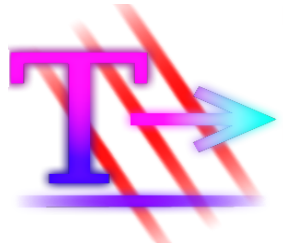


Guide to Chat Macros



Are you tired of writing long RP sentences repeatedly? Want something to automate all of that? Redux V6's new feature has got you covered!

First off, what is a chat macro? It is a predefined sentence that is triggered with a customizable command. Redux's macro feature is advanced enough to take arguments, which means you can specify certain text to be inserted somewhere to complete the sentence.

How can this be used? Imagine a certain sentence:

(somebody) is a big meanie!

Using the argument system, the above saying can be set to accept one value when its invoked, being able to replace (somebody) with anything

John is a big meanie!

Alex is a big meanie!

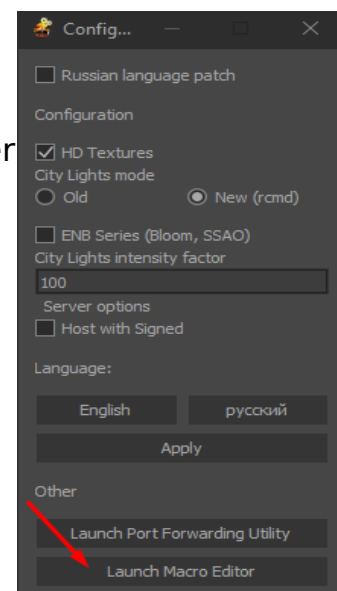
United States of America is a big meanie!

All of these are valid and can be easily done; relevant information on this, including the last one will be found later on this document.

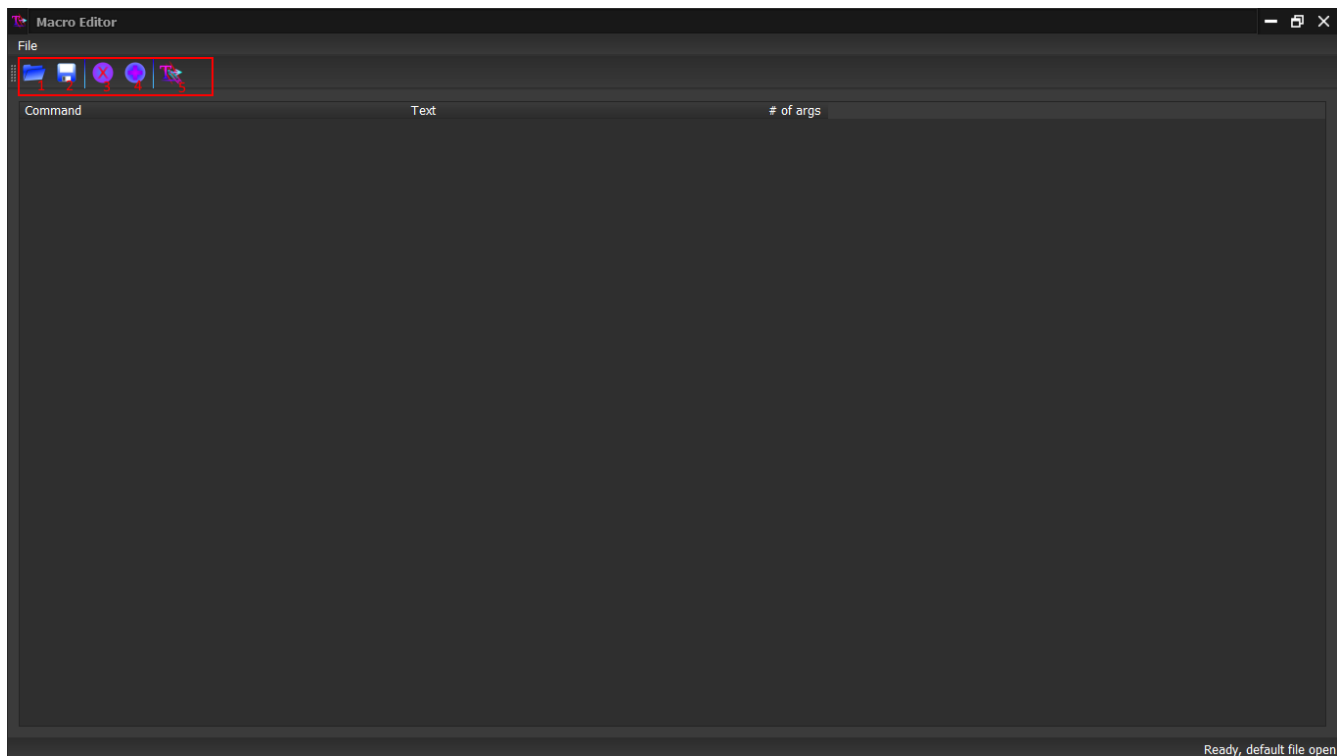
The Editor

The Macro Editor is where all editing of macros will be done.

It can be accessed by opening the Redux Config (the installer should've made a desktop shortcut) and pressing "Launch Macro Editor"



Once you launch the editor, you will see this:



1: Opens a separate file. This is not needed since the editor is set to always open the default file (the one that your game will use) when starting up. Note the status bar on the bottom right

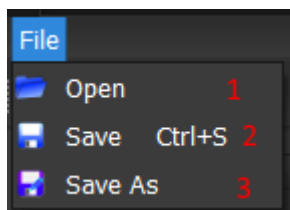
2: Saves the file

3: Removes the selected entry

4: Adds a new entry

5: Edits the currently selected entry. This can also be done by double clicking on it.

The File menu:



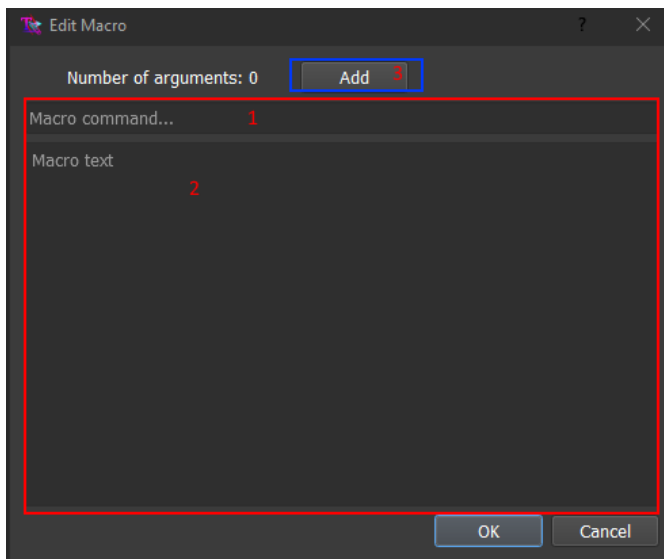
1: Same as **1** before

2: Same as **2** before

3: Saves the file to a new one in a different directory, specified by a dialog. This is normally not necessary unless you wish to exchange macro lists with other users.

Chat macros are stored in the Macro List format (.mcl). The default file is called stdmacro.mcl in the SuperPower 2 directory, where joshua.exe is located.

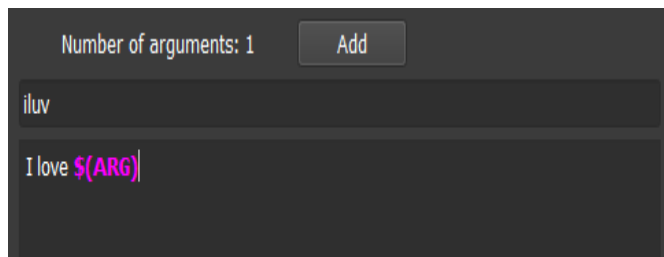
When you click Add or edit a macro, the following dialog will pop up:



1: The macro command name. It is the keyword you will use to invoke it during gameplay; select something short and simple. Cannot have spaces nor be a question mark.

2: The text this macro will say when invoked.

3: Add an argument. Arguments are marked by the specific keyword **\$(ARG)** which will be colored by the syntax highlighter. Additionally, you can just type the keyword out yourself.



A macro that contains arguments will appear incomplete in the list; do not panic, this is normal operation.

| | Command | Text | # of args |
|---|---------|--------|-----------|
| 1 | iluv | I love | 1 |

Note that if a macro takes arguments, those will be obligatory.

When you're done, don't forget to save before quitting by either clicking the icon or pressing CTRL + S

(continue in next page)

In-Game

Once you're playing SuperPower 2 in multiplayer, the macros can be used by entering their keyword followed by two forward slashes '//'. For example, if you had a command named *ilikecats*, you would type it out like this:

//ilikecats

If a macro takes arguments, you must pass them in the same message as the one in which you're issuing the commands, separated by spaces. If you've got a command named *ilike* which takes 2 arguments and you wanted these to be cats and pizza, you'd do it like this

//ilike cats pizza

Arguments cannot have spaces, even if they're escaped by double quotes.

If you type in the three-letter code in all caps of a valid country in the current server you're playing in, the parser will replace it with that country's full name. For example,

//ilike DEU SPA

DEU would become Germany, or whatever it is called at the time and SPA would become Spain when the message is issued. So, assuming the command template was *I like \$(ARG) and \$(ARG)* the final message would look like this:

I like Germany and Spain

The game reads the macro list every time it is launched, so if you make changes with the editor while the SuperPower 2 is running, you have to restart in order for it to recognize those changes.

Remember how it was stated previously that the question mark was reserved? Well, issuing the '?' command (//?) will show you a popup containing all the currently available macros and how many arguments they take.