

---

**ib3**

***Release 0.1.2.dev9+g3e44b48.d20190107***

**Jan 07, 2019**



---

# Contents

---

<b>1</b>	<b>IRC Bot Behavior Bundle (IB3)</b>	<b>3</b>
1.1	Overview . . . . .	3
1.2	Installation . . . . .	3
1.3	License . . . . .	3
1.4	Credits . . . . .	4
<b>2</b>	<b>ib3 package</b>	<b>5</b>
2.1	Module contents . . . . .	5
2.2	Submodules . . . . .	5
2.3	ib3.auth module . . . . .	5
2.4	ib3.connection module . . . . .	7
2.5	ib3.mixins module . . . . .	7
2.6	ib3.nick module . . . . .	8
	<b>Python Module Index</b>	<b>11</b>



IRC bot framework using mixins to provide commonly desired functionality.



---

## IRC Bot Behavior Bundle (IB3)

---

IRC bot framework using mixins to provide commonly desired functionality.

### 1.1 Overview

The `irc` python library's `irc.bot.SingleServerIRCBot` provides a nice base for making a new bot, but there are many common tasks needed by a robust bot that it does not handle out of the box. IB3 collects some commonly desired behaviors for a bot as [mixin](#) classes that can be used via [multiple inheritance](#):

```
from ib3 import Bot
from ib3.auth import SASL
from ib3.connection import SSL
from ib3.mixins import DisconnectOnError

class TestBot(SASL, SSL, DisconnectOnError, Bot):
    pass
```

### 1.2 Installation

- `pip install ib3` (recommended)
- `python setup.py install` (from source distribution)

### 1.3 License

IB3 is licensed under the [GNU GPLv3+](#) license.

## 1.4 Credits

Some code and much inspiration taken from Wikimedia irc bots [Adminbot](#), [Jouncebot](#), and [Stashbot](#).

## 2.1 Module contents

**class** `ib3.Bot` (*\*args, \*\*kwargs*)

Bases: `irc.bot.SingleServerIRCBot`

Basic IRC bot.

Simple subclass of `irc.bot.SingleServerIRCBot` that sets up lenient UTF-8 encoding handling for inbound messages. This is a nice base to start from when adding other IB3 mixins.

`__init__` (*\*args, \*\*kwargs*)

`x.__init__(...)` initializes x; see `help(type(x))` for signature

## 2.2 Submodules

## 2.3 `ib3.auth` module

**class** `ib3.auth.AbstractAuth` (*server\_list, nickname, realname, ident\_password, channels=[], username=None, \*\*kwargs*)

Bases: `ib3.mixins.JoinChannels`

Base class for authentication mixins.

`__init__` (*server\_list, nickname, realname, ident\_password, channels=[], username=None, \*\*kwargs*)

### Parameters

- **server\_list** – List of servers the bot will use.
- **nickname** – The bot's nickname
- **realname** – The bot's realname

- **ident\_password** – The bot’s password
- **channels** – List of channels to join after authenticating
- **username** – IRC username (default: nickname)

**class** `ib3.auth.NickServ` (\*args, \*\*kwargs)

Bases: `ib3.auth.AbstractAuth`

Authenticate with NickServ before joining channels.

`__init__` (\*args, \*\*kwargs)

**Parameters**

- **server\_list** – List of servers the bot will use.
- **nickname** – The bot’s nickname
- **realname** – The bot’s realname
- **ident\_password** – The bot’s password
- **channels** – List of channels to join after authenticating
- **username** – IRC username (default: nickname)

`_handle_privnotice` (conn, event)

Handle NOTICE sent directly to user.

Check for messages from NickServ requesting auth, warning of password failures, and acknowledging successful auth.

`_handle_welcome` (conn, event)

Handle WELCOME message.

Starts authentication handshake by sending NickServ an `identify` message.

`_identify_to_nickserv` ()

Send NickServ our username and password.

**class** `ib3.auth.SASL` (\*args, \*\*kwargs)

Bases: `ib3.auth.AbstractAuth`

Authenticate using SASL before joining channels.

`__init__` (\*args, \*\*kwargs)

**Parameters**

- **server\_list** – List of servers the bot will use.
- **nickname** – The bot’s nickname
- **realname** – The bot’s realname
- **ident\_password** – The bot’s password
- **channels** – List of channels to join after authenticating
- **username** – IRC username (default: nickname)

`_handle_903` (conn, event)

Handle 903 RPL\_SASLSUCCESS responses.

`_handle_908` (conn, event)

Handle 908 RPL\_SASLMECHS responses.

**`_handle_authenticate`** (*conn, event*)  
Handle AUTHENTICATE responses.

**`_handle_cap`** (*conn, event*)  
Handle CAP responses.

**`_handle_connect`** (*sock*)  
Send CAP REQ :sasl on connect.

**`_handle_welcome`** (*conn, event*)  
Handle WELCOME message.

## 2.4 ib3.connection module

**class** `ib3.connection.SSL` (*\*args, \*\*kwargs*)  
Bases: `object`

Use SSL connections.

**`__init__`** (*\*args, \*\*kwargs*)  
`x.__init__(...)` initializes x; see `help(type(x))` for signature

## 2.5 ib3.mixins module

**class** `ib3.mixins.DisconnectOnError` (*\*args, \*\*kwargs*)  
Bases: `object`

Handle ERROR message by logging and disconnecting.

**`__init__`** (*\*args, \*\*kwargs*)  
`x.__init__(...)` initializes x; see `help(type(x))` for signature

**class** `ib3.mixins.JoinChannels`  
Bases: `object`

Join channels one at a time to avoid flooding.

**`join_channels`** (*channels*)  
Join a list of channels, one at a time.

**class** `ib3.mixins.PingServer` (*max\_pings=2, ping\_interval=300, \*args, \*\*kwargs*)  
Bases: `object`

Add checks for connection liveness using PING commands.

**`__init__`** (*max\_pings=2, ping\_interval=300, \*args, \*\*kwargs*)

**Parameters**

- **`max_pings`** – Maximum number of missed pings to tolerate
- **`ping_interval`** – Seconds between ping attempts

**`_handle_pong`** (*conn, event*)  
Clear ping count when a pong is received.

**`_ping_server`** ()  
Send a ping or disconnect if too many pings are outstanding.

```
class ib3.mixins.RejoinOnBan(*args, **kwargs)
    Bases: object

    Handle ERR_BANNEDFROMCHAN by attempting to rejoin channel.

    __init__(*args, **kwargs)
        x.__init__(...) initializes x; see help(type(x)) for signature
```

```
class ib3.mixins.RejoinOnKick(*args, **kwargs)
    Bases: object

    Handle KICK by attempting to rejoin channel.

    __init__(*args, **kwargs)
        x.__init__(...) initializes x; see help(type(x)) for signature
```

## 2.6 ib3.nick module

```
class ib3.nick.Ghost(server_list, nickname, realname, altnick=None, *args, **kwargs)
    Bases: ib3.nick.NicknameInUse

    GHOST disconnects an old user session, or somebody attempting to use your nickname without authorization.

    This mixin assumes that you are also using a mixin from ib3.auth or another means to authenticate your IRC account.
```

```
    _recover_nick()
        Recover nick by sending GHOST command to NickServ.
```

```
class ib3.nick.NicknameInUse(server_list, nickname, realname, altnick=None, *args, **kwargs)
    Bases: object
```

```
    Handle ERR_NICKNAMEINUSE messages by changing to an alternate nick.
```

```
    __init__(server_list, nickname, realname, altnick=None, *args, **kwargs)
```

### Parameters

- **server\_list** – List of servers the bot will use.
- **nickname** – The bot’s nickname
- **realname** – The bot’s realname
- **altnick** – Alternate nickname if primary nick is taken

```
    _handle_nicknameinuse(conn, event)
        Handle ERR_NICKNAMEINUSE message.
```

```
        If failed nick matches our desired nick, switch to secondary nick, and schedule an attempt to reclaim our primary nick.
```

```
        If failed nick matches our secondary nick, die.
```

```
    _recover_nick()
        Do nothing.
```

```
        Subclasses can override this method to perform some special action to recover the bot’s primary nickname.
```

```
    has_primary_nick()
        Do we currently have our primary nick?
```

```
        Return type bool
```

**class** `ib3.nick.Regain` (*server\_list, nickname, realname, altnick=None, \*args, \*\*kwargs*)

Bases: `ib3.nick.NicknameInUse`

REGAIN disconnects an old user session, or somebody attempting to use your nickname without authorization, then changes your nickname to the given nickname. This may not work, disconnecting you, if the target client reconnects automatically.

This mixin assumes that you are also using a mixin from `ib3.auth` or another means to authenticate your IRC account.

**`_recover_nick()`**

Recover nick by sending REGAIN command to NickServ.



i

ib3, 5  
ib3.auth, 5  
ib3.connection, 7  
ib3.mixins, 7  
ib3.nick, 8



## Symbols

\_\_init\_\_() (ib3.Bot method), 5  
 \_\_init\_\_() (ib3.auth.AbstractAuth method), 5  
 \_\_init\_\_() (ib3.auth.NickServ method), 6  
 \_\_init\_\_() (ib3.auth.SASL method), 6  
 \_\_init\_\_() (ib3.connection.SSL method), 7  
 \_\_init\_\_() (ib3.mixins.DisconnectOnError method), 7  
 \_\_init\_\_() (ib3.mixins.PingServer method), 7  
 \_\_init\_\_() (ib3.mixins.RejoinOnBan method), 8  
 \_\_init\_\_() (ib3.mixins.RejoinOnKick method), 8  
 \_\_init\_\_() (ib3.nick.NicknameInUse method), 8  
 \_handle\_903() (ib3.auth.SASL method), 6  
 \_handle\_908() (ib3.auth.SASL method), 6  
 \_handle\_authenticate() (ib3.auth.SASL method), 6  
 \_handle\_cap() (ib3.auth.SASL method), 7  
 \_handle\_connect() (ib3.auth.SASL method), 7  
 \_handle\_nicknameinuse() (ib3.nick.NicknameInUse method), 8  
 \_handle\_pong() (ib3.mixins.PingServer method), 7  
 \_handle\_privnotice() (ib3.auth.NickServ method), 6  
 \_handle\_welcome() (ib3.auth.NickServ method), 6  
 \_handle\_welcome() (ib3.auth.SASL method), 7  
 \_identify\_to\_nickserv() (ib3.auth.NickServ method), 6  
 \_ping\_server() (ib3.mixins.PingServer method), 7  
 \_recover\_nick() (ib3.nick.Ghost method), 8  
 \_recover\_nick() (ib3.nick.NicknameInUse method), 8  
 \_recover\_nick() (ib3.nick.Regain method), 9

## A

AbstractAuth (class in ib3.auth), 5

## B

Bot (class in ib3), 5

## D

DisconnectOnError (class in ib3.mixins), 7

## G

Ghost (class in ib3.nick), 8

## H

has\_primary\_nick() (ib3.nick.NicknameInUse method), 8

## I

ib3 (module), 5  
 ib3.auth (module), 5  
 ib3.connection (module), 7  
 ib3.mixins (module), 7  
 ib3.nick (module), 8

## J

join\_channels() (ib3.mixins.JoinChannels method), 7  
 JoinChannels (class in ib3.mixins), 7

## N

NicknameInUse (class in ib3.nick), 8  
 NickServ (class in ib3.auth), 6

## P

PingServer (class in ib3.mixins), 7

## R

Regain (class in ib3.nick), 8  
 RejoinOnBan (class in ib3.mixins), 7  
 RejoinOnKick (class in ib3.mixins), 8

## S

SASL (class in ib3.auth), 6  
 SSL (class in ib3.connection), 7