexecname Dark Power$$target_custom_near mypet Lich"
```

...which will target your Lich (or equivalent) and then cast the buff.

(Why yes, I do run Zombies. Why do you ask?)

6.3 Controlling Pets

Controlling how your pets behave in both travel/search and combat situations can be fairly simple, or as complicated as a late-stage turn in D&D. It's up to you. But this is a place where binds make a huge difference in ease (and pleasure) of play. It's a perfect use for the numpad keys, which are largely useless for other archetypes.

☞ Bind all of your combat pet controls to layers on the numpad 1-6 keys.

Generally, you will want your pets to take one of three "stances": **`aggressive/attack`**, **`follow/defensive`** and **`stay/passive`**. You can use these control keywords in other combinations but these three are a good starting point:

```
/bind NUMPADENTER "petcom_all aggressive attack"
/bind DECIMAL "petcom_all defensive follow"
/bind CTRL+DECIMAL "petcom_all passive stay"
```

(**`decimal`** is the period/delete key on the numpad.)

And most novice/journeyman pet controllers will simply want all their pets to follow the same commands. With mastery, you might want to create binds that direct each set to do individual stances; your Lich to attack while your Zombies protect you, for example.

All pet power sets include a basic heal, somewhere. To be able to rapidly fire a heal at a pet, use the following group of binds:

```
/bind NUMPAD1 "powexec [heal power name]$$target_custom_next mypet [pet 1 name]"
                                  ...
/bind NUMPAD6 "powexec [heal power name]$$target_custom_next mypet [pet 6 name]"
```

Two taps of any of these keys will fire the heal at the selected pet; watching the pet window to see which one needs the help the most is a basic skill of Masterminding.

If you have a secondary heal, such as that from the Medicine pool power, bind it to the same keys using **CTRL+NUMPAD*n***.

And in most cases, you will want a third option: firing a green health skittle at a pet in need. The commands include this option:

```
/bind ALT+NUMPAD1 "inspexec_pet_name [pet 1 name] Respite"
                        ...
```

This command can be adapted to any other Insp type, and the compound string from "QuickInsp" in Section 8 can be used to ensure any available skittle of the desired type is used.

The very useful followup to all of the above is to bind each variation to the same level of the NUMPAD0 key that executes the same heal, buff or other power on the currently selected pet or teammate:

```
/bind NUMPAD "powexec [heal power name]"
```

…and so forth.

And finally, the above keys will often leave a heal or buff queued, and you'll waste the fire on a pet that doesn't need it. I bind all three of the unused numpad keys to cancel a queued power, so that it's quick and easy to null out any such power before it fires again:

```
/bind NUMPAD7 "powexec_unqueue"
```

Repeat for keys 8 and 9. Ya ain't using 'em anyway.

Targeting & Pets

If you use any general targeting commands (such as the Master Targeting or Name String Targeting from Section 8 or in the GABB), some fine tuning is in order for for pet controllers. First, you can create targeting binds that select only your pets:

```
/bind CTRL+ADD "target_custom_next mypet"
```

and (highly recommended) exclude your pets from most general targeting and search:

```
/bind ADD "target_custom_next base notmypet"
```

☞ It's really irritating to have targeting attempts keep sticking on your darned pets, which are always getting in the field of view.

6.4 Ve Haff Vays to Make You Talk, Lassie

Yep, you can make your pets chat and even emote. There's a set of "say" commands that follow the control command pattern:

```
/petsay string
/petsay_all string
/petsay_name pet_name string
/petsay_pow power_name string
```

All commands work the same. The first uses the targeted or selected pet; the second uses all cast or living pets; the third uses the pet specified by **`pet_name`**, and the last uses all pets cast by a specific **`power_name`** (such as "Zombie Horde" or "Lich").

The **`string`** is everything that follows, to the end of the command (line end or **`$$`** separator).

And yes, you can include emotes for pets; but there are a few quirks and bugs.

Using Emotes in `/petsay` Commands

When making pets say things and emote, there are three rules:

1. Quotes are recommended and often needed around the "say" commands to make sure the command is parsed correctly with a random string of words attached.
2. **Petsay emotes must be enclosed in <angle brackets>.** They will rarely (never?) work otherwise.
3. Don't use emotes in pet-say binds. They don't work right now.

Because of a bug in string processing (which is slated to be fixed, someday), pet emotes do not work in binds. This bug persists as of Issue 27, so a fix may not come soon.

Emotes for pets work fine in direct entry, and from macros, but not in binds. Various workarounds such as loading the binds from a text file don't work (either/any more).

The *proven* workaround is to write the commands to macros, then call the macros with binds:

```
/macro_slot 80 PD "petsayall <emote DrumDance>"

/bind ALT+1 "powexec_tray 1 9"
```

…creates the macro in Tray 9, Slot 1 and calls it from there. (Note slot-first numbering.) You can use both regular **`macro`** and **`macro_image`** commands and move things around manually, too.

☞ Not all pets will do all emotes; sometimes even not all pets of a single type will respond to an emote command. Test each class of pet with each emote before committing to a complex bind or macro.

6.5 Disposing of Pets

Pets will follow you across zones and in and out of missions. There are times, though, that it's much more convenient to just get rid of them for a while.

`/releasepets` will dismiss all your living pets instantly.

`/petcom_all dismiss` will dismiss all your living pets, but their carcasses will remain present for a time, which can be useful if you wish to use the rez or "zombie warrior" power on them.

7. Popmenus

CoX has a very cool feature that was developed mainly for, well, developers, but is accessible to players with a little skill: popmenus. You can write your own custom menus that will “pop up” at the cursor location. It’s a bit advanced unless you have some of this bind, macro, game-file experience under your utility belt, but not as hard as anything like actual programming.

☞ And before I can go any further, I have to say that every time I see a reference to popmenus, and throughout writing this section, I can’t help but hum that stupid song... *“Pop! Pop… Pop-menus! Pop! Pop… Pop-menus!”* Enjoy your earworm, now, while we... *“Talk about… Pop-menus!”*

7.1 Popmenu Fundamentals

Because they’re a little weird and work entirely “under the hood” of the game, here’s a summary of what you’re facing:

» Just like bindfiles, popmenus use ASCII text files. Don’t edit them in Word or such, use Notepad++ or a similar all-ASCII programmer’s editor. As with all game files, avoid “curly quotes” like these at all costs.

» Popmenu files must have the **`.MNU`** or **`.mnu`** extension.

» Popmenus must be placed in a specific game folder, which you may have to create.

  » The good part: that’s all you have to do to install them. If they’re there, the game loads them. And they are thereafter accessible by all alts.

  » The bad part: popmenus are only loaded at game start. To load new ones or edits, you have to restart the game from scratch. That means desktop, not the login or character screen. Gets tedious.

  » The file location is **`[GAME ROOT]\data\texts\english\menus`**. You will probably have to create everything under **`\data`**.

» Popmenu files have a highly defined structure; if you get anything wrong, the menu won’t work right.

  » Make sure to match curly braces around groups of items.

  » **The file must have at least one carriage return** (CR) **character before the first Menu keyword** or it will not load/function. Get in the habit of putting a comment line with the menu function at the beginning so as not to get bitten by this quirk. (And don’t ask me how I know.)

  » Any power or function name with spaces can substitute underscores for the spaces to avoid parsing errors: e.g. **`Speed of Sound`** should be used as **`Speed_of_Sound`** in menu files.

  » Command strings generally follow the format for bind and macro definitions, including using **`$$`** to join commands in a string, but there are many limitations on how menus parse commands.

Fair warning: Some mistakes in a popmenu definition file can crash the game client, usually when the faulty menu or command is accessed. Test new menus and changes in a safe location, so that crash-to-report-screen doesn’t mess up your alt’s day.

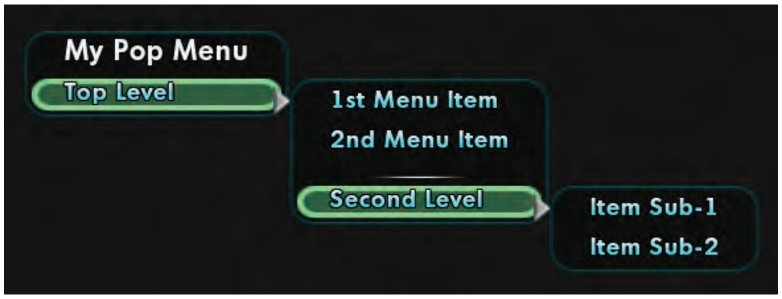
Most such crashes are caused by undefined actions (like a Menu with a null argument). Most simple coding errors just mean the menu doesn’t work right.

7.2 Basic Popmenu Structure

A basic popmenu file will look familar to anyone who's done a little programming or scripting:

```
// This is a comment line. You can use them anywhere. Putting one
// at the head of the file is a good practice (see Fundamentals!)
Menu "NameOfPopMenu"
{
Title "My Pop Menu"
    Menu "Top Level"
    {
          Option "1st Menu Item" "nop"
          Option "2nd Menu Item" "nop"
          Divider
          Menu "Second Level"
          {
                Option "Item Sub-1" "nop"
                Option "Item Sub-2" "nop"
          }
    }
}
```

And here's what that (useless) example looks like in the game, after being saved to the menu folder under the name **`DumbTest.mnu`**:



A minute or two of comparing the code and the image should make things clear. Pictures being worth 1,000 words and all that.

Let's go through it line by line:

» **`Menu "NameOfPopMenu"`** defines the internal name of the menu that you will use to call it up. It can be anything, as it's never displayed to the player. It cannot have spaces. In this case, the command **`/popmenu NameOfPopMenu`** (either directly or as part of a bind or macro) makes this menu appear at the cursor position.

As with all the menu elements, everything that follows this menu name must be enclosed in curly braces.

» **`Title "My Pop Menu"`** gives the menu its heading/name, as displayed. It should be short. As with nearly all game scripting, any spaces in the name mean the quotes are required, but they can be omitted for a one-word title. It's good practice to always use them anyway. There can be only one **`Title`** per popmenu definition.

- **`Menu "Top Level"`** defines the start of the first menu. Again, everything in this menu must be contained between curly brackets.

☞ Note that you can skip a second menu definition of any kind, and just fill the base menu definition with menu options. This is good for simple, one-level menus.

- **`Option "1st Menu Item" "nop"`** defines the first actual menu action item. Each such line must contain the keyword **`Option`**, a word or string defining the menu label that will be displayed, and a command string. In this example, the command is the do-nothing **nop**.

☞ You can include as many items in each menu as you like, but it's good practice for convenience of use and to avoid display problems to keep it to no more than a dozen (if that many), and use submenus to hold more items.

- **`Menu "Second Level"`** defines the start of a submenu. This will appear as one menu item until it's hovered over, at which point the entire submenu will be displayed. Submenus (keyword, name and pair of curly braces) must be fully enclosed in each menu they are to be displayed as part of.

☞ There is probably no reasonable limit to the number of submenus that can be nested, but confusion and display limits mean you should keep it to a few (two or three) and consider the menu organization carefully.

And that's it, except for redundancy of the commands and the careful use of closing curly braces on each menu element. Keeping these braces properly matched and nested is what leads to most menu problems.

While you can use almost any structure for the text (even running it all together on one line, I think—haven't tested!—or flat against the left margin, but the tabbed structure will help a lot in keeping the nesting, braces and code tidy (as anyone with a little code experience knows).

Oh, and **`divider`**? Does just what you see: puts a little white divider line in the menu. You can use them anywhere to keep command groups and submenus visually separated.

☞ I find it useful to put a marker character on submenus, such as **—Submenu** (that's an em dash). You may or may not.

7.3 Popmenu Command Strings

Unfortunately, writing the actual command strings for each menu item can be a headache. The menu parsing system preserves a lot of the old chat/command line faults from the earliest days of the game, and so strings that work fine for binds, bindfiles and macros often break or do unexpected things in a menu.

The biggest problem is that the menu parser wants one and only one element for the command, so anything more than one word (such as the **nop** used here) has to be in quotes. (Yes, the quotes could have been omitted… but I strongly recommend the good practice of always using them.)

All of which translates to that if you have a complex command with multiple instances of quote-wrapped elements… it may not work without a lot of tinkering, or maybe at all. Remember that you can replace spaces in command names with underscores to keep the 'grouping' issues simple.

I won't go into any further examples or workarounds at this time. If you are going to write popmenus with commands that need complex argument strings, know that you're going to have to tinker and possibly settle for less.

☞ One of the worst limitations of popmenu commands is that almost **no form of keybinding works**. You can't, for example, write a menu that will rebind your Teleport or Heal key two or three different ways. What's frustrating is that it *sort of* works… but the change won't register until you log out and return! I'm not sure about things like rewriting macros on the fly, but as macros (especially **`macro_image`** ones) require multiple arguments, I'm not sure that can be made to work, either.

7.4 Useful Popmenu Examples

I plan to add more cookbook examples you can cut, paste and use, but here's a couple of simple examples that further show how to write popmenus.

Set VisScale

This useful example is different in that it does not even have one sub-menu. The popmenu will appear with just the title and the three options, which is preferable for most simple, one-level menus.

```
// Set visscale to optimum values - low for performance,
// middling for open-world play and high for missions.
Menu "SetVisScale"
{
Title "Set VisScale"
    // Numeric values may be adjusted from 0 to 20 or more to
    // suit player preference and system capability. Higher
    // settings have HUGE impact on framerate!
    Option "VisScale Low (2)" "visscale 2"
    Option "VisScale Med (5)" "visscale 5"
    Option "VisScale Hi (10)" "visscale 10"
}
```

Dance, Chat, Emote…

The regular chat-bubble menu has almost every emote available for a search-and-click, but I find the tiny button and the vast menus a bit difficult to use, and will just whack out an "**`\em dance6`**" on the rare occasions I feel like dancin'. You can use popmenu to create your own customized (and brief!) emote and chat menu if you like:

```
// Custom Chat & Emotes - change strings as you like!
Menu "MyChatEmotes"
{
```

```
Title “Chat & Emotes”
    Option "White Guy" "em dance1"
    Option "Robot" "em dance4"
    Option "Air Guitar" "em dance9"
Divider
    Option “Bow” “em bow”
    Option “Salute” “em salute”
    Option “Laugh” “em evillaugh”
Divider
    Option "Wooo!" "l Woooooo!"
    Option "We Win!" "g We Win Again!"
    Option “Next Round" "l Next round’s on me!"
}
```

BadgeDRADIS

If you’d like to see a more complex example, download Gunner’s BadgeDRADIS from the ***HEROICA!*** website (see first page). It uses five popmenu files, six custom macros and three keybind files to implement a sophisticated badge-search tool, but it’s way too long to include here.

7.5 Advanced Popmenu Features

There is one more popmenu feature that can be used to limit menu access and content depending on game parameters such as having a certain power, badge, level, archetype etc. This function is perhaps more useful from a Dev/GM perspective so that things like the new Fast Travel menu “power” can be hidden until the player earns/needs it. It was also more useful back in the Live era when Devs needed to restrict access to game features that were only unlocked by add-on packages. Since all “add ons” are now bundled into the basic Homecoming game, there’s no real need to lock off any of these emotes, badges, powers etc.

But this function might be useful in some player-written menus to, say, optimize a UI menu to different alts or only enable certain features when a badge or accolade is achieved. (The alternative would be having to custom-write and re-write a menu for every circumstance, which would get tedious.) So a summary:

```
LockedOption
{
          DisplayName OptionName
          Command Command
          Authbit AuthIdentifiers
          Badge BadgeIdentifiers
          RewardToken RewardIdentifiers
          StoreProduct ProductIdentifiers
}
```

» **`LockedOption`** is the basic keyword; if it’s included, creates a menu item that will be grayed out and inaccessible unless one of the unlocking options is valid.

- **DisplayName** and **Command** must be defined on separate lines instead of inline with the base keyword (as for **Option**).
- At least one locking option must register as valid or 'true' for the locked option to be active and allow the **Command** to be executed:
  - **Authbit** uses an authorization code to enable features based on whether the player has one of the add-on packs for the original game. Essentially obsolete under Homecoming.
  - **Badge** checks to see if the alt has the specfied badge before unlocking the option. The list of badge names is specialized and extensive. This may be the only locking option useful in the present day.

You must use the internal badge name from the list as the parameter. For the Kings Row "Smokey" badge, for example, you would use just `KingsRowTour5`.

Props to AboveTheChemist for sorting out this frustrating glitch for me!

  - **RewardToken** checks to see if the alt has the specified reward option (mostly emotes unlocked by missions or purchased; again, this function is largely obsolete under Homecoming).
  - **StoreProduct** is another check to see if the alt has earned or purchased a specific capability, largely emotes and costume-change emotes. Obsolete under Homecoming.

A complete description of each locking option and complete lists of the 'unlock codes' for each can be found on the Homecoming wiki, at:

- `https://hcwiki.cityofheroes.dev/wiki/Popmenu_(Slash_Command)`

Sorry; other than the badge option, they're all of too-little use and too-great content length to include here. The badge name file alone is almost 300 items long. I've avoided pushing Guide users to other resources, but this is a place I gotta.

Locked Menu Example—Badge Ownership

Here's a simple example of how to include a locked menu item:

```
Menu "LetsLockSomething"
{
Title "Locked Stuff"
    Option "Dance!" "em dance3"
    LockedOption
    {
          DisplayName "Cheer!"
          Command "em victory"
          Badge "GoldClub"
    }
}
```

This menu would let you dance all you want, but cheer only if you had acquired the "Pocket D Gold VIP Member" badge (note again, badge code only: **GoldClub**) by hanging out there for an hour.

Feedback, tips and comments on popmenu stuff solicited!

8. Gunner's Targeting Secrets

Targeting in CoX is an essential adjunct to your character's eyesight—a bionic eye to help spot those pesky glowies, bosses, hostages and friendlies across vast and confusing outdoor maps.

If you've played very long, you've gotten an outdoor mission that you had to search and search to find the objectives... and you haven't played much longer if you've run into one in which the objectives remain stubbornly hidden, usually as the clock ticks down and your patience frays.

Gunner to the rescue: Here's how to use the advanced targeting commands to make those hidden suckers come out and play. As well as streamline more common needs like finding and locking onto the right foe.

8.1 Basic Targeting Commands

Okay, you probably know the targeting that's been in the game since Issue 1:

» **`target_enemy_near`**

» **`target_enemy_far`**

» **`target_enemy_next`**

» **`target_enemy_prev`**

These commands, which take no arguments, will target any foe in your visible range (about 180 degrees wide and either at map limit or about 300 yards) who is, respectively, the closest, farthest, next farthest from the one currently targeted, or next closer from the current target. The first two will select only one target at any one time, while the second two will cycle through the visible foes, one in nearest to farthest order and the other the other way around.

You can do the same thing for friendlies:

» **`target_friend_near`**

» **`target_friend_far`**

» **`target_friend_next`**

» **`target_friend_prev`**

Which does the same thing as above for any player or NPC that shows a blue or green reticle.

None of these commands will let you target objects or NPCs with a white reticle. Fortunately, an enhanced targeting system was added early on, around Issue 4 or 5.

8.2 Custom Targeting Commands

The following custom targeting options were added to expand the selectivity of the power:

- `target_custom_near`
- `target_custom_far`
- `target_custom_next`
- `target_custom_prev`

These commands work as described above with the exception that each requires one or more arguments to tell it what to target. The arguments are:

- `friend`
- `enemy`
- `mypet`
- `notmypet`
- `base`
- `notbase`
- `alive`
- `notalive` / `defeated`
- `teammate`
- `notteammate`

Some of these options are identical to the fixed targeting equivalents:

```
target_custom_near friend
```

is identical to

```
target_friend_near
```

and

```
target_custom_next enemy
```

is identical to

```
target_enemy_next
```

...and so forth. There isn't really any reason to use these longer commands in place of the fixed ones except individual preference. But you could eliminate your use of the older commands to use the more consistent and flexible custom commands all around.

☞ It is important to understand that the "custom" commands begin with an assumed "target all" and are restricted to smaller sets of targetable items by the various commands. This might seem trivial but it helps in understanding how the keywords interact and stack.

☞ The custom targeting commands REQUIRE at least one keyword. They won't work by themselves.

Custom Targeting Keywords

It's these argument keywords that add new functionality, but the actual operation of these commands and their keywords is not completely straightforward. There is a hierarchy to the commands and some co-dependency that is a bit muddy. Here, to the best of my knowledge, is an accurate description of the keyword functions:

- **`friend`** will restrict targeting to any blue-reticle (other player) or green-reticle (teammate) character.
- **`enemy`** will restrict targeting to any orange-reticle (foe) character.
- **`teammate`** will restrict targeting to any green-reticle (teammate) character, including both your and others' pets.
- **`notteammate`** will exclude all green-reticle characters from the targeting cycle.
- **`mypet`** will restrict targeting to any of your own pets.
- **`notmypet`** will exclude any of your own pets from the targeting cycle
- **`notalive`** (or **`defeated`**) will restrict targeting to any friend or enemy with zero hit points.
- **`alive`** will restrict targeting to any figure, friend, enemy or NPC with at least one hit point.

Finally, there are:

- **`base`**
- **`notbase`**

These commands are… peculiar. Both appear to function identically, for one thing, but what they do is open targeting to every single live object within view. Instead of being limited to live game elements like friends, foes and pets, using base allows you to target civilians, neutrals, NPCs, and even objects like doors, glowies and terminals.

Unfortunately, there doesn't seem to be any good way to make this selection selective; you either target all objects or none. But they can still be used to great, useful and even amusing effect.

Argument Stacking

To find friends, enemies or pet, those keywords have to be included. Again, they duplicate other fixed commands, but the rest of the keywords can add new capabilities. You can add the `alive` or `defeated/notalive` keywords to make the targeting more selective. For example, a character with a rez power could make good use of a bind that targets a defeated teammate. There may be other reasons to select pets, or even living teammates only.

So, the custom targeting commands permit more than one argument to be stacked - such as:

```
target_custom_next friend alive
```

which will target only blue- and green-reticle characters who have at least one hit point.

```
target_custom_near enemy alive
```

will target orange-reticle figures who have 1 hit point or more.

```
target_custom_near enemy defeated
```

will target enemies who have zero hit points.

Actually, those examples are backwards from any useful ones, so let's flip them around:

```
target_custom_near teammate defeated
```

will target the nearest teammate who's defeated and needs rez or tp out of the battle.

```
target_custom_near enemy alive
```

will target only enemies who have not been defeated - which would be useful for a grapple bind written with the custom targeting commands, since there's no point in a scrapper locking on to a defeated foe.

String Targeting

The final argument that the custom targeting commands will accept is strings—any character name or part of a name. It's almost impossible to overstate how useful this option can be.

Unfortunately, this won't work with foe group names or titles, so you can't seach for "Family" or "boss," for example... wouldn't THAT be nice! However, if you're on a hunt for specific types of enemy—such as that damnable hunt for Marcone Capos to get the Gangbuster badge - you can write a quick bind with the appropriate string and greatly simplify your hunting:

```
target_custom_next enemy capo
```

If you're searching for more than one exact character name, you'll have to analyze the spread of names for each foe type to see if there's a substring that will cover them all.

Any unique substring will work as well as the full name string.

All of the following are valid binds:

```
target_custom_next enemy alive sorc (Tsoo Sorcerers)
target_custom_next enemy alive outcast (Any Outcast minion)
target_custom_next enemy alive lead (Outcast Lieutenant or Boss)
```

...etc. Have fun.

The problem is that customizing this search can be time-consuming, or macro-button consuming. There's a fast workaround, though, and wouldn't you know it, it involves binds:

```
/bind SUBTRACT "targetcustomnext base alive Sorc"
/bind CTRL+SUBTRACT "beginchat /bind SUBTRACT targetcustomnext base alive "
```

…taht almost automate the process. When you need to search for a specific enemy type, hit **`Ctrl+-Subtract`** (the **–** key on the numpad) and the chat entry line will fill with the latter string. All you have to do is type the enemy identifier, or some part of it (as in the "Sorc" for Tsoo Sorcerer" of the preloaded bind). Then, whacking **`Subtract`** will hunt for that type. And it can be reset in a few moments of typing a new search name.

Note the space at the end of the chat string… saves having to type it in every time.

Note that this works for non-foes; in testing the strings, I searched for Christmas gifts, which are named "Mysterious Gift"… and "**`myst`**" found them as fast as I could turn and look in each direction. The **`base`** and **`alive`** keywords are generally recommended, since you will rarely want to find defeated foes, but all of the above custom targeting keywords can be used in the preset bind.

Cool, eh? Thanks to Veelectric Boogaloo on the Homecoming Forum for spelling out this great trick.

Now let's put it together.

8.3 Advanced Targeting

In a more general application than the string targeting above, here's how to use the custom targeting command to simplify those damned hunting missions, whether they're kill-alls, hostage rescues, glowie hunts or any other mish that requires you to laboriously search the whole darned map.

Put this "Master Targeting" bind on a key you can whack continuously while manipulating the mouse and movement keys.:

```
/bind ADD "targetcustomnext base"
```

I specify the numpad **+** key (and include it in the GABB) because I can whack it while my hand is still on the trackball. (Yeah, I use a trackball, what's it to ya?)

You will need to be able to move, control air movement and whack this key, so if another key works better for your, use it instead.

Now, when you're in a map that requires searching, or hunting in a zone...

1. **(Optional)** make your Target window more visible by moving it to a more central location and/or making it larger with the **`/windowcale target`** ***`size`*** command.
2. Get to a good central place (among obstacles) or a high place (through jump, teleport or flight).
3. Spin slowly while whacking the master targeting key. Watch the target window carefully. When you see your desired target, freeze and cycle the targeting slowly until you have it targeted.
4. Do whatever heroic or villainous thing you must.
5. Repeat steps 2 through 4 as necessary throughout the map.
6. (Alternate) Use the name-targeting command in the same way, with a string that identifies what you're looking for. (Such as "hostage" if all the scared little guys have the same name.)

9. Gunner’s Way-Cool Binds List

Over the years, I’ve found, learned and created a whole bunch of useful binds. If you’ve read through to this point, you’ve already seen dozens. This section is a compendium of those that many players might find useful.

Each uses a specific key that maps to my preferences—you’re of course free to use others.

☞ For a completely new bindset updated to modern gameplay, see the companion GABB—Gunner’s Advanced Basic Bindfile, on the same web page whereyou got this guide. Only a few samples from it are in this list; there’s much, much more!

☞ And for more, see the separate GWCBL page on the website, which I am updating more regularly than this Section.

Contributions welcome and will be credited!

Enjoy… and *to victory!* (I miss Victory...)

9.1 General Binds

Boss!

```
/bind CTRL+F9 "quit$$dialog_yes"
```

Bang, you’re at the desktop when YOUR boss walks in. Be sure your char is in a safe place, though…

☞ Frankly, I stopped using this, making the bind only “quit” and requiring a second keypress, after I accidentally banged out of the game at inappropriate times.

While you’re at it, add these:

```
/bind F9 "requestexitmission"
/bind CTRL+F9 "quittocharacterselect"
/bind ALT+F9 "quittologin"
```

The first bind exits you from a completed mission—a useful alternative to finding and clicking the teeny EXIT button. It also gives you a fast exit when you’re just doing XP mop-up and the situation turns ugly. (Also a nice insurance key if you decide to see if you really can solo a purple Aberrant...)

Last two functions should be obvious. All will give you an abort time that, unlike the first, can’t be bypassed.

Zoom!

Make faster travel easier and reduce endurance cost when necessary.

```
/bind R "powexec_toggleon Sprint$$++autorun"
/bind CTRL+R "powexec_toggleoff$Sprint$$autorun 0"
/bind SPACE "+up$$autorun 0"
/bind MOUSECHORD "+up"
```

The first bind turns on Sprint and initiates autorun on the first keypress, and will toggle autorun off and back on with successive presses. (If you have Super Speed or are using one of the prestige sprints, substitute that power for Sprint.)

Second bind cancels speed power and autorun.

Third bind cancels autorun while still providing a quick jump key. (Hitting **back** (**S**) will also halt autorun.)

The fourth bind gives jump action when both mouse keys are pressed. If you have initiated speed autorun with the first bind, you can steer and jump obstacles with just the mouse hand.

An alternative to the second bind that lets you switch off Super Speed, Super Jump and other travel powers that don't work well in missions is:

```
/bind MBUTTON "powexec_toggleoff Super [whatever]$$powexecname Sprint$$autorun 0"
```

Clicking the middle mouse button will now toggle off your speed power, cancel autorun and (with as many clicks as you like) toggle Sprint (or your other basic 'run' power) on and off. Great for those fraggin' cave missions.

If you have a mouse with extra buttons (like most newer Logitechs and many gaming mice), use these binds for the two little buttons to the left:

```
/bind BUTTON4 "emote flypose1$$autorun 1$$powexectoggleon Fly$$+up"
/bind BUTTON5 "autorun 0$$up 0$$powexectoggleoff Fly"
```

Basically, bind your commands that add up to "go like hell" to the button that is **harder** to press, and the "whoa, stop!" commands to the one that's easier. You can zoom off with just a bit of effort, and stop/drop on a dime, with a little practice. If you use a flying emote, it will take another press or two of the button to activate all the items in the command string.

There is currently a bug when `+up` is used in bind strings: up will latch on as if the command was `up 1` or `++up`. That "jump up" at the beginning of the commands is a nice touch, though, and keeps flying alts from getting snagged on ground obstacles. Simply hit any up button (`MouseChord` or Space) to stop the rise. Until the bug is fixed.

Beam Me… Over There, Scotty!

There are a number of binds that make Teleport powers much faster and easier to use. The most basic, which turns Teleport into a one-hand point-and-click travel power, is:

```
/bind LeftDoubleClick "powexecname Teleport"
```

…and travel with any succession of point-and-doubleclicks.

The companion bind is to make Recall Friend (Teleport Teammate) quick and easy:

```
/bind CTRL+LeftDoubleClick "powexecname Recall Friend"
```

This has a slight limitation in that the range limit for Recall Friend is quite short, and if you click at a point outside that range, you'll get a red targeting ring that requires repositioning and another click. So you could use this alternate keybind version:

```
/bind CTRL+G "powexec_location 0:20 Recall Friend"
```

…which will TP your teammate to a spot just in front of you, and the pointing action will be irrelevant.

To prevent confusion and allow the teleported one to opt out, throw a team chat message in there, on the keybind or this macro version:

```
/macro TTM "powexec_location 0:20 Recall Friend$$g Teleporting $target!"
```

The string "Teleporting [teammate name]" will appear on the chat and you're done; the teammate can accept or reject the 'port as they like.

Another useful and amusing bind for teleporter is this one:

```
/bind U "powexec_location up:max Teleport"
```

Punch U (or the key of your choice) and your alt teleports vertically at his or her maximum range. Useful to bounce quickly out of a bad combat situation, or to jump way high to start cross-zone travel past buildings, cliffs, etc.

Combat Teleport

One of the most amazing additions of I27 was the short-range, lightning-fast Combat Teleport. If you can get it, you want it. 'Nuf sed.

However, managing Combat Teleport and Teleport can be challenging. I have long used the LeftDoubleClick bind for Teleport, which makes travel and general translocation a snap. But compromising the speed of Combat Teleport by, say, putting it on Ctrl-LeftDoubleClick makes it harder to use in busy situations than is good. And as I found, putting CT on double-click and using Ctrl for longer-range teleport was terribly confusing, especially when a travel attempt only snapped me a hundred feet or so.

So I am now using this macro pair to manage the TP powers:

```
/macro TP "bind LeftDoubleClick powexecname Teleport$$bind CTRL+LeftDoubleClick //
powexecname Combat Teleport"
```

```
/macro CTP "bind LeftDoubleClick powexecname Combat Teleport$$bind //
CTRL+LeftDoubleClick powexecname Teleport"
```

Click the **TP** macro button, and TP is on the primary click and CTP on the secondary. Travel is easy and CTP is still readily available.

Click the **CTP** macro button, and CTP is the primary in missions and combat, where you probably won't need TP anyway (but it's right there if needed).

Ta da.

I said FROG!

Super Jumper or other jump power? Use this set:

```
/bind J "powexec_toggleon Super Jump"
/bind K "powexec_toggleon Combat Jumping"
/bind CTRL+J "powexectoggleon Super Jump$$up 1$$autorun 1"
```

The first two binds give single-key start of jump powers. Since the powers are mutually exclusive, they will toggle each other. The third bind is a variation of the Zoom! bind that sets you jumping across the zone; you can steer with the mouse. Use the **SPACE** bind above to cancel forward travel.

And while we're here, I bind Fly to the Y key for convenience:

```
/bind Y "powexecname Fly"
```

But... if you've already bound Fly (or any of its equivalents) to the small mouse buttons as above, you can use **Y** for a variation of the teleport escape key:

```
/bind Y "powexec_toggleon Fly$$up 1"
```

…which will zoom you vertically until you tap an **up** key to cancel the rise.

9.2 Combat Binds

Follow!

```
/bind F "follow"
```

A default bind, but worth mentioning here. Binds you onto the selected target, be it friend, foe or NPC. Non-melee types should be cautious with this key, or even rebind it to **`Alt-F`** so that you aren't accidentally yanked into melee range of a foe.

Engage!

```
/bind G "target_enemy_near$$follow"
```

The essential melee bind for melee alts—target the nearest enemy and lock onto him. Bind to **`G`** for "Grapple" or "Get 'em"" and keep **`F`** for Follow when you have the desired target selected. In the heat of battle, just whack **`G`** to continue dealing damage to the nearest foe. No time wasted selecting a foe or having one move/knocked out of range.

I.C.U.!

```
/bind T "target_enemy_near"

/bind CTRL+T "target_enemy_next"
```

An essential bind for all types—helps you find and target slightly hidden foes, even at a distance. The first bind finds only the closest foe; Repeated presses of the second one will cycle through all visible foes, from nearest to furthest.

You can also use this bind:

```
/bind CTRL+T "target_friend_next"
```

…which will cycle through all friendly alts, on your team or not. Or use this one:

```
/bind CTRL+T "target_custom_next teammate"
```

…to cycle through teammates as targets.

☞ Some grumpy old-school gamers insist that only Tab is the right key for targeting. Fine, whatever, bind all these to **`TAB`**, then.

QuickInsp

A universally important set of binds, built into the GABB.

```
/bind F1 "inspexec_name resurgence$$inspexec_name dramatic improvement$$inspexec_name //
respite"

/bind F2 "inspexec_name second wind$$inspexec_name take a breather$$inspexec_name //
catch a breath"
```

```
/bind F3 "inspexec_name phenomenal luck$$inspexec_name good luck$$inspexec_name luck"

/bind F4 "inspexec_name righteous rage$$inspexec_name focused rage$$inspexec_name enrage"

/bind F5 "inspexec_name uncanny insight$$inspexec_name keen insight$$inspexec_name insight"

/bind F6 "inspexec_name robust$$inspexec_name rugged$$inspexec_name sturdy "

/bind F7 "inspexec_name escape$$inspexec_name emerge$$inspexec_name break free "
```

Each of these binds will fire off the selected Inspiration type, from lowest power to highest. **Very useful for Health and Endurance**—I don't find the others as useful but you might.

Some players might prefer to reverse the order of Insps so that the most powerful ones fire first. Rearrange the specific key bindings to suit yourself—but be sure to make the first two, and perhaps "break free," easy to find and hit fast. They'll save your a… alt.

SneakyZapp!

```
/bind ALT+Z "follow$$powexecname Thunderbolt$$target_enemy_near"
```

This cutie will target the nearest enemy, trigger a ranged attack power and then move in to strike… but **stop at the absolute maximum range point to fire the attack**.

It's best used with Blaster, Defender and Corrupter ranged powers, and allows a fast, controlled attack with maximum safety. Instead of trying to figure out how close to get before firing, and possibly drawing aggro and return fire, this combo lets you slide in and attack in the most efficient way.

You can omit the targeting command if you want to choose your target ahead of time.

The best way to use this power is to target, activate… and then hit **S**-for-backwards as soon as the power fires, so you can dash back out of range and escape. There is, unfortunately, no way to add auto-runaway to the bind.

I'm So Tired…

```
/bind 0 "powexec_toggleon Rest$$g >>> I'm Resting! <<<"
/bind - "powexec_toggleoff Rest$$g <<< Rested up! >>>:
```

This pair will set your alt into Rest with `0`, and send a team message that you're momentarily out of the action. Tapping **–** next to it will take you out of Rest and inform the team you're all peppy again. The messages help prevent teammates from wading into a battle without knowing you're unavailable and highly vulnerable…

☞ Keep Rest in Slot 0 of Tray 1 to avoid confusion.

☞ Good place to use the Reference E chat text color and size enhancements for visibility.

9.3 User Interface Binds

Google (the) Map

```
/bind F12 "window_scale map 0.65"
/bind CTRL+F12 "window_scale map 3.0"
```

This bind set will let you zoom the map to huge with one key, and back to a small, out of your way helper with another. Getting rid of the map should go to the easier key.

Adjust the small value to your normal UI preference, and increase the large value up to 5.0 if you like.

☞ Works best if you park the map window in the upper left corner.

I'm Talkin' Here!

```
/bind ENTER "afk Hold on, I'm speaking to someone…$$beginchat"
```

This text will put you in the current chat entry dialog and put an AFK bubble over your head telling other players what you're doing. Different binds for current chat and tells can be used.

Hold, Please…

Tells or private chat is simple... unless you end up in a situation where you have two different conversations going.

This bindset can help, as it differentiates beween the last tell you received and the last one you **sent**. That way, if you're going back and forth with one player, a random tell from someone else won't divert your comments.

```
/bind BACKSPACE "autoreply"
```

…starts a reply to the last tell you were sent.

```
/bind CTRL+BACKSPACE "tell_last"
```

…adds a reply to the last tell you sent someone.

And finally:

```
/bind ALT+BACKSPACE "t $target, "
```

…opens a quick tell to any player you have targeted, like the one who just gave you a passing buff or heal.

☞ You can add the AFK chat bubble to each of those, with different messages to those around you.

I'm Going, I'm Going!

When you stop to check in with contacts and then your mission list, you end up with one or both of those windows plus the contact-dialog window open, and it gets tedious to close them so you can get on to heroicism or villainy. This bind slams all three closed so you can get to it:

```
/bind F11 "windowclose contact$$windowclose mission$$windowclose contactdialog"
```

Going… Going… GONE!

Whether you are a player who carefully manages your recipes and salvage for crafting or a greedy mercenary who sells them all for profit, this bind pair will make selling (and buying) these valuables much easier.

```
F8 "ah$$show salvage$$show recipes"
SHIFT+F8 "windowhide auction$$windowhide salvage$$windowhide recipes"
```

This pair works best with a little setup. Open the three windows (hey, look, you have a key that will do all that now) and carefully position and scale them so they form a nice set in the middle of the screen.

☞ Optional: save this window configuration, then dismiss the windows with the second bind and save the window configuration again. This ensures the window size and placement will not be reset by default.

Now you can occasionally pause, pop up the set, and drag your orange and purple junk to the auction list, and collect your loot from the last round. Or wheel and deal for that Very Rare you need to complete an IO.

☞ The auction window will not open inside Supergroup bases.

9.4 Healer/Buffer Binds

These bind sets are intended to put healing, buffing and general team-support commands on the keyboard numpad. As with the pet heal/buff bindset above, you'll have to go from mouse+keyboard control to two-handed keyboard control in combat, but I've found it very workable.

In each case, the numpad keys `1` through `8` are bound to select and affect a specific teammate, and numpad `9` is the same action on the currently-selected 'mate.

This works best if the Team window is large and prominent, since as a healer you will be watching it, not the battle. The map-zoom bind above can be adapted to the Team window for this purpose.

The powers referenced here are for an Empathy defender—adjust the power names and bindings to suit other models.

☞ You will usually have to tap these keys twice to select and then affect. Get used to using the Cancel button to prevent double-dealing on a teammate who doesn't need it.

Heal All

This bind uses the big, easy **0** button to fire your basic AoE healing power no matter which shift key you might have pressed.

```
/bind NUMPAD0 "powexec Healing Aura"
/bind SHIFT+NUMPAD0 "powexec Healing Aura"
/bind CTRL+NUMPAD0 "powexec Healing Aura"
/bind ALT+NUMPAD0 "powexec Healing Aura"
```

And binding your "power heal" to the **ADD** key (numpad **+**) means you can quickly add more healing to any ally you have selected. Good for followup when a basic heal isn't enough:

```
/bind ADD "powexec Absorb Pain"
/bind SHIFT+ADD "powexec Absorb Pain"
/bind CTRL+ADD "powexec Absorb Pain"
/bind ALT+ADD "powexec Absorb Pain"
```

Heal One

This bind fires your basic heal-other power at the specified teammate.

Number 9 fires the power at the targeted teammate, pet or ally.

```
/bind NUMPAD1 "unselect$$teamselect 1$$powexec Heal Other"
...
/bind NUMPAD8 "unselect$$teamselect 8$$powexec Heal Other"
/bind NUMPAD9 "powexec Heal Other"
```

Power Heal

```
/bind CTRL+NUMPAD1 "unselect$$teamselect 1$$powexec Absorb Pain"
...
/bind CTRL+NUMPAD8 "unselect$$teamselect 8$$powexec Absorb Pain"
/bind CTRL+NUMPAD9 "powexec Absorb Pain"
```

This bind fires your power-healing power at the specified teammate, and again key 9 is at the targeted ally (which is a duplicate of the **ADD** bind, but worth doing for consistency.

☞ You can repeat this bind model for the **SHIFT** and **ALT** keys for other useful team-related powers—Fortitude, Clear Mind, etc. The consistent factor is that keys 1–8 are bound to each team slot.

👍 **YOUR NAME HERE... Got a great bind or macro? Send it to me!**

REFERENCE A: Slash Commands

Slash commands are the power user's way to make the game do an endless variety of things. Many commands just parallel menu or other basic commands, but some allow access to deeper levels and alternate actions.

Most commands can be bound to keys or macros, and all but a few can be directly entered in the Chat entry field. Browse, learn and remember these commands to really supercharge your mastery of advanced gameplay!

A.1 Slash Command Listing

Many commands have multiple synonyms. I have listed all known variations and, with the 2019 update, combined all of them into single listings.

The following chart (and this whole guide) use the following conventions:

- Keywords in ***`italics`*** represent values to be specified.
- Elements in **`[brackets]`** are optional. If an element is not in brackets, it is required.
- Numbers in **`{braces}`** are required:
  - Numbers separated by vertical bars **`{0|1}`** represent the valid selections.
  - Numbers separated by a dash **`{1-4}`** or **`{0.1-2.0}`** indicate the range of acceptable values.
  - Some commands that require a numeric value will return the current state if entered without a number; others will return an error message.
- Some valid commands will return a "no such command" error if an argument is not included.
- Commands that use an underscore (_) to separate words can also be entered without the underscore, for example, **`window_hide`** and **`windowhide`** are equivalent. The underscore versions are generally used here, and you are free to use underscores in any way that makes the commands more sensible to you.
- All commands are also case-insensitive; **`UPPERCASE`** and **`CamelCase`** words are only for convenience.
- Anything highlighted in bright blue is something I have not yet verified or which I have found to be buggy—so use it cautiously and be sure to tell me anything useful you find out about it.
- Anything highlighted in green is obsolete or reported to be so. I have removed all of these from the table but a summary is in the cross-ref listing.
- Anything highlighted in red is new for I27-2 and/or also may be incomplete or buggy.

Slash Command	Description

`/ac` *`message_string`*
`/arena` *`message_string`*
	Send message on the Arena chat channel.

`/afk` *`message_string`*	Marks the player as Away From Keyboard. (An alt left idle for about one minute will automatically show 'AFK.') If no string is specified, a little balloon with "AFK" in it appears over your character's head. Otherwise, the string is displayed there. Note that an auto power can interrupt an AFK status. Note also that this command is how to put a text bubble up while you're typing a chat message… see Section W for details.

`/ah`
`/auctionhouse`
`/blackmarket`
`/wentworths`
	Open Auction (Wentworth's/Black Market) window.
	Maybe summoned anywhere except inside a supergroup base.

`/ai string`
`/arenainvite` *`name`*
	Invite player to join arena event.

`/altinvite` *`name`*	Invite your alt character by name to your current Supergroup. (Note: you must have invite privileges for this to work.)

`/alttray {0|1}`
`/alt2tray {0|1}`
	Show or hide the secondary and tertiary power trays. 1 causes the tray to raise, 0 to lower and hide. Either can be used independently (and show the tertiary tray without the secondary, for example).
	Warning: These two commands lock the tray they raise (using 1), and it will not be lowerable with the arrow icon or any other tray command. You must lower and unlock the tray with this command (using 0) before the tray can be lowered or cycled. See Section 4.2 for details.

`/alttraysticky`	Toggle the upper default trays in and out of visibility. Cycles between the base tray, base+secondary tray, base+secondary+tertiary tray, and base tray again, just as clicking the arrow icon does. Action will be blocked if either upper tray has been raised and locked with **`alttray`** or **`alttray2`**. See also **`traysticky`**, **`traystickyalt2`**
`[ARCHITECT] See section A.3 for the Architect Entertainment commands.`	
`/assist`	Set your current target to the selected ally's target.
`/assist_name` *`name`*	Set your current target to the named ally's target.
`/autoperf {0|1}`	Automatically change world detail for performance. (Function unclear; appears to do nothing. Probably an old function for original-era video performance.)
`/autorun {0|1}`	Toggle autorun on and off. Usually bound to the R key with '++' See also **`forward_mouse`**.

Slash Command	Description
/b *message_string* **/broadcast *message_string*** **/y *message_string*** **/yell *message_string***	Send message to entire zone.
/backward	Move backwards. Usually bound to the S key with '+'
[BASE] See section A.2 for base editing commands.	
/beginchat *message_string*	Starts chat-entry mode with given string. Works only in binds, not in direct entry. See also **startchat**.
/bind key *command_string*	Binds a key to a command string. See the rest of this guide for details.
/bind_load	Reads a list of keybinds from `keybinds.txt` in the default CoH directory.
/bind_load_file *filespec*	Reads a list of keybinds from a specified file location and name. Echos the file load to the status window.
/bind_load_file_silent *filespec*	Reads a list of keybinds from a specified file location and name, without echoing the action to the status window. Probably best for loading rolling bind sets and the like, where a confirmation is not useful.
/bind_save	Saves all keybinds to `keybinds.txt` in the default CoH directory.
/bind_save_file *filespec*	Saves all keybinds to specified file location and name. Echos the file save to the status window.
/bind_save_file_silent *filespec*	Saves all keybinds to specified file location and name, without an echo to the status window.
/bloomscale {2|4}	Sets bloom blur size. Valid values 2 or 4 only.
/bloomweight *n*	Sets bloom scale. Valid values 0.0—2.0.
/buffs {0|1}	Toggle display of member buffs in the team list.
/build_save	Save current character build to **BUILD.TXT** file.
/build_save_file *filespec*	Save current character build to file designated by ***filespec***.
/c *message_string* **/coalition *message_string***	Send message to the coalition chat channel. You must be a member of a supergroup that is in a coalition with another group for this function to work.
/camdist {0-120}	Sets the distance in feet that the third person camera pulls back behind the player. 0 equals first-person view; the upper limit has varied but is currently 120. Note that the mousewheel adjustment is limited to 80 feet, and if you set a larger distance with this command any touch of the mousewheel will zoom to an 80 foot view.
/camdistadjust	Adjusts the camera distance relative to the current camera distance. Reads mousewheel for input and allows a range of 0-80 feet. Probably not useful in console commands, as it appears to be permanently bound to the mouse wheel.

Slash Command	Description
`/camreset`	Resets the camera to a few feet behind the player, looking forward. Bound to the **PAGEDOWN** key by default.
`/camrotate`	Camrotate (bound to **PAGEUP** by default) allows controlled camera rotation around the player. The bound key must be pressed while the view is rotated with the mouse. This command should be bound to a suitable key, and not invoked through the console.
`/camturn`	Turns the camera to match player facing direction. Similar to `camreset` except that camera distance is not reset. See also **face** and **playerturn**.
`/canlook {0|1}`	Toggles "mouselook," which permits the character to look around using the mouse instead of moving the in-game pointer. Usually bound to right mouse key.

`/costume_change {0-9}`
`/cc {0-9}`
	Change costume instantly.
	The upper number depends on the number of active costume slots. There are currently six free slots plus four earned ones.
	Note that this is a zero-based command!
	See also **cc_emote** for a more elaborate option.

`/cc_emote {0-9}` ***emotestring*** `/cce {0-9}` ***emotestring*** *See Reference D.2 for the list of emote names and usage.*	Combines an emote and a costume change. The upper number depends on the number of active costume slots. In the Live era, there were up to four, three of which had to be earned by in-game actions. This has been expanded to six free slots plus four earned ones. Note that this is a zero-based command! The string is one of a dozen or so special costume-change emotes. Only these emotes can be used with this command, and they cannot be used as regular emotes. See **/costume_change** for a simpler option.
`/cgshaderpath` ***pathspec***	Set parent directory for /shaders/cgfx. If relative, path assumes .EXE file directory as root. Function unclear.
[CHAT] Note that there is a difference between default chat channels (which typically cannot be modified) and user-created chat channels, for which these commands apply.	
`/chan_create` ***channel***	Create a new chat channel.
`/chan_desc` ***channel string***	Set chat channel's description to ***string***.
`/chan_invite` ***channel***	Invite player or chat handle to a chat channel.
`/chan_invite_deny` ***channel name_string***	Deny/refuse chat channel invite for named player on named channel.
`/chan_invite_gf` ***channel***	Invite your entire global friends list to a chat channel.

Slash Command	Description
`/chan_invite_sg` *`channel rank`*	Invite your entire supergroup to a chat channel. Only leaders may use this command. You can invite members by rank: `0`—Invite all supergroup members. `1`—Invite captains and leaders only. `2`—Invite leaders only.
`/chan_invite_team` *`channel`*	Invite your entire team to a chat channel.
`/chan_join` *`channel`*	Join an existing chat channel.
`/chan_leave` *`channel`*	Leave a chat channel.
`/chan_members` *`channel`*	List all members of channel.
`/chan_mode` *`channel options`*	Changes default access rights for new user who joins the channel. Valid Options: **`-join`** kicks user from channel **`+send/-send`** gives/removes user ability to send messages to channel **`+operator/-operator`** gives/removes operator status from another user in the channel
`/chan_motd` *`channel string`*	Set the channel's Message Of The Day, which is sent to everyone that joins the channel.
`/chan_send` *`channel string`*	Send message to chat channel. You must be in the channel and have Send privileges. (Synonym: send)
`/chan_timeout` *`channel duration`*	Allows channel moderators to set number of days before inactive users are removed from the channel. A valid channel name for which the user is a moderator is required. The number of days can be set from 0 (no removal limit) to 30 days.
`/chan_user_mode channel` *`name`* `options`	Sets user permissions for specified user on channel. You must have operator status to set permissions. Valid Options: **`-join`** kicks user from channel **`+send/-send`** gives/removes user ability to send messages to channel **`+operator/-operator`** gives/removes operator status from another user in the channel
`/change_handle` *`name`*	Change your global user name, if allowed. There are limits on how often a global handle can be changed (e.g., it may be a one-time change for some users), so use this with caution.
`/chat`	Toggles the chat window. (Synonyms: **`toggle chat, window_toggle chat`**)
`/chat_cycle`	Cycles through the default chat channels.
`[CHAT LOAD/SAVE] See Section 4.1 for information on using chat save and load functions.`	

Slash Command	Description
/chat_load	Reads a saved chat configuration (tabs, channels, names) from the **CHAT.TXT** file in the default installation folder.
/chat_load_file *filename*	Reads a saved chat configuration (tabs, channels, names) from the specified file name in the default installation folder, or, from the file on another path if it is specified.
/chatoptions {0-4}	Opens the option menu for the specified chat window.
/chat_save	Saves the current chat configuration (tabs, channels, names) to the **CHAT.TXT** file in the default installation folder.
/chat_save_file *filename*	Saves the current chat configuration (tabs, channels, names) to the specified file name in the default installation folder, or, to the file on another path if it is specified.
/chat_set *channel*	Sets the channel to the given string. Works only for created global channels, not defaults.
/city_time	Retuns in-game time in the System chat. See **showtime** for a more permanent clock and explanation of game time cycle.
/clearAttributeView	Clear the attribute target. Function uncertain.
/clearchat	Clears all chat buffers—equivalent to executing "Clear History" in each chat tab.
/clear_tray	Clear all power trays of icons **(except for macros)**. Use with caution unless you really want to drag all your powers back into place.
/clear_petnames	Clears all names of all your named pets.
/clearRewardChoice	Choose "no reward" in the current reward choice list.
/cmdlist	Displays all console commands available in the system chat window. (Useful for finding updates and changes to this list—turn on chat logging first to save to a text file, or use **copychat** to copy all text to the clipboard.)
/coalition_cancel	Cancel coalition with selected supergroup.
/coalition_invite *player_name* **/ci *player_name***	Invites the named player to join a coalition. The player must be the leader of a supergroup for the function to work. (Synonym: **ci**)
/coalition_mintalkrank	Set the minimum rank of a supergroup who your supergroup can hear. (Values unknown.)
/coalition_nosend	Stop your supergroup from sending coalition chat to an ally supergroup.
/coalition_sg_mintalkrank	Set the minimum rank of a your supergroup who can talk on the coalition chat. (Values unknown.)
/compatible_cursors {0|1}	Shows the **status** of selection of standard Windows cursors instead of graphical cursors. (The Windows cursors are not as flexible and don't change color but may work better on some systems.) Although the command accepts an argument, it cannot be used to set the option, which must be set on the command line at game startup.

Slash Command	Description
`/contextmenu` *`menu_num`*	Activate a context menu slot. (Functionality unclear.)
`/copychat` *`tab_name`*	Copy the entire chat history from specified chat tab into the clipboard. Useful for saving extended game info passed on by other players, dumps from the name-list commands, or abuse. See also **`logchat`**.
`/copydebuginfo`	Gathers debug info, prints it and copies it into the clipboard. (Functionality unclear.)
`/cov` *`int`*	Function unknown. Probably obsolete.

`/ctm {0|1}`
`/clicktomove {0|1}`
`/ctm_invert {0|1}`
	Enable and disable click-to-move.
	When enabled, clicking on any non-clickable point with create a pretty crystalline cursor, and your character will move to it. Maximum move range is about 60 yards. Useful for zooming around missions and such.
	`ctm_invert` originally worked in reverse (0 to enable, 1 to disable) but in current issue works the same as the other two commands.
	See also **`ctmtoggle`**.

`/ctm_toggle`	Toggles click-to-move status. No argument allowed. Note that there are cases where the displayed enable/disable message gets inverted ("enabled" is actually disabled, and vice versa) if used with other CTM change commands.
`/cursorcache {0|1}`	Enable cursor cache for smoother cursor changes.
`/custom_window` *`name`*	Create a custom window. (Complex to use; requires writing window definition file.)
`/custom_window_toggle` *`name`*	Open or close a custom window.
`/demorecord` *`filename`*	Begin recording a demo with specified filename.
`/demorecord_auto`	Begin recording a demo with generated filename.
`/demostop`	Stop demo record/play.
`/demote` *`name`*	Demote supergroup member one rank.
`/dialog_answer` *`string`*	Answer dialog with button matching provided text. (Functionality unclear; assume it is for use with dialogs that have other than Yes/No options, using the following two commands.)
`/dialog_no`	Answer **OK**, **No**, or **Cancel** to current dialog.
`/dialog_yes`	Answer **OK**, **Yes**, or **Accept** to current dialog.
`/disable2D {0|1}`	Disables 2D sprite drawing. (Only effect seems to be to turn all UI elements on and off.)
`/dofweight {0.0–2.0}`	Sets DOF (depth of field scale, basically controlling distance blurring for a more realistic rendering. Setting used only if **`usedof`** is enabled.

Slash Command	Description
`/down`	Move down (if flying). Bound to X with '+' by default.
`/e3screenshot {0-?}`	Enables "Special E3 2004 screenshot mode." (Values and function unknown. Probably obsolete.)

`/e` *`emotestring`*
`/em` *`emotestring`*
`/emote` *`emotestring`*
`/me` *`emotestring`*
	Causes player to display an emote animation or emote string.
	Emote codes can be found in Reference D.
	Any string that does not match a valid emote code will be displayed in a visible thought bubble for a few seconds.

`/emaildelete` *`message_number`*	Delete message [message number]. Functionality unclear.
`/emailheaders`	Request email headers. Functionality unclear.
`/emailread message_number`	Request message [message number]. Functionality unclear.

`/emailsend` *`name subject`*
`message_body` `[`*`influence`*`]`
`/emailsendattachment` *`name`*
`subject influence attachment`
`location message_body`
 	
Thanks to Hopewarden & "Mikhail" for details on this one!
	Send email message to another player, with or without attachments, which can include Influence, Inspirations, Enhancements, Salvage and Recipes.
	The first command only sends email, despite some help notes to indicate that Influence can be added to the transmission. (The influence keyword there is probably bogus.) It is recommended that you just use the Email UI for this.
	The second command can be used to send just a message, a message with Influence, or, with care, almost any item in player inventory.
	`name` is a valid player name, preferably @global
	`subject` is the message subject line—use quotes
	`influence` is a numeric amount of Inf to send—use caution, it can't be retrieved
	`attachment` is an attachment type, of which around 17 are possibly defined:
	`2`—Inspirations
	`10`—Enhancements (may not work)
	`11`—Salvage
	`12`—Recipe
	`location` is the item's location in the corresponding inventory window, counting from 1. *Practice sending to yourself before you send an ultra-rare item to some random stranger!*
	`message_body` is the body of the message, and will include anything in or after that position, in or out of quotes.

Slash Command	Description
/enterbasefrompasscode *passcode*	Instantly enter any base with a matching entry passcode, including your own. Keeping it on hand as a macro or bind, with each base for which you have entry privileges, makes portal entry much quicker. This was a GM-only "supercommand" that allowed teleport to base from absolutely anywhere, no activation delay, no cooldown. It was restricted in I27 to working only inside bases (so you can jump to any other base) and within clicky-distance of any valid base portal.
/f *message_string*	Talk to **Friends** channel; message string will be entered immediately.
/face	Turn player to face selected target. See also **playerturn** and **camturn**.
/findmember *name*	Search for player. Appears identical to **search**.

/first {0|1}
/third {0|1}
	Toggles between first and third person camera.
	
	Each is inverse of the other; e.g. **first 0** equals **third 1**.

/fl **/friendlist**	Display friend list in chat window.
/follow	Toggle follow mode for currently selected foe, alt, NPC or other game element. Useful in melee combat and targeting.
/forward	Move forward. Bound to W with '+' by default.
/forward_mouse	Move forward; enable autorun after 2 seconds. Peculiar in that it will not force a playerturn (to face forward in direction of travel) until autorun is engaged. Bound to **mousechord** by default.
/friend *name*	Add player to friend list.
/fsaa {0-<any multiple of 2>}	Sets the amount of full screen antialiasing. The default and suggested limit is 4. Note: FSAA has more impact on framerate than nearly any other graphics setting! Even with modern video cards, high FSAA rates can bog down framerate enormously—technical implementation in the engine may be poor/outdated. Setting higher than 16 can bring UI to a near-freeze.
/fullrelight {0|1}	Disable cap on number of relit vertices per frame. (Functionality unclear.)
/fullscreen {0|1}	Sets video mode to fullscreen. If set to 0, the game will start in windowed mode next time; when set to 1, game will start in fullscreen mode. Cannot be changed during gameplay; you have to make this setting and then restart to change the view. See also **maximize**.
/g *message_string* **/group *message_string*** **/team *message_string***	Send message to group channel. Note that **t** is not a synonym for **team** (it's for **tell**).

Slash Command	Description
/gamereturn **/windowcloseextra**	Reset UI by leaving fullscreen mode, closing dialogs and closing all secondary (nonessential) windows. Using window save and load functions are more flexible, though.
/getarenastats **/getratedarenastats**	Get your arena combat statistics.
/getallarenastats	Get your arena combat statistics, more comprehensive display.
/getcomment	Get your group-search string. (See also **comment**.)
/getglobalname ***charname***	Get player's global name from character name. See **getlocalname**.
/getlocalinvite ***globalname***	Invite active character name to team using a global player name.
/getlocalleagueinvite ***globalname***	Invite active character name to legue using a global player name.
/getlocalname ***globalname***	Get currently active character name from global player name. See ***getlocalname***.
/gfriend ***name***	Add a player to your global friends list.
/gfriends	Display all members of your global friends list.
/gignore ***name***	Ignore user on global chat.
/gignoring	Display list of ignored users on global chat.
/ginvite ***player_name***	Invites the named player to join a global chat.
/ginvite_sg ***channel rank***	Invite your entire supergroup to a global chat channel. Only leaders may use this command. You can invite members by rank: **0**—Invite all supergroup members. **1**—Invite captains and leaders only. **2**—Invite leaders only. (See also **chan_invite_sg**.)
/gmotd	Recall the global (server/dev) message of the day, as displayed at first login.
[TRAY] See Section 4 for complete information on using power tray commands.	
/goto_tray {1-9}	Set the main tray to the specified virtual tray number.
/goto_tray_alt {1-9}	Set the secondary tray to the specified virtual tray number.
/goto_tray_alt2 {1-9}	Set the tertiary tray to the specified virtual tray number.
/goto_trays_tray {1-3} {1-9}	Set the specified tray (1-3) to the desired virtual tray number (0-9).

Slash Command	Description
`/graphfps {0|1|2|4|8}`	Graph current framerate. Details are not clear, and a value of 8 may show SLI-only information. `1`—FPS graph `2`—FPS graph with average numerical value `3`—Dual FPS graph `4`—FPS graph (SLI?) `0`—Off
`/gunfriend` ***`name`***	Remove a player from your global friends list.
`/gunfriend_player` ***`name`***	Remove player from global friends list.
`/gunignore` ***`name`***	Un-ignore user on global chat.

`/help` ***`message_string`***
`/helpchat` ***`message_string`***
`/h` ***`message_string`***
`/hc` ***`message_string`***
`/guide` ***`message_string`***
	Sends a message the global Help channel.

`/helpwindow`	Open Help window.

`/hide`
`/unhide`
`/ghide`
`/gunhide`
 	
`/hidefriends,/unhidefriends`
`/hidegchannels,`
`/unhidegchannels`
`/hidegfriends,/unhidegfriends`
`/hideinvite,/unhideinvite`
`/hidesearch,/unhidesearch`
`/hidesg,/unhidesg`
`/hidetell,/unhidetell`
	Each of the first four commands brings up the same Hide window with 7 click on/off buttons for each area of hiding your current login from other players. This is the recommended method and only the basic **`hide`** command needs to be remembered and used.
	
	The other **hide…** commands hide the user on the specified channel (Server Friends, Global Friends, Global Channels, Invites, Email, Searches, your Supergroup and Tells), but (may) also unhide them on all others. The parallel **unhide…** commands work similarly.
	
	It is recommended that you use **`/hide`** to bring up the Hide window to manage Hide settings, unless you use **`hideall/unhideall`** as a global invisibility command.
	
	Do not confuse player-hide commands with **`windowhide`**.

`/hideall` **`/unhideall`**	Hide or unhide your name from other users in all of the "who’s on" lists. Complete player invisibility.
`/hideprimarychat`	Toggle primary chat window. This reduces the chat window to just the chat entry line and the menu with this command. You should be sure another chat window holds whatever channels you are participating in.

`/i` ***`name`***
`/invite` ***`name`***
	Invite player to join team.

`/ignore` ***`name`***	Ignore user. See also **`unignore`** and **`ignorelist`**.

Slash Command	Description
`/ignorespammer` *`name`*	Ignore user as spammer. Automatically reports name as spammer. Functionality not verified in Homecoming.
`/ignorelist`	Displays a list of ignored users.

`/incarnate_equip` *`slot powername`*
`/incarnate_unequip`
`slot powername`
`/incarnate_unequip_by_slot` *`slot`*
`/incarnate_unequip_all`
	Equip and unequip Incarnate Ability powers.
	`slot` is the Incarnate slot number.
	`powername` is the Incarnate power by name. If the name has spaces, it must be enclosed in quotes or the spaces replaced with underscores.
	These commands do nothing if:
	– slot is locked
	– power is not possessed
	– you or team is in combat
	– currently slotted power is recharging
	– it has been less than five minutes since last IA power swap

`/info`
`/info_tab {0-6}`
	Displays the information on a selected item, same as right-clicking and selecting Info from the pop-up menu. (If no other item is selected, your alt's info panel will appear, as for the following command.)
	The second command opens the window to the named info tab. Tabs are referenced by number:
	`0`, `1`, `4`—Description.
	`2`—Powers.
	`3`—Badges.
	`5`—PvP.
	`6`—Arena.

`/info_self`
`/info_self_tab {0-6}`
	Displays your own information, the same as others see when they "info" you.
	The second command opens the window to the named info tab. Tabs are referenced by number:
	`0`, `1`, `4`—Description.
	`2`—Powers.
	`3`—Badges.
	`5`—PvP.
	`6`—Arena.

`/insp_combine` *`inspname1`*
`inspname2`
	Combines three of the first named Inspirations to create one of the second name and next power level.
	You must put quotes around multi-word Inspiration names, e.g. "break free" or "catch a breath".

Slash Command	Description
/insp_delete *inspname*	Delete named Inspiration if one exists in your tray. Might be a useful bind in combat to clear out, for example, stamina Insps on a high-stamina build. Just firing off unwanted insps or giving them to team mates seems more… useful.
/inspexec_name *inspname*	Activate an Inspiration by name.
/inspexec_pet_name *inspname* *petname*	Activate a named Inspiration on a pet by pet name.
/inspexec_pet_target *inspname*	Activate a named Inspiration on the targeted pet.
/inspexec_slot *column* **/inspirationslot *column***	Activate an inspiration slot in the first row of the specified column. Inspiration tray numbering is 1-based for both row and column.
/inspexec_tray *row column*	Activate an inspiration slot in the specified row and column.
/keybind_reset **/unbindall**	Resets all keybinds to default. See also **unbind**. Use with caution if you don't have a saved bind file to load!
/k *name* **/kick *name***	Kick player from team.
/l *message_string* **/local *message_string***	Send message to anyone in the Local chat channel (your immediate area, about a 250 foot radius).
/league_kick *name* **/lk *name***	Kick player *name* from league. Note: **lk** has been reused for this function from its prior synonym for **lackey**/**sidekick**.
/leaveteam	Quit your current team.
/left	Strafe left. Bound to A with '+' by default. See also **turnleft**.
/levelingpact *playername*	Invite named player to join a Leveling Pact.
/lfg [0|1] **/looking_for_group [0|1]**	Toggle LFG (looking for group) status. See also **lfgset**.
/lfg_event_response *string*	Accept or reject invitation to join event. Valid strings seem to be 'accept' or 1, or 'reject' or 0.
/lfg_remove_from_queue	Remove self or team from LFG queue. (Functionality uncertain.)
/lfg_request_event_list	Get LFG system event list. (Functionality uncertain.)
/lfgset {0|1}	Set LFG (looking for group) status. See also **lfg**.
/link_channel *channelname*	Activates context menu for named channel.
/link_info *channelname*	Provides info window for named channel. Works only for created channels, not defaults.
/link_interact *playername*	Activates context menu for named player interactions.

Slash Command	Description
/link_interact_global *arg arg*	Activates context menu for global player name. (Functionality uncertain.)
/localtime	Displays (your computer's) local time in the Status window. See also **servertime**, **city_time** and **showtime**.
/loc **/getpos**	Displays current XY coordinates and altitude in the System channel. A useful bind for mapping, and evaluating jump height and teleport increments. A continuous location display can be toggled with **showfps**. This command now creates a clickable coordinates link in chat.
/lodbias {0.0-2.0}	Multiplier for LOD (Loss of Detail) distances for entities. The default is 1.0. Setting this to 0.5 will cause detail switches to happen at half the distance; 2.0 will cause switches to happen at twice the default distance. Lower values improve performance; higher ones increase your character's vision. Appears to be obsolete; see also **DOFweight**.
/logchat	Toggle chat logging. Chat logs appear by date in the \logs folder under the main CoH folder. Chat can also be extracted using **copychat**.
/lookdown **/lookup**	Moves look angle down or up. Normally, these commands are used with the + modifier to permit controlled up and down looking. (If both **lookdown** and **lookup** are set to 1, or both are set to 0, you will have free look capability. Setting one or the other to 1 will force the view to a straight up or straight down view, persistent against changes. This use is not recommended.) See also **zoomin**/**zoomout**.
/lp *message_string*	Sends message to Leveling Pact channel.
/macro *macroname command_string*	Add a macro to first empty slot. The macroname will display on a gray power disc, but only up to 3 letters; the rest will appear as a tooltip on hover. See the rest of this guide for details.
/macro_image *iconimage macroname command_string*	Adds a macro to the first empty slot and (way cool!) uses the specified icon image file for the icon. The icon texture file must be an existing power-tray icon from within the game's PIGG files. There are hundreds, if not thousands, callable by using a combination of powerset and powernames. See Section 3.3 for details.
/macroslot *slotnum macroname command_string* *Thanks to hooliganj for details on this one!*	Add a macro to the specified slot of any tray. This command will overwrite any command or macro already in that slot. The value for **slotnum** is complicated in that it can be 0-89, with 0 being the first slot in the primary tray, 10 being the first slot in the second tray, and so on up to 89 being the last slot in tray 9. Calculate carefully. Note that the numbering here is zero-based.
/makeleader *name* **/ml *name***	Designated new team leader. Can be used only by current leader.

Slash Command	Description
/mailview *string*	Sets view to use on the Mail tab. (Functionality unknown.)
/manage	Toggles the Enhancement Management (slotting) window. (This is one of the few menu-window command names that does not work in the other window-control commands.)
/map	Toggles the map window. (Synonyms: **toggle map**, **window_toggle map**)
/maxcolortrackerverts *int*	Maximum number of world object vertices to relight per frame. Default seems to be 5000; much larger values accepted. (Functionality uncertain.)
/maxfps {0-?}	Set the maximum FPS (frames per second) rate. The default of 0 seems to be "maximum allowed" (or 30). This seems to be capped at 30 but now appears to accept any value. Normally you will want this maximized, but it may be useful in some circumstances to enter a slower rate. Very slow rates (1-5 or so) are NOT recommended but can be fun to play with in safe areas. See also **showfps**, **graphfps**, **maxmenufps** and **maxinactivefps**.
/maximize {0|1}	Probably sets maximized window for game (on next restart) when not fullscreen. (Functionality uncertain.) Compare with **fullscreen**.
/maxinactivefps {1-?}	Set the maximum FPS (frames per second) rate while the game is not in the foreground. Default appears to be 60. Higher values are accepted. Reducing this value may lessen the impact on other programs brought forward during gameplay. The rate should be high enough for you to be able to keep track of what is happening—no lower than 5-8 fps is recommended. See also **maxfps**.
/maxmenufps {1-?}	Set the maximum FPS (frames per second) rate while the game is in a full-screen menu. (Functionality uncertain.)
/maxrtframes	Set how many frames forward to allow buffering. (Functionality uncertain.)
/menu	Opens the main menu. (Synonyms: toggle menu; window_toggle menu)
[MISSION ARCHITECT] See section A.3 for the Architect Entertainment commands.	
/monitorattribute *string* **/stopmonitorattribute *string***	Adds a display line to the Attribute Monitor. The first command adds, or toggles any specified line; the second command removes the specified line. (The latter is thus not all that useful…) See Reference C.1 for a list of known arguments and more detailed usage information.

Slash Command	Description
`/mouse_invert {0|1}`	When active, inverts the mouse Y axis (pitch) for mouselook.
`/mouse_look {0|1}`	Mouse-look[around] command. Usually bound to right mouse key with '+'. Can be bound to another key or macro to toggle mouse-look as the default mode (looking around with mouse movement instead of pointer select. Experiment if you don't like holding down a key all the time.
`/mouse_speed {0-6}`	Mouse speed scale factor for mouse look. 1.0 is default. Values over 3 make control erratic in most cases. 0 disables mouse movement, which could be useful in some cases.
`/myhandle`	Display your global chat handle.
`/mypurchases`	Show list of purchases you have access to. Works but appears to be obsolete; for "pack" purchase history?

`/nameCaptain` ***`name_string`***
`/nameCommander` ***`name_string`***
`/nameEnforcer` ***`name_string`***
`/nameFlunky` ***`name_string`***
`/nameLeader` ***`name_string`***
`/nameLieutenant` ***`name_string`***
`/nameMember` ***`name_string`***
`/nameOverlord` ***`name_string`***
`/nameRingleader` ***`name_string`***
`/nameTaskmaster` ***`name_string`***
	Each renames the corresponding standard supergroup rank.

`/nav`	Toggles the navigation window. (Synonyms: **`toggle nav`**, **`window_toggle nav`**, **`toggle compass`**, **`window_toggle compass`**)
`/neterrorcorrection {0|1|2}`	Adjusts network error correction limits. (Functionality uncertain.)
`/netgraph {0|1|2}`	Displays network connection information. Option 1 is low-profile, Option 2 is higher-profile; not sure of other differences. See also **`graphfps`**.
`[TRAY] See Section 4 for complete information on using power tray commands.`	
`/next_tray`	Go to next primary tray. See also **`prev_tray`...** commands
`/next_tray_alt`	Go to next secondary tray.
`/next_tray_alt2`	Go to next tertiary tray.
`/next_trays_tray {1-3}`	Go to the specified tray's next tray.

Slash Command	Description
`/nojumprepeat {0|1}`	Disable jump auto-repeat. This means you'll jump only once, no matter how long the key is held down; another jump will require another keypress. Can be useful for better control indoors or with really bouncy alts.
`nop`	A null placeholder used to cancel a bind. If you enter **`/bind x nop`**, for example, any bind on X will be deleted and the key will do nothing. Useful for clearing out default binds you don't want. Note that this is not the same as a null bind argument (`""`), which erases any existing bind but can be overwritten by a default bind. Use **`NOP`** assignments to permanently blank binds in bindfiles, and `""` to erase them in game.
`/noparticles {0|1}`	Turn off particle graphics. Improves performance but many active game elements become invisible.
`/norenderthread`	See also **`renderthread`**. (Functionality uncertain.)
`/noreport {0|1}`	Do not default to error reporting window on crash. This may suppress the Windows error reporting screen after a crash; confirmation and other purpose unknown.
`/nosunflare {0|1}`	Disables sun flare (for performance debugging). Removes and restores flare/glare from sunlight (and moonlight?)
`/noversioncheck {0|1}`	Disable mapserver version check. Probably useful in this era of rogue servers and sometimes unsynchronized updates.
`See Section 4.3 for complete information on using option set, save and load functions.`	
`/option_list`	Lists option names.
`/option_load`	Reads option configuration from the file **`options.txt`** in the default installation folder.
`/option_load_file` *`filename`*	Reads option configuration from the specified filename in the default installation folder, or, if specified, in a different location.
`/option_save`	Saves window configuration to the file **`options.txt`** in the default installation folder.
`/option_save_file` *`filename`*	Saves option configuration to the specified filename in the default installation folder, or, if specified, in a different location.
`/option_set` *`optionname arg`*	Sets an option to the specified argument.
`/option_toggle` *`optionname`*	Toggles an option (binary options only).
`See Section 6 for complete information on managing pet options and control.`	

Slash Command	Description
`/petcom stance`	Set selected pet to specified action/stance. For this group of commands, the valid pet stances are: **`aggressive`**—attack any nearby foe without orders. **`defensive`**—respond to attack by any foe without orders. **`passive`**—do nothing without orders. And the valid pet actions are: **`attack`**—attack currently selected target. **`dismiss`**—dismisses pet gracefully and immediately. Compare with **`releasepets`**. **`follow`**—follow me. **`goto`**—go to the selected spot. **`stay`**—stay at the selected spot.
`/petcom_all` *`stance`*	Set all pets to specified action/stance.
`/petcom_name` *`petname stance`*	Set named pet to specified action/stance.
`/petcom_pow` *`powname stance`*	Set the stance for all pets cast by the named power.
`/pet_select` *`petnum`*	Select pet by number, starting with 0.
`/pet_select_name` *`petname`*	Select named pet.
`/petoptions`	Displays pet window options menu only if the Pet window is displayed. Use **`/show pet`** to open the Pet window.
`/petrename` *`petname`*	Renames selected pet.
`/petrename_name` *`oldname newname`*	Renames named pet.
`/petsay` *`string`*	Have selected pet say or emote ***`string`***. This and the following three targeted commands use a very specific subset of string format to work correctly. Everything in string will be "spoken" by the chosen pet, including most control characters. To have a pet perform an emote, it must be enclosed in angle brackets: **`<em Wave>`** or **`<emote Bow>`**. To combine an emote and a chat-bubble string, just run them together: **`<em Bow>At Your Service!`** Multiple emotes (in brackets) and text can be strung together. There is also a protocol for synchronizing "smash" emotes by the player and "react" emotes by the pet, so that the pet will cower as the player attacks them. Briefly, it's **`<em batreact>$$em batsmash`** **`<em slapreact>$$em smack`** and other such emote pairs. NOTE: pet emotes *do not work* in binds due to a current bug in the way strings are processed. A workaround is to write pet emote commands to macros and call the macro with a bind. See Section 6.4 for details.
`/petsay_all` *`string`*	Have all pets say or emote ***`string`***.
`/petsay_name` *`petname string`*	Have named pet say or emote ***`string`***.

Slash Command	Description
`/petsay_pow` *`powname string`*	Have all pets cast by specified power say or emote ***`string`***.
`/petition` *`subjectstring`*	Add user petition (stuck, cheated, etc.) to the database. This is more for immediate help from a game master than things like posts in Discord. Give a GM time to get the petition and help you. ***`subject_string`*** may no longer be required with I27.

`/playernote` *`playername`*
`/playernotelocal` *`playername`*
	Opens note window for specified global player name, the same as right-clicking on an alt and selecting Add Note, but can be used whether player is present (or even online).
	Note that the first command shows only the alt (local) name and stores notes only for that alt. The second shows the global name, at least the current alt name, and stores notes for the global name. Notes are separate for each alt name and the global version.

`/playerturn`	Turns player to match camera angle. Does not change camera distance. See also **`camturn`**, **`face`**.
`/popmenu` *`menuname`*	Pops up custom menu at the current mouse location. Very complex to use; see HC Wiki for developer-level info: `hcwiki.cityofheroes.dev/wiki/Popmenu_(Slash_Command)`
`/powers`	Toggles the Power inventory window.
`[POWERS] See Section 5 for complete information on using trays and powers.`	
`/powers_togglealloff`	Toggles all activated powers off.
`/powexec_abort` `/powexec_unqueue`	Cancel queued powers (to prevent unwanted attack on next foe, for example). **`powexec_unqueue`** cancels any queued power and is bound to the Z key by default. **`powexec_abort`** cancels any queued power **and de-selects the auto power**; the latter will have to be re-selected (with Ctrl-click or **`powexec_auto`**) if further auto-activation is desired. The **`unselect`** command works in a similar way; it will un-select any selected item, but if a foe is selected and a power activated, **`unselect`** will also unqueue the power.
`/powexec_auto` *`power_name`*	Sets the named power to automatically execute or activate each time it has recharged. Useful for 'booster' powers like Hasten. If no power is named, the current autoexec power assignment will be cancelled. Only a single power may be set to auto-exec at any one time, and it will not work on macros. Auto power can also be set by Ctrl-clicking its icon; a green ring will indicate which power is set. The setting is persistent until changed.
`/powexec_location` *`target power_name`*	Executes the named point-and-click power as directed, without a reference click for locating the power focus. Updated in I27-2. Complex to use: see section 5.5 for details.
`/powexec_name` *`power_name`*	Executes a power with the given name.

Slash Command	Description
`/powexec_server_slot` *`slotnum`*	Executes the specified power slot from the **server-controlled tray**. This is the unnumbered tray that appears when two-part powers like sorcerous flight/jaunt are active. Numbering is 1-based. This tray's pop-up can be disabled by talking to Null the Gull.

`/powexec_slot {1-10}`
`/powexec_altslot {1-10}`
`/powexec_alt2slot {1-10}`
	Executes the specified power slot from the current trays—primary, secondary and tertiary, in order.
	Note that slots are 1-based numbering for these commands.

`/powexec_toggleoff` *`power_name`*	Toggles a given power off. If it's already off, does nothing. See also **`powers_togglealloff`**.
`/powexec_toggleon` *`power_name`*	Toggles a given power on. If it's already on, does nothing.
`/powexec_tray` *`slot tray`*	Executes a power in the given slot and tray. Slots are numbered 1-10; trays are numbered 1-9.
`[TRAY] See Section 4 for complete information on using power tray commands.`	
`/prev_tray`	Go to previous primary tray. See also **`next_tray`…** commands.
`/prev_tray_alt`	Go to previous secondary tray.
`/prev_tray_alt2`	Go to previous tertiary tray.
`/prev_trays_tray {1-3}`	Go to the specified tray's previous tray.
`/priorityboost {0|1}`	Set game process priority to Above Normal (from Normal) when running in the foreground. (Functionality uncertain.)
`/profiler_record` *`filename`*	Record client profiler information to specified filename. (Functionality uncertain.)
`/profiler_stop`	Stop recording client profiler information.
`/promote` *`name`*	Promote supergroup member one rank. See also **`demote`**.
`/quickchat`	Pops up the Quickchat (emotes+notoriety) menu.
`/quit`	Quits game to the desktop. There is a 10 second abort unless exit button is clicked or **`dialog_yes`** is appended.
`/quittocharacterselect`	Quits game to the character selector. (5-15 second abort.)
`/quittologin`	Quits game to login screen. (15 second abort.)
`/raid_invite`	Invites selected player's supergroup to join an instant raid.

Slash Command	Description

`/recharge_indicator {0-3}`
 	
`New for Homecoming and very useful!`
	Specifies how the numeric power recharge indicator appears across power icons in the tray.
	`0`—Does not appear.
	`1`—Appears below icon.
	`2`—Appears above icon.
	`3`—Appears across icon.

`/release`	Activate medicom unit for emergency medical transport. (Equivalent to clicking "Go to Hospital" button when defeated. There does not appear to be any command equivalent for "Go to Base.")
`/release_pets`	Deactivate all current pets. They will fall dead but remain present for some time (useful for power-rezzing, etc.) Compare with the pet menu/**`petcom`** command **`dismiss`**, which makes some pet types leave more gracefully but immediately.
`/reloadgfx`	Reload all graphics textures. Useful when something has messed up your screen display. Warning: scrambles display for at least a few seconds — do not use in combat.

`/renderscale {>0-1.0}`
`/renderscalex {>0-1.0}`
`/renderscaley {>0-1.0}`
	Changes the scale at which the world is rendered, relative to your screen size. Permits you to keep your screen size the same as desktop while lowering the effective resolution.
	The first command affects both X and Y scaling simultaneously, while `renderscalex` and renderscaley affect only horizontal and vertical scaling. 1=1.0=100% resolution scaling.
	Not effective unless `userenderscale` is set to 1. (Setting this value to 0, or cycling `userenderscale` from 1 to 0, will reset renderscaling to the default of 1.0.)
	Warning: possible to set scale so low that screen is unreadable (and chat entry of a correcting command impossible). Create a "recovery" bind using **`renderscale 1`** before experimenting.

`/renderscalefilter` ***`int`***	Changes the method of filtering used in renderscaling. (Functionality uncertain.)
`/rendersize` ***`xsize ysize`***	Changes the size at which the world is rendered. Has no effect currently and may be obsolete.
`/renderthread {0-?}`	(Functionality uncertain.) See also **`norenderthread`**.

`/req` ***`message_string`***
`/request` ***`message_string`***
`/sell` ***`message_string`***
`/auction` ***`message_string`***
	Send a message to the Request channel.
	(There is no specific "Auction" channel in current issues.)

Slash Command	Description
/requestexitmission {0|1|*n*}	Leave mission map once completed. Equivalent to clicking "Mission Completed" text in Nav window. The "1" is required; "0" does nothing. Other values may have other effects—testing is required. Does not set "auto-exit" if called before end of mission.
/reply *message_string* **/r *message_string*** **/autoreply**	Reply to last *received* private message, with or without a defined message string. If a message string is specified (for, say, a "In combat... wait a minute." message), it automatically sends. If no string is entered, or autoreply is used, the window waits for you to type the message and hit send. This is differentiated from replying to the last *sent* private message, which can be replied to using the **tell_last** command. **autoreply** is a synonym for the others in starting a private reply, but does not accept (or automatically send) a defined message string.
/respec	Goes to the Respec screen if you have a "respec" available. Warning: You should only use this command with your character in a safe place… you can be attacked while in this mode.
/right	Strafe right. Bound to D with a '+' by default. See also **turnright**.
/roleplaying	Toggles "Roleplaying" tag over player name, and changes the color of the entire name block. May be tangled with the Help/Helper name flags.
/s *message_string* **/say *message_string***	Sends the given text on the **current chat channel**. (You can select channels quickly by clicking on each chat window; messages will be sent on the default chat channel for each window.)
/screen *x_dim y_dim*	Sets X and Y screen dimensions. Should be constrained to standard screen dimensions supported by your video card (1280x1024, 1600x1200, 1920x1080 etc.) Another command for which you should set a "recovery" bind before experimenting, using your current screen dimensions.
/screenshot	Save a JPEG (**.jpg**) format screenshot in the **\screenshots** directory under the default CoH directory.
/screenshottga	Save a Targa (**.tga**) format screenshot in the **\screenshots** directory under the default CoH directory. Note that TGA files are huge compared to JPEG!
/screenshottitle *filename*	Save a JPEG (**.jpg**) format screenshot in the **\screenshots** directory under the default CoH directory, using the specified filename.
/screenshotui {0|1}	Enables or disables the user interface elements for screenshots. If set to 1, the UI will be visible in screenshots; if set to 0, the UI will not be included in screenshots.
/sea **/search**	Displays a searchable list of online player alts with their name, archetype, level, zone and looking for group status. Bug? May search only the current zone when first called up, even if "All Zones" is selected. Repeat search to include other maps.

Slash Command	Description
`/selectbuild {0-n}`	Select the currently active build for your alt. The number of builds available varies with alt level and slots earned. There is a 60-second delay between build changes.
`/send channel message_string`	Send message to the named chat channel. You must be a member of the channel and have send privileges. Does not work on predefined system channels.
`/servertime`	Displays the current official (game server) time. See also **`city_time`**, **`localtime`** and **`showtime`**.

`/setdifficulty_level {-1 - 4}`
`/setdifficulty_teamsize {1-8}`*
`/setdifficulty_boss (0|1}`
`/setdifficulty_av (0|1}`
	These commands emulate, and are the basis for the "Notoriety" mission options in the quickchat menu. (That menu is a default popmenu, and so requires matching slash commands to function; those commands can be accessed directly for either convenience or to avoid the tiny menu items.)
	
	As with the in-game menu, each level can be set as follows:
	
	_level: set relative mission level between one level below the owner's level or up to four levels harder. Each level change corresponds to one level change in foes.
	
	_teamsize: set the mission size to 1 to 8 players. This setting is tricky in that **the minimum is the actual number of players**. That is, a setting of 1-3 will be ignored if there are four players on the team. It is mainly for setting the number up, so that a few players (or a solo) can encounter mob sizes and other difficulty levels appropriate to a larger team.
	
	* What is probably a bug, but one of long standing, is that numbers higher than 8 can be entered. They will be reflected in the mission window. The effect is to force the foe level up by one, which is mostly useless, but can push +4 to +5, a useful gambit for high-powered teams.
	
	(The above settings are often referred to in chat and forums by shorthand combinations such as **`+2x4`**, **`+4x8`**, etc.)
	
	_boss: set to 1 and the mission will include a Boss in a solo mission if so designed; set to 0 and the foe will be a Lieutenant instead.
	
	_av: set to 1 and the mission will include an Archvillain in a solo mission if so designed; set to 0 and the foe will be an Elite Boss instead.
	
	These commands have been in the game since the return of Homecoming; I managed to overlook them for some time. Thanks to posts by Hedgefund and SuperPlyx for the info, and Yomo Kimyata for the details.

`/set_title badgename`	Set badge title. (Must be one you have, of course.) Bug: currently does nothing but states your badge title has been cleared, no matter what string is used.

`/sg message_string`
`/supergroup message_string`
	Send message to your super group channel.

Slash Command	Description
/sgenterpasscode	Open Supergroup base entry passcode dialog. Works only when the portal entry dialog is open and cannot be used to avoid entering a passcode. See the more useful **enterbasefrompasscode**.
/sgi *name* **/sginvite *name***	Invite player to join supergroup.
/sgk *name* **/sgkick *name***	Kick player from supergroup.
/sgkickyes *name*	Kick player from supergroup, without confirmation.
/sgleave	Leave your current supergroup.
/sgmode	Toggle supergroup mode.
/sgmodeset {0|1}	Set supergroup mode.
/sgraidinvite	Invite selected player's supergroup to join raid.
/sgraidwindow *daybits hour*	Set your supergroup's raid window. (Functionality uncertain.)
/sgsetdemotetimeout	Sets supergroup demote timeout.
/sgsetdescription string	Sets supergroup description.
/sgsetmotd *message_string*	Sets supergroup MOTD.
/sgsetmotto *message_string*	Sets supergroup motto.
/sgstats	Display supergroup info in chat window. Appears to have no effect.
/shaderdetail {0|1|2}	Controls shader detail level. Variations are minor. Default: 0. Warning: changing this setting will cause the display to return to the title card for a few seconds.
/shadowvol {0-4}	Controls whether or not shadow volumes are drawn and colors them variously. Only 0 and 2 have reasonable uses; other settings appear to be dev/testing use only. **0**—off, shadows appear normally. **1**—green shadows. **2**—no shadows. **3**—white shadows. **4**—blue shadows.
/show *window_name*	Causes the given window to be shown. (Synonym: **window_show**) Has no specific opposite, although **hide** is sometimes incorrectly cited. Use **window_hide**.
/showbind *keyname*	Returns current bind string for specified key.

Slash Command	Description
`/showfps {0-3}`	Show current framerate and other information as a small boxed number on top right edge. `0`—off. `1`—show FPS. `2`—show FPS and camera POS/PYR `3`—show FPS and camera POS/PYR, large font See also **`graphfps`**.
`/shownewtray`	Opens a tear-away Tray window. May be repeated to open multiple trays. As with the **+** button on the main tray, it will open trays beginning with the last one you had open.
`/showpetnames`	Lists names of all named pets.
`/showtime {0|1}`	Show the **in-game time of day** on screen. This is a 24-hour decimal clock (each 'hour' has 100 minutes) that counts from 00.00 to 24.00, with game noon at 12.00 and midnight at 24.00. The factor is apparently scalable by the system and currently sits at 48… meaning each game day is 30 real-world minutes long. Very useful when waiting for night to hunt certain foes! Note that asking any civilian NPCs with names that begin with E or F will also get you this in-game time. You can also get a current time reading using **`/city_time`**. See also **`localtime`** and **`servertime`**.
`/slashchat`	Starts chat-entry mode and begins the chat entry with the forward slash (**/**); convenient for entering commands. Equivalent to '**`beginchat /`**' in all ways.
`/sliclear`	Clear each FBO to help SLI/CF (0 to disable). For SLI systems only; functionality uncertain.
`/sliFBOs`	Number of SLI/CF framebuffers to allocate (1 to disable). For SLI systems only; functionality uncertain.
`/slilimit`	Limit number of SLI/CF frames to allow in parallel (0 to disable limiter). For SLI systems only; functionality uncertain.
`/speed_turn {1-359}`	Set the number of degrees for each increment of rotate left/right. Used (only?) by **`turn_left`** and **`turn_right`**. Useful upper limit may be about 90, as larger turns are interrupted when the turning key is released.
`/ss {0|1}`	Controls whether or not simple shadows are drawn.
`/startchat`	Starts chat-entry mode with no preloaded text. Compare with **`slashchat`** and **`beginchat`**.
`/stopinactivedisplay {0|1}`	Stops rendering when the game is not the foreground application.

Slash Command	Description
/stuck	Tries to shift your character to the nearest unstuck position; for use when you get stuck between objects or in map flaws. (If it doesn't work, try sending a **/petition** and waiting a bit to see if a GameMaster will help you.)
/supporthardwarelights	Enable support for AlienFX/LightFX case lights. (May only function on next game start.)
/suppressCloseFx [0|1] **/suppressCloseFxDist *feet***	Hides all of **your alt's character effects** when the viewpoint is closer than the **SuppressCloseFxDist** setting. Useful when close camera viewpoint is obscured by powers effects, etc. Does not affect any other item's or character's effects. The practical limit is the maximum viewpoint camera distance, about 120 feet. Setting it at about 5 feet so that your effects disappear when very close is probably the best use. See also **noparticles**, which can suppress all particle effects from all sources.
/sync **/synch**	Try to resynchronize character/client with game server. Use when character cannot be moved, becomes invisible to teammates, etc.
/tabglobalnext	Cycle forward through all chat tabs in all windows. Will open the corresponding chat window if necessary.
/tabglobalprev	Cycle backwards through all chat tabs in all windows. Will open the corresponding chat window if necessary.
/tabnext {0-4}	Cycle forward through all chat tabs in indicated chat window (0-4).
/tabprev {0-4}	Cycle backward through all chat tabs in indicated chat window (0-4).
/tabselect *tabname*	Select the given chat tab. Will open the corresponding chat window if necessary.
/tabtoggle	Make the previously active chat tab the new active tab. Used to flip between two tabs.
/target	Toggles the target window. (Synonyms: **toggle target**, **window_toggle target**)
[TARGET] For more information on custom targeting, see Section 7.	
/target_custom_...	Powerful customizable targeting comand. There are four variants, which conclude with the following suffixes:
...near	Closest target.
...far	Farthest target.
...next	Next target, in near to far order.
...prev	Next target, in far to near order.
Each of these commands can be directed to a specific class of targetable object by one or of these keywords. (Reasonable keywords can be stacked, such as friend notalive.)	
enemy	Identical to target_enemy.
friend	Identical to target_friend.

Slash Command	Description
`defeated / notalive`	Enemy, friend or NPC with zero hit points.
`alive`	Enemy, friend or NPC with nonzero hit points.
`mypet`	Any pet spawned by you.
`notmypet`	Any pet not spawned by you
`base`	Complex, but basically target all targetable objects including doors, glowies, civilians and hidden items. Both seem to work identically (and a bit erratically).
`notbase`	
`teammate`	Any teammate.
`notteammate`	Any non-teammate player.
`/target_enemy_far`	Targets the farthest visible enemy.
`/target_enemy_near`	Targets the nearest enemy.
`/target_enemy_next` `/toggle_enemy`	Cycles through visible targetable enemies in near to far order.
`/target_enemy_prev` `/toggle_enemy_prev`	Cycles through visible targetable enemies in far to near order.
`/target_friend_far`	Targets the farthest friend. A friend is any friendly player or pet, not just teammates.
`/target_friend_near`	Targets the nearest friend.
`/target_friend_next`	Cycles through visible targetable friends in near to far order.
`/target_friend_prev`	Cycles through visible targetable friends in far to near order.
`/target_name` ***`string`***	Target any entity whose name begins with *string*. See also the ***`target_custom_`*** commands and Section 7.
`/team_accept`	Accepts an invitation to a team.
`/team_decline`	Declines an invitation to a team.
`/team_kick_internal`	Kicks a character without warning from team.
`/team_quit_internal`	Quits team without warning.
`/team_select [1-8]`	Select team member (by number in team list).
`/team_task` ***`int int int`***	Select the team task. (Functionality uncertain.)

`/tell` ***`name, message_string`***
`/p` ***`name, message_string`***
`/private` ***`name, message_string`***
`/t` ***`name, message_string`***
`/whisper` ***`name, message_string`***
	Send a message to only one player.
	See also **`telllast, reply`**.

Slash Command	Description

`/tell_last` *`message_string`*
`/tl` *`message_string`*
	Reply to the same person to whom **you last sent** a private message.
	If a message string is included (such as "In combat... wait a minute.") it will send automatically. Otherwise it will just start a return tell and wait for you to type the message and hit Enter.
	This is different from replying to the **last received private message** using **`reply`**.

`/texaniso {0|1|2|4|8|16}`	Sets amount of anisotropic filtering. UI permits only those values shown, but other integer values can be entered. Effect of these interim values uncertain.
`/texwordeditor texname`	Edit the text layout for translatable textures. (Functionality uncertain.)
`/thumbtack x z y`	Sets a thumbtack location point at the coordinates given. Note that '0 0 0' is usually somewhere near the center of most zone maps. Note that the order is X (East-West), Z (altitude), Y (North-South).
`/toggle window_`*`name`*	Show a window if hidden, hide a window if shown. (Synonym: **`window_toggle`**.)
`/trade` *`name`*	Invite player to trade.
`/trade_accept`	Accepts an offer to trade. Not validated.
`/trade_decline`	Declines an offer to trade. Not validated.
`/tray`	Toggles the tray window. (Synonyms: **`toggle tray`**, **`window_toggle tray`**)
`/traysticky {1|2} {0|1}`	Toggles the presence of the secondary (1) and tertiary (2) trays. Accepts other tray values without effect. Does not lock either tray against closure (as does **`alttray`**/**`alt2tray`**).
`/traystickyalt2`	Toggles the presence of the tertiary tray. Operates independently of the secondary tray. Similar to **`alt2tray`** but does not lock the tertiary tray open. There is no corresponding '**`traystickyalt`**' command for the secondary tray, although it appears in older command lists. Use **`traysticky`**.
`/turnleft`	Rotate left a fixed number of degrees (set by **`speed_turn`**). No default bind, could be bound to Ctrl+A as **`+turnleft`**.
`/turnright`	Rotate right a fixed number of degrees (set by **`speed_turn`**). No default bind, could be bound to Ctrl+D as **`+turnright`**.
`/uiskin` *`int`*	Functionality unknown.

Slash Command	Description
`/unbind` *`keyname`*	Unbinds a user-bound key and restores it to the default bind, if any. To unbind a key without restoring the default, use **`/bind <key> ""`** (empty string). To keep a key from having any action even with game defaults and resets, use **`/bind <key> "nop"`** (null placeholder). See also **`unbind_all`** and **`nop`**.

`/unfriend` *`name`*
`/estrange` *`name`*
	Remove player from friend list.

`/unignore` *`name`*	Stop ignoring user.
`/unlevelingpact`	Bring up dialog for quitting a leveling pact.
`/unselect`	Unselects currently selected thing. Bound to ESC by default. See also power_unqueue,
`/up`	Jump or fly up. Bound to SPACE by default.
`/usebumpmaps {0|1}`	Use bumpmaps if available. (Functionality uncertain.)
`/usecelshader {0|1}`	Turns on cel shading effect (primarly, thin black outlines around all characters and objects). Warning: changing this setting will cause the display to return to the title card for a few seconds. This option has been optimized in the Homecoming era and now has multiple adjustments in the graphics menu. (Some hate this look. I think it freshens the game a lot.)
`/usecubemap {0|1}`	Use cube map. (Functionality uncertain.)
`/usedof {0|1}`	Use Depth of Field (DOF) effects if available. Warning: enabling DOF can seriously impact rendering speed and framerate. See **`dofweight`** to set the degree of effect.
`/usefp {0|1}`	Use floating point render target for HDR effects if available. (Functionality uncertain.)
`/usehdr {0|1}`	Use HDR lighting effects (bloom, tonemapping) if available. (Functionality uncertain.)
`/usehq {0|1}`	Use high quality shader variants if available. (Functionality uncertain.)
`/usenewcolorpicker {0|1}`	Use updated color picker in game editors.
`/useenvfence {0|1}`	Use NV fences instead of ARB queries. (Functionality uncertain.)

Slash Command	Description
`/userenderscale {0|1}`	Use renderscaling if available. See also **`renderscale`**.
`/usewater {0-4}`	Use fancy water effects if available. Higher numbers render more detail and advanced effects. It is not clear what the current limits or modes are.
`/vis_scale {0.0-20.0}`	Controls draw distance. Default is 1.0. Set smaller to improve performance, larger to improve your alt's visual acuity. Higher settings reduce the annoying "slowly appearing objects" effect while traveling. Settings past 5-10 may have notable impact on framerate.
`/watching`	List all (user-created) channels that you belong to.
See Section 4.2 for complete information on using window save and load functions.	
`/wdw_load`	Reads window configuration from the file `wdw.txt` in the default installation folder.
`/wdw_load_file` *`filename`*	Reads window configuration from the specified filename in the default installation folder, or, if specified, in a different location.
`/wdw_save`	Saves window configuration to the file `wdw.txt` in the default installation folder. The default save/load commands are probably best for saving a generic window layout to be loaded to each new alt, and then resaving each under an alt-specific file name.
`/wdw_save_file` *`filename`*	Saves window configuration to the specified filename in the default installation folder, or, if specified, in a different location.
`/whereami`	Tells you server and zone/mission map. This command now creates a clickable coordinates link in chat.
`/whoall`	List who's on the current map, in the system chat window.
`/window_color` *`R G B T`*	Changes the window colors and opacity. R-G-B should each be replaced with a number from 0-255, where R=Red, G=Green, B=Blue from 0 (none) to 255 (max) T=Transparency, for which 0 equals invisible and 255=100% solid.
`/window_hide` *`window_name`*	Causes the given window to be hidden.
`/window_names`	Lists all valid window names for use with toggle and scaling commands. See Reference C for a list of names.
`/window_resetall`	Resets all window locations, sizes, and visibility to their defaults. (Use of window save/load commands recommended instead.)
`/window_scale` *`window_name`* `{0.65-5.0}`	Changes the named window to the display scale indicated. Scaling has been increased from the original 2.0 to accommodate much higher resolution screens.
`/window_show` *`window_name`*	Forces the given window to be shown. (Synonym: **`show`**)
`/window_toggle` *`window_name`*	Show a window if hidden, hide a window if shown. (Synonym: **`toggle`**.)

Slash Command	Description

`/zoomin {0|1}`
`/zoomout {0|1}`
	Controls the zooming in and out of the view. Usually used with the + modifier.
	
	As with the **`lookup`**/**`lookdown`** pair, this command pair will accept the 0/1 variable: if both are set to 1 or 0, camera zooming is unaffected; if one or the other is set to 1, the zoom will persist towards one extreme. This usage is not recommended.

A.2 Base Editing Commands

One of the great changes in the Homecoming era is that bases have been freed from the laborious grind of enough Supergroup prestige to pay for them. Any player can start a Supergroup and then build the most lavish base possible. Many have.

But now that base building is no longer the province of a few lofty base commanders, it's necessary to learn the rather quirky, odd process that leads to your own ideal of Commando Barbie's Dream Base.

Starting a Supergroup

To start a supergroup, see the Supergroup Registrar in City Hall, Atlas Park, or Arachnos Building, Port Oakes. (Gold-siders in Praetoria don't get to start a group until they choose Hero or Villain side...)

Go to any base portal (marked on each zone map, click on the portal, and click Your Supergroup Base. In you go. To your small starting room with an exit portal, and not much else. You can move this starting room around, and even isolate it, but you can't delete it.

Click on the "Edit Base" button that will be floating on your screen. (Or type in **`/editbase 1`** if you want to be a real command wonk.)

Your first step should be to expand the "plot" on which your base is built. There is really no reason not to make it the maximum size.

Base Editing Keys

When you enter base editing mode, several things happen. First, you get a god's-eye view and the ability to run right through anything but exterior walls. Second, your command bindset gets changed, with an addendum of about 15 lines that override any prior definition of those keys. In theory, these extensions are removed when you exit editing mode.

☞ The takeaway is that there are "fixed" keys for base editing, but they could be changed or extended.

A general warning: base editing is frequently buggy. One bug I ran into to was that exiting base-edit mode did not always remove those appended commands, and it's startling to have a command bound to DEL or F1 suddenly generate a mysterious "bad command" error. If it happens, reload your bindfile or save and edit it, removing the lines at the end beginning with DELETE "sell".

The default base editing keys are as follows:

» **Left-Click**—select item or option. Place selected item.

» **Left-Double-Click**—center your alt on that spot.

» **Left-Drag**—move selected object. Can be VERY tricky and erratic to move the right object in a crowded or overlapping spot.

☞ If you have trouble moving an item, move your alt so that you are standing somewhat to the right of the spot you want to place the item, and not too far back.

» **R** or **Right-Click**—rotate object 90 degrees on the Z (vertical) axis.

» **TAB**—select next object in view ("target next")

» **SHIFT-TAB**—select previous object in view ("target prev")

» **CTRL-Y**—Redo last Undo; possibly repeat last action under some conditons.

» **CTRL-Z**—**Your endless friend**: undo that last misbegotten action. You will use it frequently.

» Function keys:

  » **F1**—Toggle grid size and snap for placement (¼ , ½, 1. 2. 4, Off)
  F2—Toggle angle snap for drag rotation (Off, 1, 3, 5, 10, 15, 30, 45 degrees).
  (Note that click-rotate is fixed at 90 degrees.)
  F3—Toggle room clipping on and off for object placement. (Lets you push objects through walls.)
  F5—Toggle object placement attachment (Floor, Wall, Ceiling, Surface).

» **Esc**—Cancel selection or action. (Same as general 'abort' command in zones?)

» The shift keys operate on mouse-drag as follows, enabling and restricting object motion…

  » **Shift**: …to vertical (Z axis, up-down).

  » **Ctrl**: …to horizontal (X/Y axes, rank/file).

  » **Alt**: …to rotation on Z axis (vertical axis). (Drag rotation controlled by the **F2** setting above.)

  » **Ctrl+Alt**: …to rotation on X axis (crosswise axis).

  » **Shift+Alt**: …to rotation on Y axis (perpendicular axis).

☞ To find base items (once you start to know their names), you can search through the dozens of tabs and hundreds of items when in "Place Item" mode. It's a bit obscure that **the black oval in the Item menu is a search field**.

Base Editing Slash Commands

The relevant but mostly useless base editing slash commands are as follows. Advanced or addicted base editors could create a loadable bindfile to optimize keys and add commands.

Most of these commands are not in the cross-reference.

Base Edit Slash Command	**Description** (Default Key)
`/editbase {0-3}`	Enable base editing. Arguments are as follows: 0—Exit base editing. Works from anywhere but will reposition alt in Entrance Room. 1—Enter base editor. Works from anywhere but will reposition alt in Entrance Room. 2—Open isometric view of plot, without editing privileges. 3—Open plan (overhead) view of plot, without editing privileges. Warning: some users have reported that calling this command in regular zones results in permanent map breakage—falling through floor even after game restart. The problem may be obsolete, though.

Base Edit Slash Command	Description (Default Key)
`/angle_snap {0-359}`	Set angle snap to degrees for drag rotation.
`/angle_snap_cycle`	Toggle angle snap for drag-rotation (Off-45 degrees). (F2)
`/attach_cycle`	Toggle object placement attachment (F5)
`/base_redo`	Undo "Undo" and/or repeat action (Ctrl-Y)
`/base_select`	Select base object. (Left-click) Works as slash command at point of cursor.
`/base_undo`	Undo last action. Number of steps saved unknown. (Ctrl-Z)
`/center`	Center editing alt on spot indicated. (Left-Doubleclick)
`/grid_snap {0-n}`	Set grid snap for object placement in grid units. Limits unknown; works from small fractions to 50.
`/grid_snap_cycle`	Toggle object placement grid. (F1)
`/room_clip {0,1}`	Set wall clipping on and off.
`/room_clip_cycle`	Toggle wall clipping on and off. (F3)
`/mousedrag`	Enable dragging object. (May work in regular zones, but with no effect or possibly disastrous ones.) (Left-Drag)
`/quit`	Exit/cancel selection or action. (Esc)
`/rotate {0,1}`	Rotate object 90 degrees. `0`–CCW (Right-Click) `1`–CW
`/see_everything {0,1}`	Turn block outlines of all objects on and off. Also shows inherent ‘objects’ like lighting grids.
`/select_next`	Select next present object in series. (Tab)
`/select_last`	Select previous present object in series. (Shift-Tab)
`/sell`	Delete selected object. (Del) ("Sell" would put the value back in base funds, which are now moot.)
`/sg_passcode` ***`string`***	Set base entry code for non-members. The string will be suffixed by a base-specific number, e.g. COMEIN-1234. Any player with this current code can enter the base. Works in regular zones but is SG privilege-controlled.
`/stuck`	Return editing alt to Entrance Room. It can happen.

Base Edit Slash Command	**Description** (Default Key)
`/base_default_sky {0-15}`	Sets the "open sky" style for the entire base. To see it, set any square of a room to maximum ceiling height and select "Open Sky" as the texture. Options come from the various zone styles and are: `0`—Praetoria `1`—Atlas Park `2`—Boomtown `3`—Mercy Island `4`—Grandville `5`—Cimerora `6`—Night Ward `7`—Shadow Shard `8`—Storm Palace `9`—Dense Fog `10`—Rikti Invasion `11`—Zombie Apocalypse `12`—Praetorian Invasion `13`—Lighted Paths `14`—Shadowed Paths `15`—Starlit Space

In-Base Portals

Most base features are decorative or self-explanatory. The portals and beacons named with Hebrew letters (Aleph, Beta, etc.) are for in-base use: put the portal where you want to arrive, and somewhere else create any travel portal with a matching beacon attached. You can use this to create disconnected base rooms for, say, your generators and mainframes.

☞ And remember, All Your Base Are Belong To Us.

A.3 Mission Architect (AE) Commands

Commands for mission architects in Architect Entertainment. Most of these only work inside AE or while in active mission-editing mode.

And that's all I know about this subset of the game. Have to try it some day.

AE Slash Command	**Description** (Default Key)
`/architect`	Activate the mission search menu.
`/missionmake`	Activate the My Arcs menu of Mission Search.
`/missionsearch`	Open the Mission Search window.
`/mmentry`	Choose between making and starting a mission maker story arc.
`/architect_claim_tickets` ***`num`***	Claims ***num*** Architect tickets.
`/architect_invisible`	Toggles Architect test mode invisibility on or off.
`/architect_invincible`	Toggles Architect test mode invincibility on or off.
`/architect_completemission`	Completes the current mission being tested.
`/architect_nextobjective`	Takes you to the next objective in the mission being tested.
`/architect_nextcritter`	Takes you to the next hostile entity in the mission being tested.
`/architect_killtarget`	Defeats your currently selected target in the mission being tested.
`/architect_loginupdate`	Shows amount of Architect tickets available to claim.

A.4 Group List of Slash Commands

Slash commands listed by functional group. Refer to the prior section for details of use. Synonyms are separated by commas. Commands may appear in more than one group as appropriate.

Base and Architect commands are not included here.

Commands in red have issues; see their entries.

Commands in green are obsolete.

Binds & Macros

/bind
/bind_load
/bind_load_file
/bind_load_file_silent
/bind_save
/bind_save_file
/bind_save_file_silent
/keybind_reset, /unbindall
/macro
/macro_image
/macroslot
/showbind
/unbind
nop

Chat & Email

/ac, /arena
/autoreply
/b, /broadcast, /y, /yell
/beginchat
/c, /coalition
/chan_create
/chan_desc
/chan_invite
/chan_invite_deny
/chan_invite_gf
/chan_invite_sg
/chan_invite_team
/chan_join
/chan_leave
/chan_members
/chan_mode
/chan_motd
/chan_send
/chan_timeout
/chan_user_mode
/change_handle
/chat
/chat_cycle
/chat_load
/chat_load_file
/chat_save
/chat_save_file
/chat_set
/chatoptions
/clearchat
/copychat
/emaildelete
/emailheaders
/emailread
/emailsend/, emailsendattachment
/f
/g, /group, /team
/getlocalname
/gfriend
/gfriend_player
/gfriends
/ghide
/gignore
/gignoring
/ginvite
/guide , /helpchat, /h, /hc
/hide
/hideall
/hidefriends
/hidegchannels
/hidegfriends
/hideinvite
/hideprimarychat
/hidesearch
/hidesg
/hidetell
/ignore
/ignorelist
/ignorespammer
/ignorespammer
/l, /local
/link_channel
/link_info

/logchat
/lp
/mailview
/myhandle
/playernote
/playernotelocal
/quickchat
/reply, /r
/req, /request, /sell, /auction
/say, /s
/send
/startchat
/supergroup, /sg
/t, /tell, /private, /whisper
/tabglobalnext
/tabglobalprev
/tabnext
/tabprev
/tabselect
/tabtoggle
/tell_last, /tl
/unhide
/unhideall
/unhidefriends
/unhidegchannels
/unhidegfriends
/unhideinvite
/unhidesearch
/unhidesg
/unhidetell
/unignore
/watching

Movement

/backward
/clicktomove, /ctm
/ctm_invert
/ctm_toggle
/down
/follow
/forward
/forward_mouse
/left
/loc,/getpos
/lookdown
/lookup
/mouse_invert
/mouse_look
/mouse_speed
/nojumprepeat
/playerturn
/right
/target
/target_custom_far
/target_custom_near
/target_custom_next
/target_custom_prev
/target_enemy_far
/target_enemy_prev
/toggle_enemy
/toggle_enemy_prev
/target_friend_far
/target_friend_near
/target_friend_next
/target_friend_prev
/target_name
/turnleft
/turnright
/up

Character Control

/afk
/cc_emote, /cce
/costume_change, /cc
/emote. /e, /em, /me
/face
/first
/info_self
/info_self_tab
/release
/requestexitmission
/respect
/roleplaying
/selectbuild
/set_title
/stuck
/suppressCloseFx
/suppressCloseFxDist
/third
/whereami

Powers Control

/alt2tray
/alttray
/alttraysticky
/clear_tray
/goto_tray
/goto_tray_alt
/goto_tray_alt2
/goto_trays_tray
/incarnate_equip
/incarnate_unequip

/incarnate_unequip_by_slot
/incarnate_unequip_all
/insp_combine
/insp_delete
/inspexec_name
/inspexec_pet_name
/inspexec_pet_target
/inspexec_slot
/inspirationslot
/inspexec_tray
/manage
/next_tray
/next_tray_alt
/next_tray_alt2
/next_trays_tray
/powers_togglealloff
/powexec_abort
/powexec_alt2slot
/powexec_altslot
/powexec_auto
/powexec_location
/powexec_name
/powexec_server_slot
/powexec_slot
/powexec_toggleoff
/powexec_toggleon
/powexec_tray
/powexec_unqueue
/prev_tray
/prev_tray_alt
/prev_tray_alt2
/prev_trays_tray
/shownewtray
/tray
/traysticky
/traystickyalt
/traystickyalt2

Viewpoint Control

/camdist
/camdistadjust
/camreset
/camrotate
/camturn
/canlook
/zoomin
/zoomout

Pets

/clear_petnames
/inspexec_pet_name
/inspexec_pet_target
/pet_select
/pet_select_name
/petcom
/petcom_all
/petcom_name
/petcom_pow
/petoptions
/petrename
/petrename_name
/petsay
/petsay_all
/petsay_name
/petsay_pow
/release_pets
/showpetnames

Targeting

/target
/target_custom_far, /target_custom_near, /target_custom_next, /target_custom_prev
/target_enemy_far
/target_enemy_prev, /toggle_enemy_prev
/target_friend_far
/target_friend_near
/target_friend_next
/target_friend_prev
/target_name

Search & Info

/cmdlist
/comment
/sea, /search
/thumbtack
/whereami
/window_names
/who
/whoall
/window_names

Teams & Friends

/assist
/assist_name
/buffs
/fl, /friendlist
/friend
/team, /g, /group
/getlocalinvite
/getlocalleagueinvite
/gunfriend
/gunfriend_player

/gunhide
/gunignore
/invite, /i
/kick, /k
/leaveteam
/levelingpact
/lfg
/lfg
/lfg_event_response
/lfg_remove_from_queue
/lfg_request_event_list
/lfgset
/lfgset
/link_interact
/link_interact_global
/makeleader /ml
/playernote
/playernotelocal
/setdifficulty_level
/setdifficulty_teamsize
/setdifficulty_boss
/setdifficulty_av
/team_decline
/team_kick_internal
/team_quit_internal
/team_select
/team_task
/trade
/trade_accept
/trade_decline
/unfriend, /estrange
/unlackey, /unlk
/unlevelingpact
/who

Supergroups

/altinvite
/coalition, /c
/coalition_cancel
/coalition_invite, /ci
/coalition_mintalkrank
/coalition_nosend
/coalition_sg_mintalkrank
/demote
/editbase
/enterbasefrompasscode
/findmember
/getcomment
/getglobalname
/getglobalsilent
/ginvite_sg
/lfg_event_response
/lfg_remove_from_queue
/lfg_request_event_list
/nameCaptain
/nameCommander
/nameEnforcer
/nameFlunky
/nameLeader
/nameLieutenant
/nameMember
/nameOverlord
/nameRingleader
/nameTaskmaster
/promote
/raid_invite
/roleplaying
/sgenterpasscode
/sgkick, /sgk
/sgkickyes
/sgleave
/sgmode
/sgmodeset
/sg_passcode
/sgraidinvite
/sgraidwindow
/sgsetcostume
/sgsetdemotetimeout
/sgsetdescription
/sgsetmotd
/sgsetmotto
/sgstats
/sidekick
/supergroup, /sg
/sginvite, /sgi

Auctions

/auctionhouse
/ah
/blackmarket
/wentworths
/mypurchases

UI & Windows

/chat
/chat_save
/clearRewardChoice
/compatible_cursors
/contextmenu
/custom_window
/custom_window_toggle
/dialog_answer

/dialog_no
/dialog_yes
/gamereturn,
 /windowcloseextra
/graphfps
/help, /helpwindow
/info
/info_self
/info_self_tab
/info_tab
/map
/maxfps
/maximize
/maxinactivefps
/maxinactivefps
/maxmenufps
/maxmenufps
/menu
/monitorattribute
/nav
/netgraph
/popmenu
/powers
/quit
/quittocharacterselect
/quittologin
/screen
/screenshot
/screenshottga
/screenshottitle
/screenshotui
/show
/showfps
/stopmonitorattribute
/stopmonitorattribute
/tabglobalnext
/tabglobalprev
/tabnext
/tabprev
/tabselect
/tabtoggle
/thumbtack
/toggle
/unselect
/usenewcolorpicker
/wdw_load
/wdw_load_file
/wdw_save
/wdw_save_file
/window_color
/window_hide
/window_resetall
/window_scale
/window_show
/window_toggle
/window_names

Arena

/ai, /arenainvite
/arena, /ac
/getallarenastats
/getarenastats
/getratedarenastats

UI Graphics

/bloomscale
/bloomweight
/cgshaderpath
/cursorcache
/disable2D
/dofweight
/fsaa
/fullrelight
/fullscreen
/lodbias
/maxcolortrackerverts
/maxrtframes
/noparticles
/norenderthread
/nosunflare
/reloadgfx
/renderscale
/renderscalefilter
/renderscalex
/renderscaley
/rendersize
/renderthread
/screen
/screenshot
/screenshottga
/screenshottitle
/screenshotui
/shaderdetail
/shadowvol
/sliFBOs
/ss
/ss
/stopinactivedisplay
/suppressCloseFx
/suppressCloseFxDist
/sync, /synch
/texaniso
/usebumpmaps

/usecelshader
/usecubemap
/usedof
/useenvfence
/usefp
/usehdr
/usehq
/userenderscale
/usewater
/vis_scale

System

/autoperf
/autorun
/build_save
/build_save_file
/cmdlist
/copydebuginfo
/demorecord
/demorecord_auto
/demostop
/e3screenshot
/editbase
/gmotd
/localtime
/neterrorcorrection
/netgraph
nop
/noreport
/noversioncheck
/option_list
/option_load
/option_load_file
/option_save
/option_save_file
/option_set
/option_toggle
/petition
/priorityboost
/profiler_record
/profiler_stop
/servertime
/showtime
/stopinactivedisplay
/stuck
/supporthardwarelights
/netgraph
/window_names

Unknown

/clearAttributeView
/texwordeditor
/uiskin

Obsolete

/bug
/buy_coh
/chat_beta
/comment_string
/cov
/ex, /exemplar, /lackey, /sk, /rsk, /sidekick
/getglobalnamesilent
/gfriend_player
/kiosk
/lightmaplodscale
/mousepitchmode
/newspaper
/sgcreate
/sgsetcostume
/sidekickaccept, /sidekickdecline
/unmalefactor, /unmal
/unsidekick, /unex, /unexemplar, /unlackey, /unlk, /unrsk, /unsk
/who

REFERENCE B: Key & Mouse Button Names

Unless noted, all keys can be bound with the ALT+, CTRL+ and SHIFT+ modifiers.

Avoid assigning binds to both synonyms of a key; only the last stored will be used and inadvertent overwriting of the first bind will occur.

Many keys, such as the three shift sets and the "lock" keys, will have system actions as well as activating a bind. Use them sparingly if at all.

B.1 Bindable Keyboard Key Names

Key	Notes
A – Z	Main keyboard alphabetical keys. These keys are case-insensitive in bind definitions; F and f are the same key. Use SHIFT+ to bind two commands to the same alpha key based on "case."
1 – 0	Top numeric keys. Each of the symbols above the numbers is bindable as SHIFT+[number]. The numpad keys have different names (**NUMPAD*X***).
F1 – F12	Top function keys.
SPACE	Space bar.
COMMA	,
PERIOD	. (on the main keyboard; the dot on the numpad is **DECIMAL**)
/ **SLASH**	The second character on each of these keys is bindable as SHIFT+[key]
\ **BACKSLASH**	
; **SEMICOLON**	
` **APOSTROPHE**	
- **MINUS**	Top function key.
[**LBRACKET**	The { and } keys are bindable as SHIFT+[and SHIFT+].
] **RBRACKET**	

Key	Notes
[ALT] [LALT] [RALT] [CTRL] [LCTRL] [LCONTROL] [RCTRL] [RCONTROL] [SHIFT] [LSHIFT] [RSHIFT]	The three "shift" key types and their left side/right side codes, which are nine separate options. The three base shift codes can only be used in combination with another key: ALT-T, SHIFT-F9. etc. The six left/right key variants can be used as a synonym for the base code in combination with any key—that is, CTRL-R and LCTRL-R work identically and with both left and right Ctrl keys. The six left and right codes can be used as "tap" keys—for instance, /bind LALT "emote wave" will trigger that emote with a tap of only the left Alt key, and the right Alt key can be separately bound in the same way. To completely confuse things, LCTRL and LCONTROL and its right-side twins are locked synonyms… if you set one, the game will write bind lines for both. Only if you delete both with a "" definition will both disappear. I suggest not using them at all. I believe this is an archaic and potentially confusing option. I recommend against using the left and right shift keys as tap keys, to prevent accidental activation of the wrong command.
BACKSPACE	
END	
ESC	
ENTER	Main keyboard Enter/Return.
EQUALS	= key. The associated + key is bindable as **SHIFT+=**.
HOME	
INSERT	Does not appear to be (re)bindable.
PAGEUP	
PAGEDOWN	
TAB	
SYSRQ	SysReq/PrintScrn key. **ALT+SYSRQ** not functional.
DELETE	
PAUSE	Pause/Break key. (Does not insert pauses.)
NUMPAD0 — NUMPAD9	The numeric keypad number keys.
NUMPADENTER	The numeric keypad **ENTER** key.
DECIMAL	The numeric keypad **Del/.** key.
MULTIPLY	The numeric keypad * (asterisk/multiply) key.
DIVIDE	The numeric keypad / (slash/divide) key.
SUBTRACT	The numeric keypad - (minus) key.
ADD	The numeric keypad + (plus) key.

Key	Notes
UP UPARROW	Extended keyboard arrow keys. Although both keys in each set are separately listed in the default keybinds, they are synonyms in all tested situations. Because the default setup writes out both, it may be best to define both in all cases, so that unintended defaults or leftover custom binds aren't overwritten by the other, and vice versa. Very annoying bug, really.)
DOWN DOWNARROW	
RIGHT RIGHTARROW	
LEFT LEFTARROW	
CAPITAL	Caps Lock key (as tap key)—will toggle Caps Lock as well.
SCROLL	Scroll Lock key (as tap key)—will toggle Scroll Lock as well.
NUMLOCK	Num Lock key (as tap key)—will toggle Num Lock as well.

B.2 Bindable Mouse Button/Action Names

Although most lists, including the current wikis and older iterations of this guide, assumed that only basic mouse buttons could be used for binds, recent work with the base editing commands revealed that there's much, much more you can do with the pointing device.

» The right and left mouse clicks can be rebound or used in combination binds, but the game wiring will still use or attempt to use them for the basic select/open menu functions.

  » It is recommended that **LBUTTON** and **RBUTTON** not be used for any user binds.

  » Since **CTRL+LBUTTON** is wired to the "set/cancel auto power" function, it too should be avoided for user binds.

  » Since **RIGHTDRAG** is bound by default to "canlook" to enable look-around, it should not be rebound unless you are using another key (**LSHIFT**, etc.) for the look function.

» All mouse buttons can be combined with **Ctrl** / **Shift** / **Alt**.

» Note that the **powexec_location** command allows many power executions that otherwise need a mouse click, especially the new **cursor** keyword, which requires no mouse input at all.

Mouse Button	Notes
LBUTTON **LEFTCLICK**	Left mouse button (click operation). These two terms are synonyms, with **LEFTCLICK** overwriting **LBUTTON** in the default file order. User binding of either keyname, or **CTRL-LBUTTON**, is strongly discouraged.
RBUTTON **RIGHTCLICK**	Right mouse button (click operation). These two terms are synonyms, with **RIGHTCLICK** overwriting **RBUTTON** in the default file order. User binding of either keyname is strongly discouraged.
MBUTTON	Middle mouse button, or mousewheel click. May not be bindable in all systems.
MOUSECHORD	Combination of the left and right mouse keys.
MOUSEWHEEL	Mouse wheel—does not appear to be rebindable from the default use of camera distance adjustment from 0-80 feet. User binding of this keyname is strongly discouraged.
BUTTON4 **BUTTON5** **BUTTON6** **BUTTON7** **BUTTON8**	Additional buttons on advanced pointing devices. Not clear how well this feature is implemented, nor what the limits on multi-button gaming mice might be; possibly driver-dependent. No secondary (drag, double-click) options appear to be supported.
LEFTDOUBLECLICK **RIGHTDOUBLECLICK** **MIDDLEDOUBLECLICK**	Double-click of the specified key. Extremely powerful options when combined with the powexec_location command.
LEFTDRAG **RIGHTDRAG**	Pointer-drag with either key depressed. User binds to **RIGHTDRAG** may have unpredictable results and should be avoided.

B.3 Joystick/Controller Button Names

CoX is not a controller-friendly game. It is designed entirely around a keyboard+mouse, WASD control system.

That said, keywords do exist to map binds and macros to joystick and controller buttons. This table lists the key names; *vaya con Statesman* in getting them to work with your controller. It will take a combination of button mapping, game adjustments and controller-driver tuning.

A useful trick is to load a sub-bindfile that maps every one of these keynames to a short Local-channel ID string: **`/bind JOY1 "l JOY1"`** and so forth. Then you can park somewhere very remote, load the binds, and map out your controller's keys while entertaining NPCs within range. Doing this in a base is perhaps the best idea. (Doing it so it spams a global channel, as I once managed, is a *bad* idea.)

A set of such test files can be downloaded from the ***Heroica!*** website.

Controller interfacing was improved in I27, including a virtual mouse mode. This mode and some new slash commands have not been tested.

Joystick Button Names		
JOY1	JOY9	JOY17
JOY2	JOY10	JOY18
JOY3	JOY11	JOY19
JOY4	JOY12	JOY20
JOY5	JOY13	JOY21
JOY6	JOY14	JOY22
JOY7	JOY15	JOY23
JOY8	JOY16	JOY24
		JOY25
Joypad Key Names		
JOYPAD_UP		
JOYPAD_DOWN		
JOYPAD_LEFT		
JOYPAD_RIGHT		
POV Hat Key Names		
POV1_UP	POV2_UP	POV3_UP
POV1_DOWN	POV2_DOWN	POV3_DOWN
POV1_LEFT	POV2_LEFT	POV3_LEFT
POV1_RIGHT	POV2_RIGHT	POV3_RIGHT
X/Y Joystick Commands		
JOYSTICK1_UP	JOYSTICK2_UP	JOYSTICK3_UP
JOYSTICK1_DOWN	JOYSTICK2_DOWN	JOYSTICK3_DOWN
JOYSTICK1_LEFT	JOYSTICK2_LEFT	JOYSTICK3_LEFT
JOYSTICK1_RIGHT	JOYSTICK2_RIGHT	JOYSTICK3_RIGHT

B.4 Virtual Mouse Mode

- **(from the I27 patch notes)**
- Implemented virtual mouse mode for XInput gamepads
- Holding both triggers now enters virtual mouse mode:
  - The left stick moves the mouse pointer. This is analog, with a small amount of acceleration past halfway pressed
  - Moving the right stick up and down moves the mouse wheel
  - Moving the right stick left and right passes through to your normal binds, allowing you to turn the camera
  - The A/Cross button is the left mouse button
  - The B/Circle button is the right mouse button
  - The Y/Triangle button is the middle mouse button
  - The d-pad will snap your mouse pointer around the screen
  - The X/Square button will snap your mouse pointer to the center of the screen

Controller Slash Commands

Added with I27. Not tested.

Controller Slash Command	Description
`/controller_modifiers <first> <second>`	Allows setting two controller buttons as modifiers
`/controller_vmouse <LMB> <RMB> [MMB] [Snap]`	Configures virtual mouse mode buttons
`/extra_modifiers [mod1] [mod2] [mod3] [mod4]`	Allows setting up to four extra modifiers.

Frankly, I'm gonna wuss out here and leave you to the patch notes and community support until I can experiment with my own DS4. CoX just isn't a very controller-friendly game in any case… I have heard of few who claim real success with controllers.

REFERENCE C: Window & Menu Names

- All these window names can be used with the window control commands—**`toggle`**, **`windowscale`**, etc.
- All windows are scalable from 0.65 to 5.0 using the **`windowscale`** command. It may be possible to scale non-callable windows only when they are open; grab your chances!
- Windows with a tick in the **O** column can be opened with a command. Some windows cannot be opened except under certain conditions (e.g, if you're not a Mastermind, you can't open a Pet window).
- Windows with a tick in the **C** column can be closed with a command.
- Windows with names highlighted in green can be used as direct slash commands. For example, the map window can be controlled with window commands or toggled with **`/map`**. A few instances of parallel window commands have been included (in green) as well.
- Windows with names highlighted in red are problematic or unknown.

C.1 Window Names

Window Name	O	C	Description
`actions`	x	x	Actions window.
`auction`		x	Auction window. Window name can only be used to close the window. May be summoned anywhere except inside bases.
`/ah` **`/auctionhouse`** **`/blackmarket`** **`/wentworths`**	x	x	
`badge`	x	x	Badges window.
`badgemonitor`			Badge award window?
`browser`			*Unknown.*
`chansearch`	x	x	Channel Search window

`chat, chat0`
`chat1`
`chat2`
`chat3`
`chat4`
	x	x	Chat windows (not tabs).
			Chat windows will be opened even if they have no tabs assigned.

`clue` **`clues`**	x	x	Clues window.
`combatmonitor`			Combat Monitor display—a line-by-line configurable display window for character and combat attribute numbers. Cannot be summoned or closed directly. See Section C.2, below, for detailed usage information.

Window Name	O	C	Description
`combatnumbers`	x	x	Combat Numbers window, opened with the Combat Attributes menu item on the Powers window. Provides the source attributes for the Combat Monitor window.
`compose`	x	x	Email Composition window.
`contact` `contacts`	x	x	Contacts window.
`contactdialog`		x	Contact mission description window.
`contactfinder`	x	x	Contact Finder window. Comes up null if called directly.
`convertenhancement`		x	Convert Enhancement window.
`costume` `costumeselect`	x	x	Costume window.
`cvg`		x	Custom Villain Group window (AE).
`defeat`			Defeated window.
`email`	x	x	Email window.
`enhancements`	x	x	Enhancement tray.
`friend` `friends`	x	x	Friends window.
`health` `status`			Health/Level/Main Menu bar. Cannot be summoned or hidden. This window's size is directly coupled to the menu "Window Scale" slider. Changing either will result in a corresponding change.
`help`	x	x	Help window. Not the same as **`/help`** command.
`incarnate`	x	x	Incarnate window.
`info`		x	Object Info window.

`insp`
`insps`
`inspiration`
`inspirations`
	x	x	Inspirations tray.

`league`	x	x	League window. Not the same as **`/league`** command.
`lfg`	x	x	LFG window. Not the same as **`/lfg`** command.
`lfgdialog`			LFG Dialogue window.
`loyaltytreeaccess`			*Unknown.*
`lwcui`			*Unknown.*

Window Name	O	C	Description
mainstoreaccess			*Unknown.*
map	x	x	Map window.
mission	x	x	Mission window.
missionmaker	x	x	Mission editor window (AE).
missionreview	x	x	Mission Review window (AE).
missionsearch	x	x	AE mission search window (AE).
missionsummary		x	Mission Summary window (AE).
nav compass	x	x	Navigation window.
newfeatures		x	New Game Features window.
options	x	x	Options dialog.
paragonrewards	x	x	Paragon Rewards window.
pet	x	x	Pet window. Useful when new MMs lose their window. The Pet Options window can only be opened with **/petoptions**.
petition	x	x	GM Petition dialog.
playernote	x	x	Player note window. Opens with slash command if player name is specified.
power powers tray	x	x	Powers tray. Note inconsistency of **powers** window name and **/powers** command, below!
powerlist /powers	x	x	Power List window.
quit	x	x	Quit dialog. Flashes quit countdown when called as slash command—bug?
razertray			*Unknown—Razer mouse features?*
recipe recipes	x	x	Recipe window.
salvage	x	x	Salvage window.
salvageopen			*Unknown.*
scriptui	x	x	*Unknown.*
search /sea	x	x	Player Search window.
sg super supergroup	x	x	Supergroup window. Not same as **/sg** or **/supergroup** command.
store		x	Store dialog.

Window Name	O	C	Description
`target`	x	x	Target window.
`team` `group`	x	x	Team window. Not the same as **`/team`** or **`/group`**.

`tray`
`tray1`
`tray2`
`tray3`
`tray4`
`tray5`
`tray6`
`tray7`
`tray8`
	x	x	Power trays.
			`tray` is a synonym for `power` etc.
			`tray`*`n`* controls one of eight tear-off power trays like those created by the + menu item. There does not seem to be a way to call up an individual number of tray except by using the tray clickers. However, if each tray is set to a specific tray, that instance will be persistent.
			Floating trays may be closed by right-clicking on the tray number. Note also that you can reconfigure the tray layout from this menu!

`vault`	x	x	Vault window. May be summoned anywhere!

C.2 Using the Combat Monitor Window & `/monitorattribute` Command

One of the most obscure user-interface options is a tiny window that can be used to list any of a vast number of character/build attribute values, one per line. All of these numbers can be looked up in the Combat Numbers window (and health and endurance are of course shown in the main display bars) but advanced players may find it useful to have a customized window displaying realtime health, defense, combat and other numbers of their choosing.

The Combat Monitor window consists of one or more lines with a label and a value, taken from your current status. If there is a limit to the number of lines that can be displayed, I don't know it. Lines have to be added to the window one at a time, using one of two methods.

Window Commands

The most direct way to build an Combat Monitor window is to open the Combat Numbers menu (using **`/show combatnumbers`**, or the menu item on top of the Powers window). From the list of attributes, right-click on the ones you want displayed.

With one or more attributes in the Combat Monitor window, right-click on that window for commands to remove and re-order the attributes shown, or to close all of them.

The Combat Monitor window has some peculiar limits for control. It cannot be called up or closed using the usual window commands (**`show`**, **`toggle`**). It must be opened by adding lines, as above, or closed by either removing all lines or using the right-click | Close All command.

The window can, however, be scaled using the **`windowscale`** command… but only when it is currently displayed. The window name is **`combatmonitor`**.

The window-click method of configuration is simple but can be tedious to keep adjusting and re-calling displays. So…

Slash Commands & Binds

Attributes can be added and removed from the Combat Monitor window by executing a slash command to call them. The basic usage is:

```
/monitorattribute attribute_name
```

which will add the named line to the window. Since this is a toggle command, using it with the same argument will add and then remove the relevant line from the window.

The corresponding command to remove lines from the window is:

```
/stopmonitorattribute attribute_name
```

to turn off and remove each window line. Since the first command works well as a toggle, there may not be any good use for the 'stop' command.

☞ Given the clumsy method and long command strings required, and the slow process of using the window method, this is a *really* good place to use a set of binds to open and close selected lines.

The arguments that can be used to specify each line of data are **extensive**. Serious players should review the list that follows and the many panels of the Combat Numbers menu and choose which values will be useful in a small, realtime, heads-up display.

☞ Many entries in the table include a breakdown for base value and those added by archetype, powerset and buff enhancement.

The argument value for for each display value is the full string name of the equivalent line in the Combat Numbers list; multi-word names must be fully spelled out. Case does not matter and quotes are not necessary.

A few of the most useful lines that can be added to the Attribute Monitor are as follows:

Combat Number	Display Line Contents
`Current Hit Points`	Current hit points (green bar)
`Current Endurance`	Current endurance level (blue bar)
`Endurance Consumption`	Current endurance consumption rate
`Recovery Rate`	Endurance recovery rate (blue bar)
`Regeneration Rate`	Hit Points regeneration rate (green bar)
`Experience Debt`	Current experience debt
`Experience to Next Level`	Experience needed to reach next level
`Influence`	Current Influence/Infamy

A basic setup for the Combat Monitor is included in the GABB, and is toggled with the **`F8`** key.

There are many more detail values in the Combat Numbers window. A complete list, by window category, follows.

Combat Number Listing

Base Numbers
- `Current Hit Points`
- `Max Hit Points`
- `Current Endurance`
- `Max Endurance`
- `Regeneration Rate`
- `Recovery Rate`
- `Endurance Consumption`
- `To Hit Bonus`
- `Accuracy Bonus`
- `Last Hit Chance`
- `Damage Bonus`
- `Healing Bonus`
- `Healing Receive Bonus`
- `Recharge Time Bonus`
- `Endurance Discount`
- `Stealth Radius (PvE)`
- `Stealth Radius (PvP)`
- `Perception Radius`
- `Range Bonus`
- `Threat Level`
- `Inherent`
- `Level Shift`
- `Experience to Next Level`
- `Experience Debt`
- `Influence`

Movement Numbers
- `Running Speed`
- `Flying Speed`
- `Jumping Speed`
- `Max Jump Height`

Damage Resistance Numbers
- `Smashing Resistance`
- `Lethal Resistance`
- `Fire Resistance`
- `Cold Resistance`
- `Energy Resistance`
- `Negative Energy Resistance`
- `Psionic Resistance`
- `Toxic Resistance`

Defense Numbers
- `Base Defense`
- `Ranged Defense`
- `Melee Defense`
- `AOE Defense`
- `Smashing Defense`
- `Lethal Defense`
- `Fire Defense`
- `Cold Defense`
- `Energy Defense`
- `Negative Energy Defense`
- `Psionic Defense`

Debuff Resistance Numbers
- `Regeneration Resistance`
- `Recovery Resistance`
- `To Hit Resistance`
- `Recharge Time Resistance`
- `Defense Resistance`

Status Effect Protection Numbers
- `Hold Protection`
- `Immobilize Protection`
- `Stun Protection`
- `Sleep Protection`
- `Knockback Protection`
- `Confuse Protection`
- `Terrorize Protection`
- `Repel Protection`
- `Teleport Protection`

Status Effect Resistance Numbers
- `Hold Resistance`
- `Immobilize Resistance`
- `Stun Resistance`
- `Sleep Resistance`
- `Knockback Resistance`
- `Confuse Resistance`
- `Terrorize Resistance`
- `Placate Resistance`
- `Taunt Resistance`

REFERENCE D: Emote Codes

Emotes: the essential element of roleplaying and endless fun for everyone else!

D.1 Emote Basics

CoX has hundreds of emote codes that can be executed by almost any alt and many pets. These codes can be executed at almost any time using the slash code **`/emote`**, **`/em`**, **`/e`** or (amusingly) **`/me`**. A subset can also be selected from the QuickChat menu, which is raised by clicking the small button at the right end of the chat text entry window, or by the slash code **`/quickchat`**.

To use emotes in a bind or macro, use **`/em`** ***`emotename`*** as the command string.

The best way to see what each emote does is to find a quiet corner of the map, use camera rotate (default: **`PAGEDOWN`** plus the mouse; GABB, **`SHIFT`** plus the mouse) to spin around so you're looking at your character from the front, and try each one out. Set the chat input to something null like team so mistakes won't get hoots from the other players.

☞ It is common to see emotes in discussion using the **`;`*`emotename`*** format, which is bound to the default keybind set of using the semicolon to start slashchat. I recommend you do not follow this practice unless you never plan to use an advanced keybind set. Use **`/em`** ***`emotename`*** instead.

Some codes change by basic alt body type (male and female sitting positions, mainly) and whether the alt is flying or not.

Note that many of the QuickChat options are similarly named, but include fixed chat bubbles as well.

If you use any emote string besides one of these valid codes, the string will appear in a thought bubble over your head, visible to others and appearing in the Emote chat channel, preceded by your character name. (**`/em wishes he had a beer.`** == "Shenanigunner: *wishes he had a beer.*")

You can combine an emote (usually just one) with a text string as well. (**`/em burp$$wishes he had a beer.`** == "Shenanigunner: *wishes he had a beer." *burrp**)

Fully updated December 2020, after a poke by AboveTheChemist!

This Guide vs the Homecoming Wiki Page

Emotes are one area where the old Paragon wiki page and the newer Homecoming wiki page are useful alternatives as resources. I reviewed the latter in preparing this update and while I only found a few new emotes in all, details such as the column of "fly" variations led to useful updates of this resource.

The Homecoming page is an excellent reference and probably updated more often than this section, but is (IMVHO) a bit harder to use. For reasons that make more sense as a record of game changes than as a useful reference, it's still broken into many sections to separate the various bonus, earned and add-on emotes that are now universally available. It does have much collateral info such as what Issue or add-on each emote was introduced in, and some animated examples.

All of the entries in the chart below, including the new "fly" variation column, were individually validated. I believe my section here is still unique in noting the different sit emotes and other details, and is the first complete listing of the "hybrid" emotes, which are probably a parsing bug.

But you can never have enough references to a game this massive, right?

- See the following subsections for information on:
  - Using costume-change emotes.
  - How the different body types “sit.”
  - The new “hybrid” emotes.
- See Section 6.4 for the somewhat tricky process of using pet emotes in binds.

D.2 Emote Keywords

In the table (starting on the next page):

- **STAT**[ic] means the position holds, persists or repeats until interrupted by a movement key or an auto power like Hasten. Mouselook can sometimes be used to look around and turn the alt while the emote continues
- **SND** means the animation is accompanied by audio, sometimes very soft. Some quite loud.
- **FLY** means that the emote:
  - **Y** — has a different form while floating or flying
  - **S** — is the same pose (standing up or kneeling in midair)
  - **N** — does not work while flying.
  - It’s worth noting that the **`flypose`** emotes ONLY work while flying…
- **LOCKED** emotes must be unlocked by a certain mission completion or badge. Many emotes once tied to add-on packs or locked access are now freely available, but a few remain “earned.”

I have combined some codes out of alphabetical order, and under generic headings, for clarity.

☞ The I27-3 update included a subtle fix for the emotes that generate an object (**`alakazamreact`** and all the **`eat`** variations) that synchronize the object for all viewers.

Emote Code	STAT?	SND?	FLY?	Animation/Artifact	Notes & Description
afk newspaper	Y	N	S	Read newspaper	Good basic "I'm waiting" or AFK emote. The **/newspaper** slash code no longer works.
afraid cower fear scared	Y	N	Y	Cower in fear	You too can be a civilian.
airguitar dance10	Y	N	S	Play mad air guitar riff	
alakazam	N	N	N	Dramatic magician gesture	
alakazamreact	N	N	Y	Turn into a random object (and back)	A major hoot. Just try it.
akimbo wings	Y	N	N	Stand with hands on hips	See also STANCES.
angry	N	N	N	Akimbo with an attitude	
assumepositionwall	Y	N	Y	Stand against wall as if to be searched	Looks pretty stupid unless you stand facing a wall or other surface as closely as you can before executing. Interesting while flying, though.
atease	Y	N	S	Stand at ease	
attack	N	N	S	One-arm motion forward	
backflip flip	N	N	Y	Perform a backflip	Looks great flying.
batsmash	Y	N	S	Animated lay about you with a baseball bat	
batsmashreact	Y	N	S	Animated react to getting hit with a baseball bat	
bb boombox dropboombox	Y	Y	N	Character places boombox in front of him/her	The basic socialization, showoff and time waster emote—haul out the boombox and dance. The tune will be randomly selected from those listed under **bb…**
beatchest tarzan	N	Y	Y	Chest-pounding	Growl for male/huge. Scream and drumbeats for female.
biglaugh laugh2 laughtoo	N	N	S	Hearty laugh or chuckle	
bigwave overhere	N	N	S	Animated big wave	

Emote Code	STAT?	SND?	FLY?	Animation/Artifact	Notes & Description

`bbAltitude`
`bbBeat`
`bbCatchMe`
`bbDance`
`bbDiscoFreak`
`bbDogWalk`
`bbElectroVibe`
`bbHeavyDude`
`bbInfoOverload`
`bbJumpy`
`bbKickIt`
`bbLooker`
`bbMeaty`
`bbMoveOn`
`bbNotorious`
`bbPeace`
`bbQuickie`
`bbRaver`
`bbShuffle`
`bbSpaz`
`bbTechnoid`
`bbVenus`
`bbWahWah`
`bbWindItUp`
`bbYellow`
	Y	Y	N	Boombox + dance	Using these codes will select specific boombox tunes instead of randomly choosing one of them.
					Avoid newbie zones where as many newbies as possible attempt to set up competing-tune boomboxes. For one thing, it can crash your client. For another, it can crash your brain.
					See also `drumdance`.

`binoculars`	Y	N	S	Look through binoculars	
`blankfiller`				NA	Appears to be the emote equivalent of "nop" for slash commands. Does nothing but generate error or interrupt current emote.
`boast` `showoff`	N	N	Y	Wave arms to sides and nod	"Ain't I great!"
`bow` `sorry`	N	N	N	Bow	Asian-style bow.
`bowdown`	N	N	S	Demand person before you bow down	
`burp`	N	Y	N	Burp	Audible. Look, I'm a *rude* Warwolf!
`buzzoff` `goaway`	N	N	S	Shooing motion with hand.	
`calculate`	Y	N	Y	Write and regard foggy runes in front of you	

Emote Code	STAT?	SND?	FLY?	Animation/Artifact	Notes & Description
camera	Y	N	Y	Take pictures (continuously) with old-fashioned Speed Graphic camera.	Cannot be interrupted until first part of animation is completed. See also **cameraphone**, **smartphone**, **slrcamera**
cameraphone	N	N	Y	Take out your smartphone and snap a picture.	Cannot be interrupted until first part of animation is completed.
cardtrick	Y	N	Y	Produce a deck of cards, do a trick, toss them in the air	The toss effect repeats every few seconds. See also **juggle**.
catchbreath	Y	N	Y	Take big breath and rest hands on knees.	
cellphone smartphone	Y	N	Y	Talk on old cel phone or new smartphone	Wow, that's a big phone! See also **text** and **walkietalkie**.
champion	N	N	S	Clasped-hands victory shake	See also **victory**.
cheer	Y	N	S	Shake-fists encouragement	
chicken	N	N	S	Do the chicken dance	
clap	N	Y	S	Applaud	Audible over Local distance.
clipboard	Y	N	S	Write on clipboard	
coffee ddcoffee	Y	Y	S	Coffee with lid	Must have the **Old Fashioned** badge (fly through donut in Faultline) to unlock. See also **drink**, **teabag**.
coffee2 ddcoffee2				Coffee without lid	
collapse swoon	Y	Y	N	Wobble, take a step and fall down backward or forward	
[DECIDE] cointoss coin flipcoin	Y	N	S	Animated coinflip motion; show head or tail coin overhead	Make a choice for the team or group, or yourself. Coin remains until interrupted. See also **dice** and **paper**.
crack knuckle knuckles	N	N	S	Crack knuckles	Loud sound effects!
crossarms	Y	N	N	Cross arms	
crouch	Y	N	N	Crouch down, frog style	
curseyou noooo	N	N	S	Animated shaking fist at the heavens in dismay	Note four O's on latter.

Emote Code	STAT?	SND?	FLY?	Animation/Artifact	Notes & Description
dance	Y	N	S	Animated dancing	Several random dances; repeat emote for others. You can also use the dance*n* commands following to select specific dances. See also **drumdance** and **bb**. These also combine for the "hybrid" emotes, see Section D.5 below.
dance1	Y	N	S	Cha-cha dance	
dance2	Y	N	S	Rah-rah dance	
dance3	Y	N	S	Twist dance	
dance4	Y	N	S	Hands waving in air dance	
dance5	Y	N	S	Hands in air hop dance	
dance6	Y	N	S	High-energy twist dance	
dance7 dancerobot robodance	Y	N	S	Elaborate robot dance	
dance8 karatedance	Y	N	S	Martial-arts dance	
dance9 popdance	Y	Y	S	Moonwalk dance	
dance10 airguitar	Y	N	S	Air guitar	
dance11 discodance	Y	N	S	Singer dance	
despair	Y	N	S	Fall to knees in despair	
[DECIDE] dice rolldice	N	N	S	Dice roll motion; show die overhead	Make a choice for the team or group, or yourself. Die fades after a few seconds. See also **cointoss** and **paper**.
dice7	N	N	S	Roll dice, always getting 7	LOCKED Burkholder's Bane badge (Hero) Deuces Wild badge (Villain)
dig	Y	Y	N	Dig hole with shovel	
disagree	N	N	S	"No" wave with short lecture animation	

Emote Code	STAT?	SND?	FLY?	Animation/Artifact	Notes & Description

`donut`
`eatdonut`
`donut1`
	Y	N	S	Pink-glazed with sprinkles
				Note: **`donut1`** is somewhat higher resolution than **`donut`**.
					Eat a donut.
					See also **`eat`**.

`donut2`				Chocolate glazed	
`donut3`				Chocolate glazed with sprinkles	
`donut4`				Plain/sugar glazed	
`donut5`				White glazed with blue sprinkles	
`jellydonut`				Jelly donut with cherry filling	
`jellydonut2` `lemondonut`				Jelly donut with lemon filling	
`halloweendonut` `spookydonut`				Chocolate glazed with black/orange sprinkles	

`christmasdonut`
`holidaydonut`
`winterdonut`
				White glazed with red/green sprinkles	

`drat`	Y	N	S	Thump both fists and stomp	Express frustration in a friendly way.
`drink`	Y	N	N	Drink from glass	See also **`eat`** and **`donut`**.
`drinkenriche`	Y	N	S	Drink from bottle	Works in fly, but as **`drink`**.
`drum`	Y	Y	N	Pound on huge tribal drum	Loud sound effect.
`drumdance`	Y	N	N	Raindance	See also **`bb`** and **`dance`**.
`drumlow`	Y	Y	S	Pound on small tribal drum	Bongo sound effects.
`dustoff`	N	N	S	Brush off hands	

`eat`
`eatfood`
`food`
	Y	N	S	Eat food item	Alternates between burger, hot dog and sandwich.

`hotdog`				Eat hot dog	
`hamburger`				Eat burger	
`sandwich` `sandwitch`				Eat sandwich	

`evillaugh`
`elaugh`
`muahahaha`
`villainlaugh`
`villainouslaugh`
	N	N	S	"Bwah-ha-hah" villain laugh	(How many synonyms are needed for one emote!?)
					Note three "ha"'s.

`experiment` `mixformula`	Y	Y	Y	Pour liquid between two flasks	

Emote Code	STAT?	SND?	FLY?	Animation/Artifact	Notes & Description
`explain`	N	N	S	Animated "hold it," with short lecture animation	Cannot be interrupted until first part of animation is completed. See also **`lecture`**.
`facepalm` `doublefacepalm`	N	Y	Y	Smack own face with one or two hands	Choose based on just how stupid that noob's move was.
`fancybow` `elegantbow`	N	N	N	Animated elaborate bow	

`[EXERCISE]`
`jumpingjacks`
`kata/martialarts`
`pushup`
	Y	Y	S	Do jumping jacks, martial arts kata, or pushups	See also **`SPORTS`**. The last looks stupid in fly.

`fireworkbloom`
`fireworkrocket`
`fireworksparkle`
	N	Y	Y	Shoot off roman candles: one "pop," four "pops" and one "crackle."	

`fishing`	Y	N	S	Fish with long pole	

`flashlight`
`flashlightdown`
`flashlightup`
	Y	N	S	Looking around with large flashlight over shoulder. Specific commands control pointing direction; the first seems to be random or alternate.	Does not appear to project light.

`flex1/flexa`
`flex/flex2/flexb`
`flex3/flexc`
	Y	N	S	Animated bodybuilder poses	Impress newbies and that cute controller by doing your Arnie impression. Three different poses for your convenience.

`flippingcoin`	Y	N	S	Flip coin gambler style	Not same as **`flip`**; does not generate "result."
`floatbooks`	Y	N	S	Float three books in front of you and appear to study them	Really should work right in fly....
`[FLY]` These four emotes work only when you are already flying. If you pause, your character will revert to the standard flying pose. There is no emote to return to the standard flying posture. A fly-forward plus emote keybind is recommended for regular use, or a keybind that cycles through the options.					
`flypose1`	Y	N	Y	Fly with fists out front	
`flypose2`	Y	N	Y	Fly with one fist out front	Superman pose
`flypose3`	Y	N	Y	Fly with hands flat out front	Swan dive pose
`flypose4`	Y	N	Y	Fly with fists to sides	Invisible hang glider pose
`frustrated`	Y	N	S	Animated shake both fists	Stays in fist-clenched posture after shake.
`getsome` `kissit`	N	N	S	Turn fanny to front, pat it	Ruuuuude. Love it.
`ghoulflex` `tantrum`	Y	?	Y	Praetorian ghoul animation, and death throe animation	LOCKED — not validated The Great Escape event (Praetoria).

Emote Code	STAT?	SND?	FLY?	Animation/Artifact	Notes & Description
grief	Y	N	S	Grief on knees	Stays on knees after initial animation. Not to be confused with **propose**, which also leads to grief. See also **despair**.
hand talktohand	N	N	S	Hand out in "talk to the hand!" style	Yeah, right, enough outta you.
handsup surrender	Y	N	S	Hands in the air, alternate kneeling and standing	
hi wave	N	N	S	Wave	
hiss	N	Y	Y	Hiss and spit like a cat	See also **sniff**, **roar**, **howl** and **savage**.
holdtorch	Y	Y	S	Hold a tall flaming torch	Not sure if it actually projects any light.
hottemper	N	Y	S	Rage until steam comes out of your ears	
howl	N	Y	N	Howl like a Warwolf	See also **sniff**, **roar**, **hiss** and **savage**.
hmmm plotting	N	N	S	Stare into space and rub chin	
huh shrug what	N	N	S	Shrug	
innertube	Y	*	N	Float around in a vinyl innertube. Base command cycles among the alternatives.	Only work in water deep enough to swim in, such as the Ouroboros pool, Founders Falls canals and oceans. Canceled on movement; you can't scoot around in them. * Sound effects when called.
innertube1				Strawberry donut innertube	
innertube2				Chocolate donut innertube	
innertube3				Neon green sport innertube	
innertube4				Cow innertube (just go see)	
innertube5				Naval Academy innertube (blue-gray camo)	
innertube6				Lifeguard innertube	
inspiration	N	N	Y	Think and get bright idea	
invent	Y	N	Y	Manipulate a cool luminescent grid thingy	That or it's a new-gen Rubik's Cube. Used whenever a character is interacting with an invention table.
jackhammer	Y	Y	N	Use jackhammer	Noisy.

Emote Code	STAT?	SND?	FLY?	Animation/Artifact	Notes & Description
[JUGGLE] juggle juggleelectricity jugglefire jugglemagic	Y	N	Y	Juggle three balls. The first command conjures three colored balls. The other three add blue electricity, fire and glow effects; magic also adds sparkly auras.	See also **cardtrick**. For fun, keep changing among these options.
kneel	Y	N	S	Kneel down	
laptop	Y	N	S	Work on laptop that appears on pedestal	Occasionally seem to experience computer trouble. (Is this a backhanded joke at a "boss" key?) See also **type**.
laugh	N	N	S	Hands-on-hips laugh	Why, yes, I *am* Errol Flynn!
lecture	N	N	S	Deliver a lecture	See also **explain**.
liedown	Y	N	Y	Lie down on ground as if sleeping.	See also **sleep**.
listenpoliceband	Y	Y	N	Whip out your way-cool holographic police radio	Used for the police band mission contact. Can be used by both hero and villain.
listenstolen- policeband	Y	Y	N	Whip out your stolen holographic police radio	LOCKED — not validated Outlaw badge (villain only). (no hyphen in use...)
lookup	Y	N	S	Look up at something.	
lotus yoga	Y	N	Y	Animated lotus position	Sophisticated resting posture.
[LOYALTY] heroloyal rogueloyal vigilanteloyal villainloyal	Y	N	Y	Pose on one of four specific medallions, with other animations	All can be used by any alt type or alignment.
marriageproposal propose	Y	N	S	Down-on-knee proposal	Was originally part of the wedding pack. See also **throw...**.
matablet	Y	N	Y	Use data pad	LOCKED — not validated Mission Engineer badge.
no dontattack	N	N	S	Animated wave-hands "no"	See also **disagree**.
nod	N	N	S	Animated nod	

Emote Code	STAT?	SND?	FLY?	Animation/Artifact	Notes & Description
none	N	N	Y	No effect by itself; cancels any emote an returns to normal stance.	
observedice	Y	N	N	Closely watch dice game on ground	See also **dice**.
offergift **receivegift** **opengift**	N	N	*	Produce and offer a gift; Express surprise and take gift; Open gift to a spray of confetti	Open only standing in flight. (May only be active during Winter event.)
pamphlet	Y	N	N	Hand out flyers or pamphlets from your stack	
panhandle	Y	N	N	Animated sit with cup, offering as to passersby, occasionally looking it it disappointedly	One way to bug inf off of high-level players.
[DECIDE] **paper** **rock** **scissors**	N	N	S	Play rock-paper-scissors or Rochambeau; choices appear overhead.	Settle disputes. Animation shows all three icons for five seconds, then your selected one. See also **cointoss** and **dice**.
peerin	Y	N	S	Animated peering in window with hands cupped around face	Occasional look-around to see who's watching.
picklock	Y	Y	N	Kneel and bang on something	
plot **scheme**	Y	N	S	Hunch and rub hands together as if making evil scheme	
point	N	N	S	Animated one-hand point straight ahead	
praise	Y	N	S	Animated salaam on knees	
protest	Y	N	S	Shake large protest sign	Appear to be three different signs that come up at random. All are illegible except for a large STOP, NO and red circle/slash over an indistinct outline of something.
protestloyalist	Y	N	N	Shake large protest sign	Appear to be different signs that come up at random, All combine "no" with a yellow star.
protestresistance	Y	N	N	Shake large protest sign	Appear to be different signs that come up at random. All combine "no" with a blue chevron insignia.
raisehand **stop**	Y	N	S	Animated raise one hand	
readbook	Y	N	N	Read from book	

Emote Code	STAT?	SND?	FLY?	Animation/Artifact	Notes & Description
research	Y	N	S	Animated refer to book, then examine what's in front of you	Circle of Thorns seen doing this in Hollows and elsewhere.
researchlow	Y	N	S	Animated refer to book, then examine what's in front of you, while squatting down	Circle of Thorns seen doing this in Hollows and elsewhere.
roar	N	Y	N	Roar like a Warwolf	See also **sniff**, **hiss**, **howl** and **savage**.
[ROBOT] **robotpowerup** **robotpowerdown**	Y	N	S	Power up emulates a robot being powered up, with aura effects, and then ends. Power down makes the alt hang forward, arms loose, and stay in that position.	See also **robotdance/dance7**.
rooting **wavefist**	N	Y	S	Animated wave fist, hands-to-face shout and clap with sound	Only clapping has sound. Not interruptible.
[SALUTE] **salute** **militarysalute** **praetoriansalute**	*	N	*	Three saluting options. The first delivers a salute and ends. The second holds a salute until interrupted. The third is a very elaborate Roman-style salute.	The first two salutes can be executed (standing) in fly. The third cannot.
savage	N	Y	N	Hop around like an ape	See also **sniff**, **roar**, **howl** and **hiss**. (Is name of this emote perhaps just a tad racist?)
score1 **score2** ... **score9** **score10**	Y	N	S	Hold up score card with 1 to 10 on it, Olympics-style	Show your opinion of another player's move. Fun to use with costume contests, etc. (What, no score zero?)
screen **touchglass**	Y	N	S	Reach out and touch surface in front of you as if not sure it's there, or touch wall-screens	Fabulous animation if character is in a bubble or if you pivot viewpoint so that you're looking right into character's face. Fun.
shocked	Y	N	S	React, then look at hands, shivering	"What have I done!"
shucks	N	N	S	Animated thump one fist	Aw, it was nothing. See also **drat**.
sit	Y	N	S	Sit on ground.	Take a load off. Fun to do on benches, trees, etc. Takes some practice in positioning.
ledgesit	Y	N	N	Sit as if on ledge.	
See section D.3 below for other, advanced **sit** and **ledgesit** commands, which are differentiated by alt gender.					
slap **smack**	N	Y	N	Animated forehand slap or backhand smack	With light burst and slap sound. Combine with **slapreact** from other character for more fun.
slapreact	N	N	S	Reaction to being slapped or struck	

Emote Code	STAT?	SND?	FLY?	Animation/Artifact	Notes & Description
slash slashthroat	N	N	S	Draw finger across throat.	Stop; Shut up, dude; *or* You're dead, you know.
sleep	Y	N	S	Fall asleep standing up, with stream of Z's rising	
slrcamera	Y	N	Y	Use (huge) modern camera	See also **camera, cameraphone**.
smackyou threathand	N	N	S	Threaten to backhand someone	See **slap**.
sniff	N	Y	Y	Sniff around noisily	See also **howl** and **savage**.
[SPORTS] basketball pool soccer	Y	*	N	Standing animations that do fancy basketball dribbles, fancy soccer ball moves, or just stand talking while holding a pool cue	Ball emotes have sound. See also **EXERCISE**.
spraypaint	Y	Y	N	Take out can of spray paint, paint on surface in front of you.	
[COMMON STANCES] idle1 idle2 batlookout [HERO STANCES] stancehero1 stancehero2 [VILLAIN STANCES] stancevillain1 stancevillain2	Y	N	N	Four different standing poses, all with arms at sides. The two idle variants are identical, with arms slightly away, but turn differently. The two stance... variants are identical, with arms closer to the body, and turn differently. The first villain pose is a sort of menacing crouch; the second is with arms tightly folded. The batlookout code has the alt holding a bat in a menacing posture.	All emotes can be used with either hero or villain alts, but (like the sit emotes) have slightly different results for each type. **idle1** is the standard alt posture. See also **akimbo** and **crossarms**.
talk	Y	N	S	Talk as if in conversation	
taunt taunt2 tauntb	N	Y	Y	Two-hand taunt with "hoooah" sound (all identical)	Character stays in combat pose after taunt
taunt1 taunta	Y	Y	Y	One-hand taunt with "aaaaah" sound	Character continues to pound fists with sound effect after taunt
teabag	Y	N	S	Dunk teabag in a teacup	See also **drink, eat**. No, not *that* kind of teabag.
text	Y	N	Y	Take out your smartphone and thumb-chat away.	See also **cellphone** and **cameraphone**.
thanks thankyou	N	N	S	Left-hand gesture	See also **yourewelcome**, which is a mirror-image gesture.
thewave	N	N	S	Vertical "wave" animation	

Emote Code	STAT?	SND?	FLY?	Animation/Artifact	Notes & Description
throwconfetti throwrice throwrosepetals throwsnowflakes* snowflakes	Y	*	Y	Throw confetti, rice or rose petals	*The snowflake may only be active during a Winter event and is the only emote to make a sound.
thumbsup yes	N	N	S	Thumbs-up animation with nod	
trainwhistle	N	Y	Y	Pull cord on train whistle that magically appears	
type typing	Y	N	N	Type as if on keyboard—same as `laptop` but without prop	Great for consoles in missions.
ultimatepower	N	Y	Y	Dramatic "change" or "use power" effect with auras	Matching costume-change emote.
vendor	Y	N	S	Call out like a carnival barker, then make your pitch	
victory	N	N	S	Victory arm wave	See also **`champion`**.
waiting	Y	N	S	Various impatient waiting actions	
walkie-talkie	Y	N	N	Talk on walkie-talkie.	See also **`cellphone`**.
warmhands	Y	N	N	Rub hands, shiver and warm hands over fire	
welcome	N	N	S	Two-hand welcome	
whistle	N	Y	S	One-hand whistle with piercing sound	Audible over Local distance—loud!
winner	N	N	S	Animated clasped-fist victory wave	See also **`victory`** and **`champion`**.
walllean	Y	N	Y	Relaxed lean back against wall	Alternate positions: hands in pockets or arms crossed. Stand as close to wall or object as possible before executing. Looks weird in flight. You can also get into amusing positions if you do it back-to-back with static NPCs—it looks as if your character and the NPC are in a *very* close embrace.
wounded	Y	N	N	Wobble woozily	Like a weebelo. Remember Weebelos?
yatayata yata	N	N	S	Animated "talk-talk-talk" with hand	
yourewelcome	N	N	S	Animated right-hand gesture	See also **`thanks`**, which is a mirror-image gesture.

☞ **`/yourwelcome`**, by the way.

D.3 Advanced Sit Emotes

A huge selection of fancy sit emotes was added with Issue 8. They are somewhat complicated to list, because they are different for male/huge and female characters. (*Ladies sit differently, guys, in case you've never noticed…*) A variety of ledge-sit emotes were added Post-Live.

All are static.

Note: I have not tested these with Huge characters. I assume they are the same as male but if someone wants to test things and report back…

Emote	Male & Huge action	Female action
`ledgesit1` `sitledge1`	Hands to sides, sitting straight.	Same.
`ledgesit2` `sitledge2`	Slump sideways on one hand, one knee up.	Same.

`ledgesit3`
`sitledge3`
`ledgesit`
`sitledge`
	Sit hunched forward.	Same.

`ledgesit4` `sitledge4`	Hands behind, kick your feet a bit.	Same.
`sitbench1`	Legs out straight, hands straight behind	Same
`sitbench2`	Sideways sprawl with one leg up and one arm along bench back	Same as `sitchair1`
`sitbench3`	Sprawled back, feet flat, arms on bench back	Same as `sitchair1`
`sitbench4`	Same as `sitchair3`	Same as `sitchair1`—elevates over surfaces, though.
`sitchair1`	Straight back, feet flat, hands on knees	Straight back, knees crossed, hands center
`sitchair2`	Leaning forward, feet flat, hands loose in middle	Leaning back, feet flat, hands on thighs
`sitchair3`	Straight back, feet flat, hands on thighs	Same as `sitchair1`
`sitexecutivechair`	Lean back, hands on chair arms, feet flat	Same but legs crossed
`sitstool`	1 foot down, 1 foot on rungs, 1 hand on knee	Feet up on rungs, legs crossed, hands clasped on knee
`sittable1`	Straight back, knees loose, 1 arm on table, other hand to face	Same but knees together
`sittable2`	Same as `sittable1`, but hands loose on table	Same as `sittable1`, but lower table surface

D.4 Costume Changes & Emotes

One of the last features added to the Live version of the game was the ability to fire off an emote as you changed costumes. I am not sure this was ever fully functional in the Live game, but might have been on the last iteration to hit the Test server. It is fully functional now.

In the Live era, you had up to four different costumes, slots for three of which had to be unlocked through game achievements. The Homecoming update gives you six free slots and four that can be earned, so your alts can do the full Ken and Barbie wardrobe thing if you're so inclined.

Changing costumes is simple. Either open the Costume window and click on the one you want, or use:

```
/change_costume 1
/cc 1
```

to select the second costume in your set. Note that this is another of the 'zero based' lists, with your default costume being number 0. This command can be bound to any key or macro.

There is a delay before a costume can be changed again—I believe it was a full minute on Live, and was 30 seconds for a time, but is now 15 seconds.

If you use the slash code, your costume will change instantly with no fuss.

If you use the Costume window, however, you have an interesting option. The small menu at the bottom lets you choose one of over two dozen special emotes that will bridge the costume change, from fairly simple salutes and puffs of smoke to some of the most dazzling effects in the game. Since the window is in your way, it's hard to get the full effect of the emote, but your teammates and passing noobs will be very impressed.

If you want to have more control and actually see your change emote, you can use a slash command, which again can be bound to a key or macro for convenience… or even a rolling macro or bind for variety:

```
/cc_emote 1 ccSalute
/cce 1 ccHowl
```

The emotes used for costume change are special, begin with **`'cc'`** and can only be used for this purpose; regular emotes can't be specified and the costume ones can't be used on their own. All of them include sound effects.

The choices, which are mostly fairly self-explanatory, can be found in the Costume window menu (where you might look for updates and changes), and are as follows:

cc Emote Code	Effect
`ccBackFlip`	Do a backflip and land in your new costume.
`ccCast`	Make a Dr. Strange/Constantine spell cast to change.
`ccConfettiThrow`	Throw a giant burst of confetti from your pocket and change.
`ccDimensionShift`	Spread into multiple dimensions and reassemble changed.
`ccDrinkFormula`	Drink a flask of formula to effect the change.

cc Emote Code	Effect
`ccEnergyMorph`	Crouch and fire off an energy burst to change.
`ccEvilLaugh`	Evil laugh and burst into flame to change.
`ccFeatherBurst`	Change in a spray of feathers.
`ccFeatherSpin`	Change in a spinning spray of feathers.
`ccFireworks`	Change in a burst of fireworks.
`ccFurBurst`	Change in a spray of fur.
`ccFurSpin`	Change in a spinning spray of fur.
`ccGiftBurst`	Gift falls on you and explodes open to reveal change.
`ccHowl`	Howl and change.
`ccIceBlock`	Disappear into an ice block and emerge in your new costume.
`ccInnerWill`	Focused energy in your chest triggers the change.
`ccLightMagic`	Glowing ground sigil and vertical effects make the change.
`ccLightning`	Change in a furious burst of lightning.
`ccMurderOfCrows`	Change by bird … sort of an evil Cinderella thing.
`ccNinjaLeap`	Leap high into the air and land changed.
`ccNuke`	Impressive nuclear blast changes your looks.
`ccOilStrike`	Giant gusher of oil changes you.
`ccPeacebringer`	Peacebringer form change animation.
`ccPressureRelease`	Stomp ground to release geyser that changes you.
`ccPrestoChango`	Change with a dramatic magical gesture.
`ccPureEnergy`	Change in an energy burst.
`ccRainbow`	Change in a magical haze at the end of a rainbow.
`ccRapidBoil`	Go to a bubbling boil of green sewer ick and emerge changed.
`ccSalute`	Give a full salute and change to your new costume.
`ccSmokeBomb`	Throw down a smoke bomb and emerge changed.
`ccSpin`	Spin rapidly and stop in your new costume.
`ccStoneBlock`	Disappear into a stone block and emerge in your new costume.
`ccSuperSerum`	Inject super serum, beat chest and emerge changed.
`ccVanguardSigil`	Vanguard ground sigil and green haze leave you changed.
`ccWarshade`	Warshade form change animation.

The ParagonWiki page on these emotes was well filled out and useful in figuring out this feature. Kudos to the contributors! The Homecoming wiki added more, in the v2.85 update!

D.5 Hybrid Emotes

Okay, time for a walk on the wild side. These combined emote codes came to my attention in a forum discussion in November 2020.

👍 Thanks to **CrudeVileTerror** and **fiend** for the info!

These appear to be a bug of the command parsing system and not "real" emote codes. That is, I think the dual codes combine somehow to point to a character action that is not normally (meant to be) available to players. So it may work when you read this, or not… and there may be more such combinations waiting to be discovered. (There are also some variations that produce these same animations.)

There has been a lot of concern that these codes do something awful to the game or other players, possibly in that a number of players using them in a small area can overload other player clients. It's been most of a year since these were first publicly discussed; if putting this info here is bad, maybe the command parser should be fixed instead of trying to keep the lid on the no-longer-secret issue.

Note: Obsolete. These have been nerfed in Issue 27-2.

The effect is erratic and it's essential that you "clear" each emote (by hitting any movement key) before trying a repeat or another hybrid code. Otherwise, the alt may just lie down or do the dance.

"Hybrid" Emote Code	Effect	Continuous?
`em dance1$$em liedown`	The "at ease" position but with alt looking to one side.	Yes
`em dance2$$em liedown`	"Raise dead" animation.	Yes
`em dance3$$em liedown`	Ends above animation gracefully.	No
`em dance4$$em liedown`	Hold out arm in "casting power" animation.	Yes
`em dance5$$em liedown`	Same as above but alt floats a few feet in the air.	Yes
`em dance6$$em liedown`	Ends above animation gracefully.	No
`em dance8$$em liedown`	Fall to knees crackling with smoke and electricity.	Yes
`em dance9$$em liedown`	Alt yanked into air, explodes with energy.	Yes
`em dance10$$em liedown`	(Continuation of above?) Alt falls to ground and faceplants.	Yes
`em liedown$$em score7`	(Predecessor of above?) Alt is struck by continuous bolt of yellow energy	Long animation with persistent stance
`em dance11$$em liedown`	Alt reacts to a spectacular psionic-like attack.	Yes

REFERENCE E: Chat Bubble Color Codes

It is possible to change the appearance of your character's chat bubble in two ways. The simplest is to set the text color and the background color in the Options menu. For some reason, though, this setting only affects some chat bubbles; many will default to black-on-white.

The second way to set chat bubble color—and other characteristics—is to use inline format codes. These codes can be used in manually entered chat strings or as parts of binds. The complete code set is:

```
<color ccode><bgcolor ccodetransparency><border ccode><scale factor><duration seconds>
```

As far as I know, each command can be used separately and in any order:

» **`color`** sets the text color.

  » The value **`ccode`** can be any standard color name (not sure of the range, but basics like **`red`**, **`yel-low`**, **`white`**, **`blue`** etc. should all work).

  » You can also use hex codes in the **`#rrggbb`** format—look up those codes anywhere on the web if you're not familiar with them. This works the same as the text slider in the Options menu.

» **`bgcolor`** sets the chat bubble background color, and works the same as color except that you can add an additional value to control the chat bubble background transparency.

  » If you use only a color code, you get 100% color (that is, no transparency).

  » If you add two digits to the end of the color code, you set the transparency, from 0 to 99%, with zero being fully transparent. This setting does not appear to have full 100-step granularity; there may be as few as 8 steps of transparency. I am not sure if strings like '**`yellow50`**' will work, but codes like '**`#FFFF0050`**' will.

» **`border`** sets the color of the bubble border. Identical in operation to color.

» **`scale`** sets the text and bubble size. It is supposed to scale from 0.0 to 4.0, with 1.0 being the default size, but it only works 0-2.0 for me. Useful for blowing up important bubbles like "Here!" when you've found more foes or a glowie.

» **`duration`** sets the persistence of the bubble in seconds. Default is about 8 seconds. You can make bub-bles like "Here!" more persistent, to give mates time to find you, by setting the value to 15 or so.

To use this method, embed the codes in a chat string, like this simple example:

```
g <color red><bgcolor black>Oh, no, dead again!
```

☞ Note that any spaces between the codes will be added to the chat string.

If you want to make all your chat bubbles a specific style, or have multiple styles for different uses, you need to bind a key to start the chat and load the codes—then you type your message after the codes. A little murky, but it works. For example, the normal Chat key is **`Enter`**, so:

```
/bind ENTER "beginchat <color white><bgcolor blue><scale 2><duration 10>"
```

And whenever you press ENTER, you'll be ready to chat in large white-on-blue text with a 10-sec-ond persistence. The same thing can be used in general binds:

```
/bind CTRL+T "g <color blue><bgcolor red>Teleporting $target to me!$$powexecname //
Recall Friend"
```

…although be warned I have found this usage to get flaky at times.

A final bind you might find useful to experiment with or frequently change the settings is:

```
/bind CTRL+F1 "beginchat /bind ENTER "<color #00000><bgcolor #FFFFFF75> //
                    <border #FF0000><scale 1.0><duration 10>""
```

This mess will, when you press **CTRL+F1**, load the chat entry window with **/bind…** and the whole string that follows. Edit it to suit, press **ENTER**, and then use **ENTER** to start new chat lines with the edited characteristics.

You've changed your keybind for ENTER by doing so. This can create a complete mess if you're not careful, so… be careful.

E.1 Option Settings for Chat Bubbles

In theory, chat bubble colors can be set using **/optionset** commands. In practice, the format of the color code arguments has completely eluded me (and a number of smart folks on the forums). It's not decimal, it's not hex and it's not color names.

And a huge shout-out to Electric Keet, who figured it out.

For reasons that passeth understanding, the color codes are hex in the inline commands, hex in the options file... and **signed integers** when using the **/optionset** command. The only siggiturs anywhere in the configuration system, as far as I can tell. If you've done any programming, you know what I'm talking about. If not... just write your desired color string in hex, as described above, and use any online tool to convert that hex value to a signed integer. (There's a good converter at **cryptii.com**, or use "hex to signed integer" in your fave search window.)

So **#ff00ff88**, half-opacity purple, would be **-16711800**. And vice versa. Bright yellow text on that background would use:

```
/optionset chatbubblecolor1 -65536
/optionset chatbubblecolor2 -16711800
```

Or maybe this approach just isn't worth using...

By the way… has anyone else noticed that chat bubbles vary slightly with the different channels? The team channels are slightly transparent and have a zig-zag pointer. All the others are solid and have a straight pointer. Odd.

Oddity. Good place to leave off. For now.

Revision History

Note that letter suffixes will be used to distinguish very minor interim updates but not noted here.

0.50	18 Feb 2005	First release.
	(Seven Live-era updates omitted here.)	
2.00	13 April 2009	(Issues 9-14 update.) Wow, getting to this a little late. Fortunately, the command and emote base has stayed relatively stable over the last several releases. This will likely be the last major update of this guide; I hope you've found it useful!
2.50	12 May 2019	Issue 14+/post-Live server update, and damn glad to do it!
2.52	14 May 2019	Added three missing binds and powexec_location usage.
2.55	16 May 2019	Added cc_emote usage.
2.56	18 May 2019	Added /monitorattribute usage and some other material.
2.60	29 May 2019	Reformatted, rewrote and extended Appendix W. Updated Slash Command group listing.
2.61	4 June 2019	Updated email slash commands, added base-edit slash commands, other tweaks.
2.65	5 June 2019	Updated the key names and mouse action names list a whole bunch.
2.70	9 June 2019	Added the controller button keybind section, expanded button names section. Corrections and cleanup, especially to the /showtime bind. Why didn't anyone tell me the footer title hadn't been updated?
2.71	15 June 2019	Added /macro_image usage in Section 3.3, and the name list on the website.
2.72	21 June 2019	Updated pet emotes in bind usage (section E.1).
2.75	24 June 2019	Thoroughly updated Section G, options saving and loading.
2.76	27 June 2019	Expanded option-set examples with character display features.
2.80	25 Nov 2020	I27 and otherwise overdue update.
2.85	17 Dec 2020	Complete update on emotes; added hybrid emotes section.
3.00	**15 Dec 2020**	**MAJOR overhaul, reformat and rewrite. I'm *tired*.**
3.01	20 Dec 2020	Added combat numbers listing. It just *never* ends.
3.02	13 Feb 2021	Added pseudo-autofire bind section.
3.05	21 Apr 2021	Added several commands and changes for Issue 27-2.
3.06	18 May 2021	Added chat bubble color info for /optionset.
3.10	11 Jul 2021	Finally got around to popmenus.
3.11	19 Sep 2021	Minor changes, plus added /setdifficulty... commands.
3.15	26 Nov 2021	Issue 27-3; innertube emote commands.
3.16	30 Nov 2021	Additional food/drink emotes.
3.20	7 Dec 2021	Added press-release bind info. (Error update as 3.20a).

YOU HAVE REACHED THE GUIDE LEVEL CAP.
TURN AROUND.
NO INCARNATE LEVELS AHEAD.