
Poezio Documentation

Release 0.13-dev

Mathieu Pasquet - Florent Le Coz - Emmanuel Gil Peyrot

Oct 12, 2018

Contents

1	Installing poezio	1
1.1	poezio in the GNU/Linux distributions	1
1.2	Install from source	1
1.3	Docker images	4
2	Configuration	5
2.1	Global section options	5
2.2	Optional section options	16
3	Usage	19
3.1	Tab list	19
3.2	Generalities	20
3.3	Contact list tab	21
3.4	Chatroom tab	21
3.5	Private tab	23
3.6	Conversation tab	23
3.7	Dataforms tab	25
3.8	List tab	25
3.9	Confirm tab	25
3.10	Bookmarks tab	27
4	Commands	29
4.1	Global commands	29
4.2	Chat tab commands	32
4.3	MultiUserChat tab commands	33
4.4	Private tab commands	34
4.5	Normal Conversation tab commands	34
4.6	Contact list tab commands	34
4.7	XML tab commands	36
5	Keys	37
5.1	Key bindings listing	37
5.2	Key configuration	40
5.3	Actions	41
6	Other topics	45
6.1	Message Carbons	45

6.2	Using client certificates to login	45
6.3	Message Correction	46
6.4	Personal Events	46
6.5	Installing python 3.5 as a user	48
6.6	Using several accounts	48
6.7	TLS in poezio	49
6.8	Troubleshooting	51
7	Quickstart guide	53
7.1	Anonymous usage	53
7.2	Normal usage	53
8	Themes	55
8.1	Create a theme	55
8.2	Use a theme	56
8.3	Change the theme directory	56
8.4	Available options	57
9	Plugins	59
9.1	Setting up plugins	59
9.2	Plugin autoload	59
9.3	Manual plugin load	59
9.4	Plugin configuration	60
9.5	Plugin index	60
10	Development documentation	85
10.1	About plugins	85
10.2	About Poezio	98
11	Indices and tables	105
	Python Module Index	107

Warning: Python 3.5 or above is **required**. To install it on a distribution that doesn't provide it, see [pyenv](#).

1.1 poezio in the GNU/Linux distributions

As far as I know, Poezio is available in the following distributions, you just have to install it by using the package manager of the distribution, if you're using one of these.

- **Archlinux:** [poezio](#) and [poezio-git](#) packages are in the AUR (use your favourite AUR wrapper to install them)
- **Gentoo:** It's uncertain, but the [bgo-overlay](#) appears to contain poezio and slxmpp packages.
- **Fedora:** The stable poezio package was out of date for a long time in Fedora, but now thanks to Casper, there is an [up-to-date package](#) in the repos since F19.
- **Debian:** A stable package is provided since [buster](#) thanks to debacle.
- **Nix (and NixOS):** The last stable version of poezio is available in the unstable branch of *nixpkgs*. Use `nix-env -f "<nixpkgs>" -iA poezio` to install poezio for the current user.
- **OpenBSD:** a poezio [port](#) is available

(If another distribution provides a poezio package, please tell us and we will add it to the list)

Thank to all the maintainers who took time to make and maintain those packages!

1.2 Install from source

1.2.1 Stable version

Stable version packages are available in standalone (dependencies provided) and poezio-only packages (both with prebuilt html doc for convenience).

Those versions are also available on [pypi](#) (using `pip3`, for example), and it is recommended to install them this way if you absolutely want to **install** poezio and your distribution provides no package.

1.2.2 Development version

The stable versions of poezio are more like snapshots of states of development we deem acceptable. There is always an incentive to use the development version, like new features, bug fixes, and more support. Therefore, you might want to use the git version.

```
git clone git://git.poez.io/poezio
cd poezio
```

General

Poezio is a python3.5 (and above)-only application, so you will first need that.

Packages required for building poezio and deps:

- `make`
- `gcc`
- `libidn` and `libidn-dev`, only if you want to use [cython](#) (see below)
- `python3-devel` (or equivalent)
- `python3-setuptools`

Then you can run `make` to build it the poezio C extension module. If you downloaded the standalone stable package, you are finished here and can skip to [running poezio](#).

Poezio needs two libraries to run:

- [aiodns](#)
- [slixmpp](#)
- `slixmpp` can make use of [cython](#) to compile performance-critical modules and be faster

Changed in version 0.9.

Note: We provide an `update.sh` script that creates a virtualenv and downloads all the required and optional dependencies inside it. we recommend using it with the git version of poezio, in order to keep everything up-to-date.

If you don't want to use the update script for whatever reason, install the following dependencies by hand; otherwise, skip to the [installation part](#).

slixmpp

Poezio depends on `slixmpp`, a non-threaded fork of the `SleekXMPP` library.

```
git clone git://git.poez.io/slixmpp
python3 setup.py install --user
```

aiodns

The aiodns library is required in order to properly resolve XMPP domains (with SRV records).

```
pip install --user aiodns
```

This will also install pycares, which aiodns uses.

Building

If you don't run the `update.sh` script, you need to manually build the C module used by poezio:

```
make
```

1.2.3 Installation

Note: The `update.sh` + `launch.sh` method is the recommended way of using and upgrading the devel version of poezio. Installing should only be done with stable versions. And preferably using your distribution's package manager.

If you skipped the installation of the dependencies and you only want to run poezio without a system-wide install, do, in the `poezio` directory:

```
./update.sh
```

Note: You should probably install cython (for python3) on your system using your package manager, since the installation from pypi takes a long time.

Note: If you want to use a custom directory for the virtualenv used by poezio, you can use the `$POEZIO_VENV` environment variable to set use another path (the default is `poezio-venv`).

Note: The python version used can be customized using the `$POEZIO_PYTHON` env variable.

If your distribution's python3 does not have a `venv` module, install the package corresponding to that module (probably `python3-venv`).

Changed in version 0.12: Previously there was a `$POEZIO_VENV_COMMAND` env variable to define the command. Now it is required to use `$POEZIO_PYTHON`.

If you really want to install it, run as root (or sudo in ubuntu or whatever):

```
make install
```

1.2.4 Running

If you didn't install poezio, you can run it from the source directory with:

```
./launch.sh
```

If you did, it should be in the `$PATH` as `poezio`, so run:

```
poezio
```

1.3 Docker images

`poezio` is available on the docker hub in the [poezio/poezio](#) repository in which `poezio/poezio:latest` is the latest built git version, and stable versions are tagged with their numbers. The image is based off alpine linux and we tried to keep the image size to a minimum (<100MiB).

You can therefore just fetch the images with docker pull:

```
docker pull poezio/poezio
```

In order to run `poezio` with non-temporary config and logs, and to have the right colors, you have to share the `TERM` env var and some directories that should be created beforehand:

```
mkdir -p ~/.config/poezio ~/.local/share/poezio
docker run -it -e TERM -v ~/.config/poezio:/home/poezio-user/.config/poezio -v ~/.
↳local/share/poezio:/home/poezio-user/.local/share/poezio poezio/poezio
```

If you don't trust images distributed on the docker hub, you can rebuild the image from the Dockerfile at the root of the git repository.

CHAPTER 2

Configuration

The configuration is located in the file `~/.config/poezio/poezio.cfg`. On its first startup, poezio will create that file (and its containing directories) with the default configuration. You can edit that file manually or use the `/set` command to edit some of its values directly from poezio. This file is also used to configure key bindings, but this is explained in the [Keys](#) documentation file.

That file is read at each startup and the configuration is saved when poezio is closed.

This configuration file **requires** all global options to be in a section named `[Poezio]`. Some other options can be in optional sections and will apply only to tabs having the option's name.

An option is formatted like this:

```
option = value
```

An empty value *doesn't* mean that the default value will be used. That's just an empty value. To use the default value, just comment or remove the option entirely.

Here is a list of all the available configuration options, their meaning and their default value.

2.1 Global section options

These options have a sense when they are in the global section. Some of them can also be in an optional configuration section, see the next section of this documentation.

The options here are separated thematically for convenience but they all go into the main config section.

2.1.1 Security

Options pertaining to security, such as [TLS encryption](#) and certificate validation.

ca_cert_path Default value: `[empty]`

Path to the certificate of the Certification Authority. As some services may keep different certificates, it is an alternative to the Trust On First Use model provided by the *certificate* option. This option is not affected by *ignore_certificate* and both checks may be active at the same time.

certificate **Default value:** `[empty]`

The SHA-2 fingerprint of the SubjectPublicKeyInfo of the SSL certificate as a hexadecimal string, you should not touch it, except if know what you are doing.

Note: the fingerprint was previously a fingerprint of the whole certificate, while it is now only of the SubjectPublicKeyInfo, which persists across LetsEncrypt renewals, and therefore reduces the noise generated by the alert dialog.

versionchanged:: 0.12

ciphers **Default value:** `HIGH+kEDH:HIGH+kEECDH:HIGH:!PSK:!SRP:!3DES:!aNULL`

The TLS cipher suites allowed, in [OpenSSL format](#). Modify this if you know what you are doing, see the *Ciphers* dedicated section for more details.

force_encryption **Default value:** `true`

If set to true, all connections will use TLS by default. Only turn this to false if you cannot connect to your server, and do not care about your password or the privacy of your communications.

ignore_certificate **Default value:** `false`

Skip certificate validation on connection when `true`. Useful when you are in anonymous mode and changing servers often. Dangerous in other cases, from a security perspective.

2.1.2 Account

Options related to account configuration, nickname...

alternative_nickname **Default value:** `[empty]`

If you want poezio to join the room with an alternative nickname when your nickname is already in use in the room you wanted to join, put a non-empty value. If you don't, poezio won't join the room This value will be added to your nickname to create the alternative nickname. For example, if you set `"_"`, and wanted to use the nickname "john", your alternative nickname will be "john_".

certfile **Default value:** `[empty]`

Path to a PEM certificate file to use for certificate authentication through SASL External. If set, *keyfile* **MUST** be set as well in order to login.

custom_host **Default value:** `[empty]`

A custom host that will be used instead of the DNS records for the server (anonymous or the *jid*'s) defined above. You should not need this in a "normal" use case.

custom_port **Default value:** `[empty]`

A custom port to use instead of the 5222. This option can be combined with *custom_host*. You should not need this in a "normal" use case.

default_nick **Default value:** `[empty]`

the nick you will use when joining a room with no associated nick If this is empty, the `$USER` environment variable will be used

jid Default value: [empty]

Jabber identifier. Specify it only if you want to connect using an existing account on a server. This is optional and useful only for some features, like room administration or nickname registration. The *server* option will be ignored if you specify a JID (Jabber id) It should be in the form *nickname@server.tld* or *nickname@server.tld/resource*

keyfile Default value: [empty]

Path to a PEM private key file to use for certificate authentication through SASL External. If set, *certfile* MUST be set as well in order to login.

open_all_bookmarks Default value: false

If this option is set to *true*, all remote bookmarks, even those that do not have autojoin, will be opened on startup. (the tabs without autojoin will not be joined)

password Default value: [empty]

A password is needed only if you specified a *jid*. It will be ignored otherwise If you leave this empty, the password will be asked at each startup, which is recommended.

rooms Default value: [empty]

The rooms you will join automatically on startup, with associated nickname or not.

Format: *room@server.tld/nickname:room2@server.tld/nickname2*.

The *default_nick* option will be used if “/nickname” is not specified.

server Default value: *anon.jeproteste.info*

The server to use for anonymous authentication; make sure it supports anonymous authentication.

Note that this option doesn’t do anything at all if you’re using your own JID.

status Default value: [empty]

The status (show) poezio will send when connecting. It can be *available*, *dnd*, *chat*, *xa* or *away*.

Nothing or an invalid value will mean *available*.

status_message Default value: [empty]

The status message poezio will send when connecting.

2.1.3 Connectivity

Options about general or chatroom connectivity. Reconnecting does not work very well, but you will at least want to know when you get disconnected.

auto_reconnect Default value: *true*

Auto-reconnects you when you get disconnected from the server. Poezio will try to reconnect forever, until it succeeds.

autorejoin Default value: *false*

Set to *true* if you want to automatically rejoin the room when you’re kicked.

autorejoin_delay Default value: 5

Set to the number of seconds before reconnecting after getting kicked. 0, a negative value, or no value means you reconnect instantly. This option only works if *autorejoin* is enabled.

connection_check_interval Default value: 300

A ping is sent to the server every N seconds, N being the value of that option. Change this to a low value if you want to know quickly when you are disconnected, and to a very high value if bandwidth matters so much that you can't afford 100 bytes/minute, or if you don't want to waste your battery by waking up the TCP connection too often. Disable this ping altogether by setting this value to 0.

connection_timeout_delay Default value: 30

The timeout delay of the ping referenced above, 30 should really be fine, but if your network is really unstable, it can be set higher or lower, depending of your preference.

whitespace_interval Default value: 300

Interval of the whitespace keepalive sending to the server. 300 should be fine, but change it if some services have a stricter policy on client inactivity.

2.1.4 XMPP features

These options enable, disable, or allow to configure the behavior of some non-essential XMPP features. There is a dedicated page to understand what is *carbons* or *user activity/gaming/mood/tune*.

ack_message_receipts Default value: true

Acknowledge message receipts requested by the other party.

bookmark_on_join Default value: false

If true, poezio will bookmark automatically every room that is joined with a manual /join command.

display_activity_notifications Default value: false

If set to true, notifications about the current activity of your contacts will be displayed in the info buffer as 'Activity' messages.

display_gaming_notifications Default value: false

If set to true, notifications about the games your are playing will be displayed in the info buffer as 'Gaming' messages.

display_mood_notifications Default value: false

If set to true, notifications about the mood of your contacts will be displayed in the info buffer as 'Mood' messages.

display_tune_notifications Default value: false

If set to true, notifications about the music your contacts listen to will be displayed in the info buffer as 'Tune' messages.

enable_avatars Default value: true

Display contact avatars in the roster.

enable_carbons Default value: true

Set this to false to disable Message Carbons (XEP-280), which allows transparent message delivery from and to other resources with carbons enabled. There should be no reason to disable this except if you encounter issues with your server.

enable_css_parsing Default value: true

When parsing XHTML-IM content, only keep semantic elements, and not inline text styles. Only useful if *enable_xhtml_im* is enabled.

enable_smacks Default value: `false`

Stream Management (XEP-0198) is an extension designed to improve the reliability of XMPP in unreliable network conditions (such as mobile networks). It can however increase bandwidth usage. It also requires server support.

enable_user_activity Default value: `true`

Set this to `false` if you don't want to receive the activity of your contacts.

enable_user_gaming Default value: `true`

Set this to `false` if you don't want to receive the gaming activity of your contacts.

enable_user_mood Default value: `true`

Set this to `false` if you don't want to receive the mood of your contacts.

enable_user_nick Default value: `true`

Set to `false` if you don't want your contacts to hint you their identity.

enable_user_tune Default value: `true`

If this is set to `false`, you will no longer be subscribed to tune events, and the [display_tune_notifications](#) option will be ignored.

enable_xhtml_im Default value: `true`

XHTML-IM is an XMPP extension letting users send messages containing XHTML and CSS formatting. We can use this to make colored text for example. Set to `true` if you want to see colored (and otherwise formatted) messages.

force_remote_bookmarks Default value: `false`

Try to retrieve your remote bookmarks, even when your server doesn't advertise support.

go_to_previous_tab_on_alt_number Default value: `false`

If this is set to `true`, when Alt+x is pressed, where x is a number, if you are already on the tab number x, you will jump to the previously selected tab. Otherwise you'll stay on the same tab.

group_corrections Default value: `true`

Enable a message to "correct" (replace) another message in the display if the sender intended it as such. See [Message Correction](#) for more information.

request_message_receipts Default value: `true`

Request message receipts when sending messages (except in groupchats).

send_chat_states Default value: `true`

if `true`, chat states will be sent to the people you are talking to. Chat states are, for example, messages informing that you are composing a message or that you closed the tab, etc.

Set to `false` if you don't want people to know these information Note that you won't receive the chat states of your contacts if you don't send yours.

send_os_info Default value: `true`

If `true`, information about the Operation System you're using will be sent when requested by anyone Set to `false` if you don't want people to know these information.

Note that this information will not be sent if [send_poezio_info](#) is False

send_poezio_info Default value: `true`

if `true`, information about the software (name and version) will be sent if requested by anyone Set to `false` if you don't want people to know these information

send_time Default value: `true`

If `true`, your current time will be sent if asked Set to `false` if you don't want people to know that information

use_bookmark_method Default value: `[empty]`

The method that poezio will use to store your bookmarks online. Possible values are: `privatexml`, `pep`. You should not have to edit this in a normal use case.

use_pep_nick Default value: `true`

Use the nickname broadcasted by the user if set to `true`, and if none has already been set manually.

use_remote_bookmarks Default value: `true`

Use this option to force the use of local bookmarks if needed. Anything but “false” will be counted as `true`.

2.1.5 Visual interface

All these options will change how poezio looks, either by removing parts of the interface, adding them, changing the ordering of stuff, or the way messages are displayed.

create_gaps Default: `false`

Create gaps when moving a tab or closing it. Enabling this option will help you keep the tabs at the same place during the execution of poezio. (gaps are not created when the closed tab is the last one)

deterministic_nick_colors Default value: `true`

Use a deterministic algorithm to choose the user colors in chatrooms if set to `true`. Otherwise the colors will be picked randomly.

The value of this option affects the behavior of */recolor*.

display_user_color_in_join_part Default value: `true`

If set to `true`, the color of the nick will be used in chatroom information messages, instead of the default color from the theme.

enable_vertical_tab_list Default value: `true`

If `true`, a vertical list of tabs, with their name, is displayed on the left of the screen. Otherwise, it is a horizontal bar with just the tab numbers above the input bar.

filter_info_messages Default value: `[empty]`

A list of words or sentences separated by colons (“:”). All the informational messages (described above) containing at least one of those values will not be shown.

hide_exit_join Default value: `-1`

Exact same thing than *hide_status_change*, except that it concerns the quit message, and that it will be hidden only if the value is 0.

Default setting means: - all quit and join notices will be displayed

hide_status_change Default value: `120`

Set a number for this setting. The join AND status-change notices will be displayed according to this number.

-1: the notices will ALWAYS be displayed

0: the notices will NEVER be displayed

n: On any other number, the notices will only be displayed if the user involved has talked since the last n seconds if the value is incorrect, -1 is assumed

Default setting means that status changes won't be displayed unless the user talked in the last 2 minutes

hide_user_list Default value: `false`

Whether to hide the list of user in the MultiUserChat tabs or not. Useful for example if you want to copy/paste the content of the buffer, or if you want to gain space

highlight_on Default value: `[empty]`

a list of words (separated by a colon (:)) that will be highlighted if said by someone on a room

information_buffer_popup_on Default value: `error roster warning help info`

Some informational messages (error, a contact getting connected, etc) are sometimes added to the information buffer. These settings can make that buffer grow temporarily so you can read these information when they appear.

A list of message types that should make the information buffer grow Possible values: `error, roster, warning, info, help`

information_buffer_type_filter Default value: `[empty]`

Some informational messages (error, a contact getting connected, etc) are sometimes added to the information buffer.

A list of message types separated by colons (":") that should never be displayed in the information buffer. Possible values: `error, roster, warning, info, help`

max_nick_length Default value: `25`

The maximum length of the nickname that will be displayed in the conversation window. Nicks that are too long will be truncated and have a . . . appened to them.

muc_colors (section) Default: `[empty]`

Fix a color for a nick. Whenever such a nick appears in a chatroom, it will be displayed in that color. This color won't be changed by the recolor command.

nick_color_aliases Default value: `true`

Automatically search for color of nick aliases. For example, if nick is set to red, `_nick`, `nick_`, `_nick_`, `nick__` etc. will have the same color. Aliases colors are checked first, so that it is still possible to have different colors for `nick_` and `nick`.

popup_time Default value: `4`

The time the message will be visible in the information buffer when it pops up. If the message takes more than one line, the popup will stay visible two more second per additional lines.

roster_group_sort Default value: `name`

How to sort the contact list groups. The principles are the same as *roster_sort* (see below).

Available methods are:

- `reverse`: reverse the current sorting
- `name`: sort by group name (alphabetical order)
- `fold`: sort by unfolded/folded
- `connected`: sort by number of connected contacts

- `size`: sort by group size
- `none`: put the “none” group (if any) at the end of the list

roster_show_offline Default value: `false`

Set this to `true` if you want to display the offline contacts too.

roster_sort Default value: `jid:show`

How you want the contacts to be sorted inside the contact list groups. The given methods are used sequentially (from left to right), so the last one is the one on the far right.

Available methods are :

- `reverse`: reverse the current sorting
- `jid`: sort by JID (alphabetical order)
- `show`: sort by show (available/away/xa/...)
- `name`: sort by given name (if no name, then the bare `jid` is used)
- `resource`: sort by resource number
- `online`: sort by online presence (online or not)

Those methods can be arranged however you like, and they have to be separated by colons (“:”). If there are more than 3 or 4 chained sorting methods, your sorting is most likely inefficient.

show_composing_tabs Default value: `direct`

Highlight tabs where the last activity was a “composing” chat state, which means the contact is currently typing.

Possible values are:

- `direct`: highlight only in one-to-one chats (equiv. of private & conversation)
- `private`: highlight only in private chats inside chatrooms
- `conversation`: highlight only in chats with contacts or direct JIDs
- `muc`: highlight only in chatrooms
- `true`: highlight all possible tabs (equiv. of muc & private & conversation)
- `false` or any other value: don’t highlight anything

show_inactive_tabs Default value: `true`

If you want to show all the tabs in the Tab bar, even those with no activity, set to `true`. Else, set to `false`.

show_jid_in_conversations Default value: `true`

If `false`, the JID of the contact will not be displayed in the information window in conversation tags.

show_muc_jid Default value: `false`

If set to `false`, poezio will first display the bookmark name, or if empty the user part of the address (before the @) when displaying the chatroom tab name. So `poezio@muc.poez.io` will get shortened to `poezio` unless this option is set to `true`. This will be used only if `use_tab_nicks` is set to `true`.

show_roster_jids Default value: `true`

Set this to `false` if you want to hide the JIDs in the contact list (and keep only the contact names). If there is no contact name, the JID will still be displayed.

show_roster_subscriptions Default value: `[empty]`

Select the level of display of subscriptions with a char in the contact list.

- `all` to display all subscriptions
- `incomplete` to display *from*, *to* and *none*
- one of *from*, *to*, *none* and *both* to display only that one
- no value or any other value to disable it

show_s2s_errors Default value: `true`

Show s2s errors in the contact list or not.

show_tab_names Default value: `false`

If you want to show the tab name in the bottom Tab bar, set this to `true`.

show_tab_numbers Default value: `true`

If you want to disable the numbers in the bottom Tab bar, set this to `false`. Note that if both [show_tab_names](#) and [show_tab_numbers](#) are set to `false`, the numbers will still be displayed.

show_timestamps Default value: `true`

Whether or not to display a timestamp before each message.

theme Default value: `[empty]`

The name of the theme file (without the `.py` extension) that will be used. The file should be located in the [themes_dir](#) directory.

If the file is not found (or no filename is specified) the default theme will be used instead

themes_dir Default value: `[empty]`

If [themes_dir](#) is not set, themes will be searched for in `$XDG_DATA_HOME/poezio/themes`, i.e. in `~/.local/share/poezio/themes/`. So you should specify the directory you want to use instead.

This directory will be created at startup if it doesn't exist

use_tab_nicks Default value: `true`

The tabs have a name, and a nick, which is, for a contact, its name in the contact list, or for a private conversation, the nickname in the chatroom. Set this to `true` if you want to have them shown instead of the jid of the contact.

user_list_sort Default value: `desc`

If set to `desc`, the chatroom users will be displayed from top to bottom in the list, if set to `asc`, they will be displayed from bottom to top.

vertical_tab_list_size Default value: `20`

Horizontal size of the vertical tab list.

vertical_tab_list_sort Default value: `desc`

If set to `desc`, the tabs will be displayed from top to bottom in the list, if set to `asc`, they will be displayed from bottom to top.

2.1.6 User Interaction

Options that change the behavior of poezio in a non-visual manner.

add_space_after_completion Default value: `true`

Whether or not to add a space after a completion in the middle of the input (not at the start of it)

after_completion Default value: ,

What will be put after the name, when using autocompletion at the beginning of the input. A space will always be added after that

beep_on Default value: highlight private invite disconnect

The terminal can beep on various event. Put the event you want in a list (separated by spaces).

The events can be - `highlight` (when you are highlighted in a chatroom) - `private` (when a new private message is received, from your

contacts or someone from a chatroom)

- `message` (any message from a chatroom)

separate_history Default value: false

If true, the history of inputs of the same nature won't be shared between tabs (as in weechat).

words Default value: [empty]

Personal dictionary of the words you use often, that you want to complete through recent words completion. They must be separated by a colon (:). That completion will work in chatrooms, private conversations, and direct conversations.

2.1.7 Logging

Options related to logging.

load_log Default value: 10

The number of line to preload in a chat buffer when it opens. The lines are loaded from the log files. 0 or a negative value here disable that option.

log_dir Default value: [empty]

If `log_dir` is not set, logs will be saved in `$XDG_DATA_HOME/poezio/logs`, i.e. in `~/.local/share/poezio/logs/`. So, you should specify the directory you want to use instead. This directory will be created if it doesn't exist.

log_errors Default value: true

Logs all the tracebacks and errors of poezio/slixmpp in `log_dir/errors.log` by default. `false` disables this option.

use_log Default value: true

Set to `false` if you don't want to write any message to the disk.

2.1.8 Plugins

This sections references the configuration of the plugin system; for more details, go to the [dedicated page](#).

plugins_autoload Default value: [empty]

Colon-separated list of plugins to load on startup.

plugins_conf_dir Default value: [empty]

If `plugins_conf_dir` is not set, plugin configs will be loaded from `$XDG_CONFIG_HOME/poezio/plugins`. You can specify another directory to use, it will be created if it does not exist.

plugins_dir Default value: [empty]

If `plugins_dir` is not set, plugins will be loaded from the `plugins/` dir of the poezio install directory, then `$XDG_DATA_HOME/poezio/plugins`. You can specify another directory to use. It will be created if it does not exist.

2.1.9 Other

exec_remote Default value: false

If this is set to `true`, poezio will try to send the commands to a FIFO instead of executing them locally. This is to be used in conjunction with `ssh` and the `daemon.py` file. See the [/link](#) documentation for details.

extract_inline_images Default value: true

Some clients send inline images in base64 inside some messages, which results in an useless wall of text. If this option is `true`, then that base64 text will be replaced with a `file://` link to the image file extracted in `tmp_image_dir` or `$XDG_CACHE_HOME/poezio/images` by default, which is usually `~/.cache/poezio/images`

lang Default value: en

The lang some automated entities will use when replying to you.

lazy_resize Default value: true

Defines if all tabs are resized at the same time (if set to `false`) or if they are really resized only when needed (if set to `true`). `true` should be the most comfortable value

max_lines_in_memory Default value: 2048

Configure the number of maximum lines (for each tab) that can be kept in memory. If poezio consumes too much memory, lower these values

max_messages_in_memory Default value: 2048

Configure the number of maximum messages (for each tab) that can be kept in memory. If poezio consumes too much memory, lower these values

remote_fifo_path Default value: ./

The path of the FIFO used to send the commands (see the [exec_remote](#) option). Poezio will try to create a `poezio.fifo` file in this directory.

save_status Default value: true

Save the status automatically in the [status](#) and [status_message](#) options.

send_initial_presence Default value: true

Send initial presence (normal behaviour). If `false`, you will not send nor receive any presence that is not directed (through [/presence](#)) or sent by a chatroom.

tmp_image_dir Default value: [empty]

The directory where poezio will save the images received, if [extract_inline_images](#) is set to `true`. If unset, poezio will default to `$XDG_CACHE_HOME/poezio/images` which is usually `~/.cache/poezio/images`.

2.2 Optional section options

These option can appear in optional sections. These section are named after a JID. These option will apply only for the given JID. For example if an option appears in a section named `[user@example.com]`, it will apply only for the conversations with `user@example.com`.

If an option appears in a section named `[@example.com]`, it will apply for all the conversations with people `@example.com`, except when the option is already defined in a `[user@example.com]` section.

The priority of settings is thus like this: `user@example.com` > `@example.com` > Poezio (more specific to less specific)

Note that some of these options can also appear in the global section, they will be used as a fallback value when no JID-specific option is found.

```
[Poezio]
foo = false
[user@example.com]
foo = true
[@example.com]
bar = false
```

autorejoin Default value: `false`

Set to `true` if you want to automatically rejoin the room when you're kicked or banned.

autorejoin_delay Default value: `5`

Set to the number of seconds before reconnecting after getting kicked or banned. `0`, a negative value, or no value means instant reconnection.

This option only works if `autorejoin` is `true`.

disable_beep Default value: `false`

Disable the beeps triggered by this conversation. Works in chatroom tabs, private messaging tabs, and conversation tabs.

display_activity_notifications Default value: `false`

If set to `true`, notifications about the current activity of your contacts will be displayed in the info buffer as 'Activity' messages.

display_gaming_notifications Default value: `false`

If set to `true`, notifications about the game your are playing will be displayed in the info buffer as 'Gaming' messages.

display_mood_notifications Default value: `false`

If set to `true`, notifications about the mood of your contacts will be displayed in the info buffer as 'Mood' messages.

display_tune_notifications Default value: `false`

If set to `true`, notifications about the music your contacts listen to will be displayed in the info buffer as 'Tune' messages.

display_user_color_in_join_part Default value: `false`

If set to `true`, the color of the nick will be used in chatroom information messages, instead of the default color from the theme.

eval_password Default value: [empty]

A command which execution will retrieve the password from a password manager.

E.g. with secret-tool and the gnome keyring:

```
# Storing (to do beforehand)
secret-tool store --label="My jabber password" xmpp your@jid

# Retrieving (this should be the value of the option)
secret-tool lookup xmpp your@jid
```

Note: This will only be used if the *password* option is empty.

hide_exit_join Default value: -1

Exact same thing than `hide_status_change`, except that it concerns the quit message, and that it will be hidden only if the value is 0. Default setting means: - all quit and join notices will be displayed

hide_status_change Default value: 120

Set a number for this setting. The join AND status-change notices will be displayed according to this number.

-1: the notices will ALWAYS be displayed

0: the notices will NEVER be displayed

n: On any other number, the notices will only be displayed if the user involved has talked since the last n seconds
if the value is incorrect, -1 is assumed Default setting means that status changes won't be displayed unless the user talked in the last 2 minutes

highlight_on Default value: [empty]

A list of words (separated by a colon (:)) that will be highlighted if said by someone on a room.

ignore_private Default value: false

Ignore private messages sent from this room.

load_log Default value: 10

The number of line to preload in a chat buffer when it opens. The lines are loaded from the log files. 0 or a negative value here disable that option.

No value makes poezio fall back to the global value.

notify_messages Default value: true

Only for chatroom tabs: if true the tab will change its color to notify you when a new message is received. You will still be notified of highlights. Set to false if you are not interested in a room non-highlight notifications.

password Default value: [empty]

The password needed to join the room.

private_auto_response Default value: Not in private, please.

The message you want to be sent when someone tries to message you.

self_ping_delay Default value: 0

When this option is set to a positive value n, poezio will send a ping request to its own nick in the chatroom every n seconds of inactivity (whenever no new message or presence is received from the chatroom for more than n seconds). If the chatroom service does not respond with a successful pong within 60 seconds (that is: on

an error of the type “not-allowed” which means the chatroom service doesn’t consider us to be present in that room, or on a timeout which probably means that the service is down), poezio will mark that chatroom as not joined and will try to re-join it. This is useful to know when a chatroom server crashes or becomes unavailable, because there is no mechanism to be informed of that fact in XMPP.

A value of at least 60 seconds is recommended, to avoid sending too many requests.

When set to 0 (the default value), no ping request will be sent.

send_chat_states Default value: `true`

Lets you disable/enable chatstates per-JID. Works in chatroom tabs, private messaging tabs, and normal conversation tabs.

show_useless_separator Default value: `true`

If `false`, the separator at the bottom of a chat room will not be displayed if no one spoke.

use_log Default value: `[empty]`

Use logs for this JID or not. No value will make poezio fall back to the global `use_log` value.

This page is the main page of the documentation for poezio, explaining how to use it and describing its interfaces.

Poezio is composed of tabs which can be of various types. Each tab type has a distinct interface, list of commands and list of key shortcuts, in addition to the global commands and key shortcuts.

3.1 Tab list

There are two ways of showing the open tabs:

3.1.1 Horizontal list

This is the default method.

On all tabs, you get a line showing the the list of all opened tabs. Each tab has a number, each time you open a new tab, it gets the next available number.



3.1.2 Vertical list

On all tabs, you get a pane on the left side of the screen that shows a list of the opened tabs. As stated above, each tab has a number, and each time you open a new tab, it gets the next available number.

This mode is enabled by setting the *enable_vertical_tab_list* option to `true` in the configuration file.

3.1.3 Options for the tab list

The following options are used to configure the behavior of the tab list:



```
0. Roster
1. discussion
2. codingteam
3. gajim
4. archlinux-fr
5. pcinpack
6. wikipedia-fr
7. genshiken
8. prosody
9. jabberfr
10. mcabber
11. test
12. discussion
13. batajelo
14. photo
15. duckduckgo
16. seehaus
17. jdev
18. poezio
19. sleek
20. ins
```

- *enable_vertical_tab_list*
- *vertical_tab_list_size*
- *vertical_tab_list_sort*
- *show_tab_names*
- *show_tab_numbers*
- *show_inactive_tabs*
- *use_tab_nicks*

3.2 Generalities

Global commands

Global shortcuts

The tab numbered **0** is always the **Contact list** tab, the other tabs can be of any type.



The status of a tab is represented by its color:

- **Blue** (tab **0**): an inactive tab of any type, nothing new to see there.
- **Purple** (tab **1**): a *Chatroom tab* with at least one new unread message.

- **Green** (tab 2): a tab of a private conversation (*Private tab* or *Conversation tab*) with a new message to read.
- **Cyan** (tab 3): the current tab.
- **Red** (tab 4): a *Chatroom tab* with at least one new highlight message.

You can go from one tab to another in many ways:

- `Ctrl+n` (next tab) and `Ctrl+p` (previous tab)
- `/win` command
- `/next` and `/prev` commands
- `Alt + a number` (to go to the tab with that number)
- `Alt+j` followed by a two-digits number (same)
- `Alt+e`, this will jump to the next tab with the highest priority. Priority applies in this order: private message > highlight message > normal message.
- `/close` command to close a tab and go back to the previous one

3.3 Contact list tab

Specific commands

Specific shortcuts

Note: The contact list also called a roster in XMPP terms.

This is a unique tab, always numbered **0**. It contains the list of your contacts. You can add (`/add`, `/accept`), remove (`/remove`) and search contacts from there, and you can open a conversation with them (`Enter` key).

Use the **direction arrows** (`↑↓`) to browse the list, the `Space` key to fold or unfold a group or a contact.

1. Area where information messages are displayed.
2. Actual list of contacts. The first level is group, the second is the contacts and the third is the resources of your online contacts.
3. More information about the selected contact.

3.4 Chatroom tab

Specific commands

Specific shortcuts

Chat shortcuts

Note: A chatroom is also called a MUC (for Multi-User-Chat) in XMPP terms.

This tab contains a multi-user conversation.

1. The conversation window, this is where all the messages and events related to the muc will be displayed. It can be scrolled up and down with `PageUp` and `PageDown`.

```

Roster: 3/10 contacts
[.] none (3/5)
[+] bbc-news@xmpp.lobstermonster.org (1)
[+] pubsub-bridge@broadcaster.buddycloud.com (1)
[.] louiz@louiz.org (1)
  louiz@louiz.org/coucou

20:28:25 Welcome to poezio!
20:28:25 Connected to server.
20:28:26 Authentication success.
20:28:26 Your JID is tata@louiz.org/zouzou

louiz@louiz.org (dnd)
Subscription: both

[0]1]
Enter commands with "/". "o": toggle offline show

```

```

1.discussion
8.prosody
16.seehaus
18.poezio
27.support

Documentation available on http://poezio.eu french or english are ok!English-talking
21:49:45 mathieui> no
21:49:50 louiz> ah no
21:49:53 louiz> juts a new commit
21:50:00 gio> one day i will report an err418
21:50:08 louiz> no
21:50:40 mathieui> louiz', cron please
21:50:42 gio> perhaps from my coffee plugin?
21:50:53 --->
( ) joined the
room
21:51:12 louiz> done
23:13:24 ctbot> r5e6838 ~ mathieui
https://dev.louiz.org/project/poezio/browse/diff/32f8ad/5e6838
Truncate everything in the roster if needed.

Group names, resources jids, and bare jid/roster name ...
23:14:05 louiz> plus que 46 minutes
23:14:09 mathieui> tkt
23:14:13 louiz> tkt
23:14:18 mathieui> il reste de la doc, quoi
23:14:25 mathieui> c'est relou de refaire les screens
23:14:48 louiz> oui hein
23:14:55 mathieui> putain, j'ai perdu le screen de coucou_les_copains,
fuck
23:14:58 louiz> --
23:15:02 louiz> J'ferai celles de la 0.8 :o
23:15:04 mathieui> je vais en prendre un ici et flouter les nicks
[poezio@muc.poezio.eu] mathieui (owner, moderator)

```

2. The participant list. Participants are listed by their role first, and then alphabetically. The status of each participant is symbolized using the **color** of the character on the left of its nick. That character also shows the chatstate of each participant:

- |: inactive
- X: composing
- A: active
- p: paused

The roles and affiliations of the participants are symbolized by the char before the nick and its color. The characters define the affiliations, and they mean:

- ~: Owner
- &: Admin
- +: Member
- -: None

And their color define their roles, and they mean:

- **Red** : moderator
- **Blue**: participant
- **Grey**: visitor

The nicks have a fixed color assigned using [XEP-0392](#).

3. Your information in that chatroom (the name of the room, your nick, your role and affiliation).
4. The topic of the room.

You can configure the room (if you have the rights to do it) using the `/configure` command, open a private conversation with someone using the `/query` command, change or view the topic using the `/topic` command...

3.5 Private tab

Specific commands

Chat shortcuts

This is the tab opened with the `/query` command in a *Chatroom tab*, letting you talk in private with a participant of a multi-user chat.

This is just a simple one to one conversation, with a line showing the status, name and chatstate of the participant.

3.6 Conversation tab

Specific commands

Chat shortcuts

A tab opened by interacting with the contact list or `/message`, to talk in private with one of your contacts.

This is also just a simple one to one conversation, with a line showing the status, name and chatstate of the participant, as well as a line at the top showing the status message of the contact. Plugins may add some elements to the status line.

```

21:14:28 louis'> salut, ça serait pour discuter en privé et en
                prendre une screenshot, m'voyez?
21:14:36 mathieu> hm, bof
21:14:44 mathieu> je suis pas totalement d'accord
21:14:52 mathieu> ça me semble un peu scandaleux
21:14:53 louis'> Dommage, je fais, salut
21:14:57 <--- mathieu has left the room
21:14:57 ---> mathieu joined the room
21:15:02 louis'> han
21:15:04 louis'> le fou

```

```

mathieu from room poezio@muc.poezio.eu active
[43]

```

```

PLEASE
21:18:45 louis'> et cette fois ce serait pour avoir une discussion en privé, mais ici
21:18:51 louis'> c'est quasiment pareil, mais bon
21:18:59 louis'> Tu peux te mettre en xa ou truc du genre ?
21:19:43 louis'> ALLO
21:41:03 mathieu@mathieu.net is offline
21:41:40 mathieu@mathieu.net is online
21:47:50 louis'> du coup tu peux répondre maintenant
.....
21:47:56 mathieu> oui
21:48:02 mathieu> salut
21:48:11 louis'> mets toi en xa PLEASE
21:48:40 mathieu> (imo faudrait afficher les changements de status dans les ConversationTab)

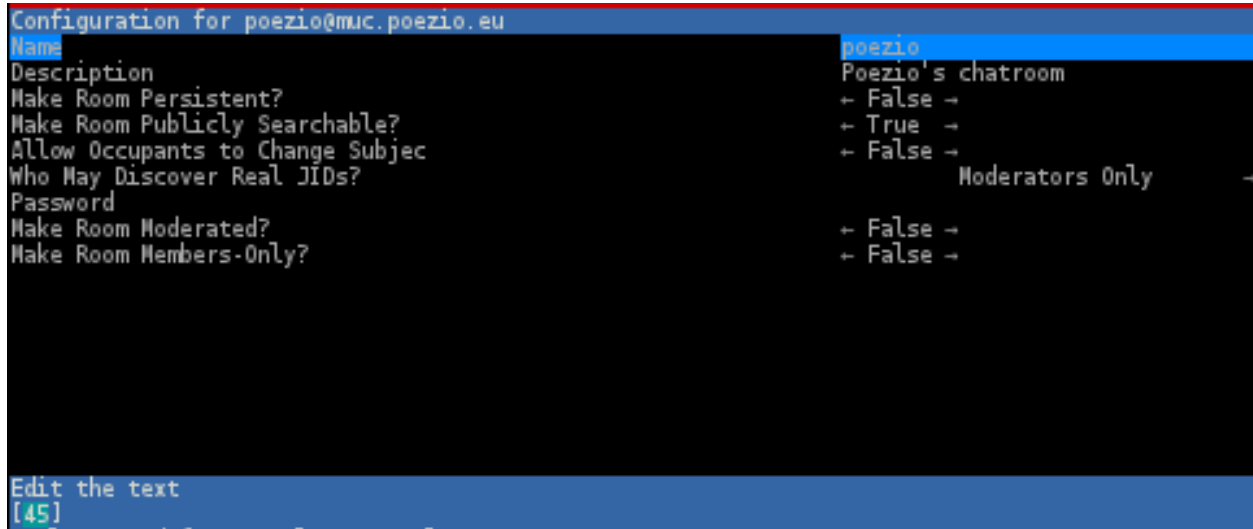
[mathieu@mathieu.net] mathieu [ ] inactive
[13|14|44]

```

3.7 Dataforms tab

Specific shortcuts

This tab lets you view a form received from a remote entity, edit the values and send everything back. It is mostly used to configure chatrooms with the `/configure` command but can actually be used for almost anything.



The screenshot shows a terminal window titled "Configuration for poezio@muc.poezio.eu". The form contains the following fields and values:

Field	Value
Name	poezio
Description	Poezio's chatroom
Make Room Persistent?	+ False -
Make Room Publicly Searchable?	+ True -
Allow Occupants to Change Subject	+ False -
Who May Discover Real JIDs?	Moderators Only
Password	
Make Room Moderated?	+ False -
Make Room Members-Only?	+ False -

At the bottom, there is a blue bar with the text "Edit the text" and a cursor at [45].

Use the Up and Down keys to go from one field to the other, and edit the value using the Space, Left or Right keys, or by entering text.

You can then send the completed form using `Ctrl+y` or cancel using `Ctrl+g`.

3.8 List tab

Specific shortcuts

This tab lists all public rooms on a chatroom service (with the `/list` command). It is currently very limited but will be improved in the future. There currently is no way to search a room.

Use the Up and Down or PageUp and PageDown keys to browse the list, and use Enter or j to join the selected room.

You can sort the rooms by moving the direction arrows (← or →) and pressing Space when you are on the appropriate column.

3.9 Confirm tab

This kind of tab is used to prompt a binary choice to the user due to external events, such as a certificate change:

Or a [XEP-0070](#) validation:

```
Chatroom list on server kikoo.louiz.org
code-part      name
ubuntu-fr-support  ubuntu-fr-support
batajelo        batajelo
chiensdegarde    chiensdegarde
quizz           quizz
noshare         noshare
xirc            xirc
test3           test3
minecraft-fr     minecraft-fr
teequest        teequest
teub            teub
asd             asd
fasc            fasc
ubuntu-fr       ubuntu-fr
sdf             sdf
coucou_les_copains  coucou_les_copains
totoland        totoland

[46]
": join room.
```

```
You need to accept or reject the certificate

WARNING: CERTIFICATE FOR jeproteste.info CHANGED

This can be part of a normal renewal process, but can also mean that an
attacker is performing a man-in-the-middle attack on your connection.
When in doubt, check with your administrator using another channel.

SHA-512 of the old certificate: 1A:C5:37:18:3F:0B:04:8C:80:C1:BF:C3:F3:4
1:5B:CE:0E:C2:E5:DC:FD:6E:94:65:57:99:0D:38:67:2F:53:D7:2E:07:22:72:37:0
6:B9:25:39:4F:EB:E4:DD:27:84:ED:1B:58:1F:2D:59:B2:93:00:4D:AC:8B:FC:DB:5
6:3F:1

SHA-512 of the new certificate: 1A:C5:37:18:3F:0B:04:8C:80:C1:BF:C3:F3:4
1:5B:CE:0E:C2:E5:DC:FD:6E:94:65:57:99:0D:38:67:2F:53:D7:2E:07:22:72:37:0
6:B9:25:39:4F:EB:E4:DD:27:84:ED:1B:58:1F:2D:59:B2:93:00:4D:AC:8B:FC:DB:5
6:3F:11

Accept  Reject

You need to accept or reject the certificate
[0]
Choose with arrow keys and press enter
```

```

An HTTP verification was requested

Someone (maybe you) has requested an identity verification
using method "GET" for the url "koin.fr".

The transaction id is: allotest
And the XMPP address of the verification service is sdf@jeproteste.info/8
4dbe821-342a-4886-96c6-123d83ab3c63.

Accept    Reject

An HTTP verification was requested
[0]1]
Choose with arrow keys and press enter

```

3.10 Bookmarks tab

This tab can be obtained using `/bookmarks`, it is a graphical interface for managing bookmarks. You can edit the bookmark address itself, its password, the storage backend, and the autojoin status. Note that local bookmarks always have autojoin set to True.

room@server/nickname	password	autojoin	storage
allo@chat.jeproteste.info		← True →	local →
example@chat.jeproteste.inf		← True →	← remote
totototo@muc.poez.io		← True →	← remote

```

Remote storage: privatexml
[0]1]
Ctrl+Y: save, Ctrl+G: cancel, ↑↓: change lines, tab: change column, M-a: add bookmark

```


CHAPTER 4

Commands

Commands start with the `/` character and can take a list of any number of arguments, separated by spaces. If an argument should contain a space, you can use the `"` character to surround this argument.

The commands described in this page are shown like this:

```
/command <mandatory argument> [optional argument]
```

You can get the same help as below from inside poezio with the `/help` command.

Note: Use command parameters like this:

- Do not use quotes if they are unnecessary (words without special chars or spaces)
 - If the command takes several arguments, you need to put quotes around arguments containing special chars such as backslashes or quotes
 - If the command always takes only one argument, then do not use quotes even for words containing special chars
-

4.1 Global commands

These commands work in *any* tab.

/activity Usage: `/activity [<general> [specific] [comment]]`

Send your current activity to your contacts (use the completion to cycle through all the general and specific possible activities).

Nothing means “stop broadcasting an activity”.

/bind Usage: `/bind <key> <eq>`

Bind a key to another key or to a “command”. For example, `/bind ^H KEY_UP` makes Control + h behave the same way as the Up key. See the *key bindings documentation page* for more details.

/bookmark Usage: /bookmark [roomname] [/nick] [autojoin] [password]

Bookmark the specified room. This command uses almost the same syntax as /join. Type /help join for syntax examples. Note that when typing /bookmark on its own, the room will be bookmarked with the nickname you're currently using in this room (instead of default_nick). You can specify an optional *autojoin* and *password* if you call it with the full line (/bookmark alone will put the room in autojoin without password). The bookmarks stored with this command are stored on your xmpp server.

/bookmark_local Usage: /bookmark_local [roomname] [/nick]

Bookmark the specified room (you will then auto-join it on each poezio start). This command uses almost the same syntax as /join. Type /help join for syntax examples. Note that when typing /bookmark on its own, the room will be bookmarked with the nickname you're currently using in this room (instead of default_nick). The bookmarks stored with this command will be stored locally. They have priority over the ones stored online.

/bookmarks Usage: /bookmarks

Open a *Bookmarks tab* in order to edit the current boookmarks.

/close Close the tab.

Note: The /close command will work everywhere, except in the Contact list tab, which can't be closed.

/destroy_room Usage: /destroy_room [room JID]

Try to destroy the room given as a parameter, or the current room is not parameter is given and the current tab is a chatroom.

You need to be the owner of a room or a server admin to destroy it.

/exit

/quit Just disconnect from the server and exit poezio.

/gaming Usage: /gaming [<game name> [server address]]

Send your current gaming activity to your contacts.

Nothing means "stop broadcasting a gaming activity".

/help Usage: /help [command]

If called without an argument, this command will list the available commands. If it has a valid command as an argument, this command will show the usage and the help for the given command.

/invitations Show the pending invitations.

/invite Usage: /invite <jid> <room> [reason]

Invite *jid* to *room* with *reason* (if provided).

/join Usage: /join [room_name] [@server] [/nick] [password]

Join the specified room. You can specify a nickname after a slash (/). If no nickname is specified, you will use the default_nick in the configuration file. You can omit the room name: you will then join the room you're looking at (useful if you were kicked). You can also provide a room_name without specifying a server, the server of the room you're currently in will be used. You can also provide a password to join the room.

Examples:

- /join room@server.tld
- /join room@server.tld/John
- /join room2

- `/join /me_again`
- `/join`
- `/join room@server.tld/my_nick password`
- `/join / password`

/last_activity Usage: `/activity <jid>`

Show the last activity of a contact or a server (its uptime, in that case).

/list Usage: `/list [server.tld]`

Get the list of public chatrooms in the specified server (open a [List tab](#))

/load Usage: `/load <plugin name> [<other plugin> ...]`

Load or reload one or several plugins.

/message Usage: `/message <jid> [optional message]`

Open a conversation with the specified JID (event if it is not in our contact list), and send a message to them, if specified.

/mood Usage: `/mood [<mood> [comment]]` Send your current mood to your contacts (use the completion to cycle through all the possible moods).

Nothing means “stop broadcasting a mood”.

/move_tab Usage: `/move_tab <source> <destination>`

Move tab `<source>` to `<destination>`. If the [create_gaps](#) option is true, then it will leave a gap at the `<source>` position, leading to usual behaviour. If `create_gaps` is not enabled, then the tabs will number from 0 to your actual tab number, without gaps (which means their number will change if you close a tab on the left of the list).

A value of `.` for a parameter means the current tab.

/next Go to the next room.

/plugins List the loaded plugins.

/presence Usage: `/presence <jid> [type] [status]`

Send a directed presence to *jid* using *type* and *status* if provided.

/prev Go to the previous room.

/rawxml Usage: `/rawxml <stanza>`

Send a custom XML stanza.

/reload Reload the config. You can achieve the same by sending SIGUSR1 to poezio.

/remove_bookmark Usage: `/remove_bookmark [room_jid]`

Remove the bookmark on *room_jid* or the one on the current tab, if any.

/runkey Usage: `/runkey <key>`

Execute the action defined for *key*. For example, `/runkey KEY_PPAGE` will scroll up, or `/runkey ^N` will go to the next tab.

/self Reminds you of who you are and what your status is.

/server_cycle Usage: `/server_cycle [server.tld] [message]`

Disconnect and reconnect in all the rooms of `server.tld`.

/set Usage: /set [plugin|][section] <option> <value>

Set the value to the option in your configuration file. You can, for example, change your default nickname by doing “/set default_nick toto” or your password with “/set password blabla”. Doing so will write in the main config file, and in the main section ([Poezio]). But you can also write to another section, with /set bindings M-i ^i, to a plugin configuration with /set mpd_client| host main (notice the |, it is mandatory to write in a plugin), or even to another section in a plugin configuration /set plugin|other_section option value. **toggle** can be used as a special value for a boolean option. It just set the option to true if it's currently false, and to false if it's currently true.

/set_default Usage: /set_default [section] <option>

Set the value of an option back to the default. For example, /set_default password will reset the password option.

/status Usage: /status <availability> [status message]

Set your availability and (optionaly) your status message. The <availability> argument is one of “available, chat, away, afk, dnd, busy, xa” and the optional [status] argument will be your status message.’

/theme Usage: /theme [theme_name]

Reload the theme defined in the config file. If *theme_name* is given, this command will act like /set theme *theme_name* then /theme.

/toggle Usage: /toggle <option>

Toggle an option, shortcut for */set <option> toggle*.

/unload Usage: /unload <plugin name> [<other plugin> ...]

Unload one or several plugins.

/version Usage: /version <jid>

Get the software version of the given JID (usually its XMPP client and Operating System).

/win

/w Usage: /win <number or string>

Go to the matching tab. If the argument is a number, it goes to the tab with that number. Otherwise, it goes to the next tab whose name contains the given string.

/xml_tab Open an XML tab.

4.2 Chat tab commands

These commands will work in any conversation tab (MultiUserChat, Private, or Conversation tabs).

/clear Clear the current buffer.

/correct Usage: /correct <corrected message>

Replace the content of the last sent message with *corrected message*.

/say Usage: /say <message>

Just send the message (only useful if you want your message to begin with a /). Note that you can also send message starting with a / by starting it with //.

/xhtml Usage: /xhtml <custom xhtml>

Send a custom xhtml message to the current tab.

4.3 MultiUserChat tab commands

/affiliation Usage: `/affiliation <nick> <affiliation>`

Sets the affiliation of the participant designated by **nick** to the given **affiliation** (can be one of owner, admin, member, outcast and none).

/clear [Chatroom version] Usage: `/clear`

Clear the messages buffer.

/color Usage: `/color <nick> <color>`

Assign a color to the given nick. The nick and all its alias (nicks are considered identical if they only differ by the presence of one or more `_` character at the beginning or the end. For example `_Foo` and `Foo__` are considered aliases of the nick `Foo`) will then always have the specified color, in all MultiUserChat tabs. This is true whatever the value of **deterministic_nick_colors** is.

Use the completion to get a list of all the available color values. Use the special color **unset** to remove the attributed color on this nick. You can also use **random** to attribute a random color.

/configure Configure the current room through a form (Open a [Dataforms tab](#)).

/cycle Usage: `/cycle [message]`

Leave the current room and rejoin it immediately. You can specify an optional quit message.

/ignore Usage: `/ignore <nickname>`

Ignore a specified nickname.

/info Usage: `/info <nickname>`

Display some information about the user in the room: his/her role, affiliation, status, and status message.

/invite [Chatroom version] Usage: `/invite <jid> [reason]`

Invite *jid* to this room with *reason* (if provided).

/kick Usage: `/kick <nick> [reason]`

Kick the user with the specified nickname. You can also give an optional reason.

/names Get the list of the users in the room, their number, and the list of the people assuming different roles.

/nick Usage: `/nick <nickname>`

Change your nickname in the current room.

/part Usage: `/part [message]`

Disconnect you from a room. You can specify an optional message.

/query Usage: `/query <nick> [message]`

Open a [Private tab](#) with `<nick>`. This nick has to be present in the room you're currently in. If you specified a message after the nickname, it will be sent to this user.

/recolor Usage: `/recolor [random]`

Re-assign a color to all the participants in the current room, based on the last time they talked. Use this if the participants currently talking have too many identical colors. If a random argument is given, the participants will be shuffled before they are assigned a color.

/role Usage: /affiliation <nick> <role>

Sets the role of the participant designated by **nick** to the given **role** (can be one of moderator, participant, visitor and none).

/topic Usage: /topic [subject]

Change the subject of the room.

Using the auto-completion of this command writes the current topic in the input, to help the user make a small change to the topic without having to rewrite it all by hand.

If no subject is specified as an argument, the current topic is displayed, unchanged.

/unignore Usage: /unignore <nickname>

Remove the specified nickname from the ignore list.

/version Usage: /version <nickname or jid>

Get the software version of the given nick in room or the given jid (usually its XMPP client and Operating System).

4.4 Private tab commands

/info Display some info about this user in the MultiUserChat.

/unquery Close the tab.

/version Get the software version of the current interlocutor (usually its XMPP client and Operating System).

4.5 Normal Conversation tab commands

/info Display the status of this contact.

/unquery Close the tab.

/version Get the software version of the current interlocutor (usually its XMPP client and Operating System).

4.6 Contact list tab commands

/accept Usage: /accept [jid]

Authorize the provided JID (or the selected contact in the contact list) to see your presence.

/add Usage: /add <jid>

Add the specified JID to your contact list and authorize them to see your presence. If they accepts you, the subscription will be mutual (and if they don't, you can still /remove them).

/deny Usage: /deny [jid]

Prevent the provided JID (or the selected contact in the contact list) from seeing your presence.

/groupadd Usage: “/groupadd (<jid> <group>|<group>)

Add the given JID to the given group (if the group does not exist, it will be created). If no jid is provided, the currently selected item on the contact list (resource or JID) will be used.

/groupmove Usage: /groupmove <jid> <old_group> <new_group>

Move the given JID from one group to another (the JID has to be in the first group, and the new group may not exist).

/groupremove Usage: /groupremove <jid> <group>

Remove the given JID from the given group (if the group is empty after that, it will get deleted).

/name Usage: /name <jid> <name>

Set the given JID's name in your contact list.

/password Usage: /password <password>

Change your password.

/reconnect Disconnect from the remote server (if connected) and then connect to it again.

/remove Usage: /remove [jid]

Remove the specified JID from your contact list. This will unsubscribe you from its presence, cancel its subscription to yours, and remove the item from your contact list.

Note: The following commands only exist if your server announces it supports them.

/block Usage: /block [jid]

Block the following JID using simple blocking. You will not receive any of his messages and won't be able to send some to him either.

/cert_add Usage: /cert_add <name> <certificate file> [management]

Add a client X.509 certificate to the list of the certificates which grant access to your account. It must have a unique name the file must be in PEM format. [management] is true by default and specifies if the clients connecting with this particular certificate will be able to manage the list of authorized certificates.

/cert_disable Usage: /cert_disable <name>

Remove a certificate from the authorized list. Clients currently connected with the certificate identified by <name> will however **not** be disconnected.

/cert_fetch Usage: /cert_fetch <name> <path>

Download the public key of the authorized certificate identified by name from the XMPP server, and store it in <path>.

/cert_revoke Usage: /cert_revoke <name>

Remove a certificate from the authorized list. Clients currently connected with the certificate identified by <name> **will** be disconnected.

/certs List the remotely stored X.509 certificated allowed to connect to your accounts.

/list_blocks List the blocked JIDs.

/unblock Usage: /unblock [jid]

Unblock a previously blocked JID using simple blocking. You will be able to send and receive messages from him again.

Note: The following commands do not comply with any XEP or whatever, but they can still prove useful when you are migrating to an other JID.

/export Usage: /export [/path/to/file]

Export your contacts into /path/to/file if specified, or \$HOME/poezio_contacts if not.

/import Usage: /import [/path/to/file]

Import your contacts from /path/to/file if specified, or \$HOME/poezio_contacts if not.

4.7 XML tab commands

/clear [XML tab version] Clear the current buffer.

/dump Usage: /dump <filename>

Write the content of the XML buffer into a file.

/filter_from Usage: /filter_from <jid>

Filter by JID for from attribute.

/filter_id Usage: /filter_id <id>

Filter by stanza id attribute.

/filter_jid Usage: /filter_jid <jid>

Filter by JID, both to and from.

/filter_reset Reset the stanza filters.

/filter_to Usage: /filter_to <jid>

Filter by JID for the to attribute.

/filter_xmlmask Usage: /filter_xmlmask <xml mask>

Filter using an XML mask

/filter_xpath Usage: /filter_xpath <xpath>

Filter with an XPath selector.

This file describes the default keys of poezio and explains how to configure them.

By default, most keys manipulating the input (where you type your messages and commands) behave like emacs does.

Note: Keys are case sensitive. Ctrl-X is not the same than Ctrl-x

5.1 Key bindings listing

Some key bindings are available only in some tabs, others are global.

5.1.1 Global keys

These keys work in **any** tab.

Ctrl-p or **F5**: Go to the previous tab.

Ctrl-n or **F6**: Go to the next tab.

Alt-number: Go to the tab with that number.

Alt-j: Waits for you to type a two-digits number. Go to tab number xx.

Alt-e: Go to the tab with a higher priority (private message > highlight > message > non-empty input).

Alt-z: Go to the previously selected tab.

Alt-r: Go to the contact list tab.

F4: Toggle the left pane.

F7: Shrink the information buffer.

F8: Grow the information buffer.

Ctrl-I: Refresh the screen.

Alt-D: Scroll the information buffer up.

Alt-C: Scroll the information buffer down.

5.1.2 Input keys

These keys concern only the inputs.

NOTE: The clipboard is common to all inputs. This lets you cut a text from one input to paste it into an other one.

Ctrl-a: Move the cursor to the beginning of line.

Ctrl-e: Move the cursor to the end of line.

Ctrl-u: Delete the text from the start of the input until the cursor and save it to the clipboard.

Ctrl-k: Delete the text from the cursor until the end of the input and save it to the clipboard.

Ctrl-y: Insert the content of the clipboard at the cursor position.

Ctrl-Enter or **Ctrl-j:** Insert a line break. Since the input is only one line, the line break is represented by the character `|` in it but will be sent as the real `\n` character.

Alt-k: Escape the next key pressed. For example if you press Alt-k, followed by Ctrl-q, this will enter “^Q” into the text input. This is useful for example in conjunction with the bind command, to help you know how to bind something to a key combination without having to remember how to write them by hand.

5.1.3 Chat tab input keys

These keys work in any conversation tab (MultiUserChat, Private or Conversation tabs).

Key Up: Use the previous message from the message history.

Key Down: Use the next message from the message history.

Page Up: Scroll up in the conversation by x lines, where x is the height of the conversation window - 1.

Page Down: Like Page Up, but down.

Ctrl-b: Go one line up in the buffer.

Ctrl-f: Go one line down in the buffer.

Ctrl-s: Go half a screen up in the buffer.

Ctrl-x: Go half a screen down in the buffer.

Alt-/: Complete what you're typing using the “recent” words from the current conversation, if any.

Alt-v: Move the separator at the bottom of the tab.

Alt-h: Scroll to the separator, if there is one.

Ctrl-c: Insert xhtml formatting.

You have to press Ctrl-c then a character listed below:

- 1: Red
- 2: Green
- 3: Yellow/Orange
- 4: Blue

- 5: Pink
- 6: Turquoise
- b: Bold
- u: Underlined
- o: Stop formatting

5.1.4 MultiUserChat tab input keys

These keys work only in the MultiUserChat tab.

Alt-u: Scroll the user list down.

Alt-y: Scroll the user list up.

Alt-p: Scroll to the previous highlight.

Alt-n: Scroll to the next highlight.

tabulation: Complete a nick.

5.1.5 MultiUserChat List tab input keys

These keys work only in the MultiUserChat List tab (obtained with */list*).

Up: Go up one row.

Down: Go down one row.

j: Join the MultiUserChat currently selected.

J: Join the MultiUserChat currently selected, without giving focus to its tab.

Ctrl-M: Join the MultiUserChat currently selected (same as **j**).

PageUp: Scroll a page of chats up.

PageDown: Scroll a page of chats down.

5.1.6 Contact list tab input keys

These keys work only in the Contact list tab (the tab number 0).

/: Open a prompt for commands.

s: Start a search on the contacts.

S: Start a (slow) search with approximation on the contacts.

Alt-u: Move the cursor to the next group.

Alt-y: Move the cursor to the previous group.

Ctrl-c: Cancel the input (search or command)

Enter on a contact/resource: open a chat tab with this contact/resource

Enter on a group: fold/unfold that group

Up: Move the cursor down one contact.

Down: Move the cursor up one contact.

PageUp: Scroll a page of contacts up.

PageDown: Scroll a page of contacts down.

Note: The following will not work if you can still write things in the input (meaning you previously typed `s` or `/`)

Space: Fold/Unfold the current item.

o: Show the offline contacts.

During a search

Enter: end the search while keeping the selected contact under the cursor (tip: press **Enter** a second time to open a chat window)

5.1.7 Data Forms tab keys

Ctrl+y: Validate the form, send it and close the tab.

Ctrl+g: Cancel that form (do not send your changes) and close the tab.

Up: Select the next field.

Down: Select the previous field.

Right/Left: Switch between possible values, in a **jid-multi**, **list-multi**, **list-single** or **text-multi** field.

Space: Select that option

5.1.8 XML tab input keys

These keys only work in the XML tab (obtained with `/xml_tab`)

Ctrl+k: Freeze or un-freeze the display in order to have a clear view of the stanzas.

5.2 Key configuration

Bindings are keyboard shortcut aliases. You can use them to define your own keys to replace the default ones. where $\wedge x$ means *Control* + *x* and $M-x$ means *Alt* + *x*

To know exactly what the code of a key is, just run

```
python3 poezio/keyboard.py
```

And enter any key.

Turn Alt-i into a tab key (completion, etc):

```
M-i = ^I
```

5.3 Actions

5.3.1 Mapping actions on keys

One may want to add keyboard shortcuts on actions that were not mapped already in poezio. To this effect, you can map the keys on actions using the *Key configuration* seen in the previous section.

The actions are pseudo-keystrokes, and have to be treated the same way. They all begin with an underscore to prevent any possible collision with things already defined.

5.3.2 Actions list

Note: Even if some of these actions are labelled as similar to other keystrokes, remapping the keystrokes will not remap the actions defined here.

`_bookmark`

Bookmarks the current room.

Similar to */bookmark*.

`_bookmark_local` Bookmarks the current room, locally.

Similar to */bookmark_local*

`_close_tab`: Closes the current tab.

This is the same as */close*. The first tab (the contact list) can not be closed.

`_disconnect`: Disconnects poezio from the server.

`_quit`: Exits poezio.

Similar to */quit*.

`_reconnect`: Disconnects then reconnects poezio, if possible.

This is similar to */reconnect*.

`_redraw_screen`: Redraws the screen.

This isn't normally useful, similar to Ctrl-l.

`_reload_theme`: Reloads the theme.

Similar to */theme*.

`_remove_bookmark`: Removes the bookmark on the current room.

Similar to */remove_bookmark*.

`_room_left`: Goes to the room on the left.

Similar to the default Ctrl-p action.

`_room_right`: Goes to the room on the right.

Similar to the default Ctrl-n action.

`_show_roster`: Goes to the contact list

Similar to Alt-r action.

_scroll_down: Scrolls down in the current buffer.

Similar to PAGEDOWN.

_scroll_up: Scrolls up in the current buffer.

Similar to PAGEUP.

_scroll_info_down: Scrolls down in the info buffer.

Similar to Alt-c.

_scroll_info_up: Scrolls up in the info buffer.

Similar to Alt-d.

_server_cycle: Cycles in the current chatroom server.

Similar to */server_cycle* in a chatroom. If you are not in a chatroom, you will get an error.

_show_bookmarks: Shows the current bookmarks.

Similar to */bookmarks*.

_show_important_room: Goes to the most important room.

Similar to Alt-e.

_show_invitations: Shows all the pending chatroom invitations.

Similar to */invitations*.

_show_plugins: Shows the currently loaded plugins.

Similar to */plugins*.

_show_xmltab: Opens an XML tab.

Similar to */xml_tab*.

_toggle_pane: Toggles the left pane.

Similar to F4.

5.3.3 Status actions

_available: Sets the status to *available*.

Similar to */status available*.

_away: Sets the status to *away*.

Similar to */status away*.

_chat: Sets the status to *chat*.

Similar to */status chat*.

_dnd: Sets the status to *dnd*.

Similar to */status dnd*.

_xa: Sets the status to *xa*.

Similar to */status xa*.

5.3.4 Command execution

With that kind of actions, you can also execute arbitrary commands, with the `_exc_` keyword.

You only have to prefix your command line with `_exc_`, and without the `/`.

/kick Partauche bound on Ctrl-w:

```
^W = _exc_kick Partauche
```

That key binding will only work in the tabs defining the command (here, the chatroom tab), and will show an error message in the others.

5.3.5 Examples

Config with user-defined actions

```
[bindings]
^W = _close_tab
M-x = _show_xmltab
M-i = _show_important_room
M-p = _toggle_pane
```

Config with commands mapped

```
[bindings]
M-c = _exc_configure
^Q = _exc_part RAGE QUIT
^J = _exc_join
^F = _exc_load figlet
^R = _exc_load rainbow
^S = _exc_say lollllllllllllll
```


Contents:

6.1 Message Carbons

Starting from poezio 0.8, poezio now supports [message carbons](#).

Message carbons are useful to duplicate chat messages between several connected devices, so that when you switch from one to another, you don't lose any message you sent or received.

To enable it, you only have to set the [enable_carbons](#) option to `true`.

Note: This feature only duplicates *chat* messages (direct conversations between two JIDs), not groupchat messages or private conversations in groupchats.

6.2 Using client certificates to login

Passwordless authentication is possible in XMPP through the use of mechanisms such as [SASL External](#). This mechanism has to be supported by both the client and the server. This page does not cover the server setup, but prosody has a [mod_client_certs](#) module which can perform this kind of authentication, and also helps you create a self-signed certificate.

6.2.1 Poezio configuration

If you created a certificate using the above link, you should have at least two files, a `.crt` (public key in PEM format) and a `.key` (private key in PEM format).

You only have to store the files wherever you want and set [keyfile](#) with the path to the private key (`.key`), and [certfile](#) with the path to the public key (`.crt`).

6.2.2 Authorizing your keys

Now your poezio is setup to try to use client certificates at each connection. However, you still need to inform your XMPP server that you want to allow those keys to access your account.

This is done through `/cert_add`. Once you have added your certificate, you can try to connect without a password by commenting the option.

Note: The `/cert_add` command and the others are only available if your server supports them.

6.2.3 Next

Now that this is setup, you might want to use `/certs` to list the keys currently known by your XMPP server, `/cert_revoke` or `/cert_disable` to remove them, and `/cert_fetch` to retrieve a public key.

6.3 Message Correction

Poezio implements the [XEP-0308](#) which allows the correction of the last message sent.

The corrections are signalled with a number append to the nick of the user, in a different color.

```
23:44:19 robert9> allo allo
```

The **9** here represents the number of times this message has been corrected.

You can show the revisions of a message by loading the *Display corrections* plugin, and you can correct your own messages with the `/correct` command.

Note: Please do not abuse of this feature, as it will simply be displayed as another plain message in the clients that do not support correction.

6.4 Personal Events

Starting from poezio 0.8, poezio now supports [user mood](#), [user activity](#), [user tune](#), and [user gaming](#).

Those extensions are standardized ways to broadcast information that might be useful to your contacts (they will receive those information only if they have indicated their interest in them).

The events are also shown in the contact list, next to the contact line:

```
[ - ] Test (1/1)  
[+] A Contact (1)AMG
```

On the above figure, A stands for Activity, M for Mood, and G for Gaming.

The details are shown with notifications if configured, and in the detailed contact information window, on the bottom left in the contact list tab.

You can see below the list of the related options and commands.

```
Subscription: both
Status: dfdsdfs
Mood: Confident
Activity: Exercising/Other
```

Note: All the *display_foo_notifications* options can be tab-specific, so you can display those notifications only for a specific contact, or the other way around.

6.4.1 User Mood

Options:

- *enable_user_mood*
- *display_mood_notifications*

Commands:

- */mood*

6.4.2 User Activity

Options:

- *enable_user_activity*
- *display_activity_notifications*

Commands:

- */activity*

6.4.3 User Gaming

Options:

- *enable_user_gaming*
- *display_gaming_notifications*

Commands:

- */gaming*

6.4.4 User Tune

Options:

- *enable_user_tune*
- *display_tune_notifications*

Note: There is no `/tune` command because it wouldn't be really useful. There was originally a way to broadcast the current tune with the `mpd` plugin, but it was no good. You should use an external script to do that, or use a player that has the feature.

6.5 Installing python 3.5 as a user

6.5.1 Building your own python 3

- Go to the [python download page](#)
- Select the “Latest Python 3 Release”
- Download a tarball and extract it
- Run `./configure && make` (takes only a few minutes even on old CPUs)
- Edit the `poezio launch.sh` script to make it call your user-compiled python binary

6.5.2 Pyenv (x86/x86_64 only)

[Pyenv](#) is a useful script that allows you to install several python versions in your user directory, and lets you manage which one you want depending on the directory you are in. It is therefore useful for people who are on distributions not providing the latest stable version, such as Debian or CentOS.

You can follow the step-by-step [installation tutorial](#) on github that will help you install it to your home directory (on step 5, you should use 3.7.0 which is the latest python release at the time of writing this page); or you can use the [automated installer](#) and use `pyenv install 3.7.0` thereafter.

Then you only need to add a `.python-version` file containing `3.7.0` in your `poezio` directory to make the python version in that directory default to the python 3.7.0 installed with `pyenv`.

6.5.3 Other

[pythonz](#) allows the same kind of version management as `pyenv`, but builds from source instead of fetching precompiled binaries, so it allows more control over what is going on.

6.6 Using several accounts

Poezio does not support multi-accounts, and we do not plan to do so in a foreseeable future. However, you can run several `poezio` instances (e.g. with `tmux` or `screen`) to have similar fonctionnality.

You can specify a different configuration file than the default with:

```
./launch.sh -f separate_config.cfg
```

The relevant options for a separate config are the following:

- `plugins_dir`: A different directory for the plugin sources (not `_that_` useful)
- `log_dir`: A different directory for logs
- `plugins_conf_dir`: A different directory for plugin configurations

Those options are detailed in the *configuration page*.

6.7 TLS in poezio

6.7.1 Security of the connection

Enabling or disabling TLS

Starting from version 0.8, poezio is configured to reject unencrypted connections by default, in accordance to the [TLS manifesto](#). Users can still allow unencrypted connections by setting the *force_encryption* option to false.

If you cannot connect to your server, maybe it does not allow encrypted connections, in which case you should reconfigure it if it is yours, or contact your admin to let him know he should try to protect your privacy and credentials, at least a little.

Ciphers

From the version 0.8, poezio offers the possibility to define your own set of ciphers.

You can set this with the *ciphers* option, the default for poezio being `HIGH+kEDH:HIGH+kEECDH:HIGH:!PSK:!SRP:!3DES:!aNULL`. You can check what ciphers are enabled by that list by running the command `openssl ciphers -v 'cipher list'`. The default list prioritizes [Forward Secrecy](#) and does not have any cipher suite providing less than 128 bits of security.

You should change this if you either cannot connect to your server (but in this case, you should notify the administrator that his XMPP server configuration is probably not great), or if you want to be even more restrictive (only allowing 256 bits of security *and* forward secrecy, for example).

For example, gmail.com (and subsequent XMPP services) only support RC4-MD5 and RC4-SHA, so you will want to set the option to RC4 (or the default with `:RC4` appended, just in case they upgrade their service, though that is very unlikely). Please consider moving to a better XMPP service provider.

6.7.2 Certificate validation

Starting from version 0.7.5, poezio offers some options to check the validity of a X.509 certificate.

TOFU

The default handling method is the [TOFU/TUFU](#) method. At your first connection, poezio will save the hash of the certificate received, and will compare the received one and the first one for the next connections.

If you are paranoid (or run poezio for the first time in an unsafe environment), you can set the *certificate* value of your config file yourself (the hash, colon-separated).

If the certificate is not the same, poezio will open a *Confirm tab* and wait for confirmation:

If you refuse, you will be disconnected.

```
You need to accept or reject the certificate

WARNING: CERTIFICATE FOR jeproteste.info CHANGED

This can be part of a normal renewal process, but can also mean that an
attacker is performing a man-in-the-middle attack on your connection.
When in doubt, check with your administrator using another channel.

SHA-512 of the old certificate: 1A:C5:37:18:3F:0B:04:8C:80:C1:BF:C3:F3:4
1:5B:CE:0E:C2:E5:DC:FD:6E:94:65:57:99:0D:38:67:2F:53:D7:2E:07:22:72:37:0
6:B9:25:39:4F:EB:E4:DD:27:84:ED:1B:58:1F:2D:59:B2:93:00:4D:AC:8B:FC:DB:5
6:3F:1

SHA-512 of the new certificate: 1A:C5:37:18:3F:0B:04:8C:80:C1:BF:C3:F3:4
1:5B:CE:0E:C2:E5:DC:FD:6E:94:65:57:99:0D:38:67:2F:53:D7:2E:07:22:72:37:0
6:B9:25:39:4F:EB:E4:DD:27:84:ED:1B:58:1F:2D:59:B2:93:00:4D:AC:8B:FC:DB:5
6:3F:11

Accept      Reject

You need to accept or reject the certificate
[0|1]
Choose with arrow keys and press enter
```

CA-Based

If you are connecting to a large server that has several front-facing endpoints, you might be bothered by having to validate the change each time, and you may want to check only if it the same authority delivered the certificate.

You can then set the `ca_cert_path` option to the path of a file containing the validation chain in [PEM format](#) ; those certificates are usually in `/usr/share/ca-certificates/` but it may vary depending of your distribution.

If the authority does not match when connecting, you should be disconnected.

None

If you do not want to bother with certificate validation at all (which can be the case when you run poezio on the same computer as your jabber server), you can set the `ignore_certificate` value to true, and let the `ca_cert_path` option empty (or even remove it).

Warning: Only do this if you know what you are doing, or you will be open to Man in The Middle attacks!

6.8 Troubleshooting

6.8.1 I cannot connect.

1. Check that you are still connected to the internet.
2. Double-check your credentials.
3. Check the *security settings*, maybe your server does not support encryption, or only with weak parameters (like gmail).
4. Maybe your DNS are wrong, try setting the *custom_host* option with the server IP.
5. Overzealous firewall?
6. Running poezio with -d file.txt (debug mode) might reveal your issues.
7. Come see us from the [web client](#) to discuss your issues further.

6.8.2 The outline of poezio is not displayed and unicode characters are broken

We believe we (or unrelated people) have reported the bug of python3 compiled against the wrong ncurses to [every significant distribution out there](#), but if there is still one with it, please go ahead and report it.

6.8.3 Poezio tracebacks with weird encoding errors

Please check your locale for utf-8 compatibility.

6.8.4 Python is too heavy

We know. It's too late to change that. If you are running your XMPP client on a toaster, please try [mcabber](#).

6.8.5 Other issues

Some things may appear in `$XDG_DATA_HOME/poezio/logs/errors.log`. (or a user-defined *log_dir*/errors.log)

This page is an attempt at providing first aid to new users, who must first follow the [Install Guide](#) to get a working poezio install.

Reading the more detailed [Usage page](#) is recommended to get a deeper understanding of poezio.

7.1 Anonymous usage

If you run poezio right after installing, you will get connected to the default anonymous server, which allows you to join rooms, and talk to people.

7.1.1 Joining rooms

The [/join command](#) allows you to join a chatroom and start talking to people right away. It opens a new [Chatroom tab](#).

7.1.2 Talking to people

You can use the [/message](#) command if you know the address of people you want to talk to. This will open a [Conversation tab](#).

7.2 Normal usage

In order to use an account, you have to edit the [Configuration](#) first, to set the account address and password (optionally). Sadly, poezio doesn't allow account creation yet, so if you don't have an account you will have to either use another client like [gajim](#) to create your account, or stay in anonymous mode.

After obtaining an account and setting the [jid](#) config option to the right value, you should go through the configuration file to get an overview of the different [configuration options](#) available. If you don't set the value of the [password](#) option, you will be prompted to enter it on startup.

7.2.1 Joining rooms

Just as in the anonymous mode, the */join command* allows you to join a chatroom and start talking to people right away. It opens a new *Chatroom tab*.

7.2.2 Talking to people

Just as in the anonymous mode, you can use the */message* command if you know the address of people you want to talk to. This will open a *Conversation tab*.

7.2.3 Adding people

However, one of the benefits of having an account is to have contacts, see when they are online and offline, see their activity, mood, etc. To this end, you should add the people you know to your *contact list*.

The *Contact list tab* is the tab numbered 0, and the only one which is always open. To add people, use */add*, to accept a contact request use */accept*.

7.2.4 Using end-to-end encryption

To use OTR end-to-end encryption, you have to *enable* the *OTR plugin*. The plugin requires python-potr for python3, so make sure you have it installed first.

After that, you can enable the OTR plugin with */load otr*. Further usage is discussed in the *plugin documentation*.

7.2.5 Exiting poezio

Use the */exit* command to quit poezio.

Themes

This page describes how themes work in poezio and how to create or modify one.

A theme contains color attributes and character definitions. Poezio can display up to **256** colors if your terminal supports it. Most of the time, if it doesn't work, that's because the **\$TERM** environment variable is wrong. For example with tmux or screen, set it to **screen-256color**, in **xterm**, set it to **xterm-256color**, etc.

If your terminal doesn't have 256 colors, only 8 colors will be available, and poezio will replace the colors by one of the 8 values available. Thus, some theme files may not work properly if you only have 8 colors, for example light gray on dark gray may be converted to black on black, making the text impossible to read).

Note: The default theme should work properly in any case. If not, that's a bug.

A theme file is a python file (with the .py extension) containing a class, inheriting the *theming.Theme* class defined into the *theming* poezio module.

To check how many colors your current terminal/\$TERM supports, do:

```
tput colors
```

8.1 Create a theme

To create a theme named foo, create a file named foo.py into the theme directory (by default it's ~/ .local/share/poezio/themes/) and add:

```
import theming

class FooTheme(theming.Theme):
    # Define here colors for that theme
theme = FooTheme()
```

To define a *color pair* and assign it to the *COLOR_NAME* option, just do

```
class FooTheme(theming.Theme):
    COLOR_NAME = (fg_color, bg_color, opt_attr)
```

You do not have to define all the *Available options*, you can decide that your theme will only change some options, the other one will just have the default value (from the default theme).

8.1.1 Colors and attributes

A color pair defines how the text will be displayed on the screen. It has a **foreground color** (`fg_color`), a **background color** (`bg_color`) and an **optional attribute** (`opt_attr`).

Colors

A color is a number between -1 and 255 . If it is -1 , this is the default color defined by your terminal (for example if your terminal displays white text on black by default, a `fg_color` of -1 is white, and a `bg_color` of -1 is black). If it's between 0 and 256 it represents one of the colors on this image:

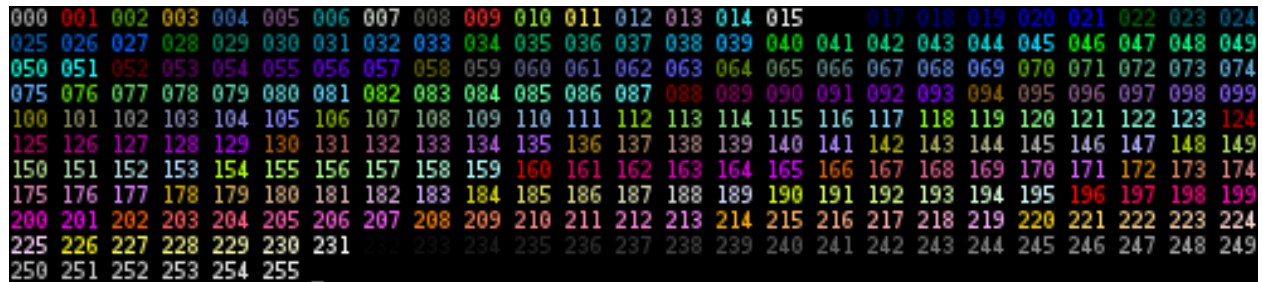


Fig. 1: The list of all 256 colors

Attributes

An attribute is a python string (so, it has to be surrounded by simple or double quotes). It can be one of the following:

- `'b'`: bold text
- `'u'`: underlined text

8.2 Use a theme

To use a theme, just define the *theme* option into the *configuration file* to the name of the theme you want to use. If that theme is not found, the default theme will be used instead.

Note that the default theme is defined directly into poezio's source code, and not in a theme file.

8.3 Change the theme directory

To change the default theme directory (`~/local/share/poezio/themes/` by default), you have to change the *themes_dir* option in the *configuration file* to the directory that contains your theme files.

8.4 Available options

Warning: This section is not complete.

All available options can be found into the default theme, which is into the **theming.py** file from the poezio's source code.

class poezio.theming.Theme

The theme class, from which all themes should inherit. All of the following values can be replaced in subclasses, in order to create a new theme.

Do not edit this file if you want to change the theme to suit your needs. Create a new theme and share it if you think it can be useful for others.

Starting from the 0.7.5 version, poezio supports plugins. Here is a quick howto and a plugin index.

9.1 Setting up plugins

Poezio seeks the plugins in the `~/.local/share/poezio/plugins/` dir (more generally, the `$XDG_DATA_HOME/poezio/plugins/` dir), but that can be changed by setting the *plugins_dir* option to the directory where you want to put your plugins.

By default, poezio will also seek the plugins in `./plugins`, in the source directory, in order to always load the latest versions. You should put a plugin in `$XDG_DATA_HOME/poezio/plugins` only if you have a custom version (that will override the one in `./plugins`), or if it is a plugin you made.

9.2 Plugin autoload

Use the *plugins_autoload* option to select which plugins should be loaded on startup. The value is a list of plugin names separated by colons, e.g.

```
plugins_autoload = tell:exec
```

9.3 Manual plugin load

Plugins can of course be loaded with the command */load* and unloaded with the command */unload*.

9.4 Plugin configuration

Most plugins will manage their configuration internally, and you do not (and should not) have to edit it, but some (e.g. `mpd_client`) require manual editing (the `/set` command can be used, but it is not pleasant to set multiple values with it).

The plugin configuration directory is located in `~/.config/poezio/plugins/` (or `$XDG_CONFIG_HOME/poezio/plugins/`) and the file related to a specific plugin is named `plugin_name.cfg`. The configuration options should usually be inside a section named after the plugin (sections are delimited with `[]`).

```
[plugin_name]
key = value
other_key = other_value
```

9.5 Plugin index

Admin *Documentation*

Creates convenient aliases for chatroom administration.

Alias *Documentation*

Allows you to create your own aliases.

Amsg *Documentation*

Allows a message to be broadcasted on all the rooms you are in. Caution: do not overuse.

Autocorrect *Documentation*

Add new ways to correct messages.

Close all *Documentation*

Close all tabs except chatrooms and the contact list.

CSI *Documentation*

Set the client state indication manually.

Cyber *Documentation*

Add a cybertouch to your messages.

Day Change *Documentation*

Logs the day change inside the buffers, to keep track of the days when backlogging.

Dice *Documentation*

Roll one or several dice using message corrections.

Disco *Documentation*

Add a `/disco` command to display the `disco#info` of a JID.

Display corrections *Documentation*

Lists old versions of a corrected message.

Double *Documentation*

Double the first word of each sentence.

Embed *Documentation*

Send an URL annotating it as embedded.

Exec *Documentation*

Runs a system command and optionally sends the output as a message.

Figlet *Documentation*

Ascii-art writing (requires the `figlet` package on your system).

IQ Show *Documentation*

Shows the received IQs, for debugging purposes.

IRC *Documentation*

Manage IRC gateways with `biboumi` more easily

Link *Documentation*

Opens links in a web browser, locally or remotely using a FIFO and SSH.

Marquee *Documentation*

Reproduce the behavior of the `<marquee/>` html tag.

MPD Client *Documentation*

Sends the current song (and optionally the progress inside the song) to the current (chat) tab.

OTR *Documentation*

Allows encrypted and deniable exchanges using OTR.

PacoKick *Documentation*

Kicks a random user in the room.

Ping *Documentation*

Sends a ping probe to an entity (XEP-0199)

Pipe Command *Documentation*

Send commands to poezio through a named pipe.

PointPoint *Documentation*

Insert dots in your messages.

Quote *Documentation*

Adds a `/quote` command to quote a message at HH:MM:SS and put it in the input (to prevent painful copy/pastes).

Rainbow *Documentation*

Sends your messages in rainbow colors using XHTML-IM.

Regex Admin *Documentation*

Add regex-based kick and ban commands.

Reminder *Documentation*

Reminds you to do something every now and then.

Reorder *Documentation*

Reorder the tabs according to a static layout.

Replace *Documentation*

Replace some patterns in your messages.

Revstr *Documentation*

Reverse everything you say.

Screen Detach *Documentation*

Changes your status to **away** if the screen (or tmux) poezio is in gets detached.

Send Delayed *Documentation*

Program the sending of futur messages.

Server Part *Documentation*

Add a `/server_part` command.

Shuffle *Documentation*

Shuffle everything you say.

Simple notify *Documentation*

Sends a notification with a command of your choice on (non-chatroom) messages.

Spam *Documentation*

Adds a subtle little advertising in your messages.

Status *Documentation*

Adds convenient aliases to `/status` (`/away`, etc).

Tell *Documentation*

Sends a message to a nick when he connects to a chatroom.

Time Marker *Documentation*

Display the time between two messages.

Title change *Documentation*

Change the title of the terminal according to the name of the current tab.

Upload *Documentation*

Add an `/upload` command to upload a file.

Uptime *Documentation*

Gets the uptime of a XMPP server or a component.

vCard *Documentation*

Add a `/vcard` command to retrieve and display a vCard.

9.5.1 Admin

9.5.2 Alias

Usage

This plugin defines two new global commands: */alias* and */unalias*.

/alias Usage: `/alias <name> <command> [args]`

This command will create a new command, named `<name>` (and callable with `/name`), that runs `/command`, with `[args]` as fixed args for the command. When you run the alias, you can also pass parameters to it, that will be given to the original command.

Example:

```
/alias toto say koin
```

Will bind `/say koin` to `/toto`, so this alias will work in any Chat tab. If someone calls it with

```
/toto koin
```

Poezio will then execute `/say koin koin`.

Also, you can rebind arguments arbitrarily, with the `{ }` placeholder. For example,

```
/alias toto say {} le {}  
/toto loulou coucou
```

Will execute `/say loulou le coucou`, because the `{ }` are replaced with the command args, in the order they are given.

Extra args are still added at the end of the command if provided (args used for the formatting are only used for the formatting).

/unalias Usage: `/unalias <name>`

This command removes a defined alias.

Config

The aliases are stored inside the configuration file for the plugin. You can either use the above commands or write it manually, and it will be read when the plugin is loaded.

Example of the syntax:

```
[alias]  
toto = say {} le {}  
j = join {}@conference.jabber.org/nick  
jp = say je proteste
```

9.5.3 Amsg

This plugin broadcasts a message to all your joined rooms.

Note: With great power comes great responsibility. Use with moderation.

Command

/amsg Usage: /amsg <message>

Broadcast a message.

9.5.4 Day change

This plugin adds a message at 00:00 in each of your chat tabs saying that the date has changed.

9.5.5 Display corrections

Lists old versions of a corrected message.

Usage

/display_corrections Usage: /display_corrections [number]

This command lists the old versions of a message.

Without argument, it will list the last corrected message if there is any. If you give an integer as an argument, /display_corrections will go back gradually in the buffer to find the message matching that number (starting from 1, for the last corrected message).

If you are scrolling in the buffer, Poezio will list the corrected messages starting from the first you can see. (although there are some problems with multiline messages).

9.5.6 Embed

Display an image URL as an embedded image in some clients like Conversations. Uses: <https://xmpp.org/extensions/xep-0066.html#x-oob>

Usage

/embed <image_url> Run this command to send the <image_url> as an embedded image in your contact's client.

9.5.7 Exec

This plugin lets you execute a system command through poezio.

Usage

Warning: Running commands that start a daemon or an interface is not a good idea.

/exec Usage: /exec [-o|-O] <command>

Execute a system command.

```
/exec command
```

Will give you the result in the information buffer.

```
/exec -o command
```

Will send the result of the command into the current tab, if possible.

```
/exec -O command
```

Will send the result of the command and the command summary into the current tab, if possible.

9.5.8 Figlet

This plugin uses figlet to transform every message into a big ascii-art message.

Usage

Say something in a Chat tab.

Note: Can create fun things when used with *The rainbow plugin*.

9.5.9 Link

Opens links in a browser.

Installation

First use case: local use

If you use poezio on your workstation, this is for you. You only have to load the plugin:

```
/load link
```

Second use case: remote use

If you use poezio through SSH, this is for you.

Note: Small explanation: Poezio will create a [Unix FIFO](#) and send the commands in, and you will have to run a daemon locally with ssh, to get those commands.

First, set the `exec_remote` option in the config file to `true`. Then select the directory you want to put the fifo in (default is the current directory, `.` / `/`), the `poezio.fifo` file will be created there.

After that, load the plugin:

```
/load link
```

And open a link with */link* (as described below), this will create the FIFO.

You need to grab poezio's sources on your client computer, or at least the `daemon.py` file.

Finally, on your client computer, run the ssh command:

```
ssh toto@example.org "cat ~/poezio/poezio.fifo" | python3 daemon.py
```

Usage

/link Usage: `/link [range] [command]`

This plugin adds a */link* command that will open the links in `firefox`. If you want to use another browser, or any other command, you can use the */set* command to change the *browser* option. You can also specify the command to execute directly in the arguments. For example */link "mpv %s"* will open the first link found using `mpv`, instead of the configured browser.

/link without argument will open the last link found in the current tab, if any is found. An optional range argument can be given, to select one or more links to open. Examples: */link 1* is equivalent to */link /link 3* will open the third link found in the current tab, starting from the bottom. */link 1:5* will open the last five links in the current tab */link :2* will open the last two links

Options

exec_remote

To execute the command on your client

browser Set the default browser started by the plugin

9.5.10 Mpd client

This plugin is here to send what you are listening to in a chat tab.

Installation

You need `python-mpd`, in its python3 version.

Then you can load the plugin.

```
/load mpd_client
```

Configuration

You have to put the following into `mpd_client.cfg`, as explained in the *Plugin configuration* section.

Note: If you do not put anything, the plugin will try to connect to `localhost:6600` with no password.

```
[mpd_client]
host = the_mpd_host
port = 6600
password = password if necessary
```

Usage

/mpd Usage: /mpd [full]

The bare command will show the current song, artist, and album

/mpd full will show the current song, artist, and album, plus a nice progress bar in color.

9.5.11 OTR

This plugin implements [Off The Record](#) messaging.

This is a plugin used to encrypt a one-to-one conversation using the OTR encryption method. You can use it if you want good privacy, deniability, authentication, and strong secrecy. Without this encryption, your messages are encrypted **at least** from your client (poezio) to your server. The message is decrypted by your server and you cannot control the encryption method of your messages from your server to your contact's server (unless you are your own server's administrator), nor from your contact's server to your contact's client.

This plugin does end-to-end encryption. This means that **only** your contact can decrypt your messages, and it is fully encrypted during **all** its travel through the internet.

Note that if you are having an encrypted conversation with a contact, you can **not** send XHTML-IM messages to them (or correct messages, or anything more than raw text). All formatting will be removed and be replaced by plain text messages.

This is a limitation of the OTR protocol, and it will never be fixed. Some clients like Pidgin-OTR try to do magic stuff with html unescaping inside the OTR body, and it is not pretty.

Installation

To use the OTR plugin, you must first install pure-python-otr and pycrypto (for python3).

You have to install it from the git because a few issues were found with the python3 compatibility while writing this plugin, and the fixes did not make it into a stable release yet.

Install the python module:

```
git clone https://github.com/afflux/pure-python-otr.git
cd pure-python-otr
python3 setup.py install --user
```

You can also use pip in a virtualenv (built-in as [pyvenv](#) with python since 3.3) with the requirements.txt at the root of the poezio directory.

Important details

The OTR session is considered for a full JID (e.g. [toto@example/client1](#)), but the trust is set with a bare JID (e.g. [toto@example](#)). This is important in the case of Private Chats (in a chatroom), since you cannot always get the real JID of your contact (or check if the same nick is used by different people).

Note: This also means that you cannot have an OTR session in the “common” conversation tab, which is not locked to a specific JID. After activating the plugin, you need to open a session with a full JID to be able to use OTR.

Usage

Command added to Static Conversation Tabs (opened with `/message foo@bar/baz` or by expanding a contact in the roster) and Private Tabs:

/otr Usage: `/otr [start|refresh|end|fpr|ourfpr|trust|untrust]`

This command is used to manage an OTR private session.

- The `start` (or `refresh`) command starts or refreshes a private OTR session
- The `end` command ends a private OTR session
- The `fpr` command gives you the fingerprint of the key of the remote entity
- The `ourfpr` command gives you the fingerprint of your own key
- The `trust` command marks the current remote key as trusted for the current remote JID
- The `untrust` command removes that trust
- Finally, the `drop` command is used if you want to delete your private key (not recoverable).

Warning: With `drop`, the private key is only removed from the filesystem, *NOT* with multiple rewrites in a secure manner, you should do that yourself if you want to be sure.

/otrsmp Usage: `/otrsmp <ask|answer|abort> [question] [secret]`

Verify the identify of your contact by using a pre-defined secret.

- The `abort` command aborts an ongoing verification
- The `ask` command start a verification, with a question or not
- The `answer` command sends back the answer and finishes the verification

Managing trust

An OTR conversation can be started with a simple `/otr start` and the conversation will be encrypted. However it is very often useful to check that your are talking to the right person.

To this end, two actions are available, and a message explaining both will be prompted each time an **untrusted** conversation is started:

- Checking the knowledge of a shared secret through the use of `/otrsmp`
- Exchanging fingerprints (`/otr fpr` and `/otr ourfpr`) out of band (in a secure channel) to check that both match, then use `/otr trust` to add then to the list of trusted fingerprints for this JID.

Files

This plugin creates trust files compatible with libotr and the files produced by gajim.

The files are located in `$XDG_DATA_HOME/poezio/otr/` by default (so `~/.local/share/poezio/otr` in most cases).

Two files are created:

- An `account_jid.key3` (`example@example.com.key3`) file, which contains the private key
- An `account_jid.fpr` (`example@example.com.fpr`) file, which contains the list of trusted (or untrusted) JIDs and keys.

Configuration

decode_entities **Default:** `true`

Decode XML and HTML entities (like `&`) even when the document isn't valid (if it is valid, it will be decoded even without this option).

decode_newlines **Default:** `true`

Decode `
` and `
` tags even when the document isn't valid (if it is valid, it will be decoded even without this option for `
`, and `
` will make the document invalid anyway).

decode_xhtml **Default:** `true`

Decode embedded XHTML.

keys_dir **Default:** `$XDG_DATA_HOME/poezio/otr`

The directory in which you want keys and fpr to be stored.

log **Default:** `false`

Log conversations (OTR start/end marker, and messages).

require_encryption **Default:** `false`

If `true`, prevents you from sending unencrypted messages, and tries to establish OTR sessions when receiving unencrypted messages.

timeout **Default:** `3`

The number of seconds poezio will wait until notifying you that the OTR session was not established. A negative or null value will disable this notification.

The *require_encryption*, *decode_xhtml*, *decode_entities* and *log* configuration parameters are tab-specific.

9.5.12 Pacokick

This plugin adds a */pacokick* command, which is a random kick.

Usage

/pacokick Run the command in a room where you are a moderator to kick someone randomly.

9.5.13 Ping

This plugin allows you to ping an entity.

Command

/ping Usage (globally): /ping <jid>

Usage (in a MUC tab): /ping <jid or nick>

Usage (in a conversation tab): /ping [jid]

Globally, you can do /ping jid@example.com to get a ping.

In a MUC, you can either do it to a JID or a nick (/ping nick or /ping jid@example.com).

In a private or a direct conversation, you can do /ping to ping the current interlocutor.

9.5.14 Quote

This plugin allows you to quote messages easily.

Usage

/quote Usage: /quote <message>

The message must exist. You can use autocompletion to get the message you want to quote easily.

Example:

```
/quote "Pouet "
```

If the message “Pouet” exists, it will be put in the input. If not you will get a warning.

Options

after_quote Default value: [empty]

Text to insert after the quote. %(nick)s and %(time)s can be used to insert the nick of the user who sent the message or the time of the message.

before_quote Default value: [empty]

Text to insert before the quote. %(nick)s and %(time)s can be used to insert the nick of the user who sent the message or the time of the message.

9.5.15 Rainbow

This plugin colors each character of a message with a random color.

Note: As ticket #3273 puts it, the final output is closer to vomit than a rainbow.

Usage

/rainbow Say something in a Chat tab.

Note: Can create fun things when used with *The figlet plugin*.

9.5.16 Reminder

Usage

This plugin defines three new global commands: */remind*, */done*, and */tasks*.

/remind Usage: `/remind <time> <todo>`

This command will remind you to do `todo` every `time`.

/done Usage: `/done <id>` Remove a reminder.

The `id` is found using */tasks*.

/tasks Print a list of the tasks, their ids, and their frequency, into the information buffer.

Time format

In seconds:

```
/remind 600 Work!
```

Will remind you to work every 10 minutes.

Defining the time in seconds is not really practical, so you can describe it with days, hours, and minutes, in a time-string, e.g:

```
/remind 1h23m "Get up"
```

Will remind you to get up every 1 hour 23 minutes.

9.5.17 Replace

Replace some patterns in a message before sending it.

Usage

Insert a pattern in the form

```
%pattern%
```

in your message, and it will be replaced by the corresponding text.

The list of provided patterns is:

- **time:** Insert the current time
- **date:** Insert the current date

- **datetime**: Insert the current date and time
- **random_nick**: Insert a random nick from the current MUC
- **dice**: Insert a random number between 1 and 6

Add your own pattern

You can easily edit this plugin to add your own patterns. For example if don't want to search for an insult everytime you're angry, you can create a curse pattern this way:

- In the `init(self)` method of the Plugin class, add something like

```
self.patterns['curse'] = replace_curse
```

- then define a function (not a method of the Plugin class) at the bottom of the file. For example:

```
def replace_curse(message, tab):  
    return random.choice(['dumb shit', 'idiot', 'moron'])
```

and you can now use something like

```
Shut up, %curse%!
```

in your everyday-conversations.

For more convenience, you can read your nice words from a file, do whatever you want in that function, as long as it returns a string.

9.5.18 Screen detach

This plugin will set your status to **away** if you detach your screen.

The default behaviour is to check for both `tmux` and `screen` (in that order).

Configuration options

use_screen **Default:** `true`

Try to find an attached screen.

use_tmux **Default:** `true`

Try to find and attached `tmux`.

use_csi **Default:** `false`

Use `client state indication` to limit bandwidth (thus CPU) usage when detached. **WARNING:** using CSI together with chatrooms will result in inaccurate logs due to presence filtering or other inaccuracies.

9.5.19 Send delayed

Send a message after a certain delay.

Usage

This plugin adds a command to the chat tabs.

/send_delayed Usage: /send_delayed <delay> <message>

Send a message after a given delay to the current tab. The delay can be either in seconds or in a classic XdXhXm format (e.g. 7h3m or 1d), some examples are given with the autocompletion.

9.5.20 Simple notify

This plugin lets you execute a command, to notify you from new important messages.

Installation and configuration

You need to create a plugin configuration file. Create a file named `simple_notify.cfg` into your plugins configuration directory (`~/ .config/poezio/plugins` by default), and fill it like this:

First example:

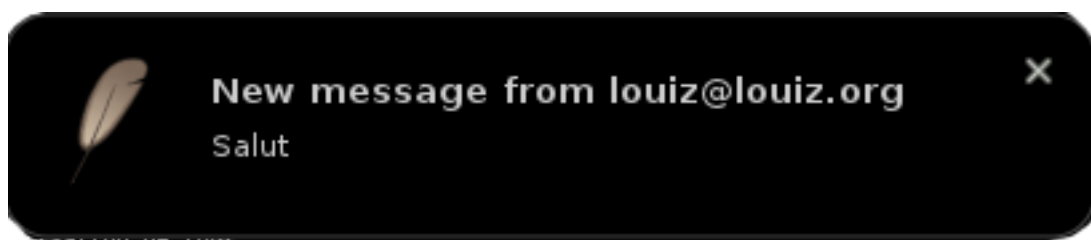
```
[simple_notify]
command = notify-send -i /path/to/poezio/data/poezio_80.png "New message from %(from)s
↳ " "%(body)s"
```

Second example:

```
[simple_notify]
command = echo \<%(from)s\> %(body)s >> some.fifo
delay = 3
after_command = echo >> some.fifo
```

You can put any command, instead of these ones. You can also use the special keywords `%(from)s` and `%(body)s` that will be replaced directly in the command line by the author of the message, and the body.

The first example shown above will display something like this:



The second example will first write the author and the message in a fifo, that fifo can locally be read by some other program (was tested with the `xmobar PipeReader` command, which displays what is read from a fifo into a status bar. Be careful, you have two different fifos in that case, don't get confused). The `delay` and `after_command` options are used to erase/delete/kill the notification after a certain delay. In our example it is used to display an empty message in our `xmobar`, erasing the notification after 3 seconds.

Third example:

```
[simple_notify]
command = notify-send -i /path/to/poezio/data/poezio_80.png "New message from %(from)s
↳ " "%(body)s"
```

(continues on next page)

(continued from previous page)

```
muc_too = true
muc_list = someroom@conference.jabber.org:someotherroom@conference.jabber.org
```

If present and set to `True`, the `muc_too` option will also trigger a notification when a new message arrives on a Multi User Chat you've joined.

If present and set to a colon separated list of muc JIDs, `muc_list` together with `muc_too = true` will only notify when a new message arrives on a Multi User Chat, you've joined if it is present on the list.

Note: If you set the `exec_remote` option to `true` into the main configuration file, the command will be executed remotely (as explained in the [Link](#) plugin help).

Options defined

after_command Command to run after [delay](#). You probably want to clean up things.

command The command to execute (with special keywords `%{from}s` and `${body}s`)

delay Delay after which [after_command](#) must be executed.

muc_too Boolean indicating whether new messages in Multi User Chat rooms should trigger a notification or not.

9.5.21 Spam

Add a subtle little advertising in your messages.

Configuration

```
[spam]
ad = I'm a happy poezio user. Get it at http://poezio.eu
```

9.5.22 Status

This plugin adds several aliases, to shorten status changes.

Aliases

/afk

/away Set your status to away

/available Set your status to available

/chat Set your status to chat

/dnd

/busy Set your status to dnd

/xa Set your status to xa

9.5.23 Tell

The command added by this plugin sends a message to someone when he next joins.

Usage

This plugin defines two new commands for chatroom tabs: */tell*, */untell*, and */list_tell*.

/list_tell Usage: */list_tell*

List all queued messages for the current chatroom.

/tell Usage: */tell* <nick> <message>

Send *message* to *nick* at his next join.

/untell Usage: */untell* <nick>

Cancel all scheduled messages to *nick*.

9.5.24 Time marker

Display the time between two messages.

Helps you identify the times of a conversation. For example if you disable the timestamps, and remove the join/quit notifications in a chatroom, you can't really distinguish when a conversation stopped and when a new one started, because you don't have a visual separation between the two.

This plugin displays a message in the conversation indicating the time that passed between two messages, if the time is bigger than X minutes (configurable, of course. Default is 15 minutes). This way you know how many time elapsed between them, letting you understand more easily what is going on without any visual clutter.

Configuration

You can configure the minimum delay between two messages, to display the time marker, in seconds. The default is 10 minutes (aka 600 seconds).

```
[time_marker]
delay = 600
```

Usage

Messages like “2 hours, 25 minutes passed...” are automatically displayed into the conversation. You don't need to (and can't) do anything.

9.5.25 Uptime

This plugin retrieves the uptime of a server.

Command

/uptime Usage: */uptime* <jid>

Retrieve the uptime of the server of *jid*.

9.5.26 Revstr

Reverse everything you say (Je proteste énergiquement will become tnemeuqigrené etsetorp eJ)

9.5.27 Double

Double the first word of any message you send in a *Chatroom tab*, making you appear retarded.

9.5.28 Shuffle

Shuffle the words in every message you send in a *Chatroom tab* (may/should confuse the reader).

9.5.29 Iq show

Show the exchanged IQs (useful for debugging).

9.5.30 Admin regex

This plugins adds a */rkick* and a */rban* command, in order to kick/ban according to a regex on a nick.

Commands

Those commands take a regular expression (as defined in the [re module documentation](#)) as a parameter.

/rban Usage: /rban <regex>

Ban a participant using a regex.

/rkick Usage: /rkick <regex>

Kick a participant using a regex.

9.5.31 Pointpoint

This plugin adds a command (that can be bound to a key) that adds a random number of dots in the input, making you look depressed, or overly thinking...

Installation

Load the plugin.:

```
/load pointpoint
```

Then use the command:

```
/pointpoint
```

But since the goal is to be able to add the dots while typing a message, entering a command is not really useful. To be useful, this plugin needs to be used through a bound key, for example like this:

```
/bind M-. _exc_pointpoint
```

You just need to press Alt+. and this will insert dots in your message.

Command

/pointpoint Usage: /pointpoint

...

9.5.32 Autocorrect

This plugin lets you perform simple replacements on the last message.

Usage

Note: the /, #, !, : and ; chars can be used as separators, even if the examples only use /

Regex replacement

Once the plugin is loaded, any message matching the following regex:

```
^s/(.+?)/(.*?)(/|/g)?$
```

will be interpreted as a regex replacement, and the substitution will be applied to the last sent message.

For example, if you sent the message:

```
This tab lists all public rooms on a MUC service. It is currently very limited but
↪will be improved in the future. There currently is no way to search a room.
```

And you now want to replace “MUC” with “multi-user chat”, you input:

```
s/MUC/multi-user chat
```

And poezio will correct the message for you.

Raw string replacement

Once the plugin is loaded, any message matching the following regex:

```
^r/(.+?)/(.*?)(/|/g)?$
```

will be interpreted as a replacement, and the substitution will be applied to the last send message.

This variant is useful if you don’t want to care about regular expressions (and you do not want to have to escape stuff like space or backslashes).

9.5.33 IRC

Plugin destined to be used together with the Biboumi IRC gateway.

For more information about Biboumi, please see the [official website](#).

This plugin is here as a non-default extension of the poezio configuration made to work with IRC rooms and logins. It also defines commands aimed at reducing the amount of effort needed to navigate smoothly between IRC and XMPP rooms.

Configuration

Every feature of this plugin is centered around its *configuration file*, so you have to make sure it is filled properly.

Global configuration

gateway **Default:** `irc.poez.io`

The JID of the IRC gateway to use. If empty, `irc.poez.io` will be used. Please try to run your own, though, it's painless to setup.

initial_connect **Default:** `true`

Set to `true` if you want to join all the rooms and try to authenticate with nickserv when the plugin gets loaded. If it set to `false`, you will have to use the */irc_login* command to authenticate, and the */irc_join* command to join preconfigured rooms.

Note: There is no nickname option because the default from poezio will be used.

Server-specific configuration

Write a configuration section for each server, with the server address as the section name, and the following options:

login_command **Default:** `[empty]`

The command used to identify with the services (e.g. `IDENTIFY mypassword`).

login_nick **Default:** `[empty]`

The nickname to whom the auth command will be sent.

nickname **Default:** `[empty]`

Your nickname on this server. If empty, the default configuration will be used.

rooms [IRC plugin] **Default:** `[empty]`

The list of rooms to join on this server (e.g. `#room1:#room2`).

Note: If no `login_command` or `login_nick` is set, the authentication phase won't take place and you will join the rooms without authentication with nickserv or whatever.

Commands

/irc_join Usage: /irc_join <room or server>

Join the specified room on the same server as the current tab (can be a private conversation or a chatroom). If a server that appears in the conversation is specified instead of a room, the plugin will try to join all the rooms configured with autojoin on that server.

/irc_login Usage: /irc_login [server1] [server2]...

Authenticate with the specified servers if they are correctly configured. If no servers are provided, the plugin will try them all. (You need to set *login_nick* and *login_command* as well)

/irc_query Usage: /irc_query <nickname> [message]

Open a private conversation with the given nickname, on the same IRC server as the current tab (can be a private conversation or a chatroom). Doing */irc_query foo "hello there"* when the current tab is *#foo%irc.example.com@biboumi.example.com* is equivalent to */message foo%irc.example.com@biboumi.example.com "hello there"*

Example configuration

```
[irc]
gateway = irc.poez.io

[irc.freenode.net]
nickname = mynick
login_nick = nickserv
login_command = identify mypassword
rooms = #testroom1:#testroom2

[irc.geeknode.org]
nickname = anothernick
login_nick = C
login_command = nick identify mypassword
rooms = #testvroum
```

9.5.34 Title change

This plugin will set the title of your terminal to the name of the current tab.

9.5.35 Pipe Command

This plugin allows commands to be sent to poezio via a named pipe.

You can run the same commands that you would in the poezio input (e.g. *echo '/message toto@example.tld Hi' >> /tmp/poezio.fifo*).

Configuration

pipename Default: /tmp/poezio.fifo

The path to the fifo which will receive commands.

9.5.36 Close all

`close_all` plugin: close all tabs except chatrooms and the contact list.

Commands

/closeall Usage: `/closeall`

Close all tabs except the roster and chatroom tabs.

9.5.37 Reorder

`reorder` plugin: Reorder the tabs according to a layout

Commands

/reorder Usage: `/reorder`

Reorder the tabs according to the configuration.

/save_order Usage: `/save_order`

Save the current tab order to the configuration.

Configuration

The configuration file must contain a section `[reorder]` and each option must be formatted like `[tab number] = [tab type]:[tab name]`.

For example:

```
[reorder]
1 = muc:toto@conference.example.com
2 = muc:example@muc.example.im
3 = dynamic:robert@example.org
```

The `[tab number]` must be at least 1; if the range is not entirely covered, e.g.:

```
[reorder]
1 = muc:toto@conference.example.com
3 = dynamic:robert@example.org
```

Poezio will insert gaps between the tabs in order to keep the specified numbering (so in this case, there will be a tab 1, a tab 3, but no tab 2).

The `[tab type]` must be one of:

- `muc` (for multi-user chats)
- `private` (for chats with a specific user inside a multi-user chat)
- `dynamic` (for normal, dynamic conversations tabs)
- `static` (for conversations with a specific resource)

And finally, the `[tab name]` must be:

- For a type `muc`, the bare JID of the room

- For a type `private`, the full JID of the user (room JID with the username as a resource)
- For a type `dynamic`, the bare JID of the contact
- For a type `static`, the full JID of the contact

9.5.38 Cyber

This plugin adds a “cyber” prefix to a random word in your chatroom messages.

Usage

Say something in a MUC tab.

Configuration options

frequency **Default:** 10

The percentage of the time the plugin will activate (randomly). 100 for every message, ≤ 0 for never.

9.5.39 CSI

This plugin lets you set the **CSI** state manually, when the autoaway plugin is not sufficient for your usage.

Commands

/csi_active **Usage:** /csi_active

Set CSI state to active.

/csi_inactive **Usage:** /csi_inactive

Set CSI state to inactive.

9.5.40 Dice

Dice plugin: roll some dice

Usage of this plugin is not recommended.

Commands

/roll [number of dice] [duration of the roll] Roll one or several unicode dice

Configuration

default_duration **Default:** 5

Total duration of the animation.

refresh **Default:** 0.5

Interval in seconds between each correction (the closest to 0 is the fastest)

9.5.41 Disco

Do a disco#info query on a JID

Usage

/disco Usage: /disco <JID>

This command queries a JID for its disco#info.

There is no cache, as this is generally used for debug more than anything user-related.

9.5.42 Marquee

Marquee plugin: replicate the html <marquee/> tag with message corrections.

Usage of this plugin is not recommended.

Commands

/marquee <text> Send the following text with <marquee/> behavior

Configuration

padding Default: 20

Padding to use to move the text.

refresh Default: 1

Interval between each correction (the closest to 0 is the fastest)

total_duration Default: 30

Total duration of the animation.

9.5.43 Server Part

This plugin adds a /server_part command to leave all rooms on a server.

Command

/server_part Usage: /server_part [<server> [message]]

Leave all rooms on <server>, if not provided and the current tab is a chatroom tab, it will leave all rooms on the current server. [message] can indicate a quit message.

9.5.44 vCard

This plugin adds a */vcard* command to all tabs, allowing you to request and display the vcard-temp of any given entity.

Command

/vcard Usage (globally): /vcard <jid>

Usage (in a chatroom tab): /vcard <jid or nick>

Usage (in a conversation or contact list tab): /vcard [jid]

Globally, you can do /vcard user@server.example to get a vcard.

In a chatroom , you can either do it on a JID or a nick (/vcard nick, /vcard room@muc.server.example/nick or /vcard user@server.example).

In a private or a direct conversation, you can do /vcard to request vcard from the current interlocutor, and in the contact list to do it on the currently selected contact.

10.1 About plugins

10.1.1 Plugin API documentation

BasePlugin

class poezio.plugin.**BasePlugin** (*plugin_api, core, plugins_conf_dir*)

Class that all plugins derive from.

init (*self*)

Method called at the creation of the plugin.

Do not override `__init__` and use this instead.

cleanup (*self*)

Method called before the destruction of the plugin.

Use it to erase or save things before the plugin is disabled.

core

The Poezio `Core` object. Use it carefully.

api

The *PluginAPI* instance for this plugin.

init ()

Method called at the creation of the plugin.

Do not overwrite `__init__` and use this instead.

Each plugin inheriting *BasePlugin* has an `api` member variable, which refers to a *PluginAPI* object.

The *PluginAPI* object is an interface through which the *BasePlugin* (and inheritors) *should* go to interact with poezio. If it is not sufficient, then the `core` member can be used.

PluginAPI

class poezio.plugin.**PluginAPI** (*core, plugin_manager*)

The public API exposed to the plugins. Its goal is to limit the use of the raw Core object as much as possible.

add_command (*module, *args, **kwargs*)

Add a global command.

Parameters

- **name** (*str*) – The name of the command (/name)
- **handler** (*function*) – The function called when the command is run.
- **help** (*str*) – The complete help for that command.
- **short** (*str*) – A short description of the command.
- **completion** (*function*) – The completion function for that command (optional)
- **usage** (*str*) – A string showing the required and optional args of the command. Optional args should be surrounded by [] and mandatory args should be surrounded by <>.

Example string: “<server> [port]”

Raises **Exception** – If the command already exists.

add_event_handler (*module, *args, **kwargs*)

Add an event handler for a poezio event.

Parameters

- **event_name** (*str*) – The event name.
- **handler** (*function*) – The handler function.
- **position** (*int*) – The position of that handler in the handler list. This is useful for plugins like OTR, which must be the last function called on the text. Defaults to 0.

A complete list of those events can be found at <https://doc.poez.io/dev/events.html>

add_key (*module, *args, **kwargs*)

Associate a global binding to a handler.

Parameters

- **key** (*str*) – The curses representation of the binding.
- **handler** (*function*) – The function called when the binding is pressed.

Raises **Exception** – If the binding is already present.

add_slix_event_handler (*module, event_name, handler*)

Add an event handler for a slxmpp event.

Parameters

- **event_name** (*str*) – The event name.
- **handler** (*function*) – The handler function.

A list of the slxmpp events can be found here http://sleekxmpp.com/event_index.html

add_tab_command (*module, *args, **kwargs*)

Add a command to only one type of tab.

Parameters

- **tab_type** (*tabs.Tab*) – The type of Tab to target.
- **name** (*str*) – The name of the command (/name)
- **handler** (*function*) – The function called when the command is run.
- **help** (*str*) – The complete help for that command.
- **short** (*str*) – A short description of the command.
- **completion** (*function*) – The completion function for that command (optional)
- **usage** (*str*) – A string showing the required and optional args of the command. Optional args should be surrounded by [] and mandatory args should be surrounded by <>.

Example string: “<server> [port]”

Raises **Exception** – If the command already exists.

add_tab_key (*module*, **args*, ***kwargs*)

Associate a binding to a handler, but only for a certain tab type.

Parameters

- **tab_type** (*Tab*) – The type of tab to target.
- **key** (*str*) – The binding to add.
- **handler** (*function*) – The function called when the binding is pressed

add_timed_event (_, **args*, ***kwargs*)

Schedule a timed event.

Parameters **event** (*timed_events.TimedEvent*) – The timed event to schedule.

all_tabs (_)

Return a list of all opened tabs

Returns list The list of tabs.

create_delayed_event (_, **args*, ***kwargs*)

Create a delayed event, but do not schedule it; *add_timed_event()* must be used for that.

A delayed event is a timed event with a delay from the time this function is called (instead of a datetime).

Parameters

- **delay** (*int*) – The number of seconds to schedule the execution
- **callback** (*function*) – The handler that will be executed
- **args** – Optional arguments passed to the handler.

Returns The created event.

Return type *timed_events.DelayedEvent*

create_timed_event (_, **args*, ***kwargs*)

Create a timed event, but do not schedule it; *add_timed_event()* must be used for that.

Parameters

- **date** (*datetime.datetime*) – The time at which the handler must be executed
- **callback** (*function*) – The handler that will be executed
- **args** – Optional arguments passed to the handler.

Returns The created event.

Return type `timed_events.TimedEvent`

current_tab (`_`)

Get the current Tab.

Returns The current tab.

del_command (`module`, `*args`, `**kwargs`)

Remove a global command.

Parameters **name** (`str`) – The name of the command to remove. That command `_must_` have been added by the same plugin

del_event_handler (`module`, `*args`, `**kwargs`)

Remove a handler for a poezio event.

Parameters

- **event_name** (`str`) – The name of the targeted event.
- **handler** (`function`) – The function to remove from the handlers.

del_key (`module`, `*args`, `**kwargs`)

Remove a global binding.

Parameters **key** (`str`) – The binding to remove.

del_slix_event_handler (`module`, `event_name`, `handler`)

Remove a handler for a slxmp event

Parameters

- **event_name** (`str`) – The name of the targeted event.
- **handler** (`function`) – The function to remove from the handlers.

del_tab_command (`module`, `*args`, `**kwargs`)

Remove a tab-specific command.

Parameters

- **tab_type** (`tabs.Tab`) – The type of tab to target.
- **name** (`str`) – The name of the command to remove. That command `_must_` have been added by the same plugin

del_tab_key (`module`, `*args`, `**kwargs`)

Remove a binding added with `add_tab_key`

Parameters

- **tab_type** (`tabs.Tab`) – The type of tab to target.
- **key** (`str`) – The binding to remove.

get_conversation_messages (`_`, `*args`, `**kwargs`)

Get all the Messages of the current Tab.

Returns The list of `text_buffer.Message` objects.

Returns None if the Tab does not inherit from ChatTab.

Return type `list`

get_status (`_`)

Get the current user global status.

Returns Status The current status.

information (_, *args, **kwargs)

Display a new message in the information buffer.

Parameters

- **msg** (*str*) – The message to display.
- **typ** (*str*) – The message type (e.g. Info, Error...)

remove_timed_event (_, *args, **kwargs)

Unschedule a timed event.

Parameters **event** (`timed_events.TimedEvent`) – The event to unschedule.

run_command (_, *args, **kwargs)

Run a command from the current tab. (a command starts with a /, if not, it's a message)

Parameters **line** (*str*) – The command to run.

send_message (_, *args, **kwargs)

Send a message to the current tab.

Parameters **msg** (*str*) – The message to send.

Example plugins

Example 1: Add a simple command that sends “Hello World!” into the conversation

```
class Plugin(BasePlugin):
    def init(self):
        self.add_command('hello', self.command_hello, "Send 'Hello World!'")

    def command_hello(self, arg):
        self.core.send_message('Hello World!')
```

Example 2: Adds an event handler that sends “tg” to a groupchat when a message is received from someone named “Partauche”

```
class Plugin(BasePlugin):
    def init(self):
        self.add_event_handler('muc_msg', self.on_groupchat_message)

    def on_groupchat_message(self, message, tab):
        if message['mucnick'] == "Partauche":
            tab.command_say('tg')
```

10.1.2 Event Index

The following events are poezio-only events, for Slixmpp events, check out [their index](#).

changing_nick

- **presence:** *Presence* to be sent

Triggered when the user changes his/her nickname on a MUC. The presence can thus be modified before being sent.

conversation_chatstate

- **message:** *Message* received

- **tab:** ConversationTab source

Triggered when a chatstate is received in a ConversationTab.

conversation_msg

- **message:** *Message* received
- **tab:** ConversationTab source

Triggered when a message is received in a ConversationTab.

conversation_say

- **message:** *Message* that will be sent
- **tab:** ConversationTab source

Triggered whenever the user sends a message to a ConversationTab.

conversation_say_after:

- **message:** *Message* that will be sent
- **tab:** ConversationTab source

Same thing than *conversation_say*, but after XHTML generation of the body, if needed. This means you must not insert any colors in the body in the handler, since it may lead to send invalid XML. This hook is less safe than *conversation_say* and you should probably not need it.

highlight

- **message:** *Message* that was received
- **tab:** MucTab source of the event

ignored_private

- **message:** *Message* received
- **tab:** PrivateTab source

Triggered when a private message (that goes in a PrivateTab) is ignored automatically by poezio.

tab is always None, except when a tab has already been opened.

joining_muc

- **presence:** *Presence* to be sent

Triggered when joining a MUC. The presence can thus be modified before being sent.

muc_ban

- **presence:** *Presence* received
- **tab:** MucTab source

Triggered when a user from a MucTab gets banned.

muc_chatstate

- **message:** *Message* received
- **tab:** MucTab source

Triggered when a chatstate is received in a MucTab.

muc_join

- **presence:** *Presence* received

- **tab:** MucTab source

Triggered when a user joins a MucTab

muc_kick

- **presence:** *Presence* received
- **tab:** MucTab source

Triggered when a user from a MucTab gets kicked.

muc_msg

- **message:** *Message* received
- **tab:** MucTab source

Triggered when a message is received in a MucTab.

muc_nickchange

- **presence:** *Presence* received
- **tab:** MucTab source

Triggered when a user in a MucTab changes his nickname.

muc_presence

- **presence:** *Presence* received
- **tab:** MucTab source

Triggered when a presence is received from someone in a MucTab.

muc_say

- **message:** *Message* that will be sent
- **tab:** MucTab source

Triggered whenever the user sends a message to a MucTab.

muc_say_after

- **message:** *Message* that will be sent
- **tab:** MucTab source

Same thing than *muc_say*, but after XHTML generation of the body, if needed. This means you must not insert any colors in the body in the handler, since it may lead to send invalid XML. This hook is less safe than *muc_say* and you should probably not need it.

normal_presence

- **presence:** *Presence* received
- **resource:** *Resource* that emitted the *Presence*

Triggered when a presence is received from a contact.

private_chatstate

- **message:** *Message* received
- **tab:** PrivateTab source

Triggered when a chatstate is received in a PrivateTab.

private_msg

- **message:** *Message* received
- **tab:** `PrivateTab` source

Triggered when a message is received in a `PrivateTab`.

private_say

- **message:** *Message* that will be sent
- **tab:** `PrivateTab` source

Triggered whenever the user sends a message to a `PrivateTab`.

private_say_after

- **message:** *Message* that will be sent
- **tab:** `PrivateTab` source

Same thing than *private_say*, but after XHTML generation of the body, if needed. This means you must not insert any colors in the body in the handler, since it may lead to send invalid XML. This hook is less safe than *private_say* and you should probably not need it.

send_normal_presence

- **presence:** *Presence* sent

Triggered when poezio sends a new *Presence* stanza. The presence can thus be modified before being sent.

tab_change

- **old_tab:** `int` Old current tab.
- **new_tab:** `int` New current tab.

Triggered whenever the user switches between tabs.

10.1.3 SleekXMPP classes

class `slixmpp.Message` (**args*, ***kwargs*)

XMPP's `<message>` stanzas are a “push” mechanism to send information to other XMPP entities without requiring a response.

Chat clients will typically use `<message>` stanzas that have a type of either “chat” or “groupchat”.

When handling a message event, be sure to check if the message is an error response.

Example `<message>` stanzas:

```
<message to="user1@example.com" from="user2@example.com">
  <body>Hi! </body>
</message>

<message type="groupchat" to="room@conference.example.com">
  <body>Hi everyone! </body>
</message>
```

Stanza Interface:

- **body:** The main contents of the message.
- **subject:** An optional description of the message's contents.
- **mucroom:** (Read-only) The name of the MUC room that sent the message.

- **mucnick**: (Read-only) The MUC nickname of message's sender.

Attributes:

- **types**: May be one of: normal, chat, headline, groupchat, or error.

chat()

Set the message type to 'chat'.

del_mucnick()

Dummy method to prevent deletion.

del_mucroom()

Dummy method to prevent deletion.

del_parent_thread()

Delete the message thread's parent reference.

get_mucnick()

Return the nickname of the MUC user that sent the message.

Read-only stanza interface.

Return type `str`

get_mucroom()

Return the name of the MUC room where the message originated.

Read-only stanza interface.

Return type `str`

get_parent_thread()

Return the message thread's parent thread.

Return type `str`

get_type()

Return the message type.

Overrides default stanza interface behavior.

Returns 'normal' if no type attribute is present.

Return type `str`

normal()

Set the message type to 'normal'.

reply(body=None, clear=True)

Create a message reply.

Overrides StanzaBase.reply.

Sets proper 'to' attribute if the message is from a MUC, and adds a message body if one is given.

Parameters

- **body** (`str`) – Optional text content for the message.
- **clear** (`bool`) – Indicates if existing content should be removed before replying. Defaults to True.

Return type `Message`

set_mucnick(value)

Dummy method to prevent modification.

set_mucroom (*value*)

Dummy method to prevent modification.

set_parent_thread (*value*)

Add or change the message thread's parent thread.

Parameters *value* (*str*) – identifier of the thread

class `slixmpp.Presence` (**args*, ***kwargs*)

XMPP's <presence> stanza allows entities to know the status of other clients and components. Since it is currently the only multi-cast stanza in XMPP, many extensions add more information to <presence> stanzas to broadcast to every entry in the roster, such as capabilities, music choices, or locations (XEP-0115: Entity Capabilities and XEP-0163: Personal Eventing Protocol).

Since <presence> stanzas are broadcast when an XMPP entity changes its status, the bulk of the traffic in an XMPP network will be from <presence> stanzas. Therefore, do not include more information than necessary in a status message or within a <presence> stanza in order to help keep the network running smoothly.

Example <presence> stanzas:

```
<presence />

<presence from="user@example.com">
  <show>away</show>
  <status>Getting lunch.</status>
  <priority>5</priority>
</presence>

<presence type="unavailable" />

<presence to="user@otherhost.com" type="subscribe" />
```

Stanza Interface:

- **priority**: A value used by servers to determine message routing.
- **show**: The type of status, such as away or available for chat.
- **status**: Custom, human readable status message.

Attributes:

- **types**: One of: available, unavailable, error, probe, subscribe, subscribed, unsubscribe, and unsubscribed.
- **showtypes**: One of: away, chat, dnd, and xa.

del_type ()

Remove both the type attribute and the <show> element.

get_priority ()

Return the value of the <presence> element as an integer.

Return type `int`

get_type ()

Return the value of the <presence> stanza's type attribute, or the value of the <show> element.

reply (*clear=True*)

Create a new reply <presence/> stanza from `self`.

Overrides StanzaBase.reply.

Parameters `clear` (*bool*) – Indicates if the stanza contents should be removed before replying. Defaults to True.

set_priority (*value*)

Set the entity's priority value. Some server use priority to determine message routing behavior.

Bot clients should typically use a priority of 0 if the same JID is used elsewhere by a human-interacting client.

Parameters `value` (*int*) – An integer value greater than or equal to 0.

set_show (*show*)

Set the value of the <show> element.

Parameters `show` (*str*) – Must be one of: away, chat, dnd, or xa.

set_type (*value*)

Set the type attribute's value, and the <show> element if applicable.

Parameters `value` (*str*) – Must be in either self.types or self.showtypes.

class `slixmpp.Iq` (*args, **kwargs)

XMPP <iq> stanzas, or info/query stanzas, are XMPP's method of requesting and modifying information, similar to HTTP's GET and POST methods.

Each <iq> stanza must have an 'id' value which associates the stanza with the response stanza. XMPP entities must always be given a response <iq> stanza with a type of 'result' after sending a stanza of type 'get' or 'set'.

Most uses cases for <iq> stanzas will involve adding a <query> element whose namespace indicates the type of information desired. However, some custom XMPP applications use <iq> stanzas as a carrier stanza for an application-specific protocol instead.

Example <iq> Stanzas:

```
<iq to="user@example.com" type="get" id="314">
  <query xmlns="http://jabber.org/protocol/disco#items" />
</iq>

<iq to="user@localhost" type="result" id="17">
  <query xmlns='jabber:iq:roster'>
    <item jid='otheruser@example.net'
      name='John Doe'
      subscription='both'>
      <group>Friends</group>
    </item>
  </query>
</iq>
```

Stanza Interface:

- **query:** The namespace of the <query> element if one exists.

Attributes:

- **types:** May be one of: get, set, result, or error.

del_query ()

Remove the <query> element.

get_query ()

Return the namespace of the <query> element.

Return type `str`

reply (*clear=True*)

Create a new <iq> stanza replying to `self`.

Overrides `StanzaBase.reply`

Sets the ‘type’ to ‘result’ in addition to the default `StanzaBase.reply` behavior.

Parameters **clear** (*bool*) – Indicates if existing content should be removed before replying.
Defaults to `True`.

send (*callback=None, timeout=None, timeout_callback=None*)

Send an <iq> stanza over the XML stream.

A callback handler can be provided that will be executed when the Iq stanza’s result reply is received.

Returns a future which result will be set to the result Iq if it is of type ‘get’ or ‘set’ (when it is received), or a future with the result set to `None` if it has another type.

Overrides `StanzaBase.send`

Parameters

- **callback** (*function*) – Optional reference to a stream handler function. Will be executed when a reply stanza is received.
- **timeout** (*int*) – The length of time (in seconds) to wait for a response before the `timeout_callback` is called, instead of the regular callback
- **timeout_callback** (*function*) – Optional reference to a stream handler function. Will be executed when the timeout expires before a response has been received for the originally-sent IQ stanza.

Return type `asyncio.Future`

set_payload (*value*)

Set the XML contents of the <iq> stanza.

Parameters **value** (*list or XML object*) – An XML object or a list of XML objects to use as the <iq> stanza’s contents

set_query (*value*)

Add or modify a <query> element.

Query elements are differentiated by their namespace.

Parameters **value** (*str*) – The namespace of the <query> element.

unhandled ()

Send a feature-not-implemented error if the stanza is not handled.

Overrides `StanzaBase.unhandled`.

10.1.4 XEP

Table of all XEPs implemented in poezio.

XEP number	XEP name	Implementation status
0004	Data Forms	100%
0012	Last Activity	100%
0027	Current Jabber OpenPGP Usage	100%
0030	Service Discovery	~100%

Continued on next page

Table 1 – continued from previous page

XEP number	XEP name	Implementation status
0045	Multi-User Chat	~90%
0048	Bookmarks	90%
0049	Private XML Storage	100%
0050	Ad-Hoc Commands	70%
0054	vcard-temp	70%
0060	Publish-Subscribe	10%
0066	Out of Band Data	10%
0070	HTTP Request Verification	~100%
0071	XHTML-IM	~70%
0077	In-Band Registration	50%
0084	User Avatar	50%
0085	Chat State Notifications	80%
0092	Software Version	100%
0107	User Mood	100%
0108	User Activity	100%
0115	Entity capabilities	~20%
0118	User Tune	90%
0153	vCard-Based Avatars	50%
0163	Personal Eventing Protocol	100%
0172	User Nickname	100%
0175	Best Practices for Use of SASL ANONYMOUS	100%
0178	Best Practices for Use of SASL EXTERNAL with Certificates	100%
0184	Message Delivery Receipts	100%
0191	Blocking Command	95%
0196	User Gaming	70%
0198	Stream Management	100%
0199	XMPP Ping	100%
0202	Entity time	100%
0203	Delayed Delivery	100%
0224	Attention	100%
0231	Bits of Binary	30%
0245	The /me Command	80%
0249	Direct MUC Invitations	90%
0257	Client Certificate Management for SASL EXTERNAL	100%
0270	Compliance Suites 2010	Advanced Client
0280	Message Carbons	100%
0296	Best Practices for Resource Locking	0%
0297	Stanza Forwarding	100%
0302	Compliance Suites 2012	Advanced Client
0308	Last Message Correction	100%
0319	Last User Interaction In Presence	100%
0334	Message Processing Hints	100%
0352	Client State Indication	100%
0363	HTTP File Upload	100%
0364	Current OTR Usage	100%
0375	Compliance Suites 2016	Advanced Client + Core Mobile
0378	OTR Discovery	100%
0380	Explicit Message Encryption	90%
0392	Consistent Color Generation	100%

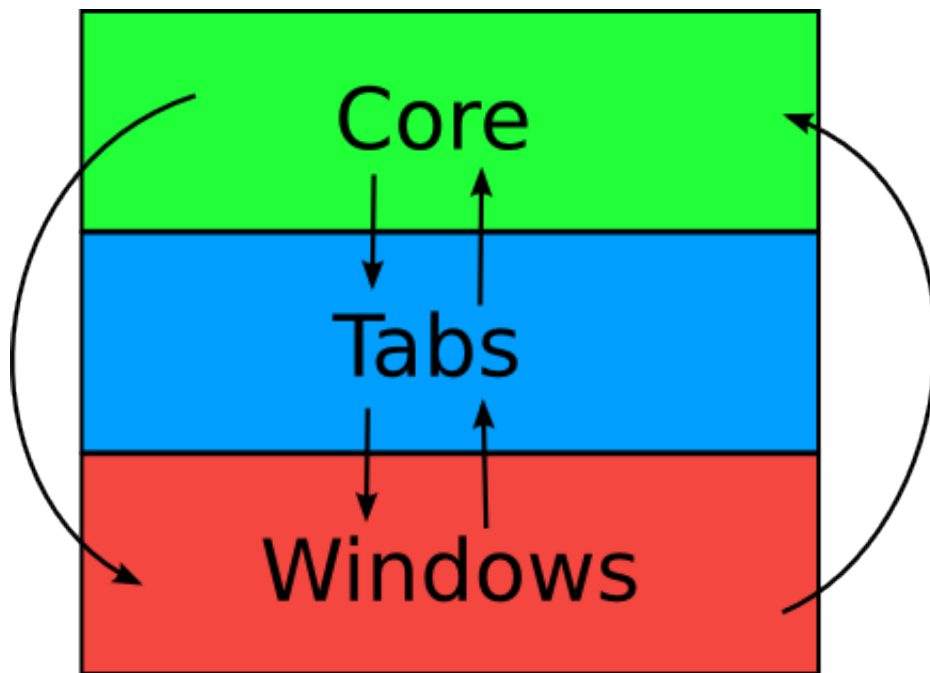
10.2 About Poezio

10.2.1 Overview

Note: This is not an introduction to XMPP, but to how poezio works.

Global overview

Poezio is an application that has three main layers, mostly separated in three different python modules: `core`, `tabs`, and `windows`. An UML diagram of Poezio would be inefficient, cluttered, or incomplete, so there is none, if that bugs you.



Core is mostly a “global” object containing the state of the application at any time, it contains the global commands, the xmpp event handlers, the list of open tabs, etc. Most objects in poezio have a `self.core` attribute referencing the **Core** (it’s a singleton, so there is never more than one instance). **Core** also contains the main loop of the application, which then dispatchs the I/O events (keypress) to the appropriate methods.

But the main loop is not the most important thing in poezio; because it is an IM client, it is essentially event-driven. The event part is handled by `slixmpp`, which is our fork of `sleekxmpp` to use `asyncio` instead of threads.

Tabs are the second layer of poezio, but the first dealing with the UI: each **Tab** is a layout of several **windows**, it contains tab-specific commands, tab-specific keybinds, and it has methods in order for core to interact with it, and some methods are only proxies for the methods of a **window**.

Example scenario: If someone presses the key `PageUp`, then **Core** will call the appropriate method on the current `_Tab_`, which will in turn, if it implements the method (inherited empty from the `Tab` class), call a scrolling method from the appropriate **window**.

All tabs types inherit from the class **Tab**, and the tabs featuring chat fonctionnality will inherit from **ChatTab** (which inherits from **Tab**).

Examples of **tabs**: MUCTab, XMLTab, RosterTab, MUCListTab, etc...

Event handlers

The events handlers are registered right at the start of poezio, and then when a matching stanza is received, the handler is called. The handlers are in **Core**, and then they call the appropriate methods in the corresponding **tabs**.

Example scenario: if a message is received from a MUC, then the **Core** handler will identify the **Tab**, and call the relevant handler from this **Tab**, this tab will in turn, add the message to the buffer, which will then add it to the relevant **windows**.

Note: All the `_windows_` that deal with received or generated text are linked to a **text_buffer**, in order to rebuild all the display lines from the sources if necessary. This also enables us to have several **windows** presenting the same text, even if they are not of the same size and layout.

Commands and completion

Commands are quite straightforward: those are methods that take a string as a parameter, and they do stuff.

From a user point of view, the methods are entered like that:

```
/command arg1 arg2
```

or

```
/command "arg1 with spaces" arg2
```

However, when creating a command, you will deal with `_one_` str, no matter what. There are utilities to deal with it (`common.shell_split`), but it is not always necessary. Commands are registered in the **commands** dictionary of a tab structured as key (command name) -> tuple(command function, help string, completion).

Completions are a bit tricky, but it's easy once you get used to it:

They take an **Input** (a `_windows_` class) as a parameter, named `the_input` everywhere in the sources. To effectively have a completion, you have to create a `poezio.core.structs.Completion` object initialized with the completion you want to call (**the_input.auto_completion()** or **the_input.new_completion()**) with the relevant parameters and return it with the function. Previously you would call the function directly from the completion method, but having side effects inside it makes it harder to test.

```
class Input(Win):
    # ...
    def auto_completion(completion_list, after='', quotify=True):
        # ...

    def new_completion(completion_list, argument_position, after='', quotify=True):
        # ...
```

Set the input to iterate over **completion_list** when the user hits tab, to insert **after** after the completed item, and surround the item with double quotes or not.

To find the current completed argument, use the **input.get_argument_position()** method. You can then use **new_completion()** to select the argument to be completed.

You can look for examples in the sources, all the possible cases are covered (single-argument, complex arguments with spaces, several arguments, etc...).

Note: Only `new_completion()` used together with `get_argument_position()` allow completing arguments that are not at the end of the command line, therefore it is preferable to use that and not `auto_completion()`.

Dealing with the command line

For convenience's sake, poezio includes a **decorators** module containing a **command_args_parser**, which can be used to filter the input easily.

Examples:

```
from decorators import command_args_parser
class MyClass(object):

    @command_args_parser.raw
    def command_raw(self, raw):
        # the "raw" parameter will be the raw input string

    @command_args_parser.ignored
    def command_ignored(self):
        # no argument is given to that function

    @command_args_parser.quoted(mandatory=1, optional=0)
    def command_quoted_1(self, args):
        # the "args" parameter will be a list containing one argument
```

See the source of the `CommandArgParser` for more information.

class poezio.decorators.**CommandArgParser**

Modify the string argument of the function into a list of strings containing the right number of extracted arguments, or None if we don't have enough.

10.2.2 Contributing

Conventions

We don't have a strict set of conventions, but you should respect PEP8 mostly (e.g. 4 spaces, class names in CamelCase and methods lowercased with underscores) except if it means less-readable code (80 chars is often a hassle, and if you look inside poezio you'll see lots of long lines, mostly because of strings).

As explained in the [Overview](#), "global" code goes in `core.py`, tab-related code goes in `tabs.py`, and ui-related code goes in `windows.py`. There are other modules (e.g. `xhtml.py`) but they do not matter for the application as a whole.

Commit guidelines

Commits **should** have a meaningful title (first line), and *may* have a detailed description below. There are of course exceptions (for example, a single-line commit that takes care of a typo right behind a big commit does not need to say fix a typo ("azre" → "are") in `toto.py` line 45454, since the metainfos already take care of that.), but if you do not have commit access on the poezio trunk, you can still reset and commit again.

Try to do atomic commits: since git is a DVCS, it doesn't hurt to `git add -p` and split the commit into several meaningful small commits ; on the contrary, it helps to track the changes on different levels.

If you have a conflict, solve it with rebase and not merge if the fast-forwards do not resolve it automatically in your case. This helps to avoid creating useless merges (and polluting the commit history) when none is needed.

```
git fetch origin
git rebase origin/master
git push origin master
```

If your commit is related to an issue on our [tracker](#) (or fixes such an issue), you can use `Fix #BUGID` or `References #BUGID` to help with the tracking.

Getting your code into poezio

If you have code you want to contribute, you can:

- Give us a patch and a description of what it does
- Give us a link to a **git** repo from which we can pull

The code is of course reviewed and tested a bit, but we trust the contributors to submit good code. If we can't integrate the given code into poezio (if it crashes or has some issues), if the size is small, we may tweak it ourselves and integrate it, and if not, you are of course free to take our advice into account and submit it again.

If you have already submitted some code and plan to do more, you can ask us direct commit access on the main repo.

10.2.3 Theming module

Define the variables (colors and some other stuff) that are used when drawing the interface.

Colors are numbers from -1 to 7 (if only 8 colors are supported) or -1 to 255 if 256 colors are available. If only 8 colors are available, all colors > 8 are converted using the `table_256_to_16` dict.

XHTML-IM colors are converted to -1 -> 255 colors if available, or directly to -1 -> 8 if we are in 8-color-mode.

A `pair_color` is a background-foreground pair. All possible pairs are not created at startup, because that would create 256*256 pairs, and almost all of them would never be used.

A theme should define color tuples, like `(200, -1)`, and when they are to be used by poezio's interface, they will be created once, and kept in a list for later usage. A color tuple is of the form `(foreground, background, optional)`. A color of -1 means the default color. So if you do not want to have a background color, use `(x, -1)`. The optional third value of the tuple defines additional information. It is a string and can contain one or more of these characters:

- `b`: bold
- `u`: underlined
- `x`: blink

For example, `(200, 208, 'bu')` is bold, underlined and pink foreground on orange background.

A theme file is a python file containing one object named 'theme', which is an instance of a class (derived from the Theme class) defined in that same file. For example, in `pinkytheme.py`:

```
import theming
class PinkyTheme(theming.Theme):
    COLOR_NORMAL_TEXT = (200, -1)

theme = PinkyTheme()
```

if the command `/theme pinkytheme` is issued, we import the `pinkytheme.py` file and set the global variable `'theme'` to `pinkytheme.theme`.

And in poezio's code we just use `theme.COLOR_NORMAL_TEXT` etc

Since a theme inherits from the `Theme` class (defined here), if a color is not defined in a theme file, the color is the default one.

Some values in that class are a list of color tuple. For example `[(1, -1), (2, -1), (3, -1)]` Such a list SHOULD contain at least one color tuple. It is used for example to define color gradient, etc.

class `poezio.theming.Theme`

The theme class, from which all themes should inherit. All of the following values can be replaced in subclasses, in order to create a new theme.

Do not edit this file if you want to change the theme to suit your needs. Create a new theme and share it if you think it can be useful for others.

10.2.4 Timed events documentation

Timed events are the standard way to schedule events for later in poezio.

Once created, they must be added to the list of checked events with `Core.add_timed_event()` (within poezio) or with `PluginAPI.add_timed_event()` (within a plugin).

class `poezio.timed_events.TimedEvent` (*date: datetime.datetime, callback: Callable, *args*)

An event with a callback that is called when the specified time is passed.

The callback and its arguments should be passed as the last arguments.

__init__ (*date: datetime.datetime, callback: Callable, *args*) → None
Create a new timed event.

Parameters

- **date** (*datetime.datetime*) – Time at which the callback must be run.
- **callback** (*function*) – The handler that will be executed.
- **args** – Optional arguments passed to the handler.

class `poezio.timed_events.DelayedEvent` (*delay: Union[int, float], callback: Callable, *args*)

A `TimedEvent`, but with the date calculated from now + a delay in seconds. Use it if you want an event to happen in, e.g. 6 seconds.

__init__ (*delay: Union[int, float], callback: Callable, *args*) → None
Create a new `DelayedEvent`.

Parameters

- **delay** (*int*) – The number of seconds.
- **callback** (*function*) – The handler that will be executed.
- **args** – Optional arguments passed to the handler.

10.2.5 Common operations documentation

Various useful functions.

`poezio.common.find_argument(pos: int, text: str, quoted=True) → int`

Split an input into a list of arguments, return the number of the argument selected by pos.

If the position searched is outside the string, or in a space between words, then it will return the position of an hypothetical new argument.

See the doctests of the two methods for example behaviors.

Parameters

- **pos** (*int*) – The position to search.
- **text** (*str*) – The text to analyze.
- **quoted** (*bool*) – Whether to take quotes into account or not.

Return type *int*

`poezio.common.find_delayed_tag(message: slxmpp.stanza.message.Message) → Tuple[bool, Optional[datetime.datetime]]`

Check if a message is delayed or not.

Parameters **message** (*slxmpp.Message*) – The message to check.

Returns A tuple containing (True, the datetime) or (False, None)

Return type *tuple*

`poezio.common.format_gaming_string(infos: Dict[str, str]) → str`

Construct a string from a dict containing “user gaming” information. (for now, only use address and name)

Parameters **infos** (*dict*) – Gaming information

Returns The formatted string

Return type *str*

`poezio.common.format_tune_string(infos: Dict[str, str]) → str`

Construct a string from a dict created from an “User tune” event.

Parameters **infos** (*dict*) – Tune information

Returns The formatted string

Return type *str*

`poezio.common.get_local_time(utc_time: datetime.datetime) → datetime.datetime`

Get the local time from an UTC time

`poezio.common.get_os_info() → str`

Returns a detailed and well formatted string containing information about the operating system

Return type *str*

`poezio.common.get_utc_time(local_time: Optional[datetime.datetime] = None) → datetime.datetime`

Get the current UTC time

Parameters **local_time** (*datetime*) – The current local time

Returns The current UTC time

`poezio.common.parse_secs_to_str(duration=0) → str`

Do the reverse operation of `parse_str_to_secs()`.

Parse a number of seconds to a human-readable string. The string has the form XdXhXmXs. 0 units are removed.

Parameters `duration` (*int*) – The duration, in seconds.

Returns A formatted string containing the duration.

Return type `str`

```
>>> parse_secs_to_str(3601)
'1h1s'
```

`poezio.common.parse_str_to_secs(duration="")` → `int`

Parse a string of with a number of d, h, m, s.

Parameters `duration` (*str*) – The formatted string.

Returns The number of seconds represented by the string

Return type `int`

```
>>> parse_str_to_secs("1d3m1h")
90180
```

`poezio.common.safeJID(*args, **kwargs)` → `slxmpp.jid.JID`

Construct a `slxmpp.JID` object from a string.

Used to avoid tracebacks during is stringprep fails (fall back to a JID with an empty string).

`poezio.common.shell_split(st: str)` → `List[str]`

Split a string correctly according to the quotes around the elements.

Parameters `st` (*str*) – The string to split.

Returns A list of the different of the string.

Return type `list`

```
>>> shell_split('"sdf 1" "toto 2"')
['sdf 1', 'toto 2']
```

CHAPTER 11

Indices and tables

- `genindex`
- `modindex`
- `search`

a

alias, 63
amsg, 63
autocorrect, 77

c

change_title, 79
close_all, 80
csi, 81
cyber, 81

d

day_change, 64
dice, 81
disco, 82
display_corrections, 64
double, 76

e

embed, 64
exec, 64

f

figlet, 65

i

iq_show, 76
irc, 78

l

link, 65

m

marquee, 82
mpd_client, 66

o

otr, 67

p

pacokick, 69
ping, 70
pipe_cmd, 79
poezio.common, 102
poezio.plugin, 85
poezio.theming, 101
poezio.timed_events, 102
pointpoint, 76

q

quote, 70

r

rainbow, 70
regex_admin, 76
reminder, 71
reorder, 80
replace, 71
revstr, 76

s

screen_detach, 72
send_delayed, 72
server_part, 82
shuffle, 76
simple_notify, 73
slixmpp, 92
spam, 74
status, 74

t

tell, 75
time_marker, 75

u

uptime, 75

v

vcard, 82

Symbols

`/accept`, 34
`/activity`, 29
`/add`, 34
`/affiliation`, 33
`/afk`, 74
`/alias`, 63
`/amsg`, 64
`/available`, 74
`/away`, 74
`/bind`, 29
`/block`, 35
`/bookmark`, 30
`/bookmark_local`, 30
`/bookmarks`, 30
`/busy`, 74
`/cert_add`, 35
`/cert_disable`, 35
`/cert_fetch`, 35
`/cert_revoke`, 35
`/certs`, 35
`/chat`, 74
`/clear`, 32
`/clear [Chatroom version]`, 33
`/clear [XML tab version]`, 36
`/close`, 30
`/closeall`, 80
`/color`, 33
`/configure`, 33
`/correct`, 32
`/csi_active`, 81
`/csi_inactive`, 81
`/cycle`, 33
`/deny`, 34
`/destroy_room`, 30
`/disco`, 82
`/display_corrections`, 64
`/dnd`, 74
`/done`, 71
`/dump`, 36
`/embed <image_url>`, 64
`/exec`, 64
`/exit`, 30
`/export`, 36
`/filter_from`, 36
`/filter_id`, 36
`/filter_jid`, 36
`/filter_reset`, 36
`/filter_to`, 36
`/filter_xmlmask`, 36
`/filter_xpath`, 36
`/gaming`, 30
`/groupadd`, 34
`/groupmove`, 35
`/groupremove`, 35
`/help`, 30
`/ignore`, 33
`/import`, 36
`/info`, 33, 34
`/invitations`, 30
`/invite`, 30
`/invite [Chatroom version]`, 33
`/irc_join`, 79
`/irc_login`, 79
`/irc_query`, 79
`/join`, 30
`/kick`, 33
`/last_activity`, 31
`/link`, 66
`/list`, 31
`/list_blocks`, 35
`/list_tell`, 75
`/load`, 31
`/marquee <text>`, 82
`/message`, 31
`/mood`, 31
`/move_tab`, 31
`/mpd`, 67
`/name`, 35

- /names, [33](#)
- /next, [31](#)
- /nick, [33](#)
- /otr, [68](#)
- /otrsmp, [68](#)
- /pacokick, [69](#)
- /part, [33](#)
- /password, [35](#)
- /ping, [70](#)
- /plugins, [31](#)
- /pointpoint, [77](#)
- /presence, [31](#)
- /prev, [31](#)
- /query, [33](#)
- /quit, [30](#)
- /quote, [70](#)
- /rainbow, [71](#)
- /rawxml, [31](#)
- /rban, [76](#)
- /recolor, [33](#)
- /reconnect, [35](#)
- /reload, [31](#)
- /remind, [71](#)
- /remove, [35](#)
- /remove_bookmark, [31](#)
- /reorder, [80](#)
- /rkick, [76](#)
- /role, [34](#)
- /roll [number of dice] [duration of the roll], [81](#)
- /runkey, [31](#)
- /save_order, [80](#)
- /say, [32](#)
- /self, [31](#)
- /send_delayed, [73](#)
- /server_cycle, [31](#)
- /server_part, [82](#)
- /set, [32](#)
- /set_default, [32](#)
- /status, [32](#)
- /tasks, [71](#)
- /tell, [75](#)
- /theme, [32](#)
- /toggle, [32](#)
- /topic, [34](#)
- /unalias, [63](#)
- /unblock, [35](#)
- /unignore, [34](#)
- /unload, [32](#)
- /unquery, [34](#)
- /untell, [75](#)
- /uptime, [75](#)
- /vcard, [83](#)
- /version, [32](#), [34](#)
- /w, [32](#)

- /win, [32](#)
- /xa, [74](#)
- /xhtml, [32](#)
- /xml_tab, [32](#)
- __init__() (poezio.timed_events.DelayedEvent method), [102](#)
- __init__() (poezio.timed_events.TimedEvent method), [102](#)

A

- ack_message_receipts, [8](#)
- add_command() (poezio.plugin.PluginAPI method), [86](#)
- add_event_handler() (poezio.plugin.PluginAPI method), [86](#)
- add_key() (poezio.plugin.PluginAPI method), [86](#)
- add_slix_event_handler() (poezio.plugin.PluginAPI method), [86](#)
- add_space_after_completion, [13](#)
- add_tab_command() (poezio.plugin.PluginAPI method), [86](#)
- add_tab_key() (poezio.plugin.PluginAPI method), [87](#)
- add_timed_event() (poezio.plugin.PluginAPI method), [87](#)
- Admin, [60](#)
- after_command, [74](#)
- after_completion, [14](#)
- after_quote, [70](#)
- Alias, [60](#)
- alias (module), [63](#)
- all_tabs() (poezio.plugin.PluginAPI method), [87](#)
- alternative_nickname, [6](#)
- Amsg, [60](#)
- amsg (module), [63](#)
- api (poezio.plugin.BasePlugin attribute), [85](#)
- auto_reconnect, [7](#)
- Autocorrect, [60](#)
- autocorrect (module), [77](#)
- autorejoin, [7](#), [16](#)
- autorejoin_delay, [7](#), [16](#)

B

- BasePlugin (class in poezio.plugin), [85](#)
- beep_on, [14](#)
- before_quote, [70](#)
- bookmark_on_join, [8](#)
- browser, [66](#)

C

- ca_cert_path, [5](#)
- certfile, [6](#)
- certificate, [6](#)
- change_title (module), [79](#)
- changing_nick, [89](#)
- chat() (slixmpp.Message method), [93](#)
- ciphers, [6](#)

cleanup() (poezio.plugin.BasePlugin method), 85
 Close all, 60
 close_all (module), 80
 command, 74
 CommandArgParser (class in poezio.decorators), 100
 connection_check_interval, 8
 connection_timeout_delay, 8
 conversation_chatstate, 89
 conversation_msg, 90
 conversation_say, 90
 conversation_say_after, 90
 core (poezio.plugin.BasePlugin attribute), 85
 create_delayed_event() (poezio.plugin.PluginAPI method), 87
 create_gaps, 10
 create_timed_event() (poezio.plugin.PluginAPI method), 87
 CSI, 60
 csi (module), 81
 current_tab() (poezio.plugin.PluginAPI method), 88
 custom_host, 6
 custom_port, 6
 Cyber, 60
 cyber (module), 81

D

Day Change, 60
 day_change (module), 64
 decode_entities, 69
 decode_newlines, 69
 decode_xhtml, 69
 default_duration, 81
 default_nick, 6
 del_command() (poezio.plugin.PluginAPI method), 88
 del_event_handler() (poezio.plugin.PluginAPI method), 88
 del_key() (poezio.plugin.PluginAPI method), 88
 del_mucnick() (slixmpp.Message method), 93
 del_mucroom() (slixmpp.Message method), 93
 del_parent_thread() (slixmpp.Message method), 93
 del_query() (slixmpp.Iq method), 95
 del_slix_event_handler() (poezio.plugin.PluginAPI method), 88
 del_tab_command() (poezio.plugin.PluginAPI method), 88
 del_tab_key() (poezio.plugin.PluginAPI method), 88
 del_type() (slixmpp.Presence method), 94
 delay, 74
 DelayedEvent (class in poezio.timed_events), 102
 deterministic_nick_colors, 10
 Dice, 60
 dice (module), 81
 disable_beep, 16
 Disco, 60

disco (module), 82
 Display corrections, 60
 display_activity_notifications, 8, 16
 display_corrections (module), 64
 display_gaming_notifications, 8, 16
 display_mood_notifications, 8, 16
 display_tune_notifications, 8, 16
 display_user_color_in_join_part, 10, 16
 Double, 60
 double (module), 76

E

Embed, 61
 embed (module), 64
 enable_avatars, 8
 enable_carbons, 8
 enable_css_parsing, 8
 enable_smacks, 9
 enable_user_activity, 9
 enable_user_gaming, 9
 enable_user_mood, 9
 enable_user_nick, 9
 enable_user_tune, 9
 enable_vertical_tab_list, 10
 enable_xhtml_im, 9
 eval_password, 17
 Exec, 61
 exec (module), 64
 exec_remote, 15
 extract_inline_images, 15

F

Figlet, 61
 figlet (module), 65
 filter_info_messages, 10
 find_argument() (in module poezio.common), 102
 find_delayed_tag() (in module poezio.common), 103
 force_encryption, 6
 force_remote_bookmarks, 9
 format_gaming_string() (in module poezio.common), 103
 format_tune_string() (in module poezio.common), 103
 frequency, 81

G

gateway, 78
 get_conversation_messages() (poezio.plugin.PluginAPI method), 88
 get_local_time() (in module poezio.common), 103
 get_mucnick() (slixmpp.Message method), 93
 get_mucroom() (slixmpp.Message method), 93
 get_os_info() (in module poezio.common), 103
 get_parent_thread() (slixmpp.Message method), 93
 get_priority() (slixmpp.Presence method), 94
 get_query() (slixmpp.Iq method), 95

get_status() (poezio.plugin.PluginAPI method), 88
 get_type() (slixmpp.Message method), 93
 get_type() (slixmpp.Presence method), 94
 get_utc_time() (in module poezio.common), 103
 go_to_previous_tab_on_alt_number, 9
 group_corrections, 9

H

hide_exit_join, 10, 17
 hide_status_change, 10, 17
 hide_user_list, 11
 highlight, 90
 highlight_on, 11, 17

I

ignore_certificate, 6
 ignore_private, 17
 ignored_private, 90
 information() (poezio.plugin.PluginAPI method), 88
 information_buffer_popup_on, 11
 information_buffer_type_filter, 11
 init() (poezio.plugin.BasePlugin method), 85
 initial_connect, 78
 Iq (class in slixmpp), 95
 IQ Show, 61
 iq_show (module), 76
 IRC, 61
 irc (module), 78

J

jid, 7
 joining_muc, 90

K

keyfile, 7
 keys_dir, 69

L

lang, 15
 lazy_resize, 15
 Link, 61
 link (module), 65
 load_log, 14, 17
 log, 69
 log_dir, 14
 log_errors, 14
 login_command, 78
 login_nick, 78

M

Marquee, 61
 marquee (module), 82
 max_lines_in_memory, 15

max_messages_in_memory, 15
 max_nick_length, 11
 Message (class in slixmpp), 92
 MPD Client, 61
 mpd_client (module), 66
 muc_ban, 90
 muc_chatstate, 90
 muc_colors (section), 11
 muc_join, 90
 muc_kick, 91
 muc_msg, 91
 muc_nickchange, 91
 muc_presence, 91
 muc_say, 91
 muc_say_after, 91
 muc_too, 74

N

nick_color_aliases, 11
 nickname, 78
 normal() (slixmpp.Message method), 93
 normal_presence, 91
 notify_messages, 17

O

open_all_bookmarks, 7
 OTR, 61
 otr (module), 67

P

PacoKick, 61
 pacokick (module), 69
 padding, 82
 parse_secs_to_str() (in module poezio.common), 103
 parse_str_to_secs() (in module poezio.common), 104
 password, 7, 17
 Ping, 61
 ping (module), 70
 Pipe Command, 61
 pipe_cmd (module), 79
 pipename, 79
 PluginAPI (class in poezio.plugin), 86
 plugins_autoload, 14
 plugins_conf_dir, 14
 plugins_dir, 15
 poezio.common (module), 102
 poezio.plugin (module), 85
 poezio.theming (module), 101
 poezio.timed_events (module), 102
 PointPoint, 61
 pointpoint (module), 76
 popup_time, 11
 Presence (class in slixmpp), 94
 private_auto_response, 17

private_chatstate, [91](#)
 private_msg, [91](#)
 private_say, [92](#)
 private_say_after, [92](#)

Q

Quote, [61](#)
 quote (module), [70](#)

R

Rainbow, [61](#)
 rainbow (module), [70](#)
 refresh, [81](#), [82](#)
 Regex Admin, [61](#)
 regex_admin (module), [76](#)
 Reminder, [61](#)
 reminder (module), [71](#)
 remote_fifo_path, [15](#)
 remove_timed_event() (poezio.plugin.PluginAPI method), [89](#)
 Reorder, [62](#)
 reorder (module), [80](#)
 Replace, [62](#)
 replace (module), [71](#)
 reply() (slixmpp.Iq method), [96](#)
 reply() (slixmpp.Message method), [93](#)
 reply() (slixmpp.Presence method), [94](#)
 request_message_receipts, [9](#)
 require_encryption, [69](#)
 Revstr, [62](#)
 revstr (module), [76](#)
 rooms, [7](#)
 rooms [IRC plugin], [78](#)
 roster_group_sort, [11](#)
 roster_show_offline, [12](#)
 roster_sort, [12](#)
 run_command() (poezio.plugin.PluginAPI method), [89](#)

S

safeJID() (in module poezio.common), [104](#)
 save_status, [15](#)
 Screen Detach, [62](#)
 screen_detach (module), [72](#)
 self_ping_delay, [17](#)
 Send Delayed, [62](#)
 send() (slixmpp.Iq method), [96](#)
 send_chat_states, [9](#), [18](#)
 send_delayed (module), [72](#)
 send_initial_presence, [15](#)
 send_message() (poezio.plugin.PluginAPI method), [89](#)
 send_normal_presence, [92](#)
 send_os_info, [9](#)
 send_poezio_info, [10](#)

send_time, [10](#)
 separate_history, [14](#)
 server, [7](#)
 Server Part, [62](#)
 server_part (module), [82](#)
 set_mucnick() (slixmpp.Message method), [93](#)
 set_mucroom() (slixmpp.Message method), [93](#)
 set_parent_thread() (slixmpp.Message method), [94](#)
 set_payload() (slixmpp.Iq method), [96](#)
 set_priority() (slixmpp.Presence method), [95](#)
 set_query() (slixmpp.Iq method), [96](#)
 set_show() (slixmpp.Presence method), [95](#)
 set_type() (slixmpp.Presence method), [95](#)
 shell_split() (in module poezio.common), [104](#)
 show_composing_tabs, [12](#)
 show_inactive_tabs, [12](#)
 show_jid_in_conversations, [12](#)
 show_muc_jid, [12](#)
 show_roster_jids, [12](#)
 show_roster_subscriptions, [12](#)
 show_s2s_errors, [13](#)
 show_tab_names, [13](#)
 show_tab_numbers, [13](#)
 show_timestamps, [13](#)
 show_useless_separator, [18](#)
 Shuffle, [62](#)
 shuffle (module), [76](#)
 Simple notify, [62](#)
 simple_notify (module), [73](#)
 slixmpp (module), [92](#)
 Spam, [62](#)
 spam (module), [74](#)
 Status, [62](#)
 status, [7](#)
 status (module), [74](#)
 status_message, [7](#)

T

tab_change, [92](#)
 Tell, [62](#)
 tell (module), [75](#)
 theme, [13](#)
 Theme (class in poezio.theming), [57](#), [102](#)
 themes_dir, [13](#)
 Time Marker, [62](#)
 time_marker (module), [75](#)
 TimedEvent (class in poezio.timed_events), [102](#)
 timeout, [69](#)
 Title change, [62](#)
 tmp_image_dir, [15](#)
 total_duration, [82](#)

U

unhandled() (slixmpp.Iq method), [96](#)

Upload, [62](#)
Uptime, [62](#)
uptime (module), [75](#)
use_bookmark_method, [10](#)
use_csi, [72](#)
use_log, [14](#), [18](#)
use_pep_nick, [10](#)
use_remote_bookmarks, [10](#)
use_screen, [72](#)
use_tab_nicks, [13](#)
use_tmux, [72](#)
user_list_sort, [13](#)

V

vCard, [62](#)
vcard (module), [82](#)
vertical_tab_list_size, [13](#)
vertical_tab_list_sort, [13](#)

W

whitespace_interval, [8](#)
words, [14](#)