PyPickupBot Documentation

Release 0.1a

epsy

May 14, 2013

CONTENTS

Contents:

CHAPTER

ONE

INDICES AND TABLES

- *genindex*
- *search*
- *commands*
- *settings*

CHAPTER

TWO

TABLE OF CONTENTS

2.1 Quick Setup

PyPickupBot is a small-sized IRC bot primarily aimed at gaming channels.

The main purpose of PyPickupBot is to run pickup games: Players sign up for one, and once a game is deemed to have enough players, captains will be picked from the signed-up players and will start picking players in the aim of playing an organized team game.

PyPickupBot however isn't limited to this activity, which can be in fact disabled entirely, at the same title as any loadable modules. It can be extended at will with plugins that contribute to different tasks: channel management, etc.

This guide is a tutorial in the configuration and basic use of PyPickupBot.

2.1.1 Configuring your bot

PyPickupBot's config is separated into two files: `init.cfg` and `config.cfg`. The former typically includes general bot configuration(nickname, command prefix, ..) connection details such as irc server, channels to join and most importantly modules to load. The latter is where all plugin configuration goes, more or less everything else.

Creating a config dir

You should first create a directory to contain the config files:

```
mkdir ~/my_shiny_pypickupbot_config/
cd ~/my_shiny_pypickupbot_config/
```

Note that it will also host the bot's persistent database where a lot of stuff will be saved, for instance started games logs or channel bans. This makes it convenient because the config directory effectively becomes all the bot's knowledge, making it easy to save or transfer. Always remember however, that you are likely to have a password or two in the config(NickServ/Q authentication).

init.cfg

As precised earlier, `init.cfg` only contains few items.

```
[Bot]
nickname=ZeroBot

[Server]
host=irc.quakenet.org
```

```
port=6667
channels=#qlpickup.eu
username=ZeroBot
password=secret

[Modules]
modules+=q_auth
```

We will describe here the most important ones.

Bot section

```
[Bot]
nickname=JohnBot
```

The bot's nickname, as it will appear to other people.

Server Section

```
[Server]
host=irc.quakenet.org
port=6667
channels=#example
```

host IRC Server address to connect to.

port IRC Server port.

channels Channels to automatically join once connected.

If you plan on running your bot on a network that allows authenticating via server username/password(like FreeNode), or if you connect to a bouncer, these might be of use:

```
[Server]
...
username=johnbotaccount
password=secret
```

username Username given to the server when connecting. It isn't necessarily the *nickname* that people will see.

password Password given to the server when connecting.

Modules Section

PyPickupBot's main functionality comes from modules(also known as plug-ins). In fact, without modules, it could only join a channel then sit there and do nothing. Hopefully, several modules are preloaded:

- ban — manages channel bans
- chanops — makes channel operators bot admins
- help — gives command lists and help about specific commands
- info — gives general information about the bot
- pickup — used to run pickup games
- pickup_playertracking — records started games and maintains a top 10 of active participants

- topic — allows the admins to change parts of the bot-maintained channel topic

Translated into PyPickupBot's config, it would read like this(you don't have to type it):

```
[Modules]
modules=ban, chanops, help, info, pickup, pickup_playertracking, topic
```

If you plan on running your bot on a network using UnrealIRCd, such as QuakeNet, you can use the q_auth module to allow your bot to auth with Q before joining channels. You can tell PyPickupBot to load q_auth in addition to the preloaded modules like this:

```
[Modules]
modules+=q_auth
```

You don't need to repeat the preloaded modules at all in your config, as long as you use the `modules+=` syntax.

config.cfg

`config.cfg` however contains most of the settings:

```
[Q Auth]
username=JohnBot
password=secret

[Pickup games]
tdm= 4v4 TDM
ctf= 4v4 CTF

[Pickup: tdm]
captains=2
players=8

[Pickup: ctf]
captains=2
players=8
```

Q Auth

If you enabled the q_auth module as described earlier, you need to configure it. Typically, you only need to enter the account's username and password:

```
[Q Auth]
username=JohnBot
password=secret
```

Pickup

First, define in the `Pickup games` section the games that can be played:

```
[Pickup games]
ctf= 5v5 CTF
tdm= 4v4 TDM
```

Each config option here is treated as a game. The left-hand value is the short name for the game, that people will use to join it. The right-hand value is a title for the game. It can be anything you like, it isn't interpreted anyway. It is shown in the output of `!pickups` and when a game fills up and starts.

For each game, you can have an optional section defining the game's settings:

```
[Pickup: ctf]
captains=2
players=10
```

You can omit it if the game's settings matches the defaults, which are as follows:

```
[Pickup: tdm]
captains=2
players=8
```

Your bot is now configured!

Running the bot

To start the bot, run the `pickupbot` script from the main folder as follows:

```
/path/to/pickupbot -c ~/my_shiny_pypickupbot_config/
```

It will do it's thing and connect, currently with a lot of output you probably don't care about.

You usually want to run it within GNU screen or a similar application, so the bot can continue running when you close the terminal or when you disconnect from the server the bot is hosted on.

2.1.2 Using the bot

Players interested in playing a game say `!add <game>` in the bot's main channel, replacing `<game>` with the short name of the game they want to play, for instance `!add ctf`. You can list or search game types with the `!pickups` command.

Once the maximum number of players for a game is reached, players are removed from all the queues (for instance if they added up for two different games at once), and the bot announces the upcoming game with it's players and it's randomly-selected captains.

Players then proceed to the game server, captains each go on one team and they start picking players turn by turn. Once that step is done, they are ready to start their game.

Common commands

!add [game [game ...]] **!add** ([*game*[, *game*, ...]])
: Signs you up for one or more games. If no game is provided, you are signed up for all games.

!remove [game [game ...]] **!remove** ([*game*[, *game*, ...]])
: Removes you from one or more games. Same as with `!add` for when no games are provided.

!who [game [game ...]] **!who** ([*game*[, *game*, ...]])
: Lists who is currently signed up for a game.

!lastgame **!lastgame**
: Shows information about the last played game, such as time elapsed since it started and participating players.

!motd [message] **!motd** ([*message*])
: Sets a message to appear at the end of the channel topic. If no message is provided, it shows the current one. If you type `!motd --` is provided as message, the current message is removed.

Note that `!motd` is only available for channel admins. By default, channel admins are the channel's operators. There are many more commands listed in the user manual.

2.2 Using PyPickupBot

This part of the documentation is intended for people who have already installed PyPickupBot and would like to know more about what they can do with it and its included plugins.

2.2.1 Configuration files and syntax

Configuration files

PyPickupBot's configuration is done via a pair of configuration files, `init.cfg` and `config.cfg`. They follow INI syntax: Each setting is defined by a line such as `name=value`. Settings have to be grouped under sections, which are defined by a line with the section's name in square brackets. Here's an example:

```
[First section]
setting 1=1st value

[Second section]
setting 2=2nd value
setting 3=3rd value
```

init.cfg

`init.cfg` is read right the moment PyPickupBot is started. It contains start-critical information such as connection details and the list of modules to load.

config.cfg

`config.cfg` is read right before modules are loaded, but after each module's default settings. This is where you should keep your module settings.

Settings in this manual

In this manual, for the sake of concise-ness, settings appear with their appropriate section on the same line. (Normally the section would appear on it's own line and only once.) If a setting is optional, the default value will be shown after the = sign:

[Example Section]Example setting = no (bool) `[Example Section]` **`Example setting` = no** (`Bool`)
: `Example setting`'s default value is `no`, you are not forced to have it in your config unless that value doesn't suit your needs.

For settings that are required, no default value is provided and *Required* appears next to it.

[Example Section]Required setting = #REQUIRED (int) `[Example Section]` **`Required setting` (Required)** (`Int`)
: This setting is required.

If a setting is needed at start and needs to be put in `init.cfg`, it will appear like this:

[Server]host = #REQUIRED (string) `[Server]` **`host` (Required) (Must be set in init.cfg)** (`String`)
: This setting must appear in `init.cfg`.

For non-required settings:

[Server]port = 6667 (int) `[Server]` **`port` = 6667 (Must be set in init.cfg)** (`Int`)
: This setting must appear in `init.cfg` if you need to change it.

In parenthesis is the type of value the setting requires. The next section will describe each of them briefly, and you can access each type's description by clicking it in the setting's signature.

For a quick reference of all settings, you can visit the *setting index*, which is also accessible from every page of this manual in the top and bottom right-hand corners.

Setting types

bool
: Can be `yes` or `no`, or any equivalent.

int
: An integer, like `0`, `3` or `140`.

float
: A floating point number, like `0.5` or `5.3`

string
: Text.

list
: A space or comma-separated list.

```
modules=pickup help
teamnames=Team 1, Team 2, Team 3
```

You can also extend a list *once* by defining a + setting. For instance:

```
modules=pickup help
modules+=ban pickup_playertracking
```

is equivalent to:

```
modules=pickup help ban pickup_playertracking
```

You can also remove items with −:

```
modules=pickup help q_auth ban
modules-=q_auth ban
```

Alternatively, you can use the one-item-per-line style:

```
modules[]=5
modules[0]=pickup
modules[1]=help
modules[2]=q_auth
modules[3]=ban
modules[4]=pickup_playertracking
```

The `[]` entry tells the number of items in the list, each `[n]` line defines one item.

dict
: Same as a `List`, except that each entry is in the form of `key:value`:

```
channel passwords=#priv1:secret,#worldgov:topsecret
```

Each key must be unique.

duration
: A duration, expressed in a format like `1 day 2 hours` or `1d2h`.

Each duration string is composed of one or more number/unit pairs. Units are, along with their short name:

- `years` (`y`, 365 days)
- `months` (`mo`, 30 days)
- `weeks` (`w`)
- `days` (`d`)
- `hours` (`h`)
- `minutes` (`m`)
- `seconds` (`s`)

One pair could be something like `5 days`. Different pairs can be separated by a comma and/or a space. All spaces and commas can be safely removed. Most commands which use the same format as this will require you to use the compact format (no spaces, no commas).

Some examples and their short forms:

- `1 year`: `1y`
- `3 months`: `3mo`
- `10 minutes`: `10m`
- `2 hours, 30 minutes`: `2h30m`

2.2.2 Connection settings

Defining a server

[Server]host = #REQUIRED (string) `[Server]` **`host` (Required) (Must be set in init.cfg)** (`String`)
The IRC host to connect to.

[Server]port = 6667 (int) `[Server]` **`port` = 6667 (Must be set in init.cfg)** (`Int`)
The port to connect to.

[Server]ssl = no (bool) `[Server]` **`ssl` = no (Must be set in init.cfg)** (`Bool`)
Use SSL? Don't forget to adjust the port if you do.

[Server]username = (string) `[Server]` **`username` = (Must be set in init.cfg)** (`String`)
Username when connecting. (**Not** the nickname.)

[Server]password = (string) `[Server]` **`password` = (Must be set in init.cfg)** (`String`)
Password when connecting.

[Server]realname = (string) `[Server]` **`realname` = (Must be set in init.cfg)** (`String`)
`ircname`/`realname` to appear when people `/whois` the bot.

[Server]channels = #REQUIRED (list) `[Server]` **`channels` (Required) (Must be set in init.cfg)** (`List`)
What channels to join? (Currently limited to one)

[Server]channel passwords = (dict) `[Server]` **`channel passwords` = (Must be set in init.cfg)** (`Dict`)
Channel passwords in `channel: password` format, if needed.

The bot's nickname is defined `in the Bot section`

Howto

Username/Password authentication

Some servers allow you to authenticate using user/password at connect time. This is the case of most bouncer software and of FreeNode's NickServ:

```
[Server]

username=username
password=secret
```

Q Authentication

See *q_auth: Q authentication* after adding `q_auth` to your `module list`.

2.2.3 Invocation

2.2.4 Usage

Using commands

Commands in this manual

Bot Settings

[Bot]nickname = pypickupbot (string) `[Bot]` **`nickname = pypickupbot (Must be set in init.cfg)`** (`String`)
: The bot's nickname when connecting to IRC

[Bot]command prefix = ! (string) `[Bot]` **`command prefix = ! (Must be set in init.cfg)`** (`String`)
: Short prefix to address commands to the bot.

[Bot]allow mentions = no (bool) `[Bot]` **`allow mentions = no (Must be set in init.cfg)`** (`Bool`)
: Allow the bot to be addressed by mentionning its nickname.

[Bot]warn on unknown command = yes (bool) `[Bot]` **`warn on unknown command = yes (Must be set in init.cfg)`** (`Bool`)
: Warn users when they address the bot with an unknown command.

[Bot]max reply splits before waiting = 2 (int) `[Bot]` **`max reply splits before waiting = 2 (Must be set in init.cfg)`** (`Int`)
: When messages are too long to fit in one IRC message, the bot splits it into multiple messages. This is how many messages the bot will send before asking the user to use the `!more` command.

[Bot]debug = no (bool) `[Bot]` **`debug = no (Must be set in init.cfg)`** (`Bool`)
: Wether to enable debug mode. Mainly consists of printing every message received from IRC to standard output. Can be forced on at runtime with the `-d` switch.

2.2.5 Modules

The main functionality of the bot is provided by modules.

`pickup`: Organized games

`pickup` is the module that handles games to be signed up for. It lets players add up for each of them until one of them is full, at which point the game is started and the bot announces who is signed up. Rudimentary game recording – namely the `!lastgame` and `!top10` commands – is provided by the `pickup_playertracking` module(See *pickup_playertracking: Simple game recording*).

Quick setup for this plugin is explained in the *Quick Setup guide*.

Usage

!pickups [search] **`!pickups`** ([*search*])
: Lists available games.

!add [game [game ...]] **`!add`** ([*game*[, *game*, ...]])
: Lets you sign up for available games. Specify one or more games by their short handle(the name outside the parenthesis in `!pickups`). If no game is specified, signs up for all available games.

!remove [game [game ...]] **`!remove`** ([*game*[, *game*, ...]])
: Removes you from one or more games. Removes you from all games if no game is specified. You are automatically removed from all games when you leave.

!who [game [game ...]] **`!who`** ([*game*[, *game*, ...]])
: Lists people who are signed up. If you specify one or more games it will filter the result.

!promote game **`!promote`** (*game*)
: Promotes the specified game with a message inviting people to join.

By default, the channel topic will be updated each time someone adds/removes to reflect current player counts.

Admin commands

!pull player [game [game ...]] **`!pull`** (*player*[, *game*[, *game*, ...]])
: Same as `!remove`, except it lets you specify another player as you.

!start game **`!start`** (*game*)
: Immediately starts a game even if it isn't full yet. However, there should at least be enough players for captains to be chosen.

!abort [game [game ...]] **`!abort`** ([*game*[, *game*, ...]])
: Removes all players from one or more games.

Settings

Settings are organized under 3 headings:

- `[Pickup]`, which contains
- The games list, `[Pickup games]`
- Individual settings for each game

General Settings

```
[Pickup]
promote delay=180
PM each player on start=yes
implicit all games in add=yes
topic=1
```

[Pickup]promote delay = 3min (duration) `[Pickup]` **`promote delay = 3min`** (`Duration`)
: Minimum delay in seconds between two `promote` calls, to prevent spam. Regardless of this setting, players also have to be signed up for a game in order to promote it. Channel admins bypass these restrictions.

[Pickup]PM each player on start = yes (bool) `[Pickup]` **`PM each player on start = yes`** (`Bool`)
: If set to `yes`, the bot will private-message signed up players when their game starts. While useful, it requires the bot to take the time to send each of these PMs with a delay in order not to be killed from the chat network for spam, which makes it appear as slow.

[Pickup]implicit all games in add = yes (bool) `[Pickup]` **`implicit all games in add = yes`** (`Bool`)
: If set to no, the ability for players to use `add` without an argument (to sign up for all available games) is removed.

[Pickup]topic = 1 (int) `[Pickup]` **`topic = 1`** (`Int`)
: As mentioned previously, the bot regularly updates the channel topic in order to reflect player counts for each game. This is the default behavior, with `topic` set to `1`. Set `topic` to `0` to entirely disable that feature. Set `topic` to `2` to only show games which have at least one player signed up.

Games List Each game is to be defined in the games list by a `Short name=Full name pair`, under the `[Pickup games]` section:

```
[Pickup games]
ctf=4v4 Capture The Flag
tdm=5v5 Team Deathmatch
```

A player would add up for one of these with `!add ctf` or `!add tdm`.

Additionally, you can set an order for the games to appear by in the channel topic with the `order` setting in the same section as the games. Order is otherwise unpredictable.

[Pickup games]order = (list) `[Pickup games]` **`order =`** (`List`)
: This setting should be written at the beginning of the section. It should list the order you want to give to the games. Name each game by their short name:

  ```
  [Pickup games]
  order=tdm,ctf
  ctf=4v4 Capture The Flag
  tdm=5v5 Team Deathmatch
  ```

  If a game is not mentioned in the order but is defined later, it will be appended to the end of it. If a game appears in the `order` setting but isn't defined, it is ignored. If you place the `order` setting at the end of the section, games not listed in the setting will never appear in the channel topic.

Game-specific settings For each game, you can have a section named `[Pickup: shortname]` with settings specific to it.

For instance, to define the 5v5 TDM game from the previous example, one would have:

```
[Pickup: tdm]
captains=2
players=10
```

If the section is omitted, the bot will assume the game is to be run with 8 players, 2 of which are captains, with no automatic team picking. This suits the 4v4 CTF example, and therefore you wouldn't have to define a section with settings for it.

Here are all the settings available in this section:

[Pickup: shortname]players = 8 (int) `[Pickup: shortname]` **players = 8** (`Int`)
Amount of players needed for the game to be full.

[Pickup: shortname]captains = 2 (int) `[Pickup: shortname]` **captains = 2** (`Int`)
Number of captains to be chosen. If set to `0`, the game will have no captains. If autopick is enabled(see below), this is the number of teams to be created.

[Pickup: shortname]autopick = no (bool) `[Pickup: shortname]` **autopick = no** (`Bool`)
If set to yes, the bot will automatically pick team members when the game is full. There are no captains, but the `captains` setting is used to determine how many teams should be created.

[Pickup: shortname]teamnames = Team 1, Team 2, ... (list) `[Pickup: shortname]` **teamnames = Team 1, Team 2, ...** (`List`)
Set this to a list of team names if you want custom team names in autopick mode:

```
[Pickup: 4wctf]
autopick=yes
teamnames=Team blue, Team gold, Team red, Team green
```

If `autopick` is enabled and this setting isn't given, generic team names will be used, such as *Team 1* and *Team 2*.

`pickup_playertracking`: Simple game recording

Usage

!lastgame [game [game ...]] **`!lastgame`** (*[game[, game, ...]]*)
!lastgame #id **`!lastgame`** (*#id*)
Tells you what the last started game was, and who played in it. Specify one or more games to filter the search.

The *#id* form lets you lookup a started game by it's ID. Everytime a game is started, the bot will print publicly the game's ID number. You can also find game IDs with the `!lastgames` command.

!lastgames [game [game ...]] **`!lastgames`** (*[game[, game, ...]]*)
Lists recently started games.

!top10 **`!top10`**
Lists most participating players over the course of one week (or `top10 spread`).

Admin commands

!purgegames [keep] **`!purgegames`** (*[keep]*)
Clears the bot's database of all games recorded longer than *keep* ago. (Defaults to `top10 spread`)

!cleargames **`!cleargames`**
Clears the database of all recorded games.

Settings

[Pickup player tracking]top10 spread = 1week (duration) `[Pickup player tracking]` **top10 spread = 1week** (`Duration`)
Duration `!top10` goes back to.

`ban`: Enhanced banmask tracking

All commands described here are only available to bot admins. `!ban` and `!unban` are only enabled once the bot is a channel operator.

Creating bans

!ban subject [duration] [reason] **!ban** (*subject*[, *duration*][, *reason*])
Creates a ban on subject for the given duration and reason and kicks anyone who matches the created ban.

See the `Duration` type for the *duration* option. *reason* can be anything. It will show in the banlist and in the kick message(s) if applicable.

The bot will do its best to find out the correct hostmask to ban:

- If *subject* is a full hostmask and contains wildcards like `*`, the bot will create a ban on that hostmask.
- If *subject* is a full hostmask and contains no wildcards, the bot will apply a wildcard on the username part and another on the ident part if no identd authentified it(~), leaving just the host part, sometimes with the identd.
- If *subject* is the nickname of someone present in the channel, the bot will select the user's full hostmask and run the transformation above.
- If not, it creates a ban on the nickname itself (ie. `nick!*@*`).

Some examples:

!ban paul 5d ragequitter

- If paul is present in the channel, it will do a host/ip ban on him.
- If paul is not present in the channel, the bot will create a ban on `paul!*@*`, meaning that nobody with the nickname paul can enter the channel. However, an offending user can circumvent this ban simply by changing nickname.

In both cases, the ban will stand for 5 days and have *ragequitter* as reason.

!ban paul!*@* 5d ragequitter Forces a nickname ban regardless of whether paul is online or not.

!ban roger!~rfeder@1.2.3.4 5d too good for us Creates a ban on `*!*@1.2.3.4`.

!ban frank!feinst@4.3.2.1 5d aimbotter Creates a ban on `*!feinst@4.3.2.1`.

!ban *!*@225.70.* Creates a ban on `*!*@225.70.*`. (IP range ban from `225.70.0.0` to `225.70.255.255`)

!ban **!ban**
Bans the last kicked person for the `default duration` with the kick message as the reason.

Searching bans

!banlist **!banlist**
!banlist #id **!banlist** (*#id*)
!banlist search **!banlist** (*search*)
Show active or recently expired bans.

If search is provided, the bot will search against all data on the bans. Also, if you provide a hostmask as search text, it will be matched against existing banmasks. Useful if you know someone's hostmask and want too know why they are banned.

If the search returns only one result (for instance if you used the *#id* form), all details will be shown, including the ban reason.

!banhistory [search|#id] **!banhistory** (*[search|#id]*)
Show expired bans.

Editing bans

!unban search|#id **!unban** (*search|#id*)
Lifts a ban. Use a hostmask as search to your advantage here.

!ban search|#id [duration] [reason] **!ban** (*search|#id*[, *duration*][, *reason*])
Edits a ban.

Configuration

[Ban]default duration = permanent (duration) `[Ban]` **default duration = permanent** (`Duration`)
Default duration for bans in case it isn't specified.

[Ban]keep expired bans for = 1 week (duration) `[Ban]` **keep expired bans for = 1 week** (`Duration`)
Purge the database of bans which expired more than this time ago.

[Ban]check interval = 2 min (duration) `[Ban]` **check interval = 2 min** (`Duration`)
Interval between two of the bot's periodic checks on bans to lift. You don't really need to touch it.

`topic`: Channel topic management

This module lets bot admins set the channel message of the day in the channel topic.

!motd message **!motd** (*message*)
Sets the message of the day.

!motd **!motd**
Shows the current message of the day.

!motd – **!motd** (–)
Empties the message of the day.

`chanops`: Sets channel operators as bot admins

This module, when enabled, will mark channel operators from the bot's home channel as bot admins, allowing them full access to admin commands. No questions asked.

`q_auth`: Q authentication

When this module is enabled, the bot will do QuakeNet-style Q auth before joining channels.

Configuration

[Q Auth]username = #REQUIRED (string) `[Q Auth]` **username (Required)** (`String`)
Your Q username, like in

```
/msg Q@CServe.quakenet.org AUTH username password
```

[Q Auth]password = #REQUIRED (string) `[Q Auth]` **`password`** **(Required)** (`String`)
Your Q password.

[Q Auth]Q username = Q!TheQBot@CServe.quakenet.org (string) `[Q Auth]` **`Q username`** **= Q!TheQBot@CServe.quakenet.org** (
The username to message the auth command to. The default suits QuakeNet, if you are on another network that uses Q, change this.

`help`: Quick reference

This module provides a quick reference to the bot's commands.

!commands **`!commands`**
Lists all available commands.

!help command **`!help`** (*command*)
Shows help about a particular command.

`info`: General info about the bot

!source **`!source`**
Tells where to find the bot's source code.

!version **`!version`**
Tells the bot's version.

2.3 Extending PyPickupBot

COMMAND INDEX

SETTINGS INDEX