Ərk IRC Client, Version 0.860

Scripting & Commands

A guide for using commands and writing scripts for the Ərk IRC Client

https://github.com/nutjob-laboratories/erk

Summary

Ərk functions much like other popular graphical IRC clients, such as mIRC (for Windows), Textual (for OSX), or HexChat (for Linux). If you've used any of these, much of the functionality of Ərk will feel familiar.

In this document, the area where chat text or server information is displayed to the user, and allows users to enter chat or commands, is referred to as a "window" (for more details, see **The Ərk Interface** on page 7). Ərk features three kinds of "windows": server consoles, channels, and private chat. The most common kind of window is the channel window; it features a chat display, a graphical user list, and a text input. This is the window where channel chat (called "public chat") will occur. Text entered into the text input of a channel window will be sent to the channel as chat, unless it contains a command (see **IRC Commands** on page 9 and **Ərk Commands** on page 12). Private chat windows work the same way, and look almost the same; the only thing missing is the graphical user list. Text entered into the text input of a private chat window will be sent to the user being chatted with, unless it contains a command. Server console windows look like private chat windows, but are not used for chatting with the server. Consoles will display messages from the server; text entered into the text input of a server console will *not* be sent as chat to the server, and any text not containing a command will be ignored. Commands entered into the text input of *any* window will either be processed locally or sent to the server associated with that window.

Ərk has two tools built into it for customizing the client:

- **Stylər**. A tool for editing a window's "style"; that is, the way text is colored, formatted, and displayed. It can be accessed from the "Settings & Tools" menu.
- **Scriptər**. A script editor for Ərk, also accessible from the "Settings & Tools" menu. For more information, see page 8 and the next section.

Scripting

To execute multiple commands, one after the other, without having to enter them manually, Ərk also features a scripting engine. Multiple commands (one per line) can be entered into a text file, and Ərk will execute the commands in that file. Scripts can either be executed directly by using the **`/script`** command (see page 16), or can be executed automatically on connection to a specific server.

If a script contains a line without a command, and only contains text, the text will be sent as a message to the window that was used to call the script; if it's a channel window, it will be sent as a public message to the channel, and it's a private chat window, it will be sent as a chat message to the user you're chatting with. If the script is called from a server console window (or executed with the editor or by clicking the "Run Script" button) the text will *not*

be sent as a message. Macros that "resolve" to plain text (that is, if the macro does not execute a command) will also be sent as a message if called from a channel or private chat window, and will *not* be sent if called from a server console window.

Ərk will always execute the commands on the server associated with the window that called the script (and connection scripts will always execute on the server being connected to); to execute scripts on any server the client is connected to, use the script editor, named **Scriptər**, accessible with the **`/edit`** command (see page 13), or via the "Settings & Tools" menu.

Ərk scripts have the "**`.erk`**" file extension. When using the **`/script`** command, the file extension does *not* have to be included; when looking for the file, Ərk will append the file extension if necessary. Scripts saved with **Scriptər** will *always* have the "**`.erk`**" file extension.

By default, Ərk will look for scripts in the 'scripts' subdirectory in the '.erk' directory (in your home directory). If Ərk is launched with the **`--scripts`** command-line flag (with a different directory name as an argument), Ərk will use this directory instead of the default. This will also change the default save directory for **Scriptər**, as well as where it looks for installed scripts.

Connection Scripts and "Callable" Scripts

Ərk uses two different kinds of scripts: connection scripts (which can be automatically executed upon connection to a server, and are set in the connection dialog) and "callable" scripts (scripts that are executed by the **`/script`** command or the "Run Script" button on non-chat windows).

Connection scripts are executed with *no* arguments, and, thus, can't use the **`/argcount`** command (if this command is called, it is simply ignored). Connection scripts also can't use the **`/msgbox`** or **`/unalias`** commands. Other than that, they function just like other scripts.

"Callable" scripts are executed with whatever arguments are passed to the **`/script`** command; scripts executed with the "Run Script" button are executed with no arguments (calls to **`/argcount`** *will* be executed, however, any script using the command to look for anything other than zero (0) arguments will stop execution).

Comments in Scripts

Comments can be inserted into any Ərk script by containing them between **/*** and ***/**, much like in the C, Java, and Javascript programming languages. Comments can span multiple lines. Anything in between **/*** and ***/** will be ignored by the Ərk scripting engine.

```
/*
     This is an example comment.
     Comments can span multiple lines.
*/

/alias $GREETING Hello world!   /* This is a comment, too! */

/* The following command will not be executed:
/connect localhost 6667
*/
```

*Examples of Ərk script comments, as they would look in **Scriptər***

Directories

When started for the first time, Ərk will create a directory with several subdirectories in a user's home directory:

- **.erk**. This directory will contain all the other subdirectories.
- **.erk/settings**. This directory contains the default settings file, **`settings.json`**, and the user settings file, **`user.json`**. If any macros have been created, they will be saved here, in a file named **`macro.json`**.
- **.erk/scripts**. This is the default location for scripts. Ərk will look for scripts in this directory. The location of this directory can be set by using the **`--scripts`** command line option.
- **.erk/styles**. This is the default location for style files. Ərk will look for (and store) style files in this directory; it contains the default text style file, **`default.json`**. The location of this directory can be set with the **`--styles`** command line option.
- **.erk/logs**. This is where Ərk stores chat logs. The location of this directory can be set with the **`--logs`** command line option.

Miscellany

Macros created with the **`/macro`** command (see page 14) can be optionally saved and reloaded every time Ərk is ran; this behavior is enabled by default. To turn this off (that is: do not save macros or reload them on start), uncheck the "Save macros" option in the "Preferences" dialog.

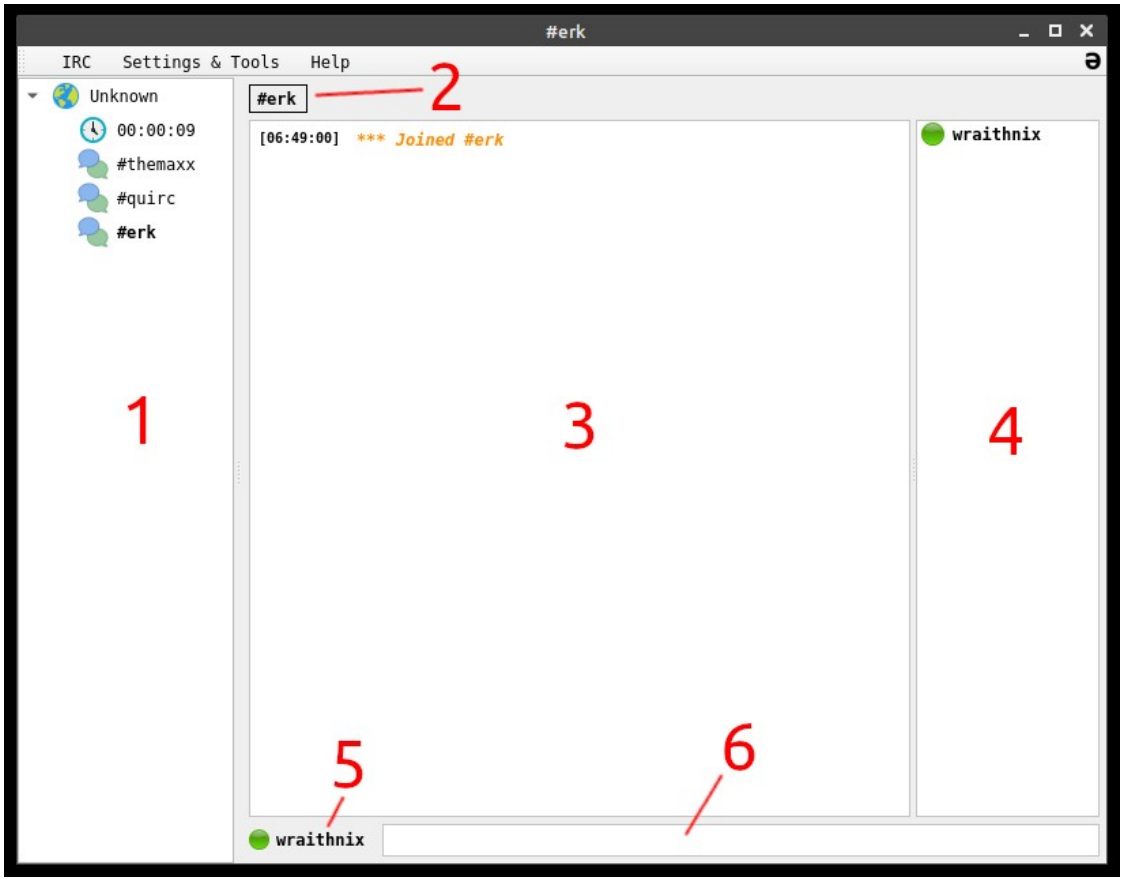
Alias variables (see **`/alias`** on page 12) only persist while Ərk is running, and are *not* saved to disk. Thus, to use specific alias variables, they must be recreated *every time Ərk is ran*. A good way to make sure that specific alias variables exist and are available to your scripts is to set them in connection scripts.

The **`/cat`** command (see page 13) *must* include the target argument if called from a server console window, or if called by a script with the "Run Script" button or by **Scriptər**. Otherwise, an error will be displayed and no messages will be sent.

All commands are preceded by a backslash, **`/`**. This is the default, but it can be changed in the Ərk configuration file, which is **.erk/settings/settings.json** by default (this can be changed with the **`--config`** command-line flag). To use a different character (or characters), change the "**`input_command_symbol`**" setting in the file to the character (or characters) you wish to use. Aspects of the GUI (such as the syntax highlighter in the connection dialog and **Scriptər**), as well as the built-in help system, will be altered to use the new character(s); scripts using the "old" character will no longer function as intended. There is no way to change this setting in the GUI, and changing this setting is unsupported.

All aliases (and other symbols for interpolation) are preceded by a dollar sign, **`$`**. This symbol is used with the **`/alias`** command, as well as with **`/script`** and **`/macro`** (for more information, see the entry for **`/alias`** on page 12). This is the default, but, like the command symbol, it can be changed in the Ərk configuration file. To use a different character (or characters), change the "**`script_interpolation_symbol`**" setting in the file to the character (or characters) you wish to use. Aspects of the GUI will be altered to use the new character(s); scripts using the "old" character will no longer function. There is no way to change this setting in the GUI, and changing this setting is unsupported.

The Ərk “Window” Interface

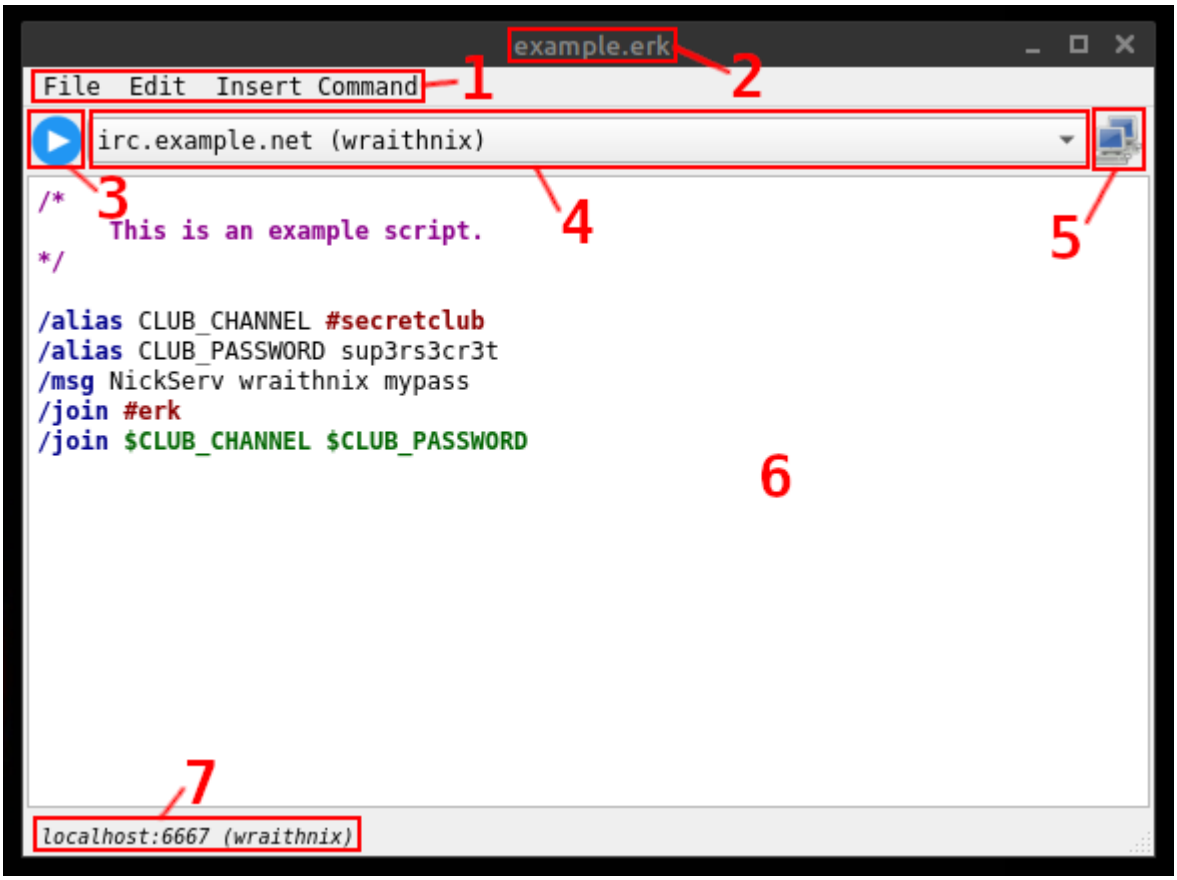


A channel chat window.

1. **Connection display.** Displays what servers the client is currently connected to, as well as what chats the client is “active” in, like channels or private chats. Click an entry to to switch to that server console, channel, or private chat window.
2. **Chat name.** Displays the name of the channel chat. For server consoles, the hostname (if set) of the server is displayed; for channels, the name and topic of the channel is displayed; private chats display the nickname of the user being chatted with.
3. **Chat.** This is where any chat text, or miscellaneous server messages, are displayed.
4. **User list.** Displays a list of users, and their status, in the current channel. This is not displayed on server console windows and private chats.
5. **Nickname.** Displays the current nickname being used, and if displayed on a channel window, the current channel status.
6. **Text input.** This is where users can enter chat text and commands.

Scriptər, the Script Editor

Ərk features a built-in script editor. It can open, edit, and save scripts, execute them on any server Ərk is connected to, and features syntax highlighting. To open the editor, use the **`/edit`** command, or click on the "**Scriptər**" entry in the "Settings & Tools" menu.



1. **Menu bar.** Here you can select scripts to open, save any new scripts or edits, and the normal tools you'd expect to find in a text editor. The "Insert Command" menu contains shortcuts to insert commonly used commands.
2. **File name.** The name of the script being edited.
3. **Execute.** Clicking this button executes the script in the editor on the selected Ərk client connection.
4. **Client selector.** This contains a list of all the servers that Ərk is connected to, allowing you to select which connection you want to execute the script on. The hostname of the server, followed by the nickname in use on that server, is displayed.
5. **Open connection script.** This will load the connection script for the currently selected client. If one does not exist, the editor will be set up to edit and save a new one.
6. **Editor.** Here, you can type your script, or edit any script you open. It features syntax highlighting (comments, commands, channel names, and alias variables are colored differently from the rest of the text). Currently, there is no option to customize the colors used; this will change in future versions.
7. **Client details.** This displays the hostname/IP address used to connect to the currently selected client, as well as the nickname in use.

IRC Commands

The these commands are for controlling the IRC connection (for example, joining and leaving channels, sending chat messages, etc); these commands should be familiar to users of other IRC clients such as mIRC or HexChat. Three commands (**/invite**, **/part** and **/topic**) will behave differently depending on where the command is entered; it will behave one way if it is entered into a chat window (for private chat or channel chat) or if it's entered into a console window (the window which displays data sent by the connected IRC server). One command, **/me**, cannot be entered into a non-chat window.

Called from any window

/away [MESSAGE...]

Sets the client to **AWAY**. If MESSAGE is included, **MESSAGE** will be sent to any user that sends a private message to the client.

Called from any window

/back

Sets the client to status to "back", and disables **/away** functionality.

Called from a chat window

/invite [CHANNEL] NICKNAME

Invites **NICKNAME** to the current channel if **CHANNEL** is omitted; invites **NICKNAME** to **CHANNEL** if a channel is included.

Called from any other window

/invite CHANNEL NICKNAME

Invites **NICKNAME** to **CHANNEL**.

Called from any window

/join CHANNEL [KEY]

Joins **CHANNEL**. If **KEY** is included, it will be passed to the server as part of the join request.

Called from any window

/list [TERMS]

Displays a list of all channels on the server if **TERMS** is omitted; displays a list of channels matching **TERMS** if **TERMS** is included. **TERMS** can contain wildcards such as ***** or **?**.

Called from a chat window

/me TEXT

Sends **TEXT** to the current chat as a CTCP action message. The command can *only* be called from a chat window.

Called from any other window

Called from a chat window

/mode [TARGET] MODE [ARGUMENTS]

Sets a **MODE** on the current channel if **TARGET** is omitted; sets a channel or user **MODE** if **TARGET** (which can be a channel or nickname) is included.

Called from any other window

/mode TARGET MODE [ARGUMENTS]

Sets a **MODE** on **TARGET** (which can be a channel or nickname).

Called from any window

/msg TARGET MESSAGE...

Sends a private message to **TARGET** (which can be a channel or a nickname).

Called from any window

/nick NICKNAME

Changes **NICKNAME**. If someone is already using **NICKNAME**, an error is displayed to the client.

Called from any window

/notice TARGET MESSAGE...

Sends a **NOTICE** to **TARGET** (which can be a channel or a nickname).

Called from any window

/oper USERNAME PASSWORD

Logs into a server operator account using **USERNAME** and **PASSWORD**.

Called from a chat window

/part [CHANNEL] [MESSAGE]

Leaves the current channel if **CHANNEL** is omitted; leaves **CHANNEL** if a channel is included. **MESSAGE** is the message displayed to other users when leaving; **MESSAGE** is optional.

Called from any other window

/part CHANNEL [MESSAGE]

Leaves **CHANNEL**. **MESSAGE** is the message displayed to other users when leaving; **MESSAGE** is optional.

Called from any window

/quit [MESSAGE...]

Disconnects from the current server. If **MESSAGE** is included, this will be sent to any channels the client is in before it disconnects.

Called from any window

/send MESSAGE

Sends **MESSAGE** to the server as a raw text; the outgoing message will not be altered. This is to allow the client to send messages to the server that the client doesn't normally support.

Called from any window

/time [SERVER]

Displays local time for the server currently connected to, or the **SERVER** specified.

Called from a chat window

/topic [CHANNEL] TEXT...

Sets the topic for the current channel if **CHANNEL** is omitted; sets the topic for another channel if **CHANNEL** is included.

Called from any other window

/topic CHANNEL TEXT...

Sets the topic for **CHANNEL**.

Called from any window

/version SERVER [SERVER...]

Requests server software version information from one or more **SERVER**s.

Called from any window

/who TEXT

Displays a list of users who's nickname matches **TEXT**.

Called from any window

/whois NICKNAME

Displays information about a specific user using **NICKNAME**.

Called from any window

/whowas NICKNAME [MAXIMUM ENTRIES] [SERVER]

Displays information about users with a **NICKNAME** that no longer exists.

Ərk Commands

The rest of the commands control the Ərk client software itself. Most of these commands can be entered into *any* window; however, the **/help** command will display slightly different output depending on what kind of window the command is called from. Six commands, **/argcount**, **/alias**, **/_alias**, **/msgbox**, **/unalias**, and **/wait**, can't be called by any window, and can only be called from scripts. Two commands, **/msgbox** and **/unalias**, can't be used in connection scripts.

If called with no arguments, the **/connect**, **/reconnect**, **/ressl**, and **/ssl** commands will open the "Connect" dialog with the appropriate options pre-selected.

If a "script only" command is called with the incorrect number of arguments or an argument in incorrect type (for example, calling **/wait** with a non-number argument), script execution will stop immediately and an error message with the reason for the error will be displayed.

Available only for scripts

/alias NAME MESSAGE...

Creates a new alias (called an "alias variable" in this document). Any script or command ran after this one (including the commands that follow the **/alias** command) that contains a dollar sign followed by **NAME** (**$NAME**) will have that instance replaced with **MESSAGE**. If the interpolation symbol (**$**) is included in **NAME**, it will be stripped, allowing for use of the alias variable normally (thus, if the command **/alias $EXAMPLE My message** is executed, using the alias variable would referenced with **$EXAMPLE**, not **$$EXAMPLE**). This command is *only* available for use in scripts; however, alias variables are interpolated into *any* input, including the text typed into the text input widget. Alias variables are "global"; once created, they are usable in any script or command input. Alias variables can also be overloaded (or overwritten) by any other script.

Available only for scripts

/_alias NAME MESSAGE...

Creates a new alias (called an "alias variable" in this document). Unlike the **/alias** command, this alias variable is *not* exported for use in other scripts/commands (thus, it is not "global"); this alias will only be interpolated for the script that it is defined it.

Available only for scripts

/argcount NUMBER MESSAGE...

Checks to make sure that a script is called with the proper **NUMBER** of arguments; if not enough or too many arguments are passed to the script, **MESSAGE** is displayed and script execution stops immediately. This command cannot be called from connection scripts.

Called from any window

/cat [TARGET] FILENAME

Reads in **FILENAME** and sends its contents to the current window as a series of messages if **TARGET** is omitted; otherwise, sends the contents to the **TARGET** user or channel. If a complete path to **FILENAME** is not provided, the client will look in the scripts directory (in **.erk/settings/scripts**, or set by the **--scripts** command-line flag) for the file; the "**.txt**" file extension will be appended to **FILENAME** while trying to find the file. Calling this command from a server console without **TARGET** will trigger an error.

Called from any window

/connect [SERVER] [PORT]

Causes the client to connect to an IRC **SERVER**. If **PORT** is omitted, a default of 6667 is used. If **SERVER** and **PORT** is omitted, the connection dialog is displayed. The client's current nickname, username, and real name is used.

Called from any window

/connectscript SERVER [PORT]

Opens the connect script for **SERVER** and **PORT** and executes it. If **PORT** is excluded, a default port of 6667 is assumed. Alternately, the **SERVER** and **PORT** can be passed to this command in the "**SERVER:PORT**" format.

Called from any window

/dictionary [WORD] ...

If called with no arguments, any words added to the spellcheck dictionary are displayed. If called with one or more arguments, **WORD** (and any **WORD**s after, separated by spaces) are added to the spellcheck dictionary. Added **WORD**s will not be marked as misspelled if the spellchecker is turned on.

Called from any window

/edit [FILENAME]

Opens up the built-in script editor, **Scriptər**; if a **FILENAME** is passed as an argument, the **FILENAME** is loaded into the script editor. If a complete path to **FILENAME** is not provided, the client will look in the scripts directory (in **.erk/settings/scripts**, or set by the **--scripts** command-line flag) for the file; the "**.erk**" file extension can be omitted.

Called from any window

/exit

Disconnects from all servers and exits the client.

Called from a chat window

/help

Displays a list of commands useful in channel and private message sessions, as well as all macros.

Called from any other window

/help

Displays a list of all commands and macros.

Called from any window

/macro NAME ARG_COUNT MESSAGE...

Creates a macro[1], allowing for new, custom command creation. **NAME** is the command that will trigger the command (in the form **/NAME**), **ARG_COUNT** is the number of arguments that the macro will accept, and **MESSAGE** is the output of the macro, which will be processed like a command. If **ARG_COUNT** is set to ***** (asterix), the macro will accept any number of arguments (including zero). If a macro is called with an incorrect number of arguments, an error is displayed, and the macro will not run.

Macros *cannot* have the same **NAME** as an existing command.

Arguments passed to the macro are interpolated into **MESSAGE** much like arguments to the **/script** command: instances of **$1** will be replaced with the first argument, **$2** with the second, and so on. There are other optional symbols that can be interpolated into **MESSAGE**:

SYMBOL	REPLACED WITH
$+	All arguments
$-	All arguments except for the first argument
$NICK	The nickname the client is using
$USERNAME	The username the client is using
$REALNAME	The real name the client is using
$HOSTNAME	The hostname of the server the client is connected to; if that is unknown, the server and port, delimited by a colon
$SERVER	The IP address or hostname used to connect to the server the client is connected to
$PORT	The port address on the server used to connect to the server the client is connected to
$WHERE	If the window the macro is called from is a server console, the hostname or server and port of the server. If the window the macro is called from is a channel or private chat, the name of the channel or the user being chatted with

If a macro using **NAME** already exists, it will be overwritten with the new macro.

For example, many IRC clients have included a macro called **/trout**, which, when called with a nickname as an argument, sends a CTCP action message to the current chat describing the user "hitting" the targeted nickname with a trout. To create this macro, you would issue this command:

```
/macro trout 1 /me slaps $1 with a trout
```

To call this macro (and "slap" your friend Bob with said trout), you could issue the command **/trout Bob**, which would send the command **/me slaps Bob with a trout**.

For more example usage, see **Macro Examples** on page 18.

1 "A macro…is a rule or pattern that specifies how a certain input should be mapped to a replacement output." *Wikipedia*

Called from any window

/macrohelp NAME MESSAGE...

Sets the text displayed for the **NAME**d macro when the **/help** command is used.

Called from any window

/macrousage NAME MESSAGE...

Sets the argument description for the **NAME**d macro when the **/help** command is used or when the macro is called with an incorrect number of arguments.

Available only for scripts

/msgbox MESSAGE...

Displays a message box with a custom **MESSAGE**.

Called from any window

/preferences

Opens the “Preferences” dialog.

Called from any window

/print MESSAGE...

Displays **MESSAGE** in the client’s current window. If **/print** is called from a window, the **MESSAGE** will be displayed in that window; if **/print** is called by any other method (like via the “Run Script” button, or from **Scriptər**), the **MESSAGE** will be displayed in the server’s console window. Text displayed with **/print** will *not* be saved in the message log.

Called from any window

/reconnect [SERVER] [PORT]

Causes the client to connect to an IRC **SERVER**, with the option to reconnect upon disconnection turned on.. If **PORT** is omitted, a default of 6667 is used. If **SERVER** and **PORT** is omitted, the connection dialog is displayed. The client’s current nickname, username, and real name is used.

Called from any window

/refresh

Requests a new list of channels from the server, and stores the new list in an internal cache.

Called from any window

/ressl [SERVER] [PORT]

Causes the client to connect to an IRC **SERVER** via SSL/TLS, with the option to reconnect upon disconnection turned on. If **PORT** is omitted, a default of 6697 is used. If **SERVER** and **PORT** is omitted, the connection dialog is displayed. The client’s current nickname, username, and real name is used.

Called from any window

/script FILENAME [ARGUMENT ...]

Opens **FILENAME**, reads its contents into memory, and executes it as a list of commands; each additional **ARGUMENT**, is passed to the script. Any errors found in the script are displayed to the current window. **ARGUMENT**s can be used in the script much like **/alias** variables: **$1** is replaced with the first argument, **$2** with the second, and so on. To interpolate all **ARGUMENT**s as a single string, use **$0**. **ARGUMENT**s are interpolated into the script before **/alias** variables. If a line in a script doesn't contain a valid command, it will be sent to the current window as chat text, just as if the user entered it as chat text; if the current window is a non-chat window, the text will be ignored. If a complete path to **FILENAME** is not provided, the client will look in the scripts directory (in **.erk/settings/scripts**, or set by the **--scripts** command-line flag) for the file; the "**.erk**" file extension can be omitted.

Called from any window

/settings

Opens the client's settings dialog.

Called from any window

/ssl [SERVER] [PORT]

Causes the client to connect to an IRC **SERVER** via SSL/TLS. If **PORT** is omitted, a default of 6697 is used. If **SERVER** and **PORT** is omitted, the connection dialog is displayed. The client's current nickname, username, and real name is used.

Called from any window

/style FILENAME

Loads a style file (created with the style editor, **Stylər**, accessible from the "Settings" menu) into the current chat. If a complete path to **FILENAME** is not provided, the client will look in the styles directory (in **.erk/settings/styles**, or set by the **--styles** command-line flag) for the file.

Called from any window

/switch [WINDOW_NAME]

Switches the current displayed channel. If **WINDOW_NAME** is omitted, a list of available channels/private chats is displayed.

Available only for scripts

/unalias NAME

Removes a global alias variable named **NAME**. The interpolation symbol (**$**) *cannot be included* in **NAME**. This command *cannot* be used to delete variables created with the **/_alias** command. This command *cannot* be called from a connection script (see **Connection Scripts and "Callable" Scripts** on page 4).

Called from any window

/undictionary WORD [WORD...]

Removes **WORD** (and any additional **WORD**s, separated by spaced) from the spellcheck dictionary.

Called from any window

/unmacro NAME

Deletes a macro.

Available only for scripts

/wait TIME

Causes a script to pause for **TIME** seconds. This command is *only* available for use in scripts.

Called from any window

/write MESSAGE...

Displays **MESSAGE** in the client's current window. If **/write** is called from a window, the **MESSAGE** will be displayed in that window; if **/write** is called by any other method (like via the "Run Script" button, or from **Scriptər**), the **MESSAGE** will be displayed in the server's console window. Text displayed with **/write** *will* be saved in the message log.

Macro examples

Greeter macro

Command `/macro hi 1 /msg $WHERE Welcome to $WHERE, $1!`

Description

Creates a macro to welcome someone to whatever channel you're in. Pass the nickname of whoever you want to welcome to the channel as the first argument, and it will send a public message to channel welcoming them. For example, if you were in a channel named **#erk**, and you wanted to welcome a new user named Alice to the channel, you'd execute:

```
/hi Alice
```

Which would send the following public message to **#erk**: "Welcome to #erk, Alice!"

Trout macro

Command `/macro trout 1 /me slaps $1 with a trout.`

Description

Creates a macro to send the old classic message. Pass the nickname of the user you'd like to slap, and it will send a CTCP action message to the current channel or private chat. For example, to slap a user named Bob:

```
/trout Bob
```

Which would send the following CTCP action message to the chat: "[your nickname] slaps Bob with a trout.

Special attention trout macro

Command `/macro trouts * /me slaps $1 with a trout, and $- too`

Description

Creates a macro to send a new classic message. Pass the nickname of the main target, and a short list of other targets to the macro to send the new CTCP action message. For example, to slap Bob, Alice, and Dave:

```
/trouts Bob Alice and Dave
```

Which would send the following CTCP action message to the chat: "[your nickname] slaps Bob with a trout, and Alice and Dave, too"