



*Ærk IRC Client, Version 0.860*

# Ærk Plugin Guide

A guide for using and writing plugins for the Ærk IRC Client

**<https://github.com/nutjob-laboratories/erk>**

<b>Ørk Plugin Guide.....</b>	<b>1</b>
<b>Summary.....</b>	<b>3</b>
<b>Using Ørk Plugins.....</b>	<b>4</b>
<b>Writing Ørk Plugins.....</b>	<b>5</b>
<b>Plugin Requirements.....</b>	<b>5</b>
<b>Event Methods.....</b>	<b>6</b>
line_in.....	6
line_out.....	6
public.....	6
private.....	6
action.....	6
mode.....	7
tick.....	7
input.....	7
notice.....	7
motel.....	8
registered.....	8
join.....	8
part.....	8
joined.....	8
parted.....	8
<b>Plugin Built-In Tools.....</b>	<b>9</b>
irc Object.....	9
Built-In Methods.....	9
console.....	9
exec.....	10
print.....	10
script.....	10
write.....	10

# Summary

Ørk plugins are Python 3 classes that inherit from a base class, named “Plugin”, that is imported from the Ørk application. They are loaded from a subdirectory in the main directory where Ørk stores settings and scripts, **.erk/plugins**, located in the user’s home directory.

A basic plugin looks something like this:

```
from erk import *

class DumpPlugin(Plugin):
    name = "Dump Plugin"
    version = "1.0"
    description = "Displays all IRC network traffic"

    def input(self,data):
        print("<- "+data)

    def output(self,data):
        print("-> "+data)
```

If this was saved to a file and placed in **.erk/plugins**, the next time Ørk is started, it would load the plugin and appear in Ørk’s “Plugins” menu like so:



Once loaded, the plugin would print all network traffic, both input and output, from all servers that Ørk is connected to, to the console.

# Using Ørk Plugins

To install an Ørk plugin, simply place it in the `.erk/plugins` directory in your home directory. The next time Ørk starts up, the plugin will be loaded automatically. If there are major errors in the plugin (for example, improper Python code), Ørk will not start, and any errors will be printed to the console. If there are minor errors (for example, if plugins are missing needed attributes), Ørk will start normally, and any errors will be displayed to the user in a dialog window. Plugins with minor errors will *not* be loaded into the client, and, thus, will be ignored by Ørk.

Uninstalling Ørk plugins is as easy as deleting the plugin from `.erk/plugins`. No other steps are required.

# Writing Ørk Plugins

## Plugin Requirements

Ørk plugins are Python 3 classes that must meet four basic requirements:

- The class must inherit from the base class **Plugin**
- The class must have a **name** attribute string
- The class must have a **version** attribute string
- The class must have at least one event method

To get access to the first requirement, simply import **Plugin** from **erk**. You can either explicitly import **Plugin**:

```
from erk import Plugin
```

or use a “splat” and import it implicitly:

```
from erk import *
```

The end result is the same either way; the only thing exported by **erk** is **Plugin**.

The second two requirements, the **name** and **version** attributes, are used by Ørk to display the plugin in the client (see page 3). These should be class attributes, rather than instance attributes<sup>1</sup>. **name** should be set to the name of the plugin, a short descriptive string; it must contain at least one character that is not whitespace. **version** should be the version number of the plugin; if no version number is required, this attribute can be set to a blank string, but it *must* exist. A third attribute, **description**, is optional; this should be a short string that describes what the plugin does.

The last requirement is for the plugin to have at least one event method. An event method is a plugin method that Ørk will execute when a specific event occurs.

---

<sup>1</sup> <https://www.geeksforgeeks.org/class-instance-attributes-python/>

## Event Methods

Ørk executes the event method of every plugin that contains that event method when a specific event occurs (like the reception of a public message, a private message, or a notice message). In short, Ørk plugins are event-driven<sup>2</sup>.

### line\_in

Arguments    **data** (string)

Description    Triggered every time Ørk receives data from an IRC server. **data** contains the data sent to the client.

### line\_out

Arguments    **data** (string)

Description    Triggered every time Ørk sends data to an IRC server. **data** contains the data sent to the server.

### public

Arguments    **channel** (string), **user** (string), **message** (string)

Description    Triggered every time Ørk receives a public message. **channel** contains the name of the channel the message was sent to, **user** contains the nickname, username, and hostname of the user that sent it (in the format **nickname!**  
**username@hostname**), and **message** contains the contents of the message.

### private

Arguments    **user** (string), **message** (string)

Description    Triggered every time Ørk receives a private message. **user** contains the nickname, username, and hostname of the user that sent it (in the format **nickname!**  
**username@hostname**), and **message** contains the contents of the message.

### action

Arguments    **target** (string), **user** (string), **message** (string)

Description    Triggered every time Ørk receives a CTCP action message. **target** contains the target of the message (if it's a public message, **target** will contain the name of the channel the message was sent to, and if it's a private message, **target** will contain the nickname the Ørk client is using), **user** contains the nickname, username, and hostname of the user that sent it (in the format **nickname!**  
**username@hostname**), and **message** contains the contents of the message.

<sup>2</sup> [https://en.wikipedia.org/wiki/Event-driven\\_programming](https://en.wikipedia.org/wiki/Event-driven_programming)

## mode

Arguments

**channel** (string), **user** (string), **mset** (bool), **modes** (string), **args** (list)

Description

Triggered every time Ørk receives a mode message. **channel** contains the name of the channel the message was sent to, **user** contains the nickname, username, and hostname of the user that sent it (in the format **nickname!** **username@hostname**), **mset** is True if a mode is being set and False if a mode is being removed/unset, **modes** contains the mode(s) being set, and **args** contains a list of the arguments to the mode being set/unset.

## tick

Arguments

**uptime** (integer)

Description

Triggered once every second that Ørk is connected to a server. **uptime** is the length of the connection to the server in seconds. Timers are specific to each connection; each server connection's tick event is independent of all the other server connections.

## input

Arguments

**name** (string), **text** (string)

Description

Triggered every time a user types something into Ørk's text input widget. **name** is the name of the window where the text was entered (a channel name for a public chat window, or a nickname for a private chat window), and **text** is the text that was entered. If the text was entered into a server console window, **name** will be set to **None**. If the user has implemented any macros, they will be interpolated into the text *before* being handed to this method. To stop Ørk from processing the user input, return **True** from this method; to allow Ørk to continue to work with the input text, return **False** or nothing. As it would be very easy to write a malicious plugin that would disable *any* input, Ørk will try to detect malicious **input()** methods and prevent the plugin that contains them from being loaded.

## notice

Arguments

**target** (string), **user** (string), **message** (string)

Description

Triggered every time Ørk receives a notice message. **target** contains the name of the target the message was sent to (either a channel or your nickname), **user** contains the nickname, username, and hostname of the user that sent it (in the format **nickname!** **username@hostname**), and **message** contains the contents of the message.

## **motd**

Arguments	<b>message</b> (list)
Description	Triggered when Ørk receives the message of the day (MOTD) from an IRC server. <b>message</b> contains the MOTD, with each entry consisting of one line.

## **registered**

Arguments	None
Description	Triggered when Ørk completes connecting to an IRC server.

## **join**

Arguments	<b>channel</b> (string), <b>user</b> (string)
Description	Triggered every time another user joins a channel that the Ørk client is “present” in. <b>channel</b> contains the channel the user joined, and <b>user</b> contains the nickname, username, and hostname of the user that joined (in the format <b>nickname!username@hostname</b> ).

## **part**

Arguments	<b>channel</b> (string), <b>user</b> (string)
Description	Triggered every time another user leaves a channel that the Ørk client is “present” in. <b>channel</b> contains the channel the user left, and <b>user</b> contains the nickname, username, and hostname of the user that left (in the format <b>nickname!username@hostname</b> ).

## **joined**

Arguments	<b>channel</b> (string)
Description	Triggered whenever Ørk joins a channel. <b>channel</b> contains the channel that the client joined.

## **parted**

Arguments	<b>channel</b> (string)
Description	Triggered whenever Ørk leaves a channel. <b>channel</b> contains the channel that the client left.

## Plugin Built-In Tools

The `Plugin` class contains some tools to make interacting with the Ørk client and any connected IRC servers easy.

### irc Object

The `irc` object is part of the `Plugin` class, and is the way plugins interact with IRC servers. It is integrated as an instance attribute of the class. This object is the instance of the Twisted IRC client<sup>3</sup> that Ørk is using for communication with the IRC server. Anything you can normally do with the Twisted IRC client, you can use `irc` for. Whenever an event method is triggered, the `irc` object is set to the Twisted instance that is connected to the server event that triggered the method. So, for example, if you wanted to write a private event method that forwards all private messages to another user with the nickname “OtherNick”, you could write:

```
def private(user,message):
    self.irc.msg("OtherNick",user+": "+message)
```

If the `irc` object is unavailable (due to a disconnection, error, or other reason), the object’s value is set to `None`.

For help on how to use the `irc` object, take a look at the documentation for the Twisted IRC Client<sup>4</sup>.

### Built-In Methods

The Plugin class also comes with several methods built into it. These methods are used to interact with the Ørk graphical user interface (GUI).

#### console

Arguments	<code>text</code> (string)
Returns	Nothing
Description	Prints <code>text</code> to the window associated with the current server connection, called the “console”.

<sup>3</sup> `twisted.words.protocols.irc.IRCClient`

<sup>4</sup> <https://twistedmatrix.com/documents/current/api/twisted.words.protocols.irc.IRCClient.html>

## exec

Arguments	<b>text</b> (string)
Returns	Nothing
Description	Executes the contents of <b>text</b> as if it were entered into the text input widget in the Ørk client. This allows plugins to execute miscellaneous scripting commands (see <i>Ørk Scripting and Commands</i> ). If <b>text</b> does not contain a command, it will be sent to the currently open window as chat.

## print

Arguments	<b>text</b> (string), ...
Returns	Nothing
Description	Prints <b>text</b> to the currently open window; if the currently open window can't be found, then the text will print to the console. To print multiple items in the same command, separate the strings to print with commas (much like Python's <b>print</b> function).

## script

Arguments	<b>filename</b> (string), <b>arguments</b> (list)
Returns	Nothing
Description	Executes the contents of <b>filename</b> as an Ørk script. This functions exactly like the Ørk command <b>/script</b> (see <i>Ørk Scripting and Commands</i> ).

## write

Arguments	<b>target</b> (string), <b>text</b> (string)
Returns	Bool
Description	Prints <b>text</b> to the window with <b>target</b> as its name. Windows are named after the chat they display; so, for example, the window displaying channel chat for #erk would be named "#erk". The window displaying private chat with a user named "Bob" would be named "Bob". If the window was found and the text was written to it, <b>write</b> will return <b>True</b> ; if the window was <i>not</i> found, <b>write</b> will return <b>False</b> .