alterNERDtive VA profiles

alterNERDtive

Table of Contents

EliteAttack

RatAttack

SpanshAttack

StreamAttack

Use

EliteAttack

Elite Dangerous VoiceAttack Profiles¶

These are various profiles for VoiceAttack (VA) I use to enhance my Elite experience. They give me important info, facilitate day-to-day gaming and do some special things for Fuel Rats and Hull Seals work.

Available Profiles¶

- EliteAttack: The main Elite VA profile. Anything related to “just” playing the game.
- RatAttack: Manages interactions with the FuelRats IRC server.
- SpanshAttack: Plots and follows trips along the neutron highway using Spansh’s neutron plotter.

- StreamAttack: Writes various status things to files that can then be read by streaming software like OBS.

They all require the `alterNERDtive-base` profile that, well, provides basic functionality common to all of the above.

Need Help / Want to Contribute?¶

Well, you are in the right place. You can find comprehensive documentation right here.

If you run into any errors, please make sure you are running the latest version of the profiles and all requirements.

If your problem persists, please file an issue. Thanks! :)

You can also say “Hi” on Discord if that is your thing.

Install

Requirements¶

VoiceAttack¶

Obviously you will need to install VoiceAttack. There is a free trial version available, but that one is limited to a single profile and a few commands. This is 4 profiles and … a lot of commands. You will need the full version, available for $10 (official site) or €11.99 (Steam, IIRC $14.99 for our US-based friends).

I recommend buying on the site. Why? Because on Steam, Valve gets a 30% cut. Unlike many other developers Gary (the developer of VoiceAttack) remedies that by having a price on Steam that ends up paying $10 to him. So basically, you are paying Valve out of your own pocket. Many other developers do not do that, and by buying from them directly instead of on Steam you are literally giving them extra money. Please do keep that in mind in the future!

You also will generally need to opt into the beta version. I am usually at the forefront of bug reports and feature requests, and I do rely on the fixes/additions in beta versions quite often.

EDDI¶

EDDI is a companion application for Elite: Dangerous, providing responses to events that occur in-game using data from the game as well as various third-party tools.

EDDI also regularly publishes beta versions. Unless a profiles release explicitly states it, you will *not* have to run EDDI beta.

bindED¶

bindED reads your Elite Dangerous binding files and makes them available to VoiceAttack as variables. That way commands can be portable and you do not have to manually go through them and change any actions that you happen to not have the standard binds for.

This plugin is *included* in the release package.

Elite Scripts¶

I have wrote a collection of Python scripts to interface with various 3rd party services like EDSM or Spansh. Those are called by the profiles for various tasks, like checking a system’s body count.

In the future they will be replaced by VoiceAttack plugin code.

The scripts are *included* in the release package.

ED-NeutronRouter¶

(required for SpanshAttack)

ED-NeutronRouter interfaces with Spansh’s neutron plotter and makes the result available to VoiceAttack.

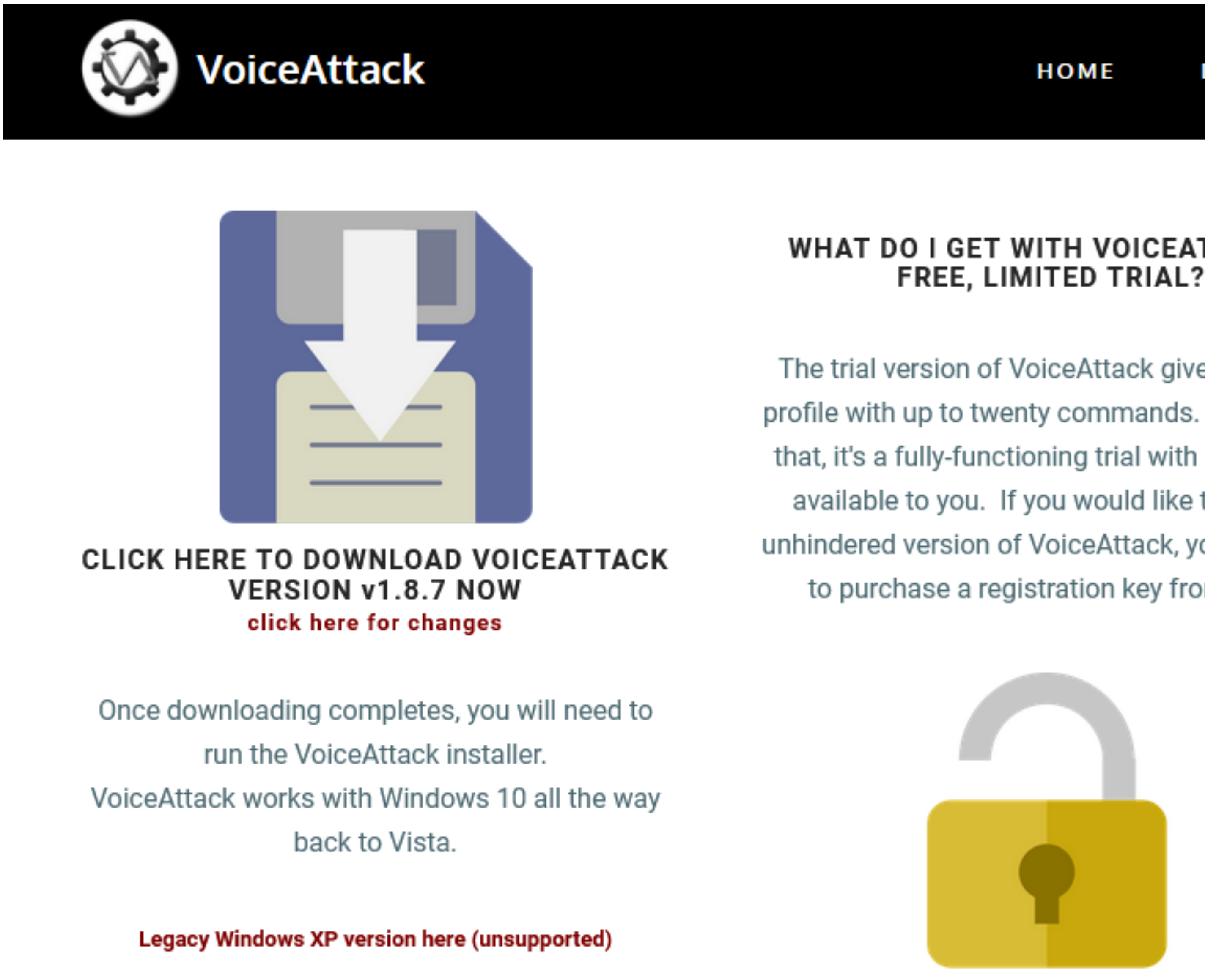
This will also eventually replaced by my own plugins.

Installing¶

Install VoiceAttack¶

Grab the most recent VoiceAttack beta version off the official site and install it. Right now 32bit vs. 64bit does not make any difference, but there is no reason not to choose 64bit unless you are running a 32bit version of Windows (is that even still a thing?).

If you are using the standalone version you should probably download the executable installer. If you are using the Steam version of VoiceAttack, you will have to download the zipped folder and replace your installed version with its contents.


VoiceAttack HOME

CLICK HERE TO DOWNLOAD VOICEATTACK VERSION v1.8.7 NOW
click here for changes

Once downloading completes, you will need to run the VoiceAttack installer.
VoiceAttack works with Windows 10 all the way back to Vista.

Legacy Windows XP version here (unsupported)

WHAT DO I GET WITH VOICEAT
FREE, LIMITED TRIAL?

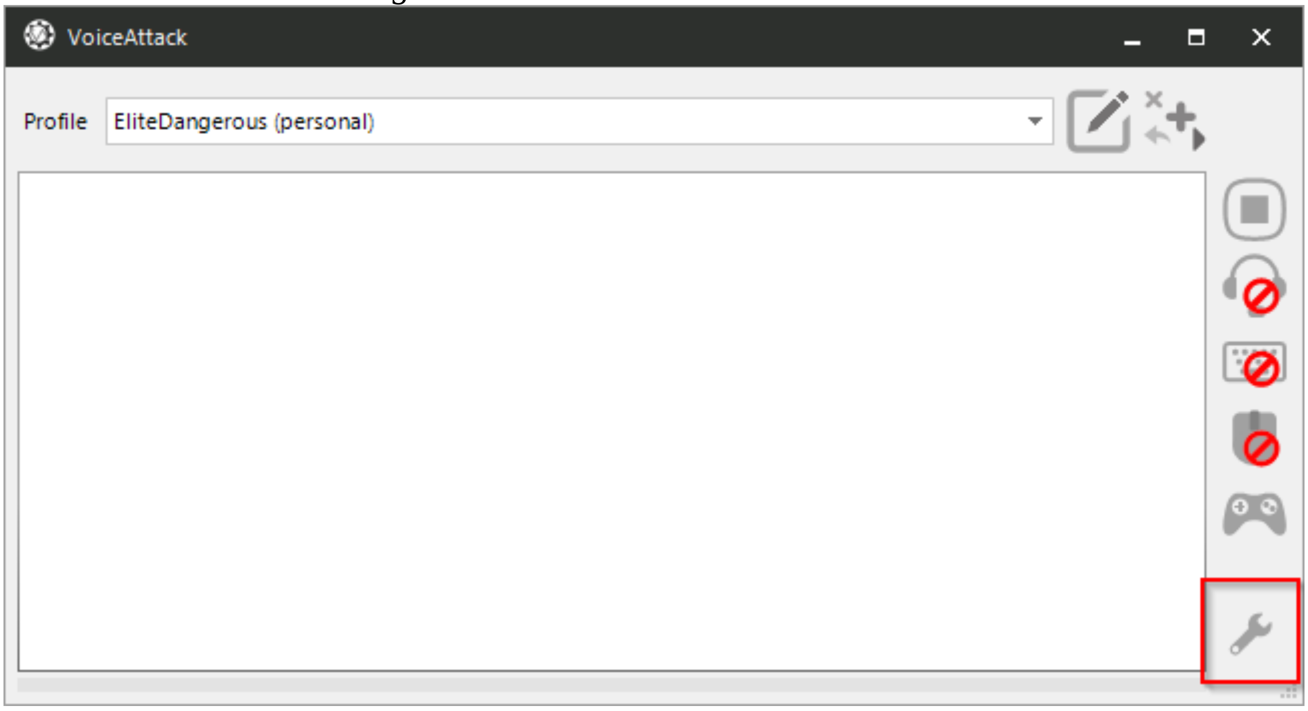
The trial version of VoiceAttack give
profile with up to twenty commands.
that, it's a fully-functioning trial with
available to you. If you would like t
unhindered version of VoiceAttack, yo
to purchase a registration key fro

[Here is the first hit on a Google search for finding the installation folder](). It is for a completely unrelated game, but the steps are the same.

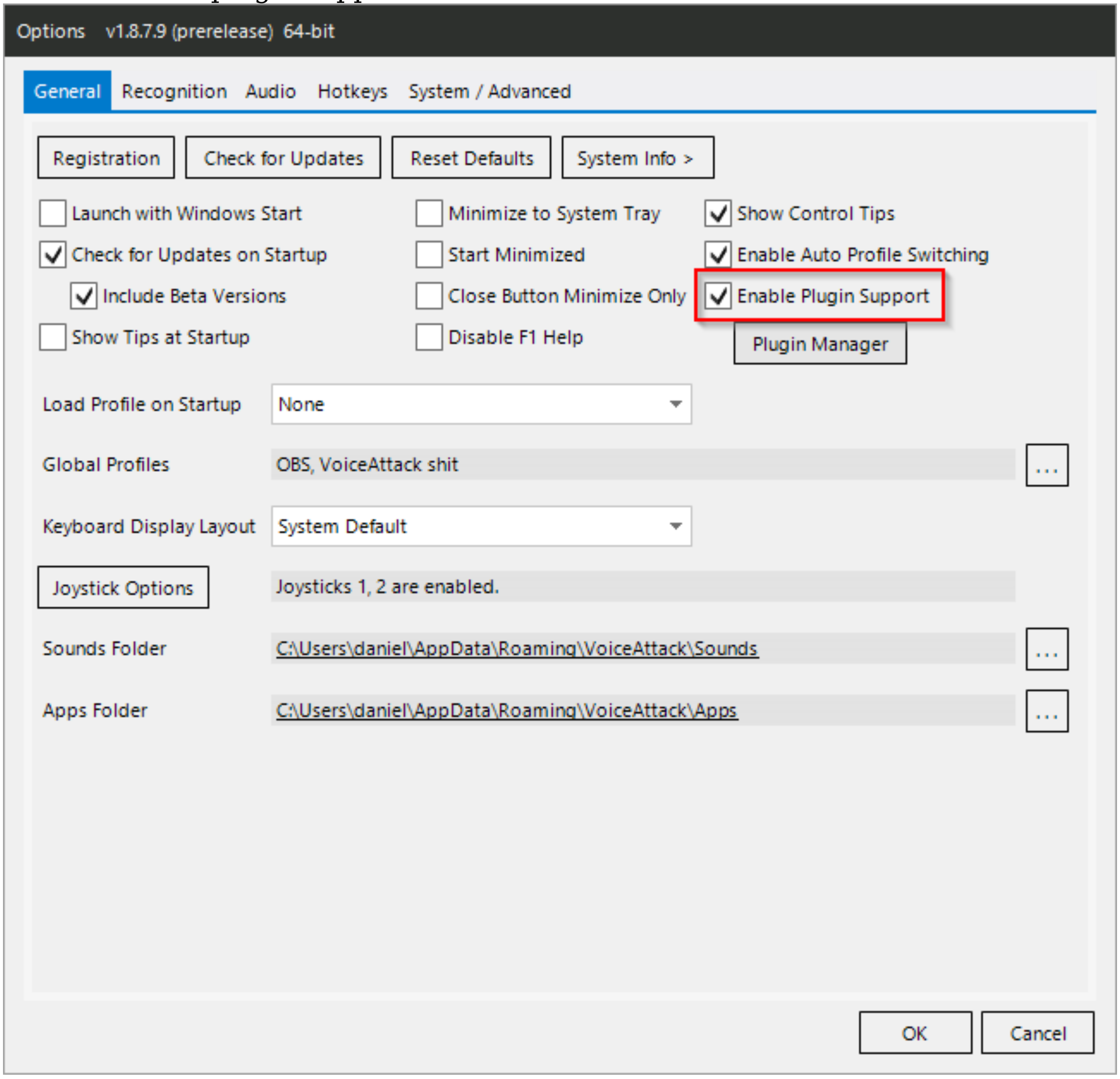
You can potentially also install the latest non-beta version; but I often use features that have only just been added to the beta, so some things might break for you if you are not using that.

Make sure you have plugin support enabled:

1. Go to VoiceAttack settings.



2. Check “enable plugin support”



Options v1.8.7.9 (prerelease) 64-bit

General | Recognition | Audio | Hotkeys | System / Advanced

Registration | Check for Updates | Reset Defaults | System Info >

[ ] Launch with Windows Start
[✓] Check for Updates on Startup
[✓] Include Beta Versions
[ ] Show Tips at Startup
[ ] Minimize to System Tray
[ ] Start Minimized
[ ] Close Button Minimize Only
[ ] Disable F1 Help
[✓] Show Control Tips
[✓] Enable Auto Profile Switching
[✓] Enable Plugin Support

Plugin Manager

Load Profile on Startup: None

Global Profiles: OBS, VoiceAttack shit

Keyboard Display Layout: System Default

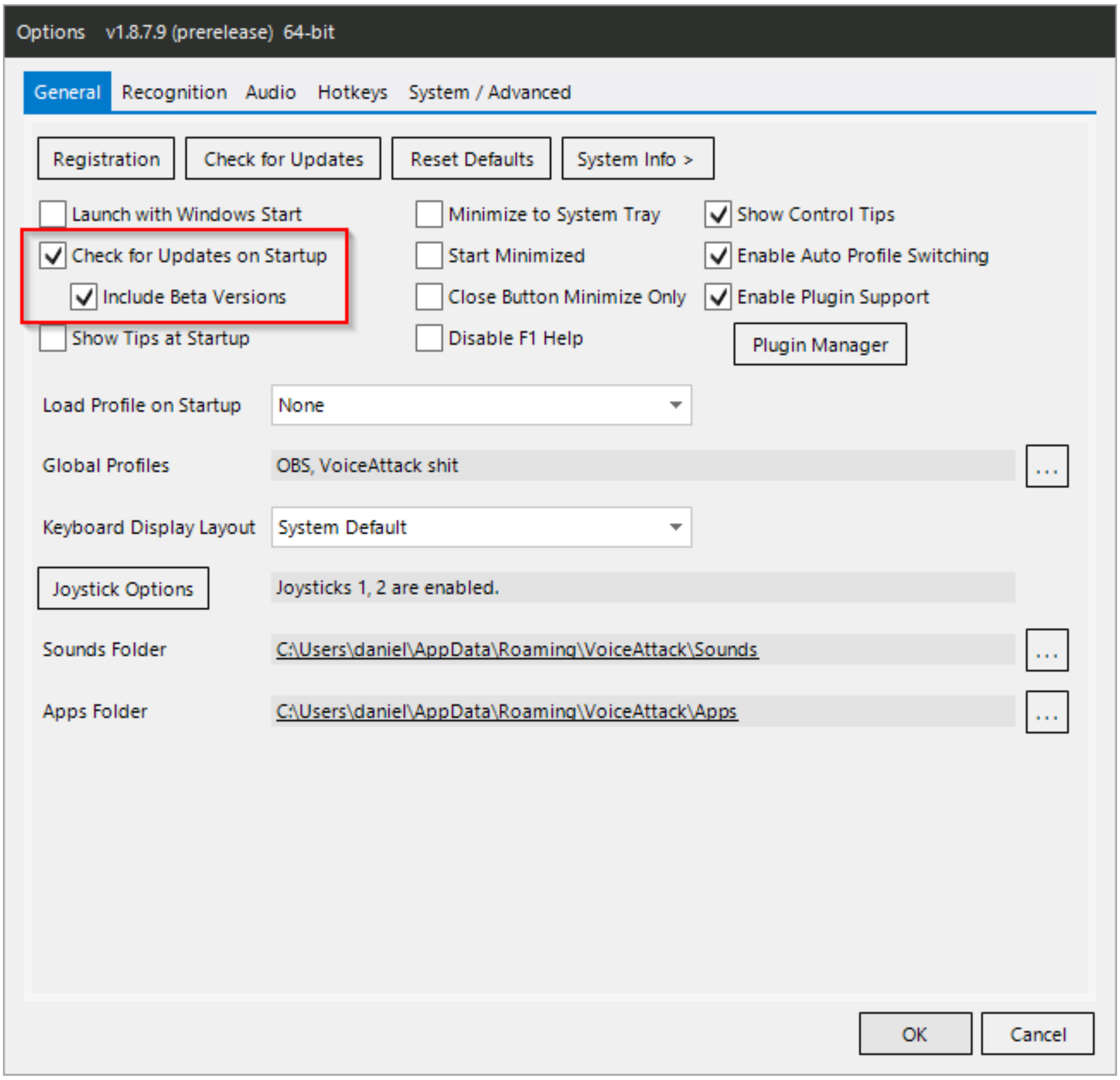
Joystick Options: Joysticks 1, 2 are enabled.

Sounds Folder: C:\Users\daniel\AppData\Roaming\VoiceAttack\Sounds

Apps Folder: C:\Users\daniel\AppData\Roaming\VoiceAttack\Apps

OK | Cancel

While you are there, you might also want to enable the automatic update checks.


Options v1.8.7.9 (prerelease) 64-bit

- [ ] Launch with Windows Start
- [✓] Check for Updates on Startup
  - [✓] Include Beta Versions
- [ ] Show Tips at Startup
- [ ] Minimize to System Tray
- [ ] Start Minimized
- [ ] Close Button Minimize Only
- [ ] Disable F1 Help
- [✓] Show Control Tips
- [✓] Enable Auto Profile Switching
- [✓] Enable Plugin Support

Load Profile on Startup: None
Global Profiles: OBS, VoiceAttack shit
Keyboard Display Layout: System Default
Joysticks 1, 2 are enabled.
Sounds Folder: C:\Users\daniel\AppData\Roaming\VoiceAttack\Sounds
Apps Folder: C:\Users\daniel\AppData\Roaming\VoiceAttack\Apps

Install EDDI¶

Install the latest release from Github (The `EDDI-X.Y.Z.exe` under “Assets”). You will need to install it as a VoiceAttack plugin.

If you do not want to install it into the VoiceAttack installation folder (or already have it installed somewhere else) you can also just make a symbolic link. Open a CMD prompt (Windows key + R, enter “cmd”, hit `Enter`) and do

```
>cd x:\path\to\VoiceAttack\Apps
>mklink /J EDDI x:\path\to\EDDI
```

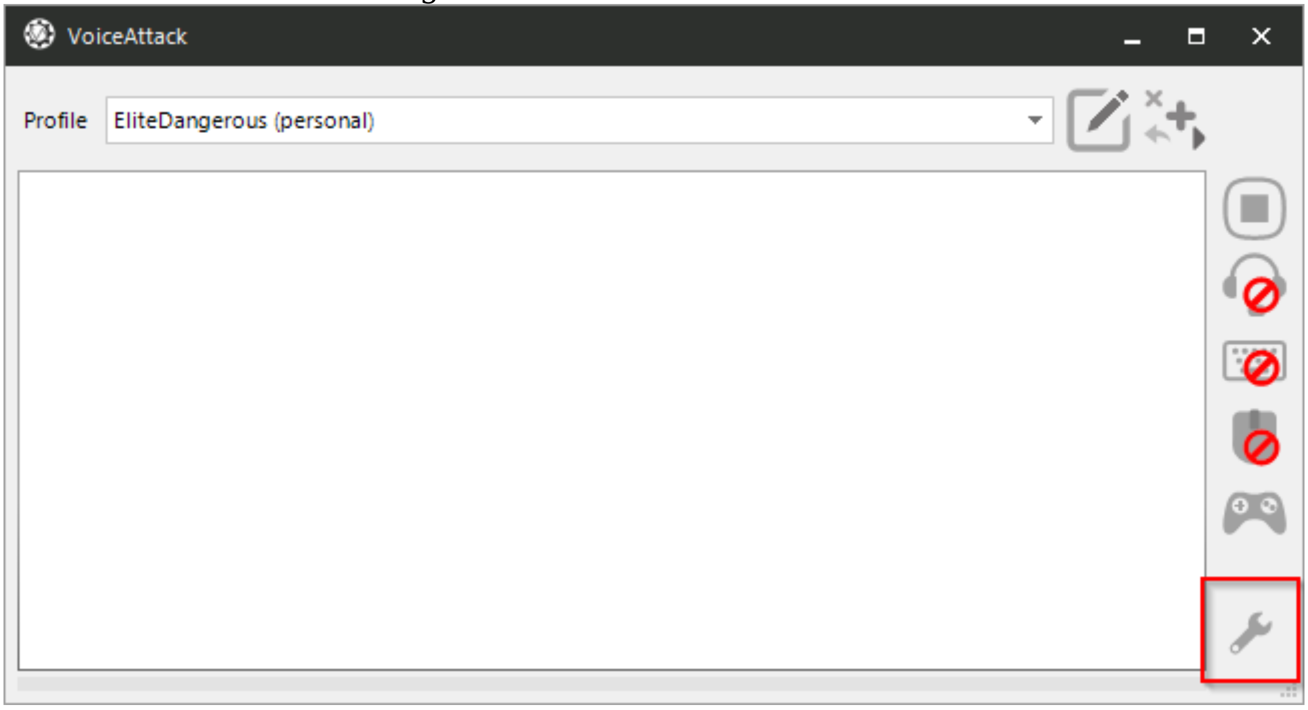
If you have installed the non-Steam version of VoiceAttack to the default folder within “ProgramFiles” you will have to run the CMD prompt as admin (Windows key + R, enter “cmd”, hit `Control` + `Shift` + `Enter`).

Install ED-NeutronRouter¶

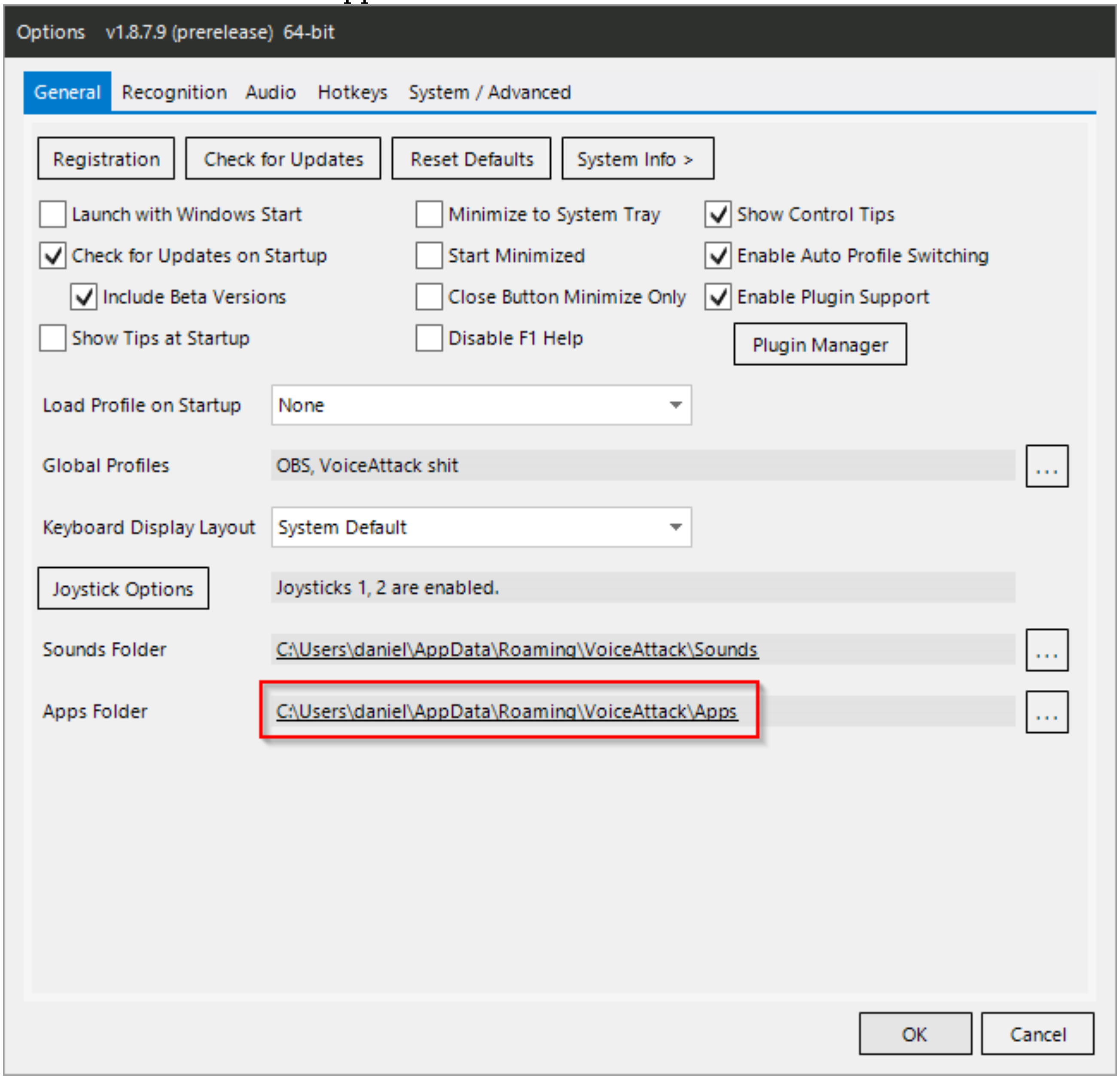
Make sure to grab the pre-release 1.02 since 1.01 has a bug with a hardcoded 50 ly jump range (the `EDNeutronRouter.vX.YZ.zip` under “Assets”).

You will have to extract the contents of the release .zip file to your VoiceAttack Apps folder:

1. Go into VoiceAttack settings.



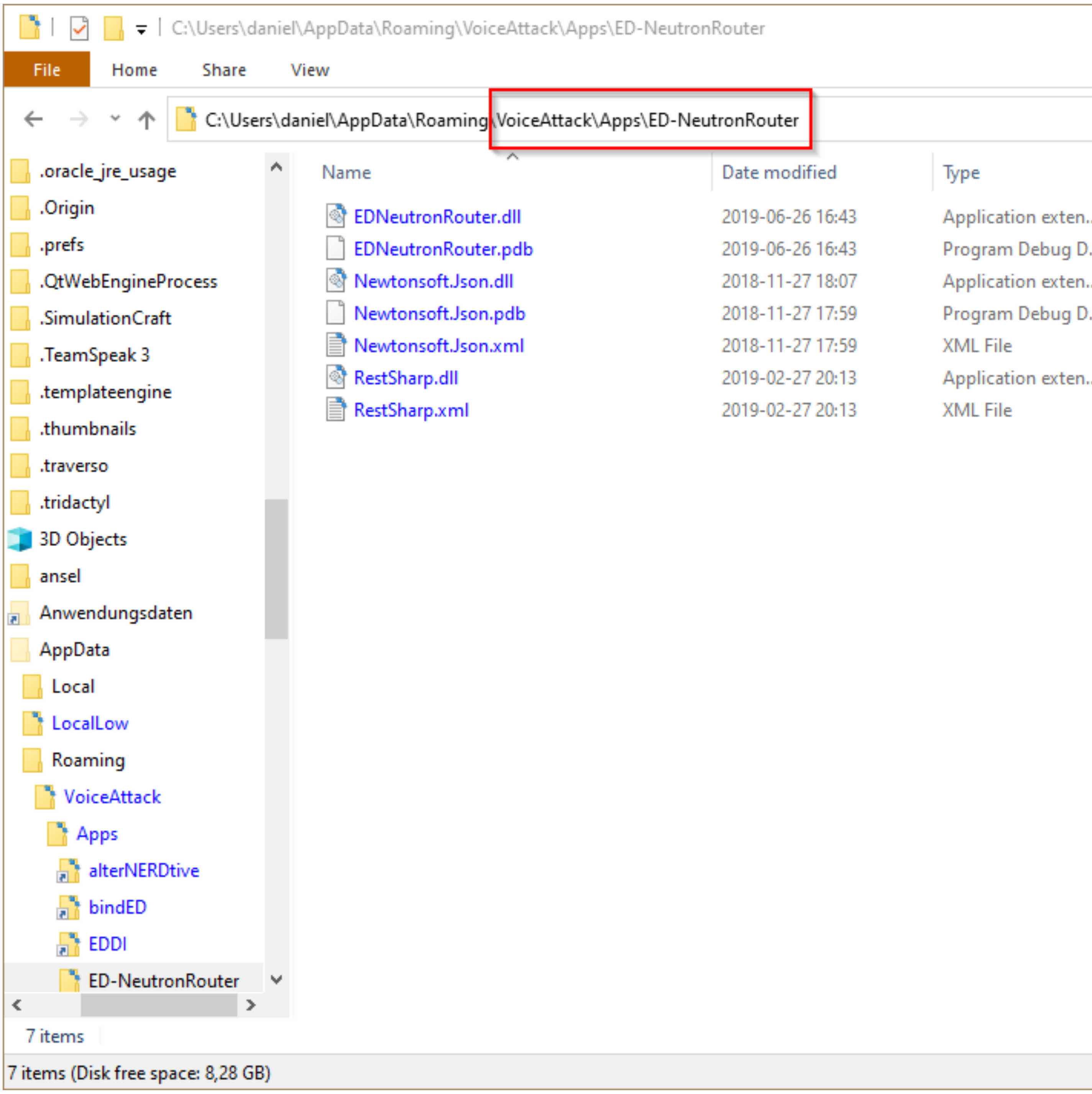
2. Click the folder set as “Apps Folder”.



- [ ] Launch with Windows Start
- [x] Check for Updates on Startup
  - [x] Include Beta Versions
- [ ] Show Tips at Startup
- [ ] Minimize to System Tray
- [ ] Start Minimized
- [ ] Close Button Minimize Only
- [ ] Disable F1 Help
- [x] Show Control Tips
- [x] Enable Auto Profile Switching
- [x] Enable Plugin Support

Load Profile on Startup	None
Global Profiles	OBS, VoiceAttack shit
Keyboard Display Layout	System Default
	Joysticks 1, 2 are enabled.
Sounds Folder	C:\Users\daniel\AppData\Roaming\VoiceAttack\Sounds
Apps Folder	C:\Users\daniel\AppData\Roaming\VoiceAttack\Apps

Now extract the contents of the downloaded file into there. Make sure that they are not naked files under “Apps”, but have their own folder “Apps\ED-NeutronRouter”! The exact folder name does not matter as long as they *are* in a subfolder. Otherwise the plugin will not load.



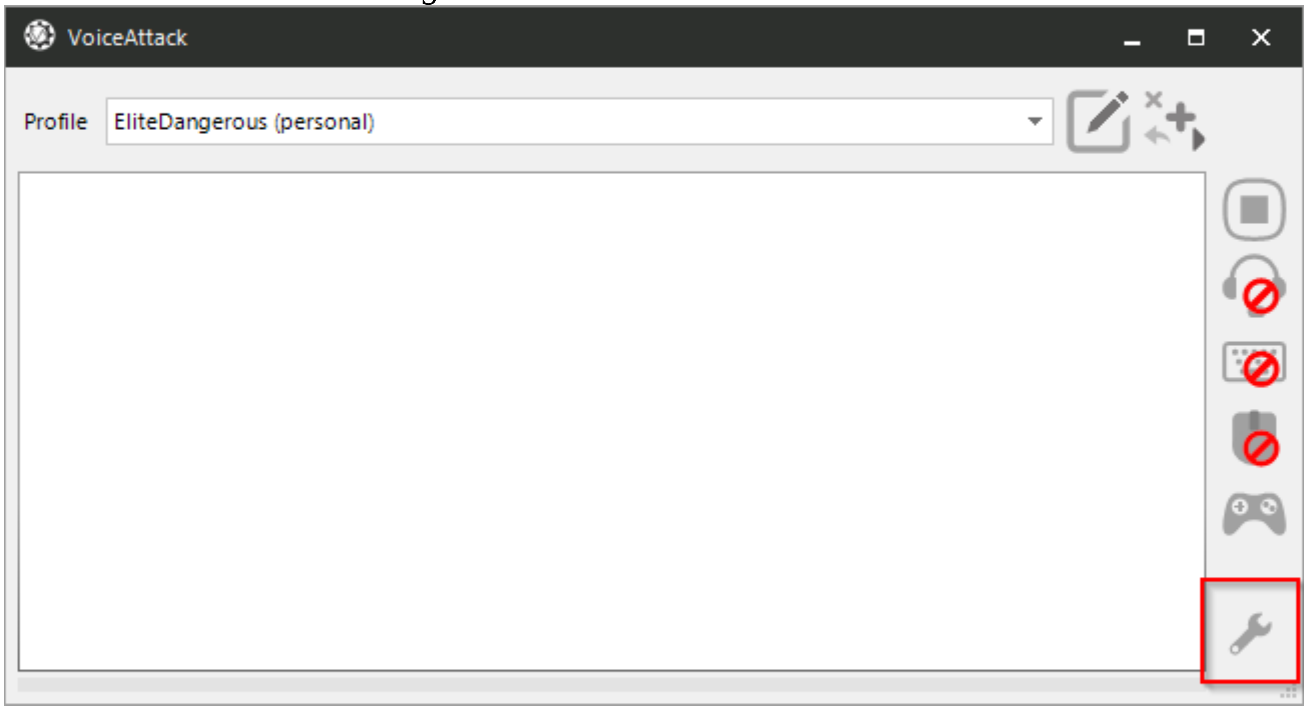
Name	Date modified	Type
EDNeutronRouter.dll	2019-06-26 16:43	Application exten
EDNeutronRouter.pdb	2019-06-26 16:43	Program Debug D
Newtonsoft.Json.dll	2018-11-27 18:07	Application exten
Newtonsoft.Json.pdb	2018-11-27 17:59	Program Debug D
Newtonsoft.Json.xml	2018-11-27 17:59	XML File
RestSharp.dll	2019-02-27 20:13	Application exten
RestSharp.xml	2019-02-27 20:13	XML File

Leave the “Apps” folder open in Windows Explorer, you will need it for the next step.

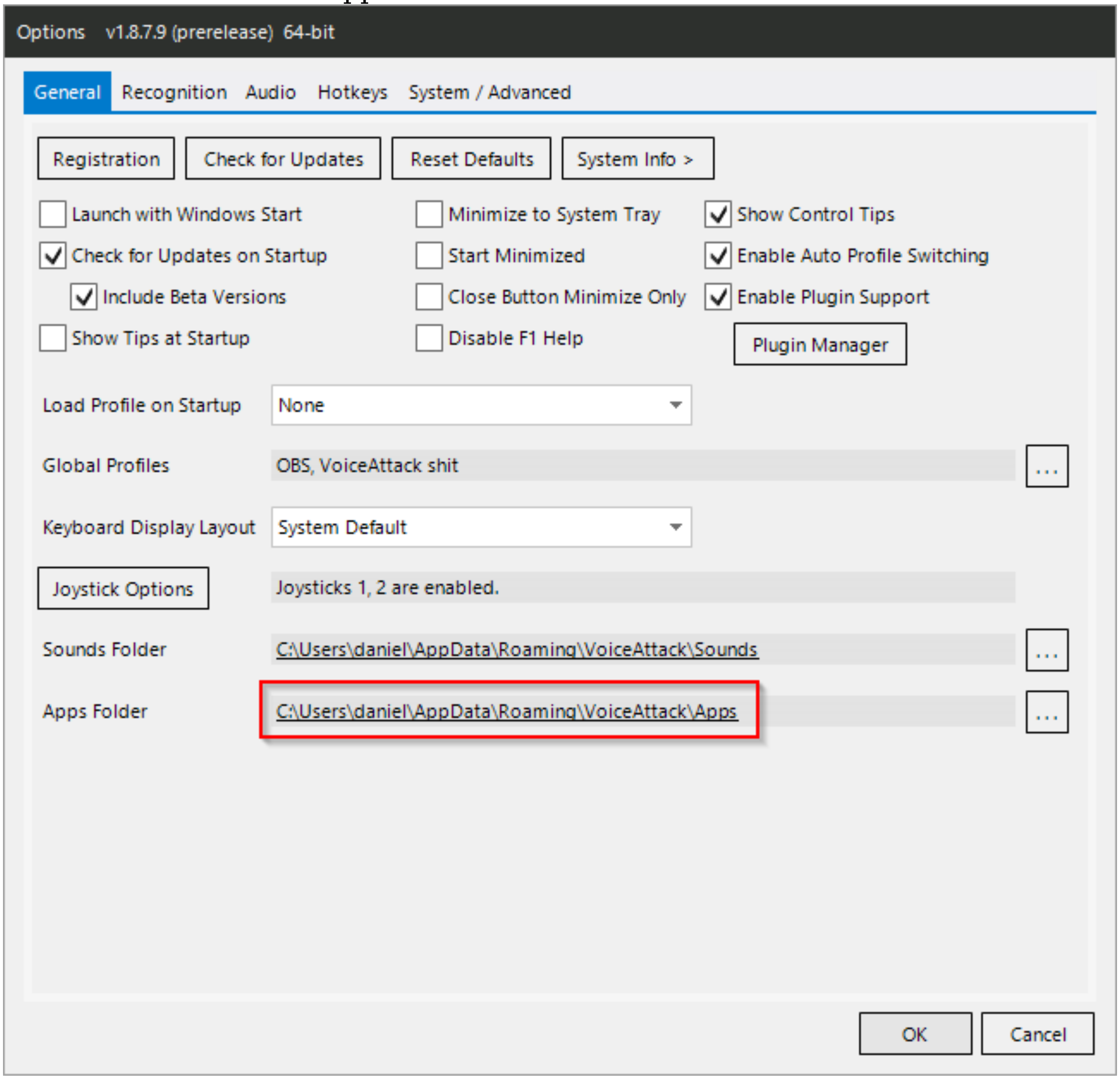
Import Profiles Package¶

Acquire the latest release from Github (the `alterNERDtive-voiceattack-profiles.vax` file under “Assets”).

1. Go into VoiceAttack settings.



2. Click the folder set as “Apps Folder”.



Options v1.8.7.9 (prerelease) 64-bit

- [ ] Launch with Windows Start
- [x] Check for Updates on Startup
  - [x] Include Beta Versions
- [ ] Show Tips at Startup
- [ ] Minimize to System Tray
- [ ] Start Minimized
- [ ] Close Button Minimize Only
- [ ] Disable F1 Help
- [x] Show Control Tips
- [x] Enable Auto Profile Switching
- [x] Enable Plugin Support

Load Profile on Startup	None
Global Profiles	OBS, VoiceAttack shit
Keyboard Display Layout	System Default
Joystick Options	Joysticks 1, 2 are enabled.
Sounds Folder	C:\Users\daniel\AppData\Roaming\VoiceAttack\Sounds
Apps Folder	C:\Users\daniel\AppData\Roaming\VoiceAttack\Apps

3. Create a sub folder named “Import” if it does not exist yet.
4. Drop the downloaded .vax file into the “Import” folder.
5. Restart VoiceAttack.
6. When prompted, import the profile package. VoiceAttack will restart when completed.

Create a Custom Profile¶

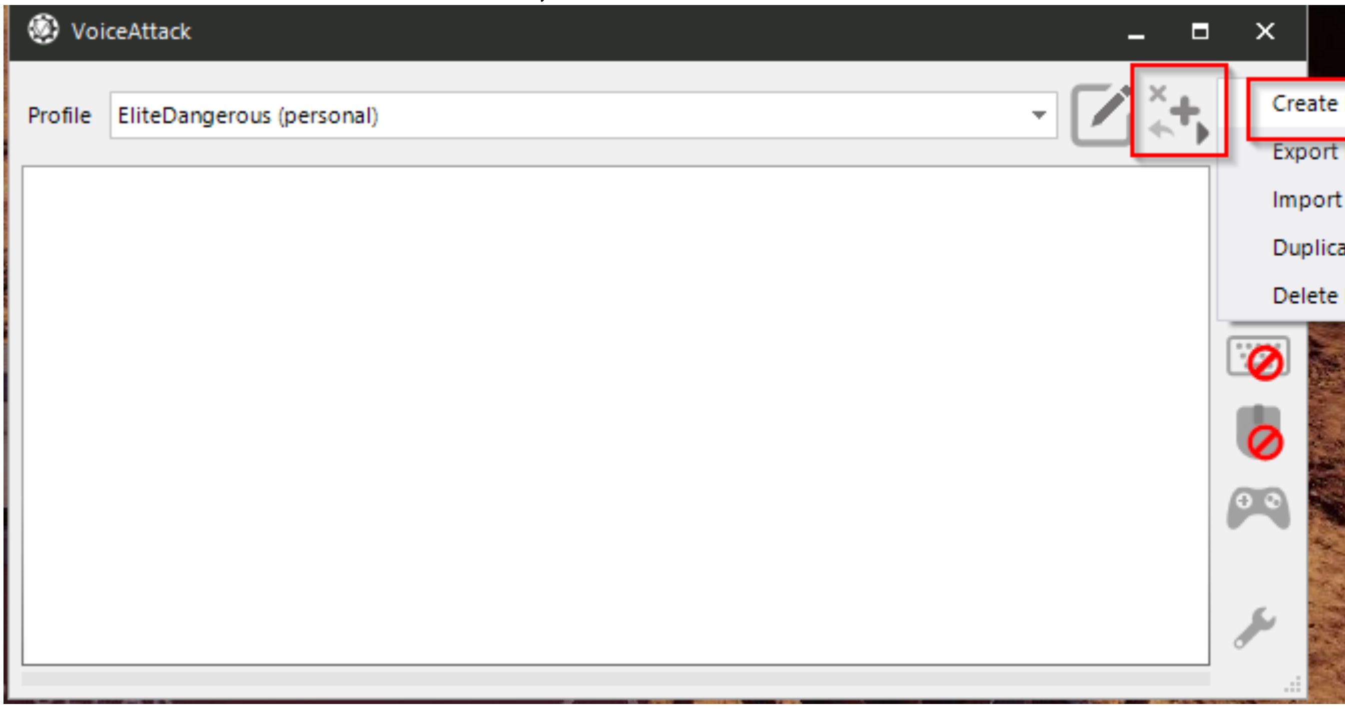
Last but not least you are going to create your own custom VoiceAttack profile for Elite. It will allow you to add your own commands, override any commands in the profiles that you want to change and add voice triggers or hotkeys.

You can either use an existing profile, create a new one or use the provided profile example as a basis.

Regardless of which way you choose, make sure to read the [#Import Profiles](#import-profiles) section and follow the instructions there!

Create a New Custom Profile¶

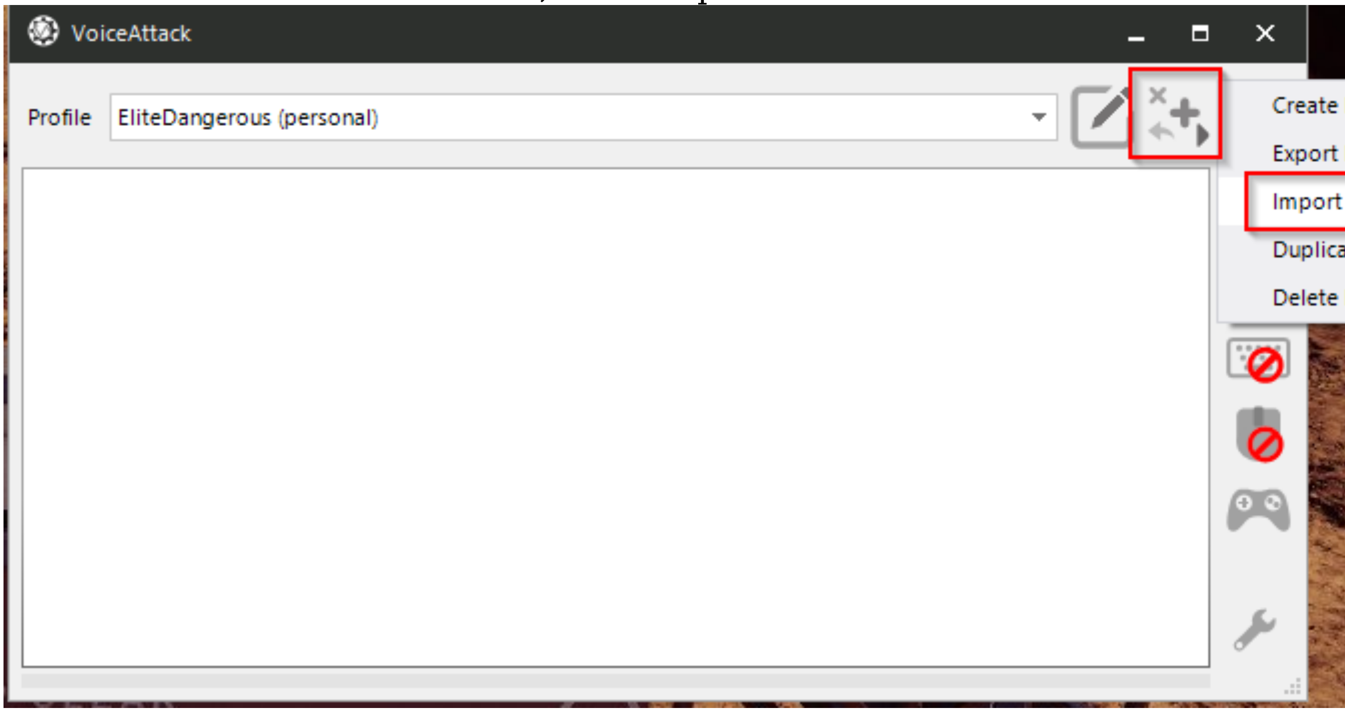
1. Click the “Profile Actions” button, then “Create New Profile”.



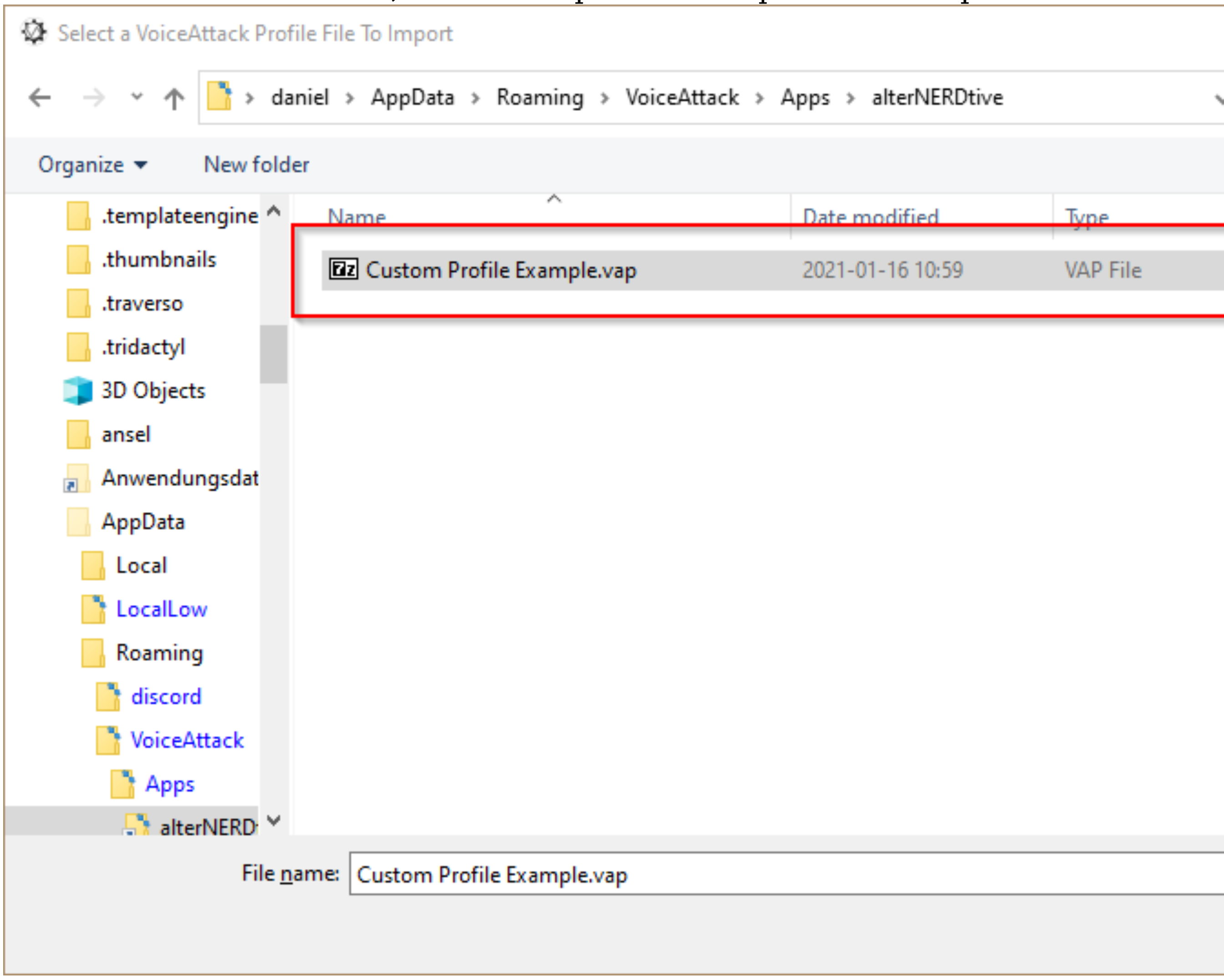
2. Give it a name and add some commands if you want to.
3. Hit “Done” to create the new profile.

Use the Profile Example¶

1. Click the “Profile Actions” button, then “Import Profile”.



2. Navigate to your VoiceAttack Apps folder (see above), go into the "alterNERDtive" subfolder, choose the profile example and hit "Open".



Once you are done with the setup and configuration process, you can find a bunch of example commands with comments on how to do things in this profile. Make sure to also rename it to something more exciting than "Custom Profile Example"!

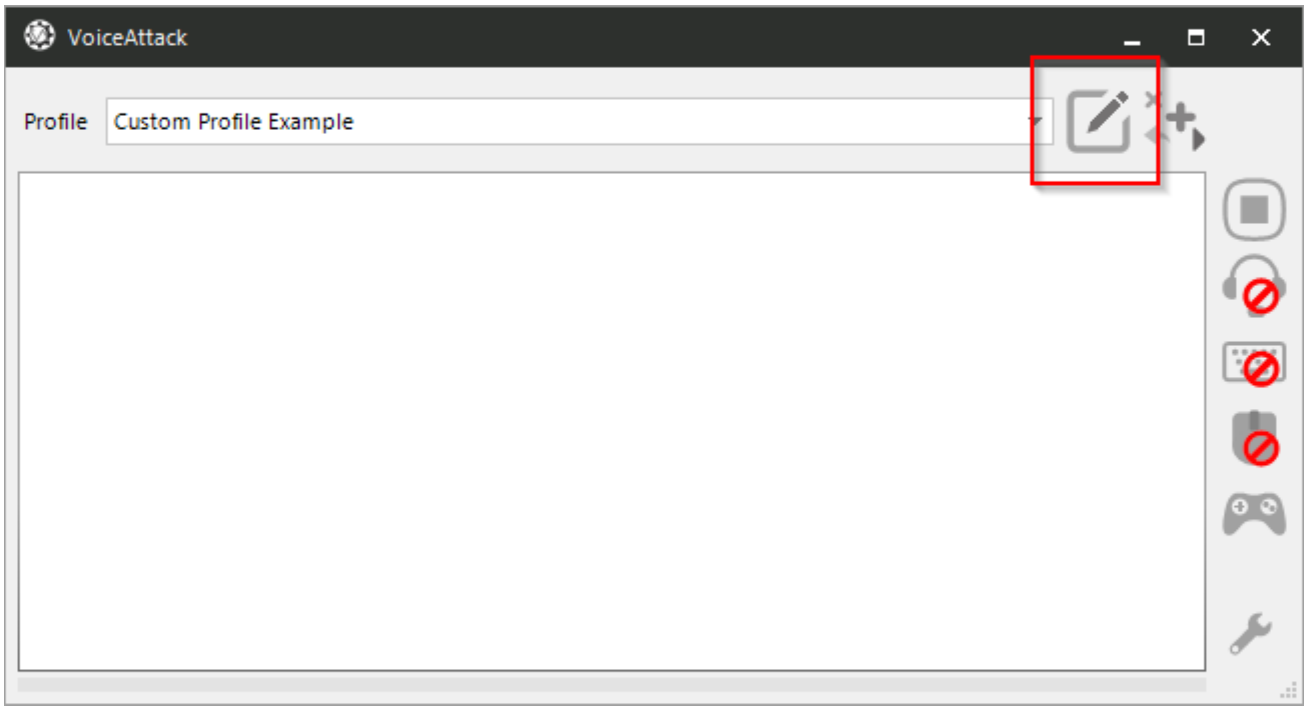
Import Profiles¶

In order to use my profiles with your custom profiles, you will need to take two additional steps:

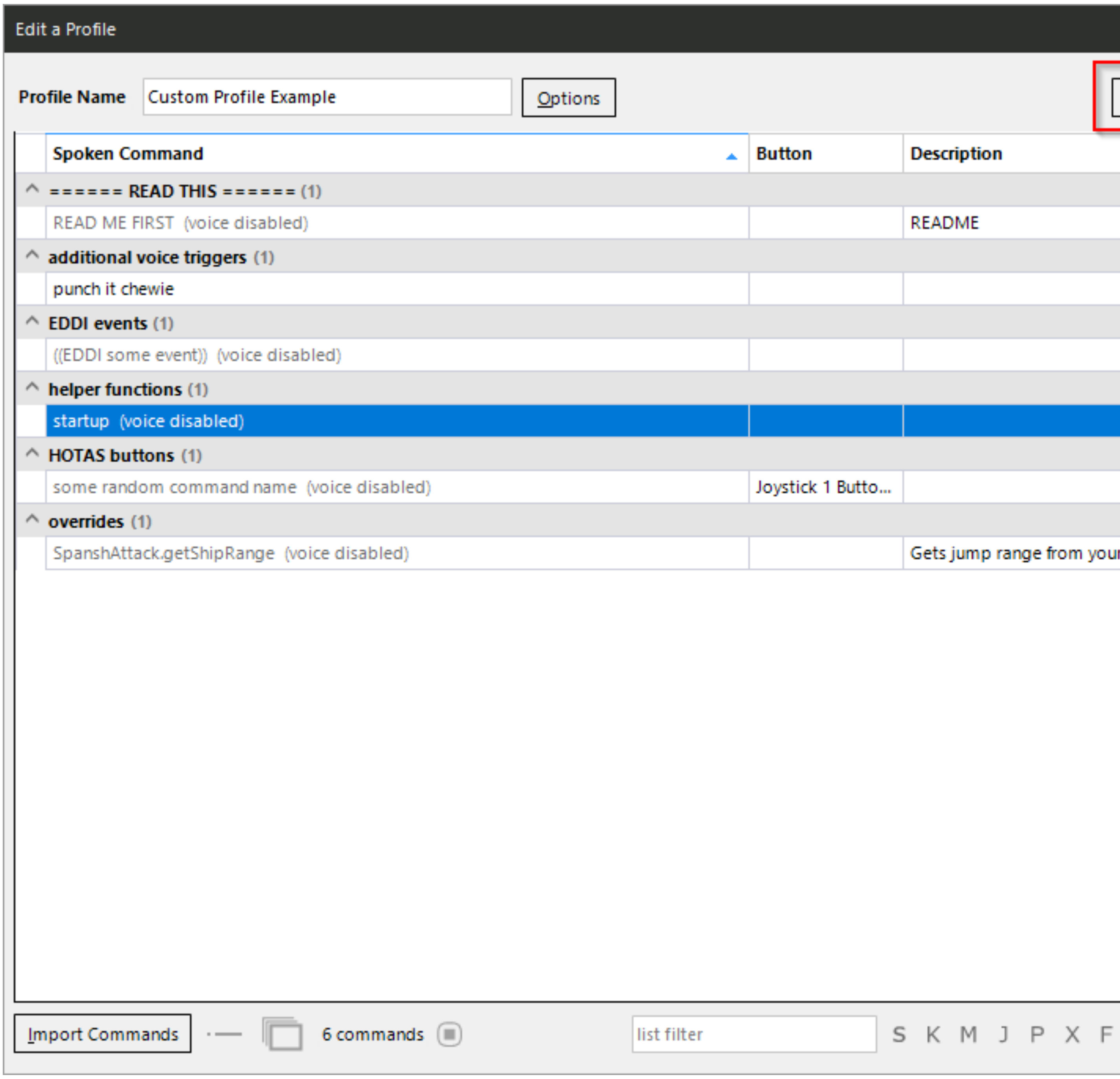
1. Include the profiles in your custom profiles. That will make all commands available when your custom profile is active.
2. Create a startup command for your custom profile. You can use it to do anything you want when your profile loads, but it will also have to run the startup command for my profiles.

Create a Startup Command¶

First off, hit the "Edit" button in VoiceAttack.



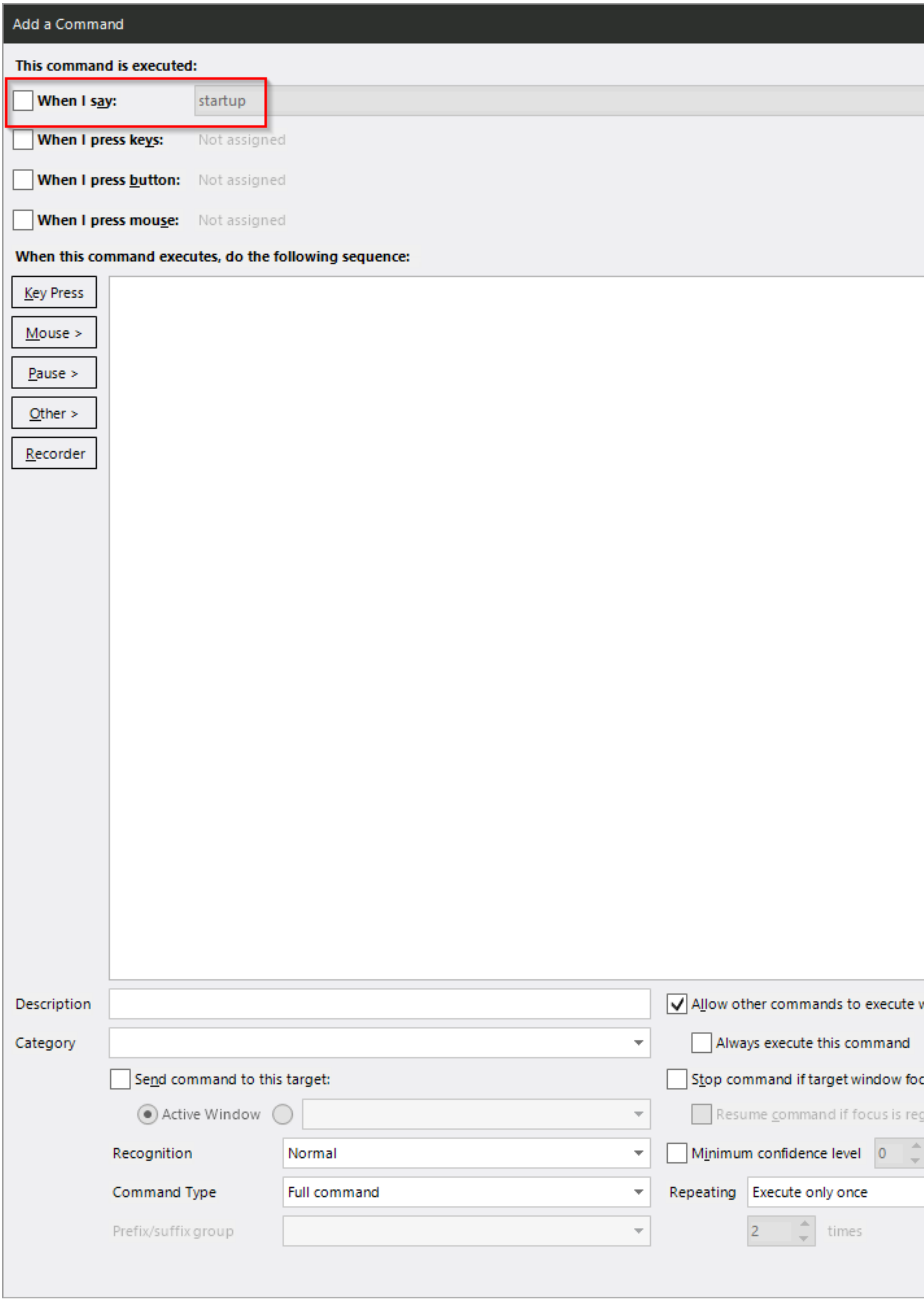
If you are using your existing profile (or have just created a fresh one) you will now have to create the startup command. Hit the “New Command” button.


Edit a Profile

Profile Name: Custom Profile Example

Spoken Command	Button	Description
====== READ THIS ====== (1)		
READ ME FIRST (voice disabled)		README
additional voice triggers (1)		
punch it chewie		
EDDI events (1)		
((EDDI some event)) (voice disabled)		
helper functions (1)		
startup (voice disabled)		
HOTAS buttons (1)		
some random command name (voice disabled)	Joystick 1 Butto...	
overrides (1)		
SpanshAttack.getShipRange (voice disabled)		Gets jump range from your

You can name it anything you want but I recommend calling it “startup” or similar, and to deactivate the “when i say”checkbox in the command options to make sure you do not accidentally run it via voice.



Add a Command

This command is executed:

☐ When I say: startup

☐ When I press keys: Not assigned

☐ When I press button: Not assigned

☐ When I press mouse: Not assigned

When this command executes, do the following sequence:

Description

Category

☑ Allow other commands to execute w

☐ Always execute this command

☐ Send command to this target:

[● Active Window] ○

☐ Stop command if target window foc

☐ Resume command if focus is reg

Recognition: Normal

☐ Minimum confidence level 0

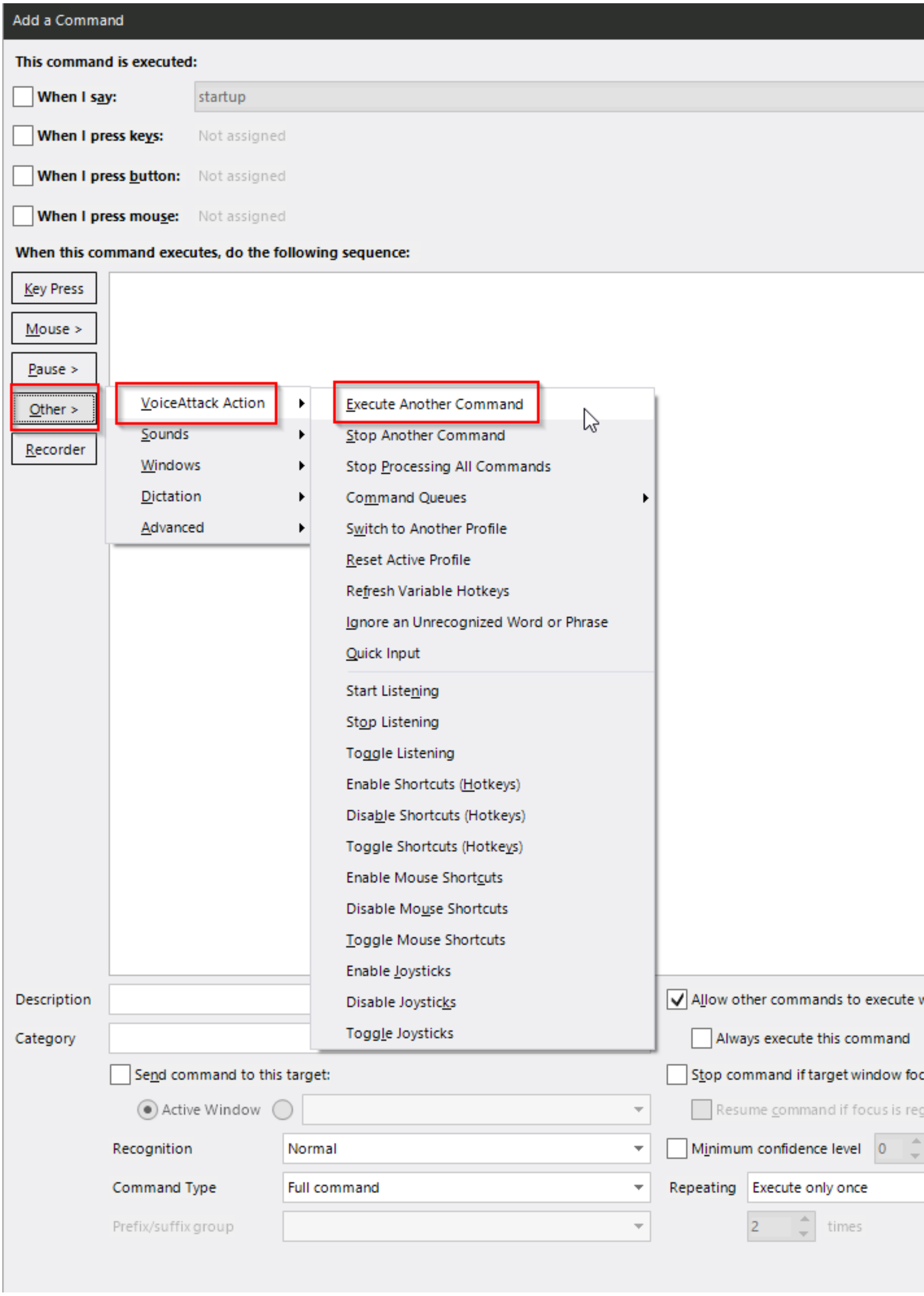
Command Type: Full command

Repeating: Execute only once

Prefix/suffix group

2 times

Add a new action using “Other” → “VoiceAttack Action”→ “Execute Another Command”.



Add a Command

This command is executed:

- [ ] When I say: startup
- [ ] When I press keys: Not assigned
- [ ] When I press button: Not assigned
- [ ] When I press mouse: Not assigned

When this command executes, do the following sequence:

Description

Category

- [ ] Send command to this target:

● Active Window ○

Recognition: Normal

Command Type: Full command

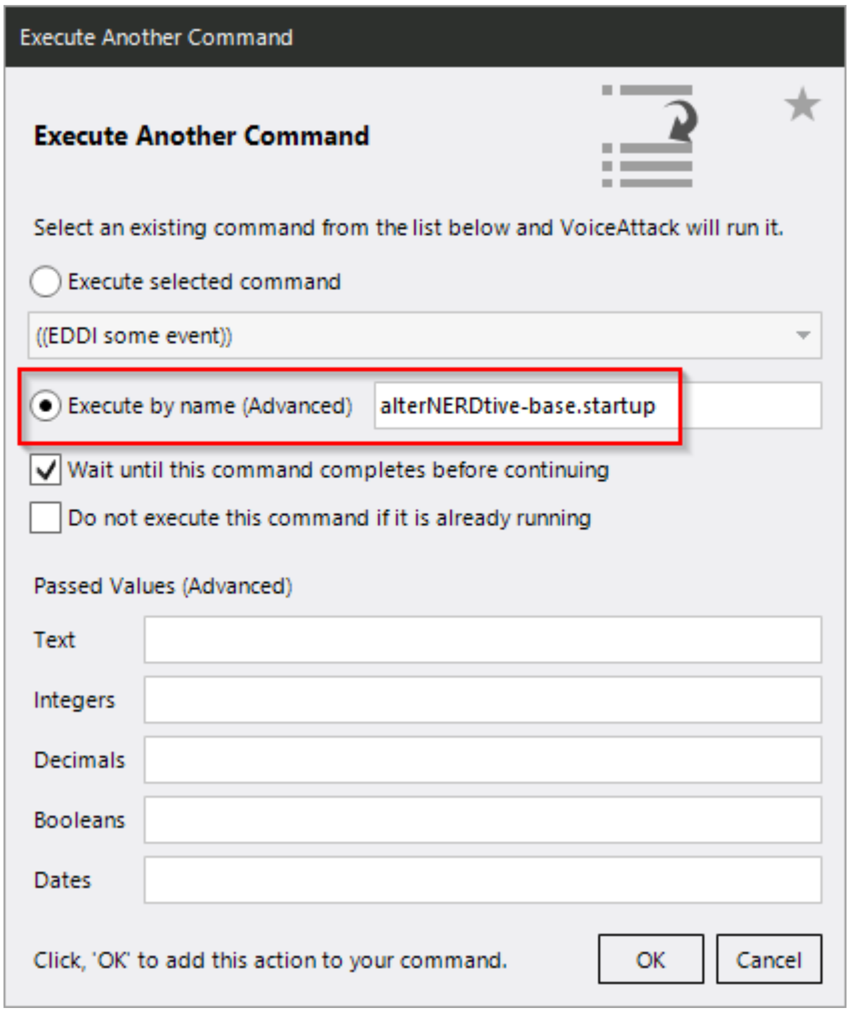
Prefix/suffix group

- [✓] Allow other commands to execute w
- [ ] Always execute this command
- [ ] Stop command if target window foc
- [ ] Resume command if focus is reg
- [ ] Minimum confidence level 0

Repeating: Execute only once

2 times

Choose “Execute by name (Advanced)” and enter “alterNERDtive-base.startup”.



Execute Another Command

Execute Another Command

Select an existing command from the list below and VoiceAttack will run it.

○ Execute selected command

((EDDI some event))

● Execute by name (Advanced) alterNERDtive-base.startup

[✓] Wait until this command completes before continuing

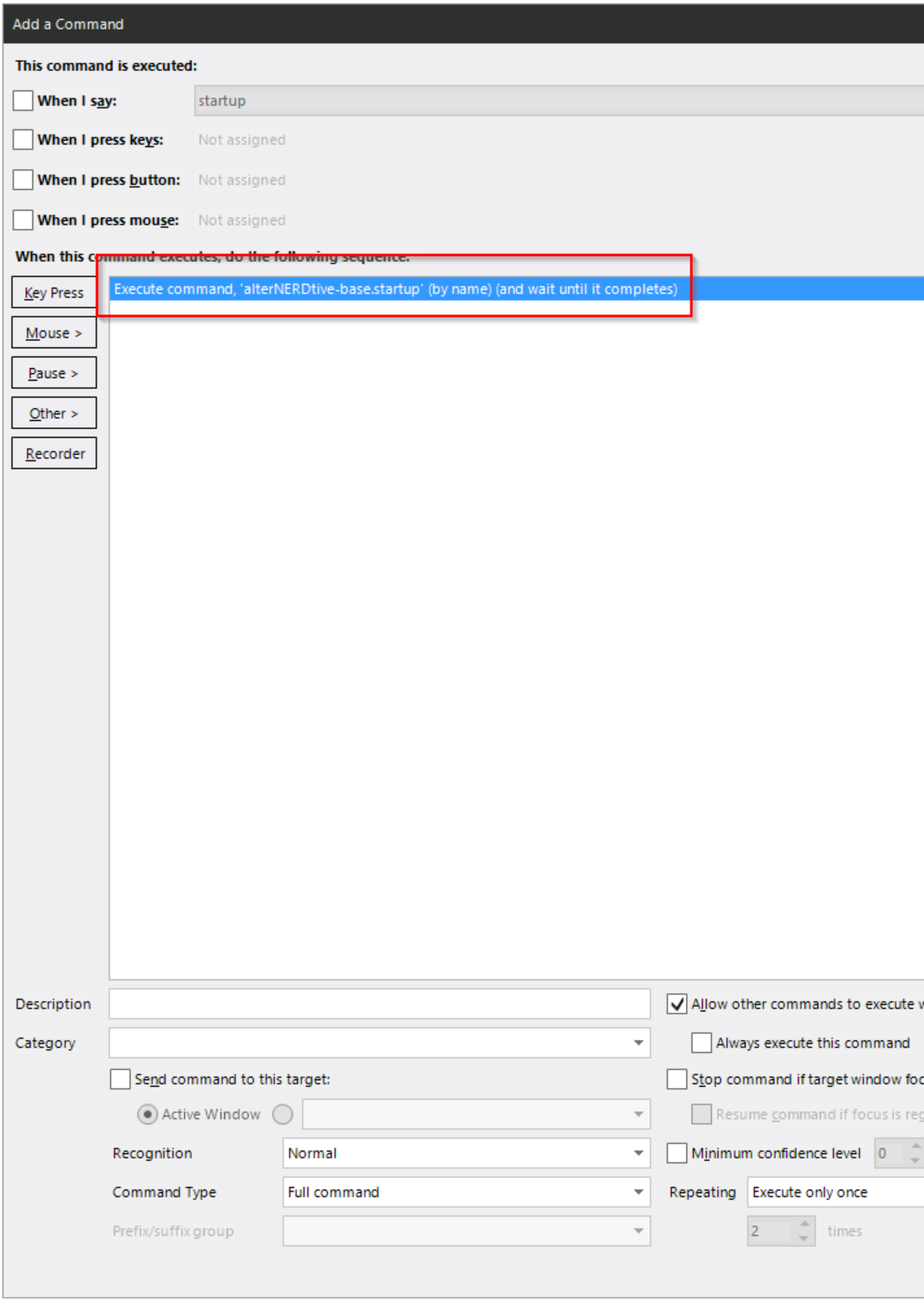
[ ] Do not execute this command if it is already running

Passed Values (Advanced)

Text	
Integers	
Decimals	
Booleans	
Dates	

Click, 'OK' to add this action to your command.

Make sure to leave “Wait until this command completes before continuing” on and have this action at the top of the action list for the command. That way you can be sure that my profiles are initialized properly before your personal startup actions are processed.



Add a Command

This command is executed:

- [ ] When I say: startup
- [ ] When I press keys: Not assigned
- [ ] When I press button: Not assigned
- [ ] When I press mouse: Not assigned

When this command executes, do the following sequence:

Execute command, 'alterNERDtive-base.startup' (by name) (and wait until it completes)

Description:

Category:

- [✓] Allow other commands to execute w
- [ ] Always execute this command
- [ ] Send command to this target:
  - [● Active Window] ○
- [ ] Stop command if target window foc
- [ ] Resume command if focus is reg

Recognition: Normal

Command Type: Full command

Prefix/suffix group:

- [ ] Minimum confidence level 0

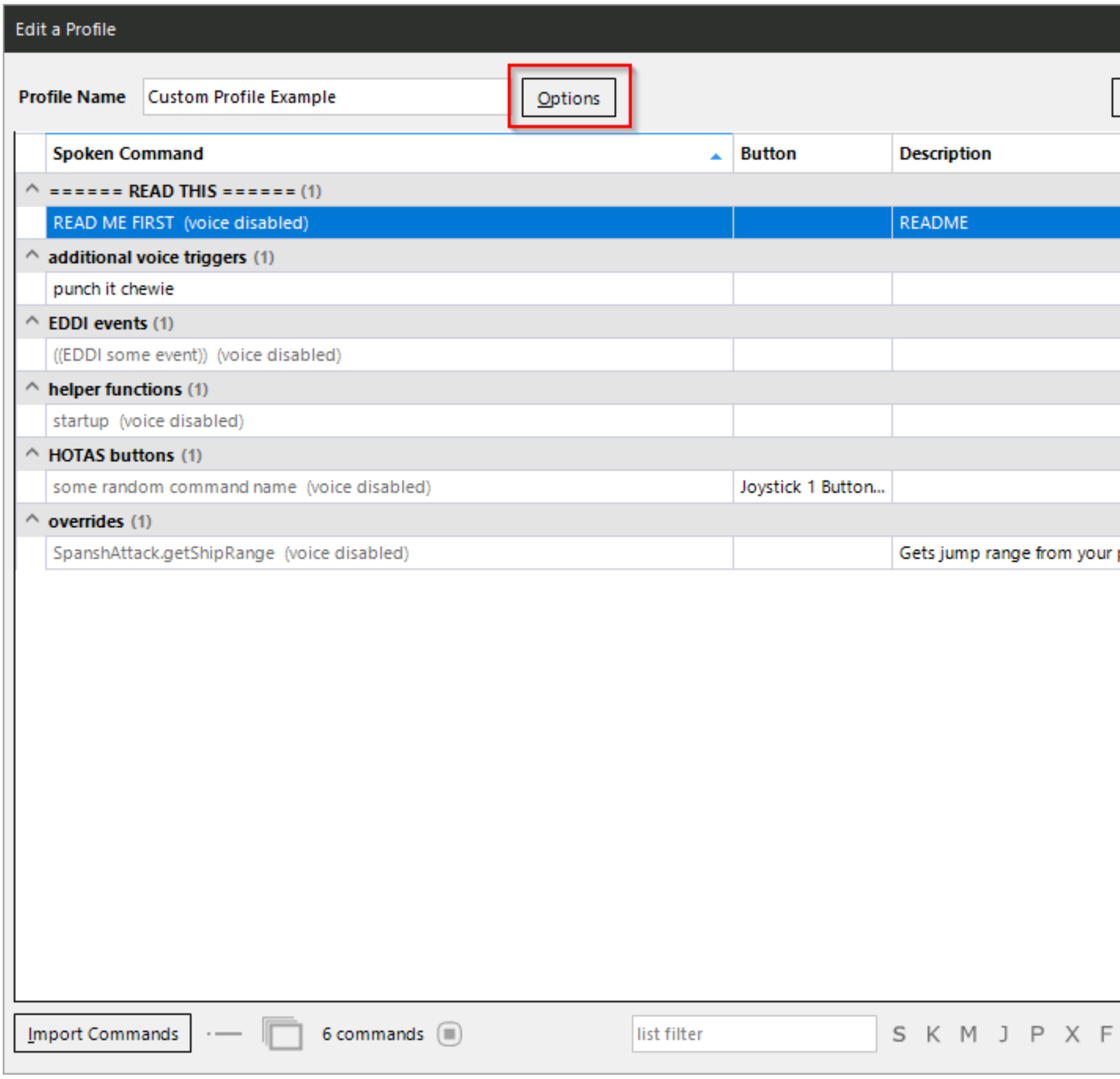
Repeating: Execute only once

2 times

You can add anything else you want your profile to do when it loads below this action. You do not have to set any configuration options, this can be done way more elegantly! More on this later on.

Set Profile Options¶

While editing the profile, hit the "Options" button.



Edit a Profile

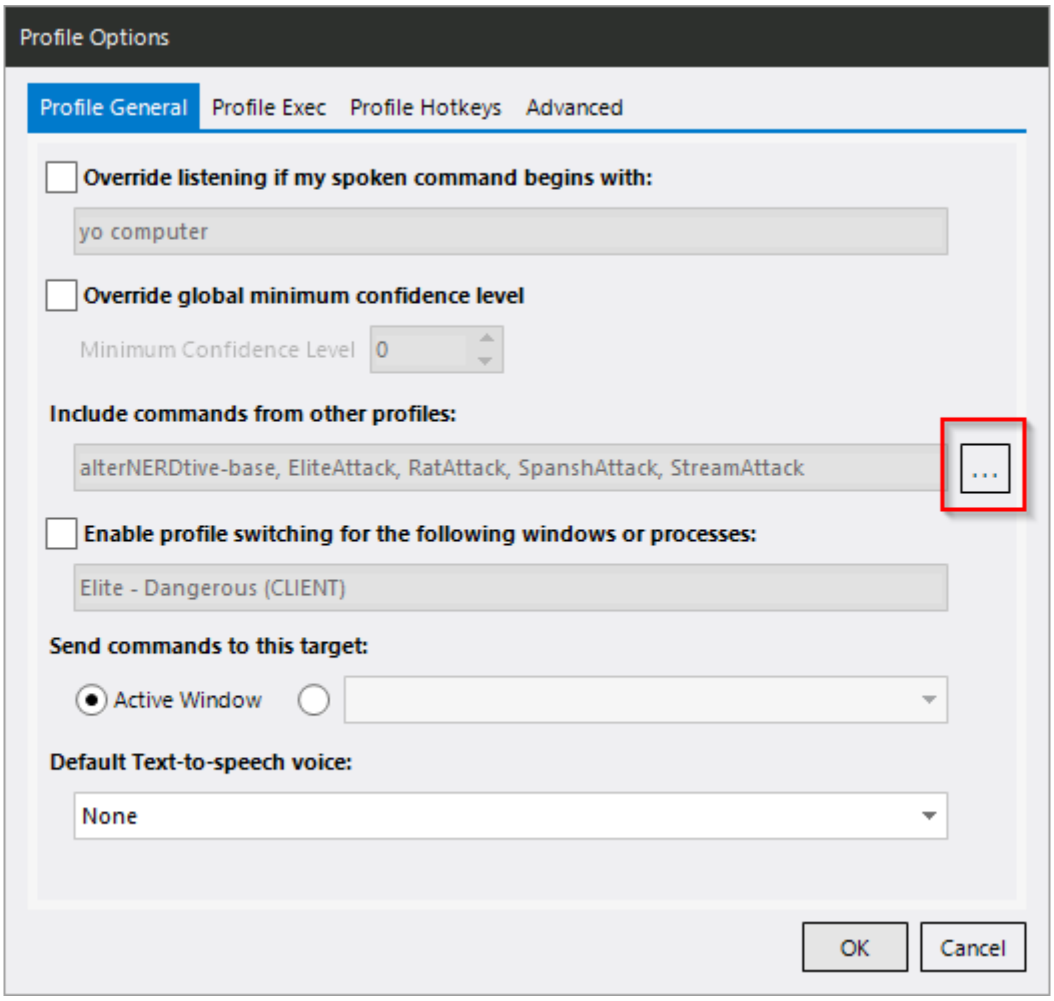
Profile Name: Custom Profile Example

Options

Spoken Command	Button	Description
====== READ THIS ====== (1)		
READ ME FIRST (voice disabled)		README
additional voice triggers (1)		
punch it chewie		
EDDI events (1)		
((EDDI some event)) (voice disabled)		
helper functions (1)		
startup (voice disabled)		
HOTAS buttons (1)		
some random command name (voice disabled)	Joystick 1 Button...	
overrides (1)		
SpanshAttack.getShipRange (voice disabled)		Gets jump range from your

Import Commands — 6 commands list filter S K M J P X F

On the section labeled"Include commands from other profiles", hit the "…" button.


Profile Options

Profile General | Profile Exec | Profile Hotkeys | Advanced

☐ Override listening if my spoken command begins with:

yo computer

☐ Override global minimum confidence level

Minimum Confidence Level: 0

Include commands from other profiles:

alterNERDtive-base, EliteAttack, RatAttack, SpanshAttack, StreamAttack

☐ Enable profile switching for the following windows or processes:

Elite - Dangerous (CLIENT)

Send commands to this target:

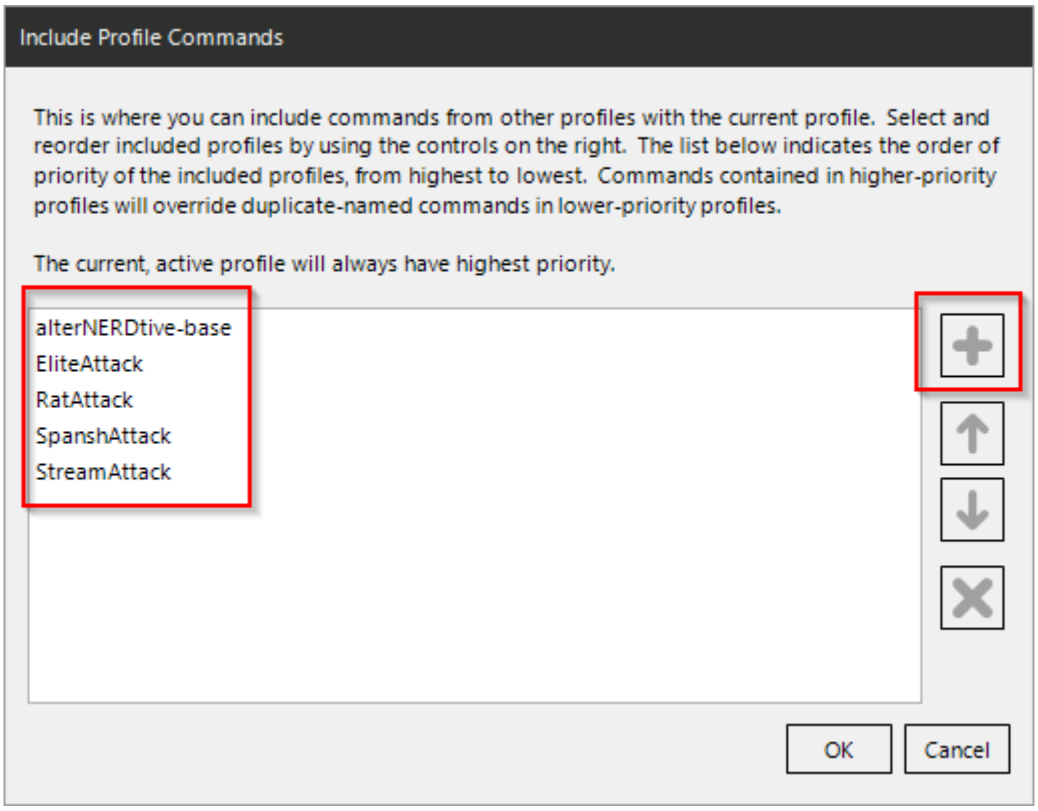
● Active Window ○

Default Text-to-speech voice:

None

OK | Cancel

Add all my profiles (“EliteDangerous”, “RatAttack”, “SpanshAttack”, `StreamAttack`).


Include Profile Commands

This is where you can include commands from other profiles with the current profile. Select and reorder included profiles by using the controls on the right. The list below indicates the order of priority of the included profiles, from highest to lowest. Commands contained in higher-priority profiles will override duplicate-named commands in lower-priority profiles.

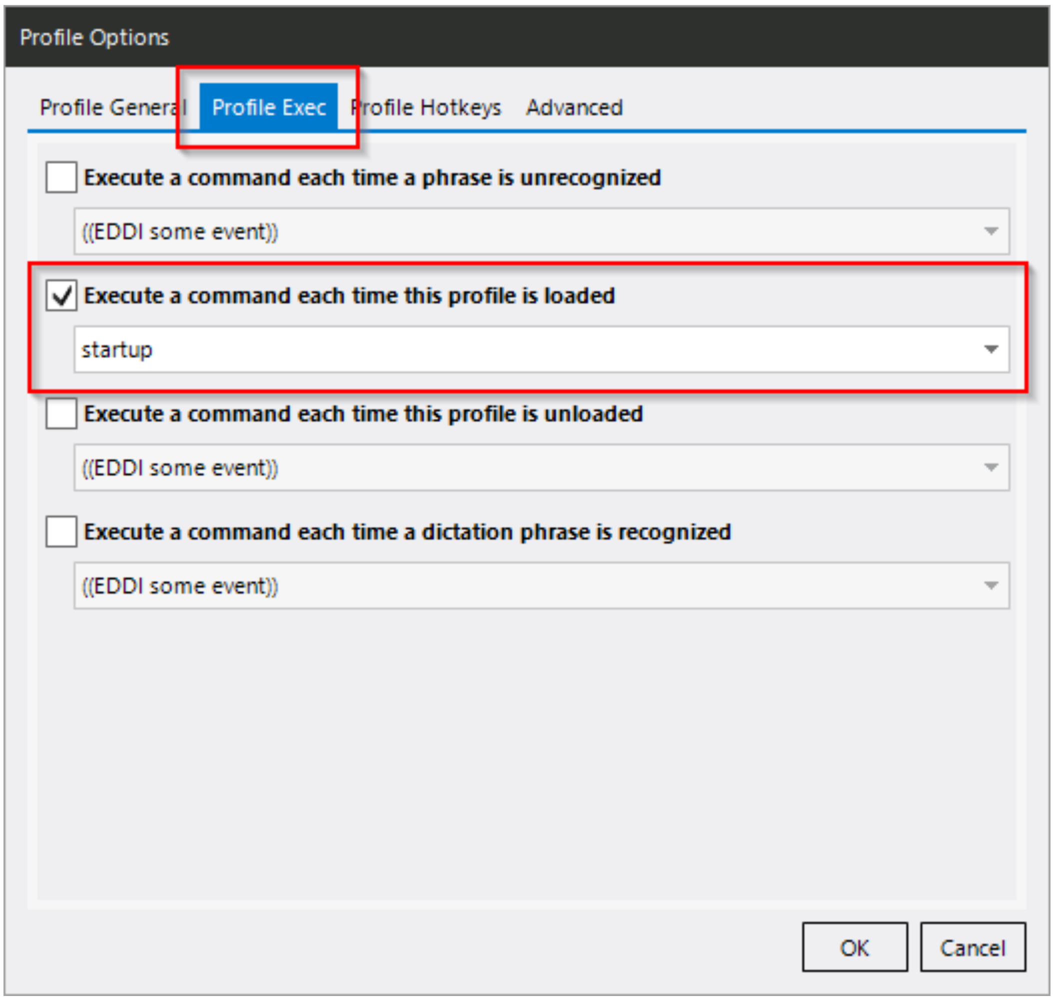
The current, active profile will always have highest priority.

alterNERDtive-base
EliteAttack
RatAttack
SpanshAttack
StreamAttack

OK | Cancel

Technically you can leave out anything you are not planning on using. Practically it probably will nott hurt you to just include everything, and it will then available for you in the future if you choose to check it out! Make sure that“alterNERDtive-base” is on top of the list, the order of the others does not matter. But I like it nice and alphabetical …

Now switch to the “Profile Exec” tab. Tick the “Execute a command each time this profile is loaded” checkbox, and select the “startup” command you have created earlier.



- [ ] Execute a command each time a phrase is unrecognized: ((EDDI some event))
- [✓] Execute a command each time this profile is loaded: startup
- [ ] Execute a command each time this profile is unloaded: ((EDDI some event))
- [ ] Execute a command each time a dictation phrase is recognized: ((EDDI some event))

Upgrading¶

4.0.0¶

Configure

VoiceAttack¶

General Configuration¶

Additionally, you need to have keyboard binds setup at least as secondary bindings in Elite’s controls options. VA *cannot* “push” joystick buttons for you, it can only do keyboard inputs. Hence its only way to interact with Elite is through keyboard emulation, even if you otherwise play the game with a controller or HOTAS. Or racing wheel. Or Rock Band set. Or bananas.

Settings¶

All profiles will load sane defaults if you haven’t changed anything. You no longer need to fiddle with the `startup` commands of each profile, instead you can use voice commands to change settings! See the `docs/` and the `_configuration` commands section of each profile.

Basically all the settings are available using the `customize settings` prefix, then saying `[enable;disable] <setting>` for on/off switches and `set <setting>` for text variables.

One caveat applies: settings will only be saved in the profile you have selected, but be preserved if you switch around.

Making changes¶

If you want to edit a command or add your own, *do not edit the profiles directly*. Instead create commands in your custom profile, and copy commands you want to change over to that before editing them. This will make sure no changes are lost if you update the profiles.

Because of limitations of VoiceAttack itself, only the first matching command found will be executed, *including EDDI events*. That means that if you create commands to handle EDDI events, you are going to have to check the imported profiles if they rely on these event handlers as well, and call them manually if they do.

E.g. if you want to create a custom `((EDDI Message sent))` handler in your profile, you will have to make it excute the `EliteDangerous.EDDI Message sent` and `RatAttack.EDDI Message sent` commands. Otherwise stuff *will* break.

If you have no idea what the previous two paragraphs were about, you can most likely just ignore them.

EliteAttack¶

This is my personal VoiceAttack profile for Elite: Dangerous. It started out ages ago as a modification of MalicVR’s public profile, then looked less and less and less like that and I added and cleaned up a lot of things while removing the stuff I didn’t use anyway. By now it would have probably been simpler to start from scratch.

Some of it has grown to a state that it might be useful to others in its own package, so I’ve separated the neutron jumping and Seals stuff into their own profiles.

The rest is a random conglomerate of all things VA and E:D; from various voice commands to lots of EDDI event handlers.

Speaking of EDDI; it has become in integral part of my Elite experience, especially the plethora of information it extracts from the game’s journal and presents to you via lots and lots of status variables and by firing various events that can then be handled through VA commands. It’s great. Check it out. (You might want to make it talk a lot less in it’s personality options, or disable the speech responder entirely like I have.)

Requirements¶

In addition to the bindED and EDDI VoiceAttack plugins, this profiles needs my Python elite-scripts to do everything properly. The release page here includes a compiled version for Windows that does not need Python installed. If you use the profile package from the release page, they will be installed automatically.

Settings¶

Because Elite’s keyboard handling is … weird you’ll have to set the key to use for pasting text into Elite:Dangerous. If you are not using a “standard” QWERT[YZ] layout, you will have to change it back to the key that is physically in the place where `v` would be on QWERTY.

For other settings, see the Configuration Variables section.

Including the Profile In Another Profile¶

This is meant to be a standalone profile, including the others in this repo (and a couple more). It was never designed to be included into your existing profile. Nevertheless, it *should* work properly if you follow some guide lines:

- Run the startup command. You will need to have a startup command in your profile (= one that is run on profile loading) and call `EliteDangerous.startup` by name from that one.
- Make sure all EDDI events that EliteDangerous needs are correctly handled. For all events used in EliteDangerous that you already have handlers for in your profile, you’ll have to include a call to `EliteDangerous.<event name>`. E.g. for “EDDI Jumped”, call `EliteDangerous.EDDI Jumped` by name from your `((EDDI Jumped))` command.

Usage¶

Chat Commands¶

There’s a bunch of commands in here to send certain things to chat. Unless stated otherwise, they will only work with the comms panel active, and you should be in the edit window ready to send. They will *not* hit Enter on their own.

- `clear [chat;text]`: Clears the chat window. Use from outside the comms panel.
- `[local;squad;system;wing] chat`: Puts you into the chosen chat channel.
- `paste text`: Pastes your clipboard into Elite. Works outside the comms panel too, e.g. on the galaxy map inside the search field.
- `salute; oh seven`: Will put “o7” into the chat.

Engineering/Materials¶

- `how many [<g5 manufactured materials list>] do i have`: Tells you how many of the given g5 manufactured material you currently have on board. I’ve restricted it to just those to not spam speech recognition with too many phrases, and because those are the ones I usually want to know while jumping around the bubble and having an eye on any HGE that might be around.
- `open e d engineer`: Opens the ED Engineer tool.
- `what [mats;materials] do i need?`: Runs the EDDI responder that tells you which materials are below wanted threshold. Needs setting those first. Gets very spammy if you do it for all of them; personally I only set them for g5 manufactured, so I can quickly check if it’s worth looking for HGE in a system I’m in.

Events¶

The main point of this profile is to react to Elite’s journal events (powered by EDDI). Each of the events listed here will trigger a `((EDDI <event>))` command in VoiceAttack which in turn triggers `EliteDangerous.EDDI <event>` and all included profiles’ similar commands, e.g. `SpanshAttack.EDDI <event>`.

The actual `((EDDI <event>))` command will only be executed once by VoiceAttack, the first one it can find. Keeping it separate from the actual code to be run makes it easier to handle (multiple) included profiles.

If the command for an event will send key presses to Elite, it focuses the client window first to make sure they get registered properly. If it is not in focus, the first key press might be swallowed otherwise.

For other commands, the profile just assumes that you are actively doing something in game and it is already focused :)

Body Mapped¶

Announces an estimate for high-value bodies’ payouts and the remaining mapping candidates in the system as given by EDDI.

Body scanned¶

Announces any interesting body traits found when scanning:

- terraformable
- Earth-like World, Ammonia World or Water World
- landable and >5 g
- semimajor axis <1.5 ls (only really interesting for planets, but there’s no proper way to separate them from moons, sadly)
- radius <300 km

Feel free to suggest more!

Carrier Cooldown¶

Tells you when your carrier is able to do its next jump. Technically only works if you were docked at it when it performed the jump, but I’m doing some behind-the-scenes magic if you weren’t :)

Might be slightly off (too early) in the latter case.

Carrier Jumped¶

Announces system and body your carrier has just jumped to.

Carrier Jump Engaged¶

This event fires when your carrier jumps but you are *not* docked at it. It provides way less information than the `Carrier Jumped` event, but hey, I don’t use most of it anyway.

Basically just calls `Carrier Jumped` (and makes sure that a `Carrier cooldown` event is triggered manually at approximately the right time).

Carrier Jump Request¶

Announces the system and body your carrier has just been scheduled to jump to. Use this to double check ingame information; I’ve had my carrier accept a body as jump target, but then end up around the star. This *might* give you a heads up on that.

Carrier Pads Locked¶

Announces your carriers lockdown procedures. This *might* only work when you are docked (which would make it pretty useless). Feel free to open in issue if that’s the case.

Discovery Scan¶

Announces the number of bodies (and non-body signals) found in the system. Also compares the number of bodies to the amount reported by EDSM (requires Python scripts).

Docked¶

Automatically gets your ship into the hangar and opens station services.

Docking Denied¶

Tells you the reason for docking denial.

Entered Normal Space¶

Throttles to 0 upon dropping from SC, if `EliteDangerous.hyperSpaceDethrottle` is set.

Fighter Launched¶

Orders your ship to hold position so it doesn’t chase after you immediately.

Jet Cone Boost¶

Sets your ship to full throttle immediately after you have supercharged.

Jumped¶

- zeroes throttle
- gets the system’s body count from EDSM (requires Python scripts)
- gets stations with outdated data (older than 1 year) from Spansh’s API
- if you haven’t visited the systems, starts a discovery scan (see the discovery scan command)
- last but not least tells you about planets worth scanning if you are on the R2R

Liftoff¶

Retracts landing gear for you. Seriously, is there any occasion in which you *don’t* immediately want to retract it after takeoff?

Low Fuel¶

Warns you when you reach 25% fuel. Also reports number of jumps you have left or the (rough) range you still have on the fumes left in your tank.

Material Threshold¶

Warns you when a monitored material falls below it’s minimum stock level. You will have to set a minimum desired amount in EDDI’s material monitor options first for all materials you wish to be monitored.

Message Sent¶

Checks any message you send for a chat prefix and sends it to the proper chat window. Probably largely useless to you without modification.

- `.nc`: Actually doesn’t send anything, but runs the `RatAttack.announceNearestCMDR` command with the system given in the rest of the message.
- `.dc`: Sends the message to the Discord window.
- `.tc`: Sends the message to my twitch channel window (IRC #alternerdtive).

There are similar event commands in RatAttack and SealAttack handling other chat windows.

Ship FSD¶

This event actually is several different events in one. Currently the following are handled:

- charging: Warns you if your target system’s main star is not scoopable, including an extra warning at low fuel levels. (**Note**: This only works if the target system is in EDSM. So it’s kind of useless for its intended use (exploration) and probably going to be removed at some point.)
- cooldown complete: Announces FSD cooldown if you are currently in normal space.

Ship interdicted¶

Tells you when you have been interdicted by a player. Is also supposed to target the interdictor automatically, but randomly sometimes just doesn’t work. Yay!

Ship targeted¶

This currently doesn’t do anything. I was fiddling around with automatically targeting a certain module on ship targeting, but it was more hassle than I had thought.

SRV Launched¶

Toggles SRV lights off after launching. Might not work if you drop particularly far after deployment because it works off a timer. Conversely might take a second to turn your lights off on a short drop and/or in high gravity.

System Scan Complete¶

Lists you all bodies EDDI considers worth mapping in the current system.

Undocked¶

Retracts landing gear for you. Seriously, is there any occasion in which you *don't* immediately want to retract it after takeoff?

VA initialized¶

Fires when the EDDI VoiceAttack plugin is loaded. Makes sure that EDDI is set to quite mode even if the profile was loaded before plugin initialization had completed.

Misc¶

The commands in here do random more or less useful things.

- `bind keys;reset key binds`: Reloads your key binds through the bindED plugin. You should do that after changing anything in the controls options.
- `copy current system`: Copies the current system name into the clipboard.
- `distance [from;to] […]`: Tells you the distance from your current position to the other thing you mentioned and is supported in the command. (requires Python scripts)
- `do a barrow roll`: WHOOOOOOO!
- `fix window dimensions`: When you start the game in VR, it forces into windowed mode with weird resolution. This changes it back. Hover the "PLAY" entry in the main menu, then run this. Will need adjustment for different graphics cards/drivers and the resolution you want.
- `neutron [jump;trip] time`: Shorter version of the same thing in SpanshAttack.
- `neutron jumps left`: Shorter version of the same thing in SpanshAttack.
- `open copied system on EDSM`: Opens the system in your clipboard on EDSM in your default browser.
- `open coriolis`: Opens Coriolis in your default browser.
- `open [current;] system on EDSM`: Opens your current system on EDSM in your default browser.
- `open EDDI options; configure EDDI`: Opens the EDDI configuration window.

- `open e d d b [station;system;faction;] [search;]`: Opens EDDB in your default browser.
- `open e d s m`: Opens EDSM in your default browser.
- `open inara`: Opens Inara in your default browser.
- `open materials finder`: Opens EDTutorials' materials finder in your default browser.
- `open miner's tool`: Opens https://edtools.ddns.net/miner in your default browser.
- `reload bindings`: Reloads your bindings for bindED.
- `shut up EDDI`: Immediately stops any ongoing (and queued) EDDI speech.
- `[start;stop] [EDISON;navigation]`: Hits `CTRL+ALT+E` which just so happens to be the start/stop hotkey I have set in E.D.I.S.O.N.
- `[bodies;what's;what is] left to [map;be mapped;scan]`: Tells you which bodies EDDI thinks are worth mapping in the system that you haven't mapped yet.

Navigation¶

There are so many navigation-focused commands now, they deserve there own category. Basically anything that helps you plot anywhere. A lot of those are powered by awesome EDDI so I don't have to do the work myself!

- `plot course;[target;] next [waypoint;way point]`: Plots a course to the system set in `~~system` or the one in your clipboard. The former way is usually used by other commands to not interfere with your clipboard.
- `[find;target] nearest [encoded;manufactured;raw] material trader`: Targets the nearest respective material trader.
- `[find;target] nearest [guardian;human] tech broker`: Targets the nearest respective tech broker.
- `[find;target] nearest [interstellar factor;mission system;scoopable star]`: Well, you know the drill by now.
- `[find;target] nearest mission system`: Targets the nearest system that has a mission target.
- `[find;target] [<system>]`: Targets the given system on the galaxy map. There's a bunch in there, the list is easily extensible. Drop me a note if you want something included.

Ship Controls¶

Basically anything that is related to directly doing something with your ship.

- `[abort;cancel;stop] jumping`: Stops a currently charging FSD jump.
- `[buggy;exploration] power`: Sets your PD to 0/4/2 or 2/4/0 respectively. Works in SRV too.
- `[close;deploy;extend;open;retract;] […] [up;down;]`: Overly complicated command to handle everything related to Cargo Scoop, Hard Points, Landing Gear. You get the gist, I guess. Works in SRV too.
- `[disco;discovery scan]`: Executes a discovery scan. To work properly, you'll have to set the Discovery Scanner to your first fire group, secondary fire.

- `[dis;]engage silent running`: Turns silent running on and off.
- `[head;spot;] lights [on;off]`: Turns your lights on and off. Works in SRV too, kinda; turning lights off there relies on the state updating fast enough, which sometimes leads to weird results.
- `[jump;engage;get me out;punch it chewie] [and scan;] [when ready;]`: Retracts everything that might be protruding from your ship, then jumps to the next system. If the FSD isn't charging within 1s, it gets you into SC instead (e.g. if your target is obstructed). If given "and scan" runs a discovery scan. If given "when ready" waits for mass lock to clear, your FSD to cool down and you to leave scoop range before jumping.
- `night vision [on;off]`: Toggles your night vision on/off. Works in SRV too.
- `power to [engines;shields;systems;weapons`: Sets 4 pips to the thing you told it, 1 to the others.
- `rapid fire lights`: Flashes your lights 5 times in a row.
- `retract [all;everything]`: Retracts, well, everything.
- `[start;stop] [firing;mining]`: Starts/stops holding down primary fire. Mostly useful when mining. When triggered with "mining", also deploys the cargo scoop.
- `[super;] cruise [when ready;]`: Retracts everything, then jumps to SC. If given "when ready" will wait for mass lock to clear and your FSD to cool down first.

SRV controls¶

Things relevant to your SRV, but not your ship.

- `[recall;dismiss] ship`: Recalls or dismisses ship. Currently does the same thing regardless of the state of your ship. I wish it would be possible to restrict it to doing one thing each, but that's currently not possible sadly.
- `[toggle;enable;disable] drive assist`: Handles all your drive assist needs!

Targeting¶

Well … targeting stuff, I guess. Not really sure why I made that it's own category, but oh well :)

- `target next system`: Selects the next system on your route.
- `target wing man [1;2;3]`: Targets your wing men.
- `target's target`: Targets your target's target.
- `wing man [1;2;3] target`: Targets your wing men's target.
- `wing man nav lock`: Toggles wing man nav lock on the selected wing member.

UI Commands¶

Everything handling stuff that’s not related to controlling your ship, but manipulating some UI element(s).

- `controls options`: Opens the controls options menu.
- `docking request;request dock[ing;]`: Sends a docking request.
- `[enter;leave] F S S`: Opens/closes FSS.
- `galaxy map`: Opens the galaxy map.
- `[main;game] menu`: Opens the ESC menu.
- `[relog;reset] to [open;solo]`: Relogs to Open or Solo mode, respectively.
- `restart from Desktop`: Quits the game and restarts from an open launcher by clicking the play button.
- `set […] filter`: Sets a nav panel filter setting. See the command or just try different things for what is possible. You need to clear filters and hover over the filter button, then run this.
- `system map`: Opens the system map.
- `take [high res;] screenshot`: Takes a (high res) screenshot.
- `toggle orbit lines`: Toggles the visibility of orbit lines.
- `[toggle;show;hide] interface`: Toggles the cockpit interface (CTRL+ALT+G). Probably needs to be adjusted if you are not playing with Neo2 keyboard layout :)

Update Commands¶

- `check for profiles update`: Does just that. Is also automatically run each time the profile is started.
- `download profiles update`: Downloads a profiles update if applicable. Will prompt you to restart VoiceAttack when the download has finished to import the updated profiles.
- `open profiles change log`: Opens `CHANGELOG.md` on Github.

Logging¶

The profile supports logging a bunch of stuff to the VoiceAttack event log. By default, logging is concise and constrained to basically error messages.

If you need more logging (usually for debugging purposes), say `enable logging`. If you want to enable verbose logging *by default*, call the `Logging.enableLogging` command from your custom profile’s `startup` command.

Configuration Variables¶

There are a bunch of configuration variables. You should not overwrite those manually, instead use the provided commands in the `_configuration` section!

Basically all the settings are available using the `customize settings` prefix, then saying `[enable;disable] <setting>` for on/off switches and `set <setting>` for text variables.

- `EDDI.quietMode` (boolean): whether or not to set EDDI to quite mode. Default: true.
- `Elite.pasteKey` (string): the key used for pasting into Elite. On QWERTY this is `v`. Default: `p`.
- `EliteDangerous.announceEdsmSystemStatus` (boolean): whether to announce the system or some of its bodies missing on EDSM. Default: true.
- `EliteDangerous.announceMappingCandidates` (boolean): whether to announce mapping candidates when they are scanned. Default: true.
- `EliteDangerous.announceOutdatedStationData` (boolean): whether to announce stations with outdated data in systems you jump to. Default: true.
- `EliteDangerous.announceR2RMappingCandidates` (boolean): whether to announce planets worth mapping when jumping into a known system. This is useful for doing some R2R on the side. Default: false.
- `EliteDangerous.enableCarrierAnnouncements` (boolean): whether or not to process fleet carrier events. Default: true.
- `EliteDangerous.enableAutoUpdateCheck` (boolean): whether or not to automatically check for updates on profile startup. Default: true.
- `EliteDangerous.flightAssistOff` (boolean): whether to automatically toggle FlightAssist off on liftoff. Default: true.
- `EliteDangerous.hyperspaceDethrottle` (boolean): same thing as the SC assist setting; if on, will throttle to 0 automatically after jumping. Default: true.
- `EliteDangerous.oldStationThreshold` (int): Age in days that will cause station data to be considered outdated. Default: 365 (1 year).
- `python.ScriptPath` (string): the path you have placed the compiled python scripts in. Default: “{VA_DIR}\Sounds\scripts” (the “\Sounds\scripts” folder in your VoiceAttack installation directory).

Delays / Pauses¶

Delays needed e.g. between key presses in UI navigation can vary wildly depending on your PC’s specs and configuration. Therefore they should be configurable, shouldn’t they?

So far those actually are:

- `EliteDangerous.delays.quitToDesktop`: Delay between quitting to desktop and hitting the play button in the launcher.

RatAttack¶

This profile facilitates Fuel Ratting. It aims to eliminate as much of the required manual task and attention switching as possible via automation and voice commands.

If you don't know what the Fuel Rats are, come hang out and ask :)

Requirements¶

Only vanilla VoiceAttack is absolutely required to use this profile. Optionally you can install EDDI and my elite scripts for advanced features.

- EDDI installed as a VoiceAttack plugin: This will give you a better (IMO) way of using TTS. Be sure to set `RatAttack.useEddiForVoice`. It will also enable you to have ingame chat be transferred to IRC; see below.
- elite-scripts: Using the Python scripts will give RatAttack a way to be aware of where your CMDRs are and give you the nearest one to a rat case. That's only really needed if you actually *have* multiple CMDRs, obviously. If you are using the profile package from the release page, they will be installed automatically.

EDDI speech responder¶

For the convenience of people that have not been using EDDI in the past, RatAttack will deactivate the speech responder automatically to not clutter them with unwanted TTS.

If you are already an EDDI user and want to keep the default speech responder functionality, you will have to disable the `EDDI.quietMode` setting by running the `customize settings disable eddi quiet mode` command.

Settings¶

There are a lot of preferences you can set, including some you really want to concern yourself with before you start using the profile. Some of the more advanced features heavily rely on you giving it the correct things to work with.

See the Configuration Variables section.

Including the Profile¶

When including the profile, be sure to

- Run the startup command. You will need to have a startup command in your profile (= one that is run on profile loading) and call `RatAttack.startup` from that one.
- Make sure all EDDI events that RatAttack needs are correctly handled. For all events used in RatAttack that you already have handlers for in your profile, you'll have to include a call to `RatAttack.<event name>`. E.g. for "EDDI Message sent", call `RatAttack.EDDI Message sent` by name from your `((EDDI Message sent))` command.

Usage¶

Going On/Off Duty¶

When you are on duty, RatAttack will automatically announce cases coming in through IRC. When off duty, it won't.

- `[enable;disable] rat duty`: puts you on/off duty.
- `open [rat;] dispatch board`: opens the web dispatch board.

Handling a Case¶

Getting Case Data From IRC¶

You can setup your IRC client to pass incoming RATSIGNALS to VoiceAttack by writing them to a file (`%appdata%\Ratattack\ratsignal.pipe`), then calling the appropriate command (`RatAttack.announceCaseFromRatsignal` for notification, `RatAttack.getInfoFromRatsignal` for silently putting it into the case list).

This has two purposes:

1. announcing a new incoming case
2. storing case data and making it available to VoiceAttack, e.g. for copying the case's system into the clipboard

You need to make your IRC client

1. wait until the file disappears (for several cases coming in at once)
2. write the RATSIGNAL to the file
3. run the VoiceAttack command

In my case I am running AdiIRC and have the following script setup for handling this:

```
on *:TEXT:RATSIGNAL - CMDR*(??_SIGNAL):#fuelrats:{
    /mkdir C:\users\<user>\appdata\roaming\RatAttack\
    /handleratsignal $1-
}
alias handleratsignal {
    if ( $exists(C:\users\<user>\appdata\roaming\RatAttack\ratsignal.pi
        /sleep 1 /handleratsignal $1-
    }
    else {
        /write C:\users\<user>\appdata\roaming\RatAttack\ratsignal.pipe
        if ( $away ) {
            /run -h "X:\path\to\VoiceAttack\VoiceAttack.exe" -nofocus -
        }
        else {
            /run -h "X:\path\to\VoiceAttack\VoiceAttack.exe" -nofocus -
        }
    }
}
```

You get the gist; if not and you don’t know how to do the same thing for your IRC client or it doesn’t support copying the control characters in the ratsignal that the profile uses to split the information, either switch to AdiIRC or bribe me to include some other way to get case data into VoiceAttack.

Note: If you are running VoiceAttack as admin you need to run your IRC client as admin, too! Otherwise it can’t talk to VoiceAttack for security reasons. You really should *not* run VoiceAttack with elevated privileges though. Or anything.

Internal Case List¶

If you have your IRC client setup properly, VoiceAttack will hold a list with all rat cases that have come in while you had it running. It will save the case number, CMDR name, system, O_2 status and platform. There are several commands you can run on this list, giving it a case number:

- `rat case number [0..19] details`: Will give you all stored info on a case.
- `[current;] rat case details`: Will give you all stored info on the currently open case.
- `distance to current rat case`: Will give you the distance from your current location to the currently opened rat case. Requires the use of my `elite-scripts` Python scripts.
- `distance to rat case number [0..19]`: Will give you the distance from your current system to a case’s system. Requires the use of my `elite-scripts` Python scripts.
- `nearest commander to rat case number [0..19]`: Will give you the nearest of your CMDRs with their distance to a case’s system. Requires some setup and the use of my `elite-scripts` Python scripts.
- `nearest commander to [the;] rat case`: Will give you the nearest of your CMDRs with their distance to the current case’s system. Requires some setup and the use of my `elite-scripts` Python scripts.

Opening a Case¶

- `open rat case number [0..19]`: Opens rat case with the given number. If there is no case data for that case (e.g. because you don’t have your IRC client set up properly), it will still open it, just not have any data on it.
- `open [latest;] rat case`: Opens the latest rat case that has come in through IRC. Will probably error out in creative ways if you don’t have your IRC client set up properly. Too tired right now to have proper error handling so just open an issue if you run into problems (it’s 7am, I haven’t slept and want to finish this doc to get the release out (yes, you are allowed to laugh at this section)).

Closing a Case¶

- `[close;clear] rat case`: Closes the currently open rat case.

Making Calls¶

There are a bunch of calls you can make for a case, the most common are modelled through VoiceAttack commands. The descriptive commands (e.g. “system confirmed”) will be shortened to the usual IRC short hands (e.g. “sysconf”). If you need something more unusual you can either still manually type it into your IRC client or use the “General IRC Integration”, see below.

- `call [1..20] jumps [and login;and takeoff;left;]`: Calls jump for the currently open case. You can optionally include that you will still have to login to the game or have to take off from your current station/port/outpost/planet.
- `call friend [positive;negative] [in pg;in private group;in solo;in main meu;sysconf;system confirmed]`: Friend request confirmations, with all the things you might want to / should call with it.
- `call [beacon;fuel;instance;pos;position;prep;sys;system;wing] [positive;negative]`: All the stuff you usually need for ratting after you have received the friend request.
- `call wing pending`: Calls “wr pending” for when it takes 30s again to actually show up.
- `call client in [exclusion zone;main menu;open;open sysconf;pg;private group;solo;super cruise]`: Callouts for all the various things a client could get themselves into.
- `call [client destroyed;client offline;sysconf;system confirmed]`: This is the command you don’t want to use. Include sysconf in your “friend+” or “in open” calls, and make sure you will never have to call “client destroyed”, would you?

General IRC Interaction¶

(requires EDDI)

Using EDDI to read the game’s journal, you can send messages to IRC from Elite’s ingame chat.

Be aware that the chat message will still appear in the ingame chat channel you send it to!

I recommend using local chat and limiting the use to instances that will probably not have other players in it (e.g. instanced in normal space with the client or in SC in some remote system out in the black on a long range rescue).

- **fuelrats: Use “.fr \<message>” to have VoiceAttack send “#\<caseNumber>¶**

  \<message>” to the #fuelrats channel – or yell at you when you are not on a case.

- **ratchat: Use “.rc \<message>” to have VoiceAttack send “\<message>” to¶**

  #ratchat.

These commands send their text to windows with “#fuelrats” and “#ratchat” in their title, respectively. If your IRC client does not do that, you will have to change the “target” window of the `RatAttack.sendToFuelrats` and `RatAttack.sendToRatchat` commands to reflect the actual window titles on your system. I will look into making this more elegant to change in the future.

Logging¶

The profile supports logging a bunch of stuff to the VoiceAttack event log. By default, logging is concise and constrained to basically error messages.

If you need more logging (usually for debugging purposes), say `enable logging`. If you want to enable verbose logging *by default*, call the `Logging.enableLogging` command from your custom profile’s `startup` command.

Exposed Variables¶

The following Variables are *global* and thus readable (and writeable! Please don’t unless it’s a config variable …) from other profiles.

Configuration Variables¶

There are a bunch of configuration variables. You should not overwrite those manually, instead use the provided commands in the `_configuration` section!

Basically all the settings are available using the `customize settings` prefix, then saying `[enable;disable] <setting>` for on/off switches and `set <setting>` for text variables.

- `EDDI.quietMode` (boolean): whether or not to set EDDI to quite mode. Default: true.
- `EDDI.useEddiForVoice` (boolean): whether to use the EDDI plugin to handle text-to-speech over VoiceAttacks built-in speech function. Default: false.
- `Elite.pasteKey` (string): the key used for pasting into Elite. On QWERTY this is `v`. Default: `v`.
- `RatAttack.announceNearestCMDR` (boolean): whether or not to automatically announce your nearest CMDR to a case. Requires the `elite-scripts` Python scripts. Will probably break in creative ways if you don't have them and turn it on anyway. Default: false.
- `RatAttack.announcePlatform` (boolean): whether or not to announce the case's platform by default. Useful to set if you are active on more than one platform. Even with this off, you will still be warned when you open a case that is *not* on one of your platforms. Default: false.
- `RatAttack.CMDRs` (string): list of your CMDR names, delimited by spaces. If your names include spaces, you will have to put them in quotes. Default: “"J Jora Jameson" NameWithNoSpace”.
- `RatAttack.confirmCalls` (boolean): whether VoiceAttack should ask you before posting to #fuelrats to make sure there hasn't been an error in voice recognition and you accidentally post the wrong thing. Default: true.
- `RatAttack.autoCloseCase` (boolean): whether or not to automatically close an open rat case on calling “fuel+”. Default: false.
- `RatAttack.onDuty` (boolean): whether or not you are currently on rat duty. Default: true.
- `RatAttack.platforms` (string): the platforms you want to be informed of incoming cases for. If you are on console, you can still have VoiceAttack running on the PC that you are using for IRC and handle calls and stuff using voice! Delimited by whatever you want. Can include “PC”, “XB”, “PS4”. Default: “PC”.
- `python.scriptPath` (string): the path you put the Python scripts in. Default: “{VA_DIR}\Sounds\scripts”.

Other Variables¶

Current case data:

- `RatAttack.caseNumber` (int): the number of the case you are currently on. Will be `Not Set` if you are not on a case.
- `RatAttack.onCase` (boolean): whether or not you are currently on a case.

Case list:

- `RatAttack.caseList.<case#>.cmdr` (string)
- `RatAttack.caseList.<case#>.system` (string)
- `RatAttack.caseList.<case#>.platform` (string)

- `RatAttack.caseList.<case#>.codeRed` (boolean)

… with `<case#>` being a number between 0 and 19.

SpanshAttack¶

This profile uses the ED-NeutronRouter plugin to plot neutron jumps using spansh. It fully does everything you need from within the game and VoiceAttack, you won't have to visit the site at any point.

Requirements¶

In addition to VoiceAttack, you will need the following plugins to use this profile:

- bindED
- EDDI installed as a VoiceAttack plugin
- ED-NeutronRouter: required for SpanshAttack. **Make sure to grab the pre-release 1.02** since 1.01 has a bug with a hardcoded 50 ly jump range.

EDDI speech responder¶

For the convenience of people that have not been using EDDI in the past, SpanshAttack will deactivate the speech responder automatically to not clutter them with unwanted TTS.

If you are already an EDDI user and want to keep the default speech responder functionality, you will have to disable the `EDDI.quietMode` setting by running the `customize settings disable eddi quiet mode` command.

Settings¶

Because Elite's keyboard handling is … weird you'll have to set the key to use for pasting text into Elite:Dangerous. If you are not using a "standard" QWERT[YZ] layout, you will have to change it back to the key that is physically in the place where `v` would be on QWERTY.

For other settings, see the Configuration Variables section.

The last "setting" in the not-so-strict sense of the word is the `SpanshAttack.getShipRange` command. Any ship listed in there will automatically have its jump range used instead of VA prompting you for it. Since, again, VA will execute the first matching command found, you can create this command in your own profile when you are using SpanshAttack by including it. You can override a saved range for your ship by using the `plot neutron [course;route;trip] with custom range` command.

The ED-NeutronRouter plugin is technically supposed to read the current jump range from EDDI; sadly a) it's bugged right now, and b) EDDI is storing the *maximum* distance for your ship instead of the current / full on fuel one.

Including the Profile¶

When including the profile, be sure to

- Run the startup command. You will need to have a startup command in your profile (= one that is run on profile loading) and call `SpanshAttack.startup` from that one.
- Make sure all EDDI events that SpanshAttack needs are correctly handled. For all events used in SpanshAttack that you already have handlers for in your profile, you'll have to include a call to `SpanshAttack.<event name>`. E.g. for "EDDI Jumped", call `SpanshAttack.EDDI Jumped` by name from your `((EDDI Jumped))` command.
- (Optional) Have a `SpanshAttack.getShipRange` command in your profile to overwrite the default one with your ship's ranges. See the default command for pointers.

Usage¶

Plotting a Route¶

1. *Target* the system you want to be routed to (target, do not plot to it).
2. Either exit the galaxy map or make sure you are on its first tab (or auto-plotting will break).
3. Trigger the `SpanshAttack.plotRoute` command either by voice (`plot neutron [course;route;trip] [with custom range;]`) or calling it from another command.
4. Enter your ship's jump range if prompted.
5. Wait for the route to be calculated. The command will automatically open the galaxy map and jump to the first waypoint on your route. If you run into weird behaviour, it's probably because your target system is not in EDDB.
6. Either target the first waypoint or plot to it.
7. Start jumping!

Plotting to a System Unknown to the Neutron Router¶

The router can only plot a route to a system that is in its data base (obviously can also only give you way points that are). If your target system is not, there are several levels of fallback handling to find a system that is.

1. Check `Next system` coordinates provided by EDDI. If the system is in EDSM, but has for some reason not been sent over EDDN to other sites including Spansh we can get coordinates here.
2. If the system is not in EDSM check EDTS. It can calculate approximate coordinates for a given procedurally generated system name.

3. If that fails prompt the user for input.
4. Query Spansh' API for the closest system to these coordinates.
5. Plot a route to the closest system.

Generally you should almost never be asked to input coordinates manually. If EDTS provides coordinates with an accuracy that is worse than ±100 ly per axis, you will be prompted to make sure you are going roughly to the right coordinates. You will find the system that is used for plotting, its coordinates and the accuracy in VoiceAttack's log window.

Neutron Jumping¶

With standard settings, just supercharge off a neutron cone. You should automatically be taken to the galaxy map with the next waypoint selected.

In case you have disabled auto-plotting to the next waypoint, manually invoke the `SpanshAttack.targetNextNeutronWaypoint` command by voice (`[target;] next neutron [waypoint; way point]`) or calling it from another command.

Additionally, you can use the `SpanshAttack.copyNextNeutronWaypoint` / `[get;copy] next neutron [waypoint;way point]` command to copy the next neutron waypoint to the clipboard.

Manual Re-Plot¶

Trigger the `SpanshAttack.replotRoute` command either by voice (`replot neutron [course;route;trip]`) or calling it from another command. This will start a re-plot of the current route with the same target system and jump range.

Refueling¶

Whenever you refuel off a scoopable star, the profile will automatically throttle back up to 100% speed. Unless you have disabled it in your configuration, you will also automatically target the next system on your route and jump to it once you leave fuel scoop range.

Clearing a Route¶

When you reach your target system, the neutron route will automatically be cleared. If you want to prematurely end your trip, call the `SpanshAttack.clearRoute` / `clear neutron [course;route;trip]` command.

Other Commands¶

Announcing Jumps Left¶

You can have VoiceAttack tell you the amount of jumps left on the current route by invoking `SpanshAttack.announceJumpsLeft` or saying `how many [neutron;] jumps [are;] left?`.

Note: Because it's pretty much impossible to calculate a 100% accurate value for the total jumps left, it will just tell you the jump count *from the current neutron waypoint*.

Announce elapsed time on the trip¶

SpanshAttack keeps track of your start time, even if you have the option to time your trip turned off. This way you can get the time you've been jumping with the `SpanshAttack.announceTripTime` or `how long have i been [jumping;on this trip;on this neutron trip]?` commands.

Reload bindings¶

If you change any relevant bindings (e.g. the galaxy map key), you should run the `reload bindings` command to make sure that SpanshAttack presses the right thing for you.

Eh, just do it every time you edit your controls without re-starting VoiceAttack, just to be sure.

Helper Functions¶

The profile contains a lot of helper functions that get called by the aforementioned commands. Have a look around, maybe learn something about VoiceAttack :)

Logging¶

The profile supports logging a bunch of stuff to the VoiceAttack event log. By default, logging is concise and constrained to basically error messages.

If you need more logging (usually for debugging purposes), say `enable logging`. If you want to enable verbose logging *by default*, call the `Logging.enableLogging` command from your custom profile's `startup` command.

Exposed Variables¶

The following Variables are *global* and thus readable (and writeable! Please don't unless it's a config variable …) from other profiles:

Configuration Variables¶

There are a bunch of configuration variables. You should not overwrite those manually, instead use the provided commands in the `_configuration` section!

Basically all the settings are available using the `customize settings` prefix, then saying `[enable;disable] <setting>` for on/off switches and `set <setting>` for text variables.

- `EDDI.quietMode` (boolean): whether or not to set EDDI to quite mode. Default: true.
- `EDDI.useEddiForVoice` (boolean): whether to use EDDI over VA's builtin `say` command. Default: false.
- `Elite.pasteKey` (string): the key used for pasting into Elite. On QWERTY this is `v`. Default: `v`.
- `SpanshAttack.timeTrip` (boolean): whether to automatically tell you at the end of a trip how long it took you to get there. Default: false.
- `SpanshAttack.announceWaypoints` (boolean): whether to announce each waypoint of the neutron route. Default: true.
- `SpanshAttack.announceJumpsLeft` (string): `;`-separated list of remaining jumps to announce when said amounts are reached. Right now only works if they are *exactly* reached when supercharging off a neutron. Note the extra `;` at the beginning and end of the string. Default: `;1;3;5;10;15;20;30;50;75;100;`
- `SpanshAttack.autoJumpAfterScooping` (boolean): whether to automatically jump after fuel scooping (and moving out of scoop range). Default: true.
- `SpanshAttack.autoPlot` (boolean): whether to automatically plot to the next waypoint on supercharging. Default: true.
- `SpanshAttack.clearOnShutdown` (boolean): whether or not to automatically clear an active neutron route on Elite client shutdown. Default: true.
- `SpanshAttack.defaultToLadenRange` (boolean): whether or not to default to your ship's laden range (as reported by EDDI) instead of asking for user input. Sadly it's with *current* fuel, not full. Setting a ship's jump range in the `SpanshAttack.getShipRange` command will still overrule this. Default: false.
- `SpanshAttack.copyWaypointToClipboard` (boolean): whether to copy the next waypoint into the Windows clipboard for use in other programs. Default: false.
- `python.scriptPath` (string): the path you put the Python scripts in. Default: "{VA_DIR}\Sounds\scripts".

Other Variables¶

These variables can be used to get information about the current neutron route. Please do not set them manually and / or from outside the SpanshAttack profile.

- `SpanshAttack.plotSystem` (string): the system actually plotted towards using the neutron router (onley used/set if the target system is not in the data base)
- `SpanshAttack.targetSystem` (string): the target system for the current neutron route
- `SpanshAttack.nextNeutronWaypoint` (string): the next waypoint on the current neutron route
- `SpanshAttack.neutronJumpMode` (boolean): neutron jump mode active/inactive
- `SpanshAttack.jumpRange` (decimal): the current ship’s jump range

StreamAttack¶

This profile uses the EDDI plugin to write a bunch of information about your commander, your current location and your ship to files that can be accessed e.g. by your streaming software to be displayed on stream.

Default folder is `%appdata%\StreamAttack\`.

Requirements¶

In addition to VoiceAttack, you will need the following plugins to use this profile:

- EDDI installed as a VoiceAttack plugin

EDDI speech responder¶

For the convenience of people that have not been using EDDI in the past, StreamAttack will deactivate the speech responder automatically to not clutter them with unwanted TTS.

If you are already an EDDI user and want to keep the default speech responder functionality, you will have to disable the `EDDI.quietMode` setting by running the `customize settings disable eddi quiet mode` command.

Settings¶

See the Configuration Variables section.

Including the Profile¶

When including the profile, be sure to

- Run the startup command. You will need to have a startup command in your profile (= one that is run on profile loading) and call `StreamAttack.startup` from that one.
- Make sure all EDDI events that StreamAttack needs are correctly handled. For all events used in StreamAttack that you already have handlers for in your profile, you'll have to include a call to `StreamAttack.<event name>`. E.g. for "EDDI Jumped", call `StreamAttack.EDDI Jumped` by name from your `((EDDI Jumped))` command.

Commands¶

- `clear jump target`: clears the current jump target.
- `set jump target`: sets the jump target to the currently targeted system. Distance will be written to the configured file.
- `[copy;open] ship build`: copies the current ship build (coriolis) or opens it in your default browser.
- `open StreamAttack folder`: opens the configured folder in Explorer.

Files the Profile Provides¶

Elite¶

Commander¶

- `Elite\cmdr\name`: the current commander's name.

Jump Target¶

- `Elite\jumpTarget\distance`: distance to current jump target in light years.
- `Elite\jumpTarget\full`: pretty-printed `<distance> ly to <name>`.
- `Elite\jumpTarget\name`: the current jump target's system name.

Location¶

- `Elite\location\full`: depending on your status, either the station you are currently docked at (+ system), the body you are currently near, or the system you are currently in.
- `Elite\location\system`: the system you are currently in.

Ship¶

- `Elite\ship\build`: your current ship’s loadout (link to coriolis).
- `Elite\ship\full`: “`<name>`” `| <model> | <build>`.
- `Elite\ship\model`: your current ship’s model.
- `Elite\ship\name`: your current ship’s name.

Logging¶

The profile supports logging a bunch of stuff to the VoiceAttack event log. By default, logging is concise and constrained to basically error messages.

If you need more logging (usually for debugging purposes), say `enable logging`. If you want to enable verbose logging *by default*, call the `Logging.enableLogging` command from your custom profile’s `startup` command.

Exposed Variables¶

The following Variables are *global* and thus readable (and writeable! Please don’t unless it’s a config variable …) from other profiles:

Configuration Variables¶

There are a bunch of configuration variables. You should not overwrite those manually, instead use the provided commands in the `_configuration` section!

Basically all the settings are available using the `customize settings` prefix, then saying `[enable;disable] <setting>` for on/off switches and `set <setting>` for text variables.

- `EDDI.quietMode` (boolean): whether or not to set EDDI to quite mode. Default: true.
- `EDDI.useEddiForVoice` (boolean): whether to use EDDI over VA’s builtin `say` command. Default: false.
- `StreamAttack.outputDir` (string): the directory StreamAttack will save its information to. Default: `%appdata%\StreamAttack\`.
- `python.ScriptPath` (string): the path you have placed the compiled Python scripts in. Default: “{VA_DIR}\Sounds\scripts” (the “\Sounds\scripts” folder in your VoiceAttack installation directory).

Other Variables¶

These variables can be used to get information about the current neutron route. Please do not set them manually and / or from outside the StreamAttack profile.

- `StreamAttack.Elite.jumpTarget` (string): the current jump target.

Use

EliteAttack¶

This is my personal VoiceAttack profile for Elite: Dangerous. It started out ages ago as a modification of MalicVR’s public profile, then looked less and less and less like that and I added and cleaned up a lot of things while removing the stuff I didn’t use anyway. By now it would have probably been simpler to start from scratch.

Some of it has grown to a state that it might be useful to others in its own package, so I’ve separated the neutron jumping and Seals stuff into their own profiles.

The rest is a random conglomerate of all things VA and E:D; from various voice commands to lots of EDDI event handlers.

Speaking of EDDI; it has become in integral part of my Elite experience, especially the plethora of information it extracts from the game’s journal and presents to you via lots and lots of status variables and by firing various events that can then be handled through VA commands. It’s great. Check it out. (You might want to make it talk a lot less in it’s personality options, or disable the speech responder entirely like I have.)

Requirements¶

In addition to the bindED and EDDI VoiceAttack plugins, this profiles needs my Python elite-scripts to do everything properly. The release page here includes a compiled version for Windows that does not need Python installed. If you use the profile package from the release page, they will be installed automatically.

Settings¶

Because Elite’s keyboard handling is … weird you’ll have to set the key to use for pasting text into Elite:Dangerous. If you are not using a “standard” QWERT[YZ] layout, you will have to change it back to the key that is physically in the place where v would be on QWERTY.

For other settings, see the Configuration Variables section.

Including the Profile In Another Profile¶

This is meant to be a standalone profile, including the others in this repo (and a couple more). It was never designed to be included into your existing profile. Nevertheless, it *should* work properly if you follow some guide lines:

- Run the startup command. You will need to have a startup command in your profile (= one that is run on profile loading) and call `EliteDangerous.startup` by name from that one.
- Make sure all EDDI events that EliteDangerous needs are correctly handled. For all events used in EliteDangerous that you already have handlers for in your profile, you'll have to include a call to `EliteDangerous.<event name>`. E.g. for "EDDI Jumped", call `EliteDangerous.EDDI Jumped` by name from your `((EDDI Jumped))` command.

Usage¶

Chat Commands¶

There's a bunch of commands in here to send certain things to chat. Unless stated otherwise, they will only work with the comms panel active, and you should be in the edit window ready to send. They will *not* hit Enter on their own.

- `clear [chat;text]`: Clears the chat window. Use from outside the comms panel.
- `[local;squad;system;wing] chat`: Puts you into the chosen chat channel.
- `paste text`: Pastes your clipboard into Elite. Works outside the comms panel too, e.g. on the galaxy map inside the search field.
- `salute; oh seven`: Will put "o7" into the chat.

Engineering/Materials¶

- `how many [<g5 manufactured materials list>] do i have`: Tells you how many of the given g5 manufactured material you currently have on board. I've restricted it to just those to not spam speech recognition with too many phrases, and because those are the ones I usually want to know while jumping around the bubble and having an eye on any HGE that might be around.
- `open e d engineer`: Opens the ED Engineer tool.
- `what [mats;materials] do i need?`: Runs the EDDI responder that tells you which materials are below wanted threshold. Needs setting those first. Gets very spammy if you do it for all of them; personally I only set them for g5 manufactured, so I can quickly check if it's worth looking for HGE in a system I'm in.

Events¶

The main point of this profile is to react to Elite's journal events (powered by EDDI). Each of the events listed here will trigger a `((EDDI <event>))` command in VoiceAttack which in turn triggers `EliteDangerous.EDDI <event>` and all included profiles' similar commands, e.g. `SpanshAttack.EDDI <event>`.

The actual `((EDDI <event>))` command will only be executed once by VoiceAttack, the first one it can find. Keeping it separate from the actual code to be run makes it easier to handle (multiple) included profiles.

If the command for an event will send key presses to Elite, it focuses the client window first to make sure they get registered properly. If it is not in focus, the first key press might be swallowed otherwise.

For other commands, the profile just assumes that you are actively doing something in game and it is already focused :)

Body Mapped¶

Announces an estimate for high-value bodies' payouts and the remaining mapping candidates in the system as given by EDDI.

Body scanned¶

Announces any interesting body traits found when scanning:

- terraformable
- Earth-like World, Ammonia World or Water World
- landable and >5 g
- semimajor axis <1.5 ls (only really interesting for planets, but there's no proper way to separate them from moons, sadly)
- radius <300 km

Feel free to suggest more!

Carrier Cooldown¶

Tells you when your carrier is able to do its next jump. Technically only works if you were docked at it when it performed the jump, but I'm doing some behind-the-scenes magic if you weren't :)

Might be slightly off (too early) in the latter case.

Carrier Jumped¶

Announces system and body your carrier has just jumped to.

Carrier Jump Engaged¶

This event fires when your carrier jumps but you are *not* docked at it. It provides way less information than the `Carrier Jumped` event, but hey, I don’t use most of it anyway.

Basically just calls `Carrier Jumped` (and makes sure that a `Carrier cooldown` event is triggered manually at approximately the right time).

Carrier Jump Request¶

Announces the system and body your carrier has just been scheduled to jump to. Use this to double check ingame information; I’ve had my carrier accept a body as jump target, but then end up around the star. This *might* give you a heads up on that.

Carrier Pads Locked¶

Announces your carriers lockdown procedures. This *might* only work when you are docked (which would make it pretty useless). Feel free to open in issue if that’s the case.

Discovery Scan¶

Announces the number of bodies (and non-body signals) found in the system. Also compares the number of bodies to the amount reported by EDSM (requires Python scripts).

Docked¶

Automatically gets your ship into the hangar and opens station services.

Docking Denied¶

Tells you the reason for docking denial.

Entered Normal Space¶

Throttles to 0 upon dropping from SC, if `EliteDangerous.hyperSpaceDethrottle` is set.

Fighter Launched¶

Orders your ship to hold position so it doesn’t chase after you immediately.

Jet Cone Boost¶

Sets your ship to full throttle immediately after you have supercharged.

Jumped¶

- zeroes throttle
- gets the system's body count from EDSM (requires Python scripts)
- gets stations with outdated data (older than 1 year) from Spansh's API
- if you haven't visited the systems, starts a discovery scan (see the discovery scan command)
- last but not least tells you about planets worth scanning if you are on the R2R

Liftoff¶

Retracts landing gear for you. Seriously, is there any occasion in which you *don't* immediately want to retract it after takeoff?

Low Fuel¶

Warns you when you reach 25% fuel. Also reports number of jumps you have left or the (rough) range you still have on the fumes left in your tank.

Material Threshold¶

Warns you when a monitored material falls below it's minimum stock level. You will have to set a minimum desired amount in EDDI's material monitor options first for all materials you wish to be monitored.

Message Sent¶

Checks any message you send for a chat prefix and sends it to the proper chat window. Probably largely useless to you without modification.

- `.nc`: Actually doesn't send anything, but runs the `RatAttack.announceNearestCMDR` command with the system given in the rest of the message.
- `.dc`: Sends the message to the Discord window.
- `.tc`: Sends the message to my twitch channel window (IRC #alternerdtive).

There are similar event commands in RatAttack and SealAttack handling other chat windows.

Ship FSD¶

This event actually is several different events in one. Currently the following are handled:

- charging: Warns you if your target system's main star is not scoopable, including an extra warning at low fuel levels. (**Note**: This only works if the target system is in EDSM. So it's kind of useless for its intended use (exploration) and probably going to be removed at some point.)

- cooldown complete: Announces FSD cooldown if you are currently in normal space.

Ship interdicted¶

Tells you when you have been interdicted by a player. Is also supposed to target the interdictor automatically, but randomly sometimes just doesn't work. Yay!

Ship targeted¶

This currently doesn't do anything. I was fiddling around with automatically targeting a certain module on ship targeting, but it was more hassle than I had thought.

SRV Launched¶

Toggles SRV lights off after launching. Might not work if you drop particularly far after deployment because it works off a timer. Conversely might take a second to turn your lights off on a short drop and/or in high gravity.

System Scan Complete¶

Lists you all bodies EDDI considers worth mapping in the current system.

Undocked¶

Retracts landing gear for you. Seriously, is there any occasion in which you *don't* immediately want to retract it after takeoff?

VA initialized¶

Fires when the EDDI VoiceAttack plugin is loaded. Makes sure that EDDI is set to quite mode even if the profile was loaded before plugin initialization had completed.

Misc¶

The commands in here do random more or less useful things.

- `bind keys;reset key binds`: Reloads your key binds through the bindED plugin. You should do that after changing anything in the controls options.
- `copy current system`: Copies the current system name into the clipboard.
- `distance [from;to] […]`: Tells you the distance from your current position to the other thing you mentioned and is supported in the command. (requires Python scripts)
- `do a barrow roll`: WHOOOOOOO!

- `fix window dimensions`: When you start the game in VR, it forces into windowed mode with weird resolution. This changes it back. Hover the “PLAY” entry in the main menu, then run this. Will need adjustment for different graphics cards/drivers and the resolution you want.
- `neutron [jump;trip] time`: Shorter version of the same thing in SpanshAttack.
- `neutron jumps left`: Shorter version of the same thing in SpanshAttack.
- `open copied system on EDSM`: Opens the system in your clipboard on EDSM in your default browser.
- `open coriolis`: Opens Coriolis in your default browser.
- `open [current;] system on EDSM`: Opens your current system on EDSM in your default browser.
- `open EDDI options; configure EDDI`: Opens the EDDI configuration window.
- `open e d d b [station;system;faction;] [search;]`: Opens EDDB in your default browser.
- `open e d s m`: Opens EDSM in your default browser.
- `open inara`: Opens Inara in your default browser.
- `open materials finder`: Opens EDTutorials’ materials finder in your default browser.
- `open miner’s tool`: Opens https://edtools.ddns.net/miner in your default browser.
- `reload bindings`: Reloads your bindings for bindED.
- `shut up EDDI`: Immediately stops any ongoing (and queued) EDDI speech.
- `[start;stop] [EDISON;navigation]`: Hits `CTRL+ALT+E` which just so happens to be the start/stop hotkey I have set in E.D.I.S.O.N.
- `[bodies;what’s;what is] left to [map;be mapped;scan]`: Tells you which bodies EDDI thinks are worth mapping in the system that you haven’t mapped yet.

Navigation¶

There are so many navigation-focused commands now, they deserve there own category. Basically anything that helps you plot anywhere. A lot of those are powered by awesome EDDI so I don’t have to do the work myself!

- `plot course;[target;] next [waypoint;way point]`: Plots a course to the system set in `~~system` or the one in your clipboard. The former way is usually used by other commands to not interfere with your clipboard.
- `[find;target] nearest [encoded;manufactured;raw] material trader`: Targets the nearest respective material trader.
- `[find;target] nearest [guardian;human] tech broker`: Targets the nearest respective tech broker.
- `[find;target] nearest [interstellar factor;mission system;scoopable star]`: Well, you know the drill by now.
- `[find;target] nearest mission system`: Targets the nearest system that has a mission target.

- `[find;target] [<system>]`: Targets the given system on the galaxy map. There's a bunch in there, the list is easily extensible. Drop me a note if you want something included.

Ship Controls¶

Basically anything that is related to directly doing something with your ship.

- `[abort;cancel;stop] jumping`: Stops a currently charging FSD jump.
- `[buggy;exploration] power`: Sets your PD to 0/4/2 or 2/4/0 respectively. Works in SRV too.
- `[close;deploy;extend;open;retract;] […] [up;down;]`: Overly complicated command to handle everything related to Cargo Scoop, Hard Points, Landing Gear. You get the gist, I guess. Works in SRV too.
- `[disco;discovery scan]`: Executes a discovery scan. To work properly, you'll have to set the Discovery Scanner to your first fire group, secondary fire.
- `[dis;]engage silent running`: Turns silent running on and off.
- `[head;spot;] lights [on;off]`: Turns your lights on and off. Works in SRV too, kinda; turning lights off there relies on the state updating fast enough, which sometimes leads to weird results.
- `[jump;engage;get me out;punch it chewie] [and scan;] [when ready;]`: Retracts everything that might be protruding from your ship, then jumps to the next system. If the FSD isn't charging within 1s, it gets you into SC instead (e.g. if your target is obstructed). If given "and scan" runs a discovery scan. If given "when ready" waits for mass lock to clear, your FSD to cool down and you to leave scoop range before jumping.
- `night vision [on;off]`: Toggles your night vision on/off. Works in SRV too.
- `power to [engines;shields;systems;weapons`: Sets 4 pips to the thing you told it, 1 to the others.
- `rapid fire lights`: Flashes your lights 5 times in a row.
- `retract [all;everything]`: Retracts, well, everything.
- `[start;stop] [firing;mining]`: Starts/stops holding down primary fire. Mostly useful when mining. When triggered with "mining", also deploys the cargo scoop.
- `[super;] cruise [when ready;]`: Retracts everything, then jumps to SC. If given "when ready" will wait for mass lock to clear and your FSD to cool down first.

SRV controls¶

Things relevant to your SRV, but not your ship.

- `[recall;dismiss] ship`: Recalls or dismisses ship. Currently does the same thing regardless of the state of your ship. I wish it would be possible to restrict it to doing one thing each, but that's currently not possible sadly.
- `[toggle;enable;disable] drive assist`: Handles all your drive assist needs!

Targeting¶

Well … targeting stuff, I guess. Not really sure why I made that it's own category, but oh well :)

- `target next system`: Selects the next system on your route.
- `target wing man [1;2;3]`: Targets your wing men.
- `target's target`: Targets your target's target.
- `wing man [1;2;3] target`: Targets your wing men's target.
- `wing man nav lock`: Toggles wing man nav lock on the selected wing member.

UI Commands¶

Everything handling stuff that's not related to controlling your ship, but manipulating some UI element(s).

- `controls options`: Opens the controls options menu.
- `docking request;request dock[ing;]`: Sends a docking request.
- `[enter;leave] F S S`: Opens/closes FSS.
- `galaxy map`: Opens the galaxy map.
- `[main;game] menu`: Opens the ESC menu.
- `[relog;reset] to [open;solo]`: Relogs to Open or Solo mode, respectively.
- `restart from Desktop`: Quits the game and restarts from an open launcher by clicking the play button.
- `set […] filter`: Sets a nav panel filter setting. See the command or just try different things for what is possible. You need to clear filters and hover over the filter button, then run this.
- `system map`: Opens the system map.
- `take [high res;] screenshot`: Takes a (high res) screenshot.
- `toggle orbit lines`: Toggles the visibility of orbit lines.
- `[toggle;show;hide] interface`: Toggles the cockpit interface (CTRL+ALT+G). Probably needs to be adjusted if you are not playing with Neo2 keyboard layout :)

Update Commands¶

- `check for profiles update`: Does just that. Is also automatically run each time the profile is started.
- `download profiles update`: Downloads a profiles update if applicable. Will prompt you to restart VoiceAttack when the download has finished to import the updated profiles.
- `open profiles change log`: Opens `CHANGELOG.md` on Github.

Logging¶

The profile supports logging a bunch of stuff to the VoiceAttack event log. By default, logging is concise and constrained to basically error messages.

If you need more logging (usually for debugging purposes), say `enable logging`. If you want to enable verbose logging *by default*, call the `Logging.enableLogging` command from your custom profile's `startup` command.

Configuration Variables¶

There are a bunch of configuration variables. You should not overwrite those manually, instead use the provided commands in the `_configuration` section!

Basically all the settings are available using the `customize settings` prefix, then saying `[enable;disable] <setting>` for on/off switches and `set <setting>` for text variables.

- `EDDI.quietMode` (boolean): whether or not to set EDDI to quite mode. Default: true.
- `Elite.pasteKey` (string): the key used for pasting into Elite. On QWERTY this is `v`. Default: `p`.
- `EliteDangerous.announceEdsmSystemStatus` (boolean): whether to announce the system or some of its bodies missing on EDSM. Default: true.
- `EliteDangerous.announceMappingCandidates` (boolean): whether to announce mapping candidates when they are scanned. Default: true.
- `EliteDangerous.announceOutdatedStationData` (boolean): whether to announce stations with outdated data in systems you jump to. Default: true.
- `EliteDangerous.announceR2RMappingCandidates` (boolean): whether to announce planets worth mapping when jumping into a known system. This is useful for doing some R2R on the side. Default: false.
- `EliteDangerous.enableCarrierAnnouncements` (boolean): whether or not to process fleet carrier events. Default: true.
- `EliteDangerous.enableAutoUpdateCheck` (boolean): whether or not to automatically check for updates on profile startup. Default: true.
- `EliteDangerous.flightAssistOff` (boolean): whether to automatically toggle FlightAssist off on liftoff. Default: true.
- `EliteDangerous.hyperspaceDethrottle` (boolean): same thing as the SC assist setting; if on, will throttle to 0 automatically after jumping. Default: true.
- `EliteDangerous.oldStationThreshold` (int): Age in days that will cause station data to be considered outdated. Default: 365 (1 year).
- `python.ScriptPath` (string): the path you have placed the compiled python scripts in. Default: "{VA_DIR}\Sounds\scripts" (the "\Sounds\scripts" folder in your VoiceAttack installation directory).

Delays / Pauses¶

Delays needed e.g. between key presses in UI navigation can vary wildly depending on your PC's specs and configuration. Therefore they should be configurable, shouldn't they?

So far those actually are:

- `EliteDangerous.delays.quitToDesktop`: Delay between quitting to desktop and hitting the play button in the launcher.

RatAttack¶

This profile facilitates Fuel Ratting. It aims to eliminate as much of the required manual task and attention switching as possible via automation and voice commands.

If you don't know what the Fuel Rats are, come hang out and ask :)

Requirements¶

Only vanilla VoiceAttack is absolutely required to use this profile. Optionally you can install EDDI and my elite scripts for advanced features.

- EDDI installed as a VoiceAttack plugin: This will give you a better (IMO) way of using TTS. Be sure to set `RatAttack.useEddiForVoice`. It will also enable you to have ingame chat be transferred to IRC; see below.
- elite-scripts: Using the Python scripts will give RatAttack a way to be aware of where your CMDRs are and give you the nearest one to a rat case. That's only really needed if you actually *have* multiple CMDRs, obviously. If you are using the profile package from the release page, they will be installed automatically.

EDDI speech responder¶

For the convenience of people that have not been using EDDI in the past, RatAttack will deactivate the speech responder automatically to not clutter them with unwanted TTS.

If you are already an EDDI user and want to keep the default speech responder functionality, you will have to disable the `EDDI.quietMode` setting by running the `customize settings disable eddi quiet mode` command.

Settings¶

There are a lot of preferences you can set, including some you really want to concern yourself with before you start using the profile. Some of the more advanced features heavily rely on you giving it the correct things to work with.

See the Configuration Variables section.

Including the Profile¶

When including the profile, be sure to

- Run the startup command. You will need to have a startup command in your profile (= one that is run on profile loading) and call `RatAttack.startup` from that one.
- Make sure all EDDI events that RatAttack needs are correctly handled. For all events used in RatAttack that you already have handlers for in your profile, you'll have to include a call to `RatAttack.<event name>`. E.g. for "EDDI Message sent", call `RatAttack.EDDI Message sent` by name from your `((EDDI Message sent))` command.

Usage¶

Going On/Off Duty¶

When you are on duty, RatAttack will automatically announce cases coming in through IRC. When off duty, it won't.

- `[enable;disable] rat duty`: puts you on/off duty.
- `open [rat;] dispatch board`: opens the web dispatch board.

Handling a Case¶

Getting Case Data From IRC¶

You can setup your IRC client to pass incoming RATSIGNALS to VoiceAttack by writing them to a file (`%appdata%\Ratattack\ratsignal.pipe`), then calling the appropriate command (`RatAttack.announceCaseFromRatsignal` for notification, `RatAttack.getInfoFromRatsignal` for silently putting it into the case list).

This has two purposes:

1. announcing a new incoming case
2. storing case data and making it available to VoiceAttack, e.g. for copying the case's system into the clipboard

You need to make your IRC client

1. wait until the file disappears (for several cases coming in at once)
2. write the RATSIGNAL to the file
3. run the VoiceAttack command

In my case I am running AdiIRC and have the following script setup for handling this:

```
on *:TEXT:RATSIGNAL - CMDR*(??_SIGNAL):#fuelrats:{
    /mkdir C:\users\<user>\appdata\roaming\RatAttack\
    /handleratsignal $1-
```

```
}
alias handleratsignal {
    if ( $exists(C:\users\<user>\appdata\roaming\RatAttack\ratsignal.pi
        /sleep 1 /handleratsignal $1-
    }
    else {
        /write C:\users\<user>\appdata\roaming\RatAttack\ratsignal.pipe
        if ( $away ) {
            /run -h "X:\path\to\VoiceAttack\VoiceAttack.exe" -nofocus -
        }
        else {
            /run -h "X:\path\to\VoiceAttack\VoiceAttack.exe" -nofocus -
        }
    }
}
```

You get the gist; if not and you don't know how to do the same thing for your IRC client or it doesn't support copying the control characters in the ratsignal that the profile uses to split the information, either switch to AdiIRC or bribe me to include some other way to get case data into VoiceAttack.

Note: If you are running VoiceAttack as admin you need to run your IRC client as admin, too! Otherwise it can't talk to VoiceAttack for security reasons. You really should *not* run VoiceAttack with elevated privileges though. Or anything.

Internal Case List¶

If you have your IRC client setup properly, VoiceAttack will hold a list with all rat cases that have come in while you had it running. It will save the case number, CMDR name, system, O_2 status and platform. There are several commands you can run on this list, giving it a case number:

- `rat case number [0..19] details`: Will give you all stored info on a case.
- `[current;] rat case details`: Will give you all stored info on the currently open case.
- `distance to current rat case`: Will give you the distance from your current location to the currently opened rat case. Requires the use of my `elite-scripts` Python scripts.
- `distance to rat case number [0..19]`: Will give you the distance from your current system to a case's system. Requires the use of my `elite-scripts` Python scripts.
- `nearest commander to rat case number [0..19]`: Will give you the nearest of your CMDRs with their distance to a case's system. Requires some setup and the use of my `elite-scripts` Python scripts.
- `nearest commander to [the;] rat case`: Will give you the nearest of your CMDRs with their distance to the current case's system. Requires some setup and the use of my `elite-scripts` Python scripts.

Opening a Case¶

- `open rat case number [0..19]`: Opens rat case with the given number. If there is no case data for that case (e.g. because you don't have your IRC client set up properly), it will still open it, just not have any data on it.
- `open [latest;] rat case`: Opens the latest rat case that has come in through IRC. Will probably error out in creative ways if you don't have your IRC client set up properly. Too tired right now to have proper error handling so just open an issue if you run into problems (it's 7am, I haven't slept and want to finish this doc to get the release out (yes, you are allowed to laugh at this section)).

Closing a Case¶

- `[close;clear] rat case`: Closes the currently open rat case.

Making Calls¶

There are a bunch of calls you can make for a case, the most common are modelled through VoiceAttack commands. The descriptive commands (e.g. "system confirmed") will be shortened to the usual IRC short hands (e.g. "sysconf"). If you need something more unusual you can either still manually type it into your IRC client or use the "General IRC Integration", see below.

- `call [1..20] jumps [and login;and takeoff;left;]`: Calls jump for the currently open case. You can optionally include that you will still have to login to the game or have to take off from your current station/ port/outpost/planet.
- `call friend [positive;negative] [in pg;in private group;in solo;in main meu;sysconf;system confirmed]`: Friend request confirmations, with all the things you might want to / should call with it.
- `call [beacon;fuel;instance;pos;position;prep;sys;system;wing] [positive;negative]`: All the stuff you usually need for ratting after you have received the friend request.
- `call wing pending`: Calls "wr pending" for when it takes 30s again to actually show up.
- `call client in [exclusion zone;main menu;open;open sysconf;pg;private group;solo;super cruise]`: Callouts for all the various things a client could get themselves into.
- `call [client destroyed;client offline;sysconf;system confirmed]`: This is the command you don't want to use. Include sysconf in your "friend+" or "in open" calls, and make sure you will never have to call "client destroyed", would you?

General IRC Interaction¶

(requires EDDI)

Using EDDI to read the game's journal, you can send messages to IRC from Elite's ingame chat.

Be aware that the chat message will still appear in the ingame chat channel you send it to!

I recommend using local chat and limiting the use to instances that will probably not have other players in it (e.g. instanced in normal space with the client or in SC in some remote system out in the black on a long range rescue).

- **fuelrats: Use ".fr \<message>" to have VoiceAttack send "#\<caseNumber>¶**

  \<message>" to the #fuelrats channel – or yell at you when you are not on a case.

- **ratchat: Use ".rc \<message>" to have VoiceAttack send "\<message>" to¶**

  #ratchat.

These commands send their text to windows with "#fuelrats" and "#ratchat" in their title, respectively. If your IRC client does not do that, you will have to change the "target" window of the `RatAttack.sendToFuelrats` and `RatAttack.sendToRatchat` commands to reflect the actual window titles on your system. I will look into making this more elegant to change in the future.

Logging¶

The profile supports logging a bunch of stuff to the VoiceAttack event log. By default, logging is concise and constrained to basically error messages.

If you need more logging (usually for debugging purposes), say `enable logging`. If you want to enable verbose logging *by default*, call the `Logging.enableLogging` command from your custom profile's `startup` command.

Exposed Variables¶

The following Variables are *global* and thus readable (and writeable! Please don't unless it's a config variable …) from other profiles.

Configuration Variables¶

There are a bunch of configuration variables. You should not overwrite those manually, instead use the provided commands in the `_configuration` section!

Basically all the settings are available using the `customize settings` prefix, then saying `[enable;disable] <setting>` for on/off switches and `set <setting>` for text variables.

- `EDDI.quietMode` (boolean): whether or not to set EDDI to quite mode. Default: true.
- `EDDI.useEddiForVoice` (boolean): whether to use the EDDI plugin to handle text-to-speech over VoiceAttacks built-in speech function. Default: false.
- `Elite.pasteKey` (string): the key used for pasting into Elite. On QWERTY this is `v`. Default: `v`.
- `RatAttack.announceNearestCMDR` (boolean): whether or not to automatically announce your nearest CMDR to a case. Requires the `elite-scripts` Python scripts. Will probably break in creative ways if you don’t have them and turn it on anyway. Default: false.
- `RatAttack.announcePlatform` (boolean): whether or not to announce the case’s platform by default. Useful to set if you are active on more than one platform. Even with this off, you will still be warned when you open a case that is *not* on one of your platforms. Default: false.
- `RatAttack.CMDRs` (string): list of your CMDR names, delimited by spaces. If your names include spaces, you will have to put them in quotes. Default: “"J Jora Jameson" NameWithNoSpace”.
- `RatAttack.confirmCalls` (boolean): whether VoiceAttack should ask you before posting to #fuelrats to make sure there hasn’t been an error in voice recognition and you accidentally post the wrong thing. Default: true.
- `RatAttack.autoCloseCase` (boolean): whether or not to automatically close an open rat case on calling “fuel+”. Default: false.
- `RatAttack.onDuty` (boolean): whether or not you are currently on rat duty. Default: true.
- `RatAttack.platforms` (string): the platforms you want to be informed of incoming cases for. If you are on console, you can still have VoiceAttack running on the PC that you are using for IRC and handle calls and stuff using voice! Delimited by whatever you want. Can include “PC”, “XB”, “PS4”. Default: “PC”.
- `python.scriptPath` (string): the path you put the Python scripts in. Default: “{VA_DIR}\Sounds\scripts”.

Other Variables¶

Current case data:

- `RatAttack.caseNumber` (int): the number of the case you are currently on. Will be `Not Set` if you are not on a case.
- `RatAttack.onCase` (boolean): whether or not you are currently on a case.

Case list:

- `RatAttack.caseList.<case#>.cmdr` (string)
- `RatAttack.caseList.<case#>.system` (string)
- `RatAttack.caseList.<case#>.platform` (string)
- `RatAttack.caseList.<case#>.codeRed` (boolean)

… with `<case#>` being a number between 0 and 19.

SpanshAttack¶

This profile uses the ED-NeutronRouter plugin to plot neutron jumps using spansh. It fully does everything you need from within the game and VoiceAttack, you won't have to visit the site at any point.

Requirements¶

In addition to VoiceAttack, you will need the following plugins to use this profile:

- bindED
- EDDI installed as a VoiceAttack plugin
- ED-NeutronRouter: required for SpanshAttack. **Make sure to grab the pre-release 1.02** since 1.01 has a bug with a hardcoded 50 ly jump range.

EDDI speech responder¶

For the convenience of people that have not been using EDDI in the past, SpanshAttack will deactivate the speech responder automatically to not clutter them with unwanted TTS.

If you are already an EDDI user and want to keep the default speech responder functionality, you will have to disable the `EDDI.quietMode` setting by running the `customize settings disable eddi quiet mode` command.

Settings¶

Because Elite's keyboard handling is … weird you'll have to set the key to use for pasting text into Elite:Dangerous. If you are not using a "standard" QWERT[YZ] layout, you will have to change it back to the key that is physically in the place where `v` would be on QWERTY.

For other settings, see the Configuration Variables section.

The last "setting" in the not-so-strict sense of the word is the `SpanshAttack.getShipRange` command. Any ship listed in there will automatically have its jump range used instead of VA prompting you for it.

Since, again, VA will execute the first matching command found, you can create this command in your own profile when you are using SpanshAttack by including it. You can override a saved range for your ship by using the `plot neutron [course;route;trip] with custom range` command.

The ED-NeutronRouter plugin is technically supposed to read the current jump range from EDDI; sadly a) it's bugged right now, and b) EDDI is storing the *maximum* distance for your ship instead of the current / full on fuel one.

Including the Profile¶

When including the profile, be sure to

- Run the startup command. You will need to have a startup command in your profile (= one that is run on profile loading) and call `SpanshAttack.startup` from that one.
- Make sure all EDDI events that SpanshAttack needs are correctly handled. For all events used in SpanshAttack that you already have handlers for in your profile, you'll have to include a call to `SpanshAttack.<event name>`. E.g. for "EDDI Jumped", call `SpanshAttack.EDDI Jumped` by name from your `((EDDI Jumped))` command.
- (Optional) Have a `SpanshAttack.getShipRange` command in your profile to overwrite the default one with your ship's ranges. See the default command for pointers.

Usage¶

Plotting a Route¶

1. *Target* the system you want to be routed to (target, do not plot to it).
2. Either exit the galaxy map or make sure you are on its first tab (or auto-plotting will break).
3. Trigger the `SpanshAttack.plotRoute` command either by voice (`plot neutron [course;route;trip] [with custom range;]`) or calling it from another command.
4. Enter your ship's jump range if prompted.
5. Wait for the route to be calculated. The command will automatically open the galaxy map and jump to the first waypoint on your route. If you run into weird behaviour, it's probably because your target system is not in EDDB.
6. Either target the first waypoint or plot to it.
7. Start jumping!

Plotting to a System Unknown to the Neutron Router¶

The router can only plot a route to a system that is in its data base (obviously can also only give you way points that are). If your target system is not, there are several levels of fallback handling to find a system that is.

1. Check `Next system` coordinates provided by EDDI. If the system is in EDSM, but has for some reason not been sent over EDDN to other sites including Spansh we can get coordinates here.
2. If the system is not in EDSM check EDTS. It can calculate approximate coordinates for a given procedurally generated system name.
3. If that fails prompt the user for input.
4. Query Spansh' API for the closest system to these coordinates.
5. Plot a route to the closest system.

Generally you should almost never be asked to input coordinates manually. If EDTS provides coordinates with an accuracy that is worse than ±100 ly per axis, you will be prompted to make sure you are going roughly to the right coordinates. You will find the system that is used for plotting, its coordinates and the accuracy in VoiceAttack's log window.

Neutron Jumping¶

With standard settings, just supercharge off a neutron cone. You should automatically be taken to the galaxy map with the next waypoint selected.

In case you have disabled auto-plotting to the next waypoint, manually invoke the `SpanshAttack.targetNextNeutronWaypoint` command by voice (`[target;] next neutron [waypoint; way point]`) or calling it from another command.

Additionally, you can use the `SpanshAttack.copyNextNeutronWaypoint` / `[get;copy] next neutron [waypoint;way point]` command to copy the next neutron waypoint to the clipboard.

Manual Re-Plot¶

Trigger the `SpanshAttack.replotRoute` command either by voice (`replot neutron [course;route;trip]`) or calling it from another command. This will start a re-plot of the current route with the same target system and jump range.

Refueling¶

Whenever you refuel off a scoopable star, the profile will automatically throttle back up to 100% speed. Unless you have disabled it in your configuration, you will also automatically target the next system on your route and jump to it once you leave fuel scoop range.

Clearing a Route¶

When you reach your target system, the neutron route will automatically be cleared. If you want to prematurely end your trip, call the `SpanshAttack.clearRoute` / `clear neutron [course;route;trip]` command.

Other Commands¶

Announcing Jumps Left¶

You can have VoiceAttack tell you the amount of jumps left on the current route by invoking `SpanshAttack.announceJumpsLeft` or saying `how many [neutron;] jumps [are;] left?`.

Note: Because it's pretty much impossible to calculate a 100% accurate value for the total jumps left, it will just tell you the jump count *from the current neutron waypoint*.

Announce elapsed time on the trip¶

SpanshAttack keeps track of your start time, even if you have the option to time your trip turned off. This way you can get the time you've been jumping with the `SpanshAttack.announceTripTime` or `how long have i been [jumping;on this trip;on this neutron trip]?` commands.

Reload bindings¶

If you change any relevant bindings (e.g. the galaxy map key), you should run the `reload bindings` command to make sure that SpanshAttack presses the right thing for you.

Eh, just do it every time you edit your controls without re-starting VoiceAttack, just to be sure.

Helper Functions¶

The profile contains a lot of helper functions that get called by the aforementioned commands. Have a look around, maybe learn something about VoiceAttack :)

Logging¶

The profile supports logging a bunch of stuff to the VoiceAttack event log. By default, logging is concise and constrained to basically error messages.

If you need more logging (usually for debugging purposes), say `enable logging`. If you want to enable verbose logging *by default*, call the `Logging.enableLogging` command from your custom profile's `startup` command.

Exposed Variables¶

The following Variables are *global* and thus readable (and writeable! Please don’t unless it’s a config variable …) from other profiles:

Configuration Variables¶

There are a bunch of configuration variables. You should not overwrite those manually, instead use the provided commands in the `_configuration` section!

Basically all the settings are available using the `customize settings` prefix, then saying `[enable;disable] <setting>` for on/off switches and `set <setting>` for text variables.

- `EDDI.quietMode` (boolean): whether or not to set EDDI to quite mode. Default: true.
- `EDDI.useEddiForVoice` (boolean): whether to use EDDI over VA’s builtin `say` command. Default: false.
- `Elite.pasteKey` (string): the key used for pasting into Elite. On QWERTY this is `v`. Default: `v`.
- `SpanshAttack.timeTrip` (boolean): whether to automatically tell you at the end of a trip how long it took you to get there. Default: false.
- `SpanshAttack.announceWaypoints` (boolean): whether to announce each waypoint of the neutron route. Default: true.
- `SpanshAttack.announceJumpsLeft` (string): `;`-separated list of remaining jumps to announce when said amounts are reached. Right now only works if they are *exactly* reached when supercharging off a neutron. Note the extra `;` at the beginning and end of the string. Default: `;1;3;5;10;15;20;30;50;75;100;`
- `SpanshAttack.autoJumpAfterScooping` (boolean): whether to automatically jump after fuel scooping (and moving out of scoop range). Default: true.
- `SpanshAttack.autoPlot` (boolean): whether to automatically plot to the next waypoint on supercharging. Default: true.
- `SpanshAttack.clearOnShutdown` (boolean): whether or not to automatically clear an active neutron route on Elite client shutdown. Default: true.
- `SpanshAttack.defaultToLadenRange` (boolean): whether or not to default to your ship’s laden range (as reported by EDDI) instead of asking for user input. Sadly it’s with *current* fuel, not full. Setting a ship’s jump range in the `SpanshAttack.getShipRange` command will still overrule this. Default: false.
- `SpanshAttack.copyWaypointToClipboard` (boolean): whether to copy the next waypoint into the Windows clipboard for use in other programs. Default: false.
- `python.scriptPath` (string): the path you put the Python scripts in. Default: “{VA_DIR}\Sounds\scripts”.

Other Variables¶

These variables can be used to get information about the current neutron route. Please do not set them manually and / or from outside the SpanshAttack profile.

- `SpanshAttack.plotSystem` (string): the system actually plotted towards using the neutron router (onley used/set if the target system is not in the data base)
- `SpanshAttack.targetSystem` (string): the target system for the current neutron route
- `SpanshAttack.nextNeutronWaypoint` (string): the next waypoint on the current neutron route
- `SpanshAttack.neutronJumpMode` (boolean): neutron jump mode active/inactive
- `SpanshAttack.jumpRange` (decimal): the current ship's jump range

StreamAttack¶

This profile uses the EDDI plugin to write a bunch of information about your commander, your current location and your ship to files that can be accessed e.g. by your streaming software to be displayed on stream.

Default folder is `%appdata%\StreamAttack\`.

Requirements¶

In addition to VoiceAttack, you will need the following plugins to use this profile:

- EDDI installed as a VoiceAttack plugin

EDDI speech responder¶

For the convenience of people that have not been using EDDI in the past, StreamAttack will deactivate the speech responder automatically to not clutter them with unwanted TTS.

If you are already an EDDI user and want to keep the default speech responder functionality, you will have to disable the `EDDI.quietMode` setting by running the `customize settings disable eddi quiet mode` command.

Settings¶

See the Configuration Variables section.

Including the Profile¶

When including the profile, be sure to

- Run the startup command. You will need to have a startup command in your profile (= one that is run on profile loading) and call `StreamAttack.startup` from that one.
- Make sure all EDDI events that StreamAttack needs are correctly handled. For all events used in StreamAttack that you already have handlers for in your profile, you'll have to include a call to `StreamAttack.<event name>`. E.g. for "EDDI Jumped", call `StreamAttack.EDDI Jumped` by name from your `((EDDI Jumped))` command.

Commands¶

- `clear jump target`: clears the current jump target.
- `set jump target`: sets the jump target to the currently targeted system. Distance will be written to the configured file.
- `[copy;open] ship build`: copies the current ship build (coriolis) or opens it in your default browser.
- `open StreamAttack folder`: opens the configured folder in Explorer.

Files the Profile Provides¶

Elite¶

Commander¶

- `Elite\cmdr\name`: the current commander's name.

Jump Target¶

- `Elite\jumpTarget\distance`: distance to current jump target in light years.
- `Elite\jumpTarget\full`: pretty-printed `<distance> ly to <name>`.
- `Elite\jumpTarget\name`: the current jump target's system name.

Location¶

- `Elite\location\full`: depending on your status, either the station you are currently docked at (+ system), the body you are currently near, or the system you are currently in.
- `Elite\location\system`: the system you are currently in.

Ship¶

- `Elite\ship\build`: your current ship's loadout (link to coriolis).
- `Elite\ship\full`: "`<name>" | <model> | <build>`.
- `Elite\ship\model`: your current ship's model.
- `Elite\ship\name`: your current ship's name.

Logging¶

The profile supports logging a bunch of stuff to the VoiceAttack event log. By default, logging is concise and constrained to basically error messages.

If you need more logging (usually for debugging purposes), say `enable logging`. If you want to enable verbose logging *by default*, call the `Logging.enableLogging` command from your custom profile's `startup` command.

Exposed Variables¶

The following Variables are *global* and thus readable (and writeable! Please don't unless it's a config variable …) from other profiles:

Configuration Variables¶

There are a bunch of configuration variables. You should not overwrite those manually, instead use the provided commands in the `_configuration` section!

Basically all the settings are available using the `customize settings` prefix, then saying `[enable;disable] <setting>` for on/off switches and `set <setting>` for text variables.

- `EDDI.quietMode` (boolean): whether or not to set EDDI to quite mode. Default: true.
- `EDDI.useEddiForVoice` (boolean): whether to use EDDI over VA's builtin `say` command. Default: false.
- `StreamAttack.outputDir` (string): the directory StreamAttack will save its information to. Default: `%appdata%\StreamAttack\`.
- `python.ScriptPath` (string): the path you have placed the compiled Python scripts in. Default: "{VA_DIR}\Sounds\scripts" (the "\Sounds\scripts" folder in your VoiceAttack installation directory).

Other Variables¶

These variables can be used to get information about the current neutron route. Please do not set them manually and / or from outside the StreamAttack profile.

- `StreamAttack.Elite.jumpTarget` (string): the current jump target.

Issues

Troubleshooting¶

Watch in Action