



# **EH Forwarder Bot Documentation**

***Rilis 2.1.1***

**Eana Hufwe, and the EH Forwarder Bot contributors**

**04 Mar, 2022**



<b>1</b>	<b>Mulai</b>	<b>3</b>
<b>2</b>	<b>Berkas Konfigurasi</b>	<b>7</b>
<b>3</b>	<b>Peluncuran kerangka kerja</b>	<b>9</b>
<b>4</b>	<b>Direktori</b>	<b>11</b>
<b>5</b>	<b>Profil</b>	<b>13</b>
<b>6</b>	<b>Support</b>	<b>15</b>
<b>7</b>	<b>Walk-through — How EFB works</b>	<b>17</b>
<b>8</b>	<b>Panduan pengembangan</b>	<b>21</b>
<b>9</b>	<b>Bagaimana cara berkontribusi</b>	<b>33</b>
<b>10</b>	<b>API documentations</b>	<b>37</b>
<b>11</b>	<b>Indeks dan tabel</b>	<b>73</b>
<b>12</b>	<b>Merasa seperti berkontribusi?</b>	<b>75</b>
<b>13</b>	<b>Artikel terkait</b>	<b>77</b>
<b>14</b>	<b>Lisensi</b>	<b>79</b>
	<b>Indeks Modul Python</b>	<b>81</b>
	<b>Indeks</b>	<b>83</b>





# EH Forwarder Bot

An extensible message tunneling chat bot framework.

*Codename* **EH Forwarder Bot** (EFB) is an extensible message tunneling chat bot framework which delivers messages to and from multiple platforms and remotely control your accounts.



Beberapa langkah sederhana untuk memulai dengan EFB.

## 1.1 Pasang EH Forwarder Bot

EH Forwarder Bot dapat dipasang dengan cara berikut:

### 1.1.1 Pasang dari PyPI

``pip`` secara default akan memasang versi stabil terbaru dari PyPI, namun versi pengembangannya tersedia di PyPI juga.

```
pip3 install ehforwarderbot
```

### 1.1.2 Pasang dari GitHub

Ini akan memasang komit terbaru dari GitHub. Kemungkinan tidak stabil, jadi lanjutkan dengan hati-hati.

```
pip3 install git+https://github.com/ehForwarderBot/ehforwarderbot.git
```

### 1.1.3 Alternative installation methods

You can find a list of alternative installation methods contributed by the community in the [project wiki](#).

For scripts, containers (e.g. Docker), etc. that may include one or more external modules, please visit the [modules repository](#).

---

**Catatan:** These alternative installation methods are maintained by the community, please consult their respective author or maintainer for help related to those methods.

---

## 1.2 A stable internet connection

Since the majority of our channels are using polling for message retrieval, a stable internet connection is necessary for channels to run smoothly. An unstable connection may lead to slow response, or loss of messages.

## 1.3 Buat direktori lokal

EFB menggunakan \*nix pengguna konfigurasi gaya, yang dijelaskan secara detail di *Direktori*. Singkatnya, jika anda menggunakan konfigurasi default, anda perlu membuat `~/.ehforwarderbot`, dan memberikan membaca dan izin menulis untuk pengguna yang menjalankan EFB.

## 1.4 Pilih, pasang dan aktifkan modul

Currently, all modules that was submitted to us are recorded in the [modules repository](#). You can choose the channels that fits your need the best.

Petunjuk tentang pemasangan setiap saluran tersedia di dokumentasi mereka masing-masing.

### 1.4.1 Set up with the configuration wizard

When you have successfully installed the modules of your choices, you can use the configuration wizard which guides you to enable channels and middlewares, and continue to setup those modules if they also have provided a similar wizard.

You can start the wizard by running the following command in a compatible console or terminal emulator:

```
efb-wizard
```

If you want to start the wizard of a module for a profile individually, run:

```
efb-wizard -p <profile name> -m <module ID>
```



## 1.4.2 Set up manually

Alternatively, you can enable those modules manually it by listing its Channel ID in the *configuration file*. The default path is `~/.ehforwarderbot/profiles/default/config.yaml`. Please refer to *Direktori* if you have configured otherwise.

Harap dicatat bahwa meskipun anda dapat memiliki lebih dari satu saluran budak yang berjalan pada saat yang sama, anda hanya dapat memiliki satu saluran utama yang berjalan dalam satu profil. Sementara itu, middlewares benar-benar opsional.

For example, to enable the following modules:

- **Saluran utama**
  - Demo Master ("foo.demo\_master")
- **Saluran Slave**
  - Demo Slave (foo.demo\_slave)
  - Dummy Slave (bar.dummy)
- **Middlewares**
  - Null Middleware ("foo.null")

`config.yaml` should have the following lines:

```
master_channel: foo.demo_master
slave_channels:
- foo.demo_slave
- bar.dummy
middlewares:
- foo.null
```

If you have enabled modules manually, you might also need configure each module manually too. Please consult the documentation of each module for instructions.

## 1.5 Luncurkan EFB

```
ehforwarderbot
```

Ini akan meluncurkan EFB langsung dalam lingkungan saat ini. Default `:doc:'profil'` yang bernama " default", untuk memulai EFB di profil yang berbeda, menambahkan `--profil <profile-name>` ke perintah.

Untuk opsi baris perintah yang lain, menggunakan `--membantu` " pilihan.

### 1.5.1 Use EFB in another language

EFB supports translated user interface and prompts. You can set your system language or locale environmental variables (`LANGUAGE`, `LC_ALL`, `LC_MESSAGES` or `LANG`) to one of our [supported languages](#) to switch language.

You can help to translate this project into your languages on [our Crowdin page](#).

---

**Catatan:** If your are installing from source code, you will not get translations of the user interface without manual compile of message catalogs (`.mo`) prior to installation.

---

### 1.5.2 Meluncurkan EFB sebagai proses daemon

Semenjak versi 2, EH Forwarder Bot telah menghapus daemon helper karena tidak stabil penggunaannya. Kami menyarankan anda untuk menggunakan solusi yang baik untuk pengelolaan daemon, seperti `systemd`, `upstart`, atau `pm2`.

---

## Berkas Konfigurasi

---

EFB has an overall configuration file to manage all enabled modules. It is located under the *directory* of current profile, and named `config.yaml`.

### 2.1 Sintaks

Berkas konfigurasi berada di sintaks YAML. Jika anda tidak akrab dengan sintaks tersebut, silahkan periksa dokumentasi dan tutorial tersebut untuk rinciannya.

- ID dari saluran induk diaktifkan di bawah tombol `saluran_induk`
- ID dari saluran slave yang diaktifkan adalah yang tercantum di bawah tombol `saluran_slave`. Ini harus dijadikan sebuah daftar meskipun hanya satu saluran yang bisa diaktifkan.
- ID dari middleware yang diaktifkan adalah yang tercantum di bawah tombol `middleware`. Ini harus dijadikan sebuah daftar meskipun hanya satu middleware yang bisa diaktifkan. Namun, Apabila anda tidak menginginkan untuk mengaktifkan middleware apapun, hanya menghilangkan bagian sepenuhnya.

### 2.2 Contoh ID

To have multiple accounts running simultaneously, you can appoint an instance ID to a module. Instance ID can be defined by the user, and if defined, it must has nothing other than letters, numbers and underscores, i.e. in regular expressions `[a-zA-Z0-9_]+`. When instance ID is not defined, the channel will run in the "default" instance with no instance ID.

To indicate the instance ID of an instance, append # following by the instance ID to the module ID. For example, slave channel `bar.dummy` running with instance ID `alice` should be written as `bar.dummy#alice`. If the channel requires configurations, it should be done in the directory with the same name (e.g. `EFB_DATA_PATH/profiles/PROFILE/bar.dummy#alice`), so as to isolate instances.

Hindari memiliki dua modul dengan penetapan yang sama dari modul ID dan contoh ID seperti itu mungkin menyebabkan hasil yang tak terduga.

For example, to enable the following modules:

- **Saluran utama**
  - Demo Master (`foo.demo_master`)
- **Slave channels**
  - Demo Slave (`foo.demo_slave`)
  - Dummy Slave (`bar.dummy`)
  - Budak tiruan (`bar.dummy`) di alt contoh
- **Middlewares**
  - Message Archiver (`foo.msg_archiver`)
  - Null Middleware (`foo.null`)

`config.yaml` should have the following lines:

```
master_channel: foo.demo_master
slave_channels:
- foo.demo_slave
- bar.dummy
- bar.dummy#alt
middlewares:
- foo.msg_archiver
- foo.null
```

## 2.3 Granulated logging control

If you have special needs on processing and controlling the log produced by the framework and running modules, you can use specify the log configuration with [Python's configuration dictionary schema](#) under section logging.

An example of logging control settings:

```
logging:
  version: 1
  disable_existing_loggers: false
  formatters:
    standard:
      format: '%(asctime)s [%(levelname)s] %(name)s: %(message)s'
  handlers:
    default:
      level: INFO
      formatter: standard
      class: logging.StreamHandler
      stream: ext://sys.stdout
  loggers:
    '':
      handlers: [default,]
      level: INFO
      propagate: true
    AliceIRCChannel:
      handlers: [default, ]
      level: WARN
      propagate: false
```

---

## Peluncuran kerangka kerja

---

EH Forwarder Bot memberikan 2 cara untuk meluncurkan framework:

- `ehforwarderbot`
- `python3 -m ehforwarderbot`

Kedua perintah tersebut sebenarnya sama, menerima bendera yang sama, menjalankan kode yang sama. Yang terakhir ini hanyalah sebuah cadangan dalam kasus pertama yang tidak bekerja.

### 3.1 Pilihan

- `-h, --help`: Menampilkan pesan bantuan
- `-p PROFILE, --profile PROFILE`: Switch *profile*

Dari versi 2, EFB mendukung jalannya berbagai contoh berbeda di bawah pengguna yang sama, yang diidentifikasi oleh profil mereka. Profil bawaan dinamakan ``bawaan``.

- `-V, --version`: Print version information

Ini menunjukkan nomor versi Python yang anda gunakan, framework EFB, dan semua saluran dan middlewares diaktifkan.

- `-v, --verbose`: Print verbose log

This option enables verbose log of EFB and all enabled modules. This, together with `--version`, is particularly useful in debugging and issue reporting.

- `--trace-threads`: Trace hanging threads

This option is useful to identify source of the issue when you encounter situations where you had to force quit EFB. When this option is enabled, once the first stop signal (`SIGINT` or `SIGTERM`) is sent, threads that are *asleep* will be identified and reported every 10 seconds, until a second stop signal is seen.

In order to use this option, you need to install extra Python dependencies using the following command.

```
pip3 install 'ehforwarderbot[trace]'
```

### 3.2 Quitting EFB

If you started EFB in a shell, you can simply press `Control-c` to trigger the quit process. Otherwise, ask your service manager to issue a `SIGTERM` for a graceful exit. The exit process may take a few second to complete.

---

**Penting:** It is important for you to issue a graceful termination signal (e.g. `SIGTERM`), and **NOT** to use `SIGKILL`. Otherwise you may face the risk of losing data and breaking programs.

---

If you have encountered any issue quitting EFB, press `Control-c` for 5 times consecutively to trigger a force quit. In case you have frequently encountered situations where you had to force quit EFB, there might be a bug with EFB or any modules enabled. You may want to use the `--trace-threads` option described above to identify the source of issue, and report this to relevant developers.

Sejak EH Forwarder Bot 2.0, sebagian besar modul harus diinstal dengan Package Manager Python ``pip``, sedangkan konfigurasi dan data disimpan di direktori data "EFB".

Secara default, direktori data adalah pengguna tertentu, terletak di direktori home pengguna, ``~/ehforwarderbot``. Ini bisa diganti dengan variabel lingkungan ``EFB\_DATA\_PATH``. Jalan yang didefinisikan di sini harus berupa jalur \*\*mutlak\*\*.

## 4.1 Struktur direktori

Dengan menggunakan konfigurasi default sebagai contoh, bagian ini akan mengenalkan tentang struktur direktori data EFB.

```
./ehforwarderbot          or $EFB_DATA_PATH
|- profiles
| |- default              The default profile.
| | |- config.yaml       Main configuration file.
| | |- dummy_ch_master   Directory for data of the channel
| | | |- config.yaml     Config file of the channel. (example)
| | | |- ...
| | |- random_ch_slave
| | | |- ...
| |- profile2             Alternative profile
| | |- config.yaml
| | |- ...
| |- ...
|- modules                Place for source code of your own channels/middlewares
| |- random_ch_mod_slave Channels here have a higher priority while importing
| | |- __init__.py
| | |- ...
```





Mulai dari EFB versi 2, profil diperkenalkan untuk memungkinkan pengguna perlu untuk menjalankan beberapa contoh EFB secara bersamaan tanpa mempengaruhi satu sama lain.

Setiap profil telah menetapkan sendiri dari file konfigurasi seperangkat saluran yang berbagi kode yang sama, tetapi memiliki file data yang berbeda, sehingga mereka dapat berjalan mereka sendiri.

Nama profil default disebut " default". Untuk beralih ke profil yang berbeda, menentukan nama profil di "--profil " bendera sementara mulai EFB.

## 5.1 Memulai profil baru

To create a new profile, you need to create a directory in the *EFB\_DATA\_PATH/profiles*, and create a new configuration file as described in chapter *Mulai*.

Ketika semuanya dikonfigurasi dengan benar, Anda baik untuk pergi.



## 6.1 Bug reports and feature requests

See *contribution guideline* for details.

## 6.2 Questions about development and usage

If you have any question about developing a module for EFB, or about usages, you can always visit our [GitHub Discussions](#) forum or join our [Telegram Group](#) for help.

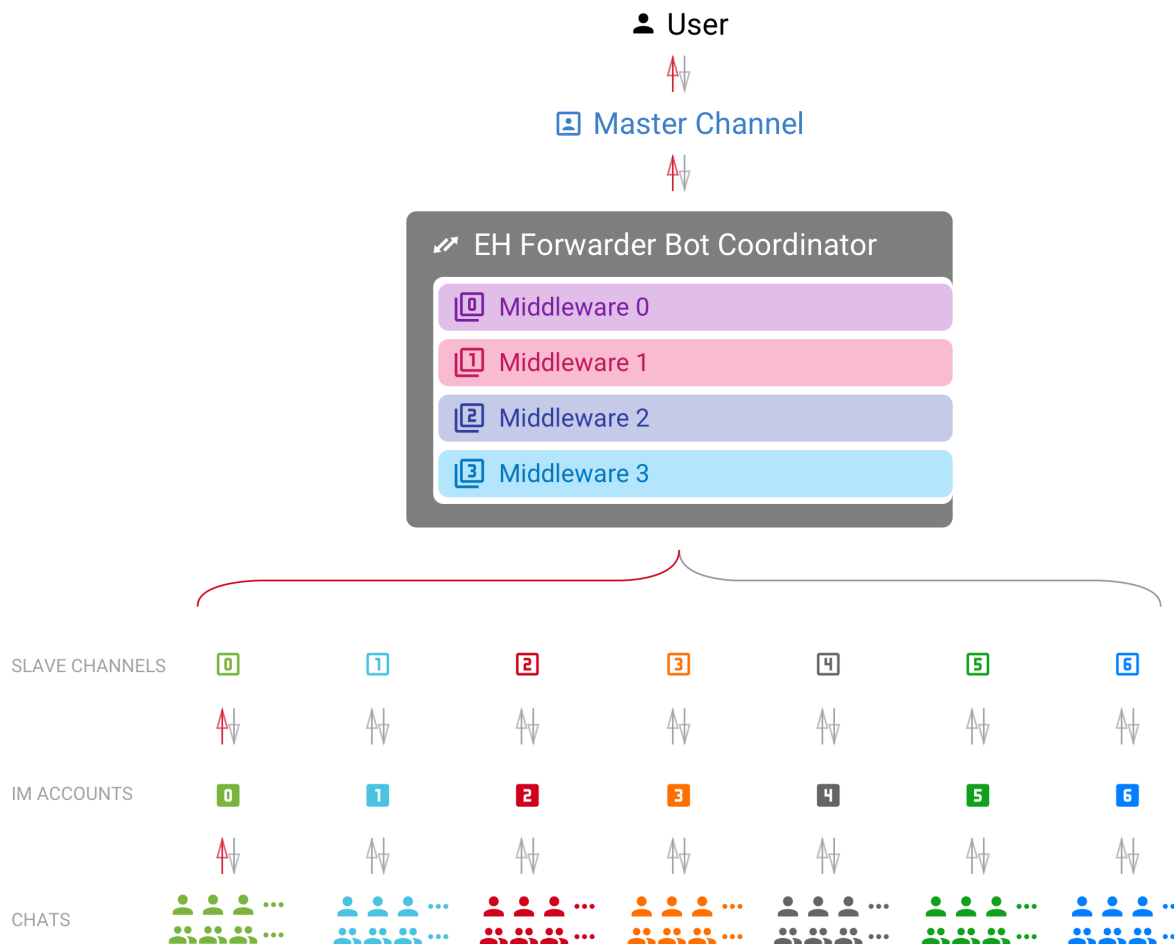


---

### Walk-through — How EFB works

---

EH Forwarder Bot adalah kerangka kerja yang bisa diperluas yang memungkinkan pengguna mengontrol dan mengelola akun dari platform obrolan yang berbeda di antarmuka yang dipersatukan. Ini terdiri dari 4 bagian: Saluran Utama, beberapa Saluran Cabang, beberapa Middleware dan Koordinator.



**master channel** The channel that directly interact with *the User*. It is guaranteed to have one and only one master channel in an EFB session.

**slave channel** The channel that delivers messages to and from their relative platform. There is at least one slave channel in an EFB session.

**coordinator** Komponen kerangka kerja yang menyimpan contoh saluran, dan mengirim pesan antar saluran.

**middleware** Modul yang memproses pesan dan status terkirim antara saluran, dan membuat perubahan dimana diperlukan.

## 7.1 Konsep untuk mengetahui

**module** A common term that refers to both channels and middlewares.

**the User**

**the User Himself** This term<sup>1</sup> can refer to the user of the current instance of EH Forwarder Bot, operating the master channel, and the account of an IM platform logged in by a slave channel.

**chat** A place where conversations happen, it can be either a *private chat*, a *group chat*, or a *system chat*.

<sup>1</sup> "Himself" here is used as a derived form of a gender-neutral singular third-person pronoun.

**private chat** A conversation with a single person on the IM platform. Messages from a private conversation shall only have an author of *the User Themselves*, the other person, or a “system member”.

For platforms that support bot or something similar, they would also be considered as a “user”, unless messages in such chat can be sent from any user other than the bot.

For chats that *the User* receive messages, but cannot send message to, it should also be considered as a private chat, only to raise an exception when messages was trying to send to the chat.

**group chat** A chat that involves more than two members. A group chat MUST provide a list of members that is involved in the conversation.

**system chat** A chat that is a part of the system. Usually used for chats that are either a part of the IM platform, the *slave channel*, or a *middleware*. *Slave channels* can use this chat type to send system message and notifications to the master channel.

**chat member** A participant of a chat. It can be *the User Themselves*, another person or bot in the chat, or a virtual one created by the IM platform, the *slave channel*, or a *middleware*.

**message** Pesan yang disampaikan secara ketat antara guru saluran dan saluran budak. Hal ini biasanya membawa informasi dari jenis tertentu.

Setiap pesan setidaknya harus memiliki ID unik yang berbeda dalam saluran budak berhubungan dengan itu. Setiap pesan yang disunting harus dapat diidentifikasi dengan ID unik yang sama.

**status** Informasi yang tidak diformat ke dalam pesan. Biasanya termasuk pembaruan obrolan dan anggota obrolan, dan penghapusan pesan.

## 7.2 Saluran Slave

Tugas dari saluran budak relatif sederhana.

1. Menyampaikan pesan ke dan dari saluran utama.
2. Mengelola sebuah daftar dari semua obrolan yang tersedia, dan anggota kelompok.
3. Monitor perubahan obrolan dan memberitahukan saluran inti.

Fitur-fitur yang tidak sesuai dengan standar EFB model Saluran Budak dapat ditawarkan sebagai budak-tambahan-fitur.

## 7.3 Saluran Inti

Master channels is relatively more complicated and also more flexible. As it directly faces the User, its user interface should be user-friendly, or at least friendly to the targeted users.

Pekerjaan dari saluran inti termasuk:

1. Menerima, memproses dan menampilkan pesan dari budak saluran.
2. Menampilkan daftar lengkap dari obrolan dari semua budak saluran.
3. Offer an interface for the User to use "extra functions" from slave channels.
4. Memproses pembaruan dari saluran budak.
5. Berikan antarmuka yang mudah digunakan sejauh mungkin.

## 7.4 Middlewares

Middlewares dapat memantau dan membuat perubahan atau membatalkan pesan dan status disampaikan antara saluran. Middlewares dieksekusi dalam urutan pendaftaran, satu demi satu. Sebuah middleware akan selalu menerima pesan-pesan yang diproses jika sebelumnya middleware tersedia. Sekali middleware membatalkan pesan atau status, pesan tidak akan diproses dan disampaikan lebih lanjut.



---

## Panduan pengembangan

---

This section includes guides on how to develop channels and middlewares for EH Forwarder Bot.

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “NOT RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in BCP 14 [RFC 2119] [RFC 8174] when, and only when, they appear in all capitals, as shown here.

### 8.1 Saluran budak

Saluran budak lebih mirip bungkus API IM, itu melampirkan pesan dari IM ke objek yang sesuai dan mengirimkannya ke saluran induk.

Meskipun kami menyarankan agar saluran budak harus sesuai dengan platform IM, akan tetapi anda dapat mencoba menyesuaikan untuk segala hal yang dapat mengirimkan informasi sebagai pesan, dan memiliki batasan daftar poin akhir untuk menyampaikan pesan ke dan dari obrolan.

In most of the cases, slave channels **SHOULD** be identified as one single user from the IM platform, instead of a bot. You should only use a bot for slave channels when:

- platform IM tidak membedakan antara pengguna dan bot, atau
- bot pada platform IM bisa melakukan hal yang sama persis, jika tidak lebih, sebagai pengguna, dan bot bisa dibuat lebih mudah daripada akun pengguna.

### 8.1.1 Fitur tambahan

Saluran budak dapat menawarkan fungsi yang lebih dari apa yang EFB membutuhkan, seperti membuat kelompok, mencari teman, dll, melalui *fitur tambahan*.

Such features are accessed by the user in a CLI-like style. An “additional feature” method **MUST** only take one string parameter aside from `self`, and wrap it with `extra()` decorator. The `extra` decorator takes 2 arguments: `name` -- a short name of the feature, and `desc` -- a description of the feature with its usage.

`desc` **SHOULD** describe what the feature does and how to use it. It's more like the help text for an CLI program. Since method of invoking the feature depends on the implementation of the master channel, you **SHOULD** use "{function\_name}" as its name in `desc`, and master channel will replace it with respective name depend on their implementation.

The method **MUST** in the end return a string, which will be shown to the user as its result, or `None` to notify the master channel there will be further interaction happen. Depending on the functionality of the feature, it may be just a simple success message, or a long chunk of results.

The callable **MUST NOT** raise any exception to its caller. Any exceptions occurred within should be *expected* and processed.

Callable name of such methods has a more strict standard than a normal Python 3 identifier name, for compatibility reason. An additional feature callable name **MUST**:

- menjadi peka
- hanya termasuk huruf besar, huruf kecil, dan garis bawah.
- tidak dimulai dengan angka.
- berada di suhu udara turun menjadi antara 1 dan 20 inklusif
- *Jadilah sesingkat dan ringkas mungkin, tapi tetap mengerti*

It can be expressed in a regular expression as:

```
^[A-Za-z][A-Za-z0-9_]{0,19}$
```

An example is as follows:

```
@extra(name="Echo",
        desc="Return back the same string from input.\n"
        "Usage:\n"
        "    {function_name} text")
def echo(self, arguments: str = "") -> str:
    return arguments
```

### 8.1.2 Perintah pesan

Perintah pesan biasanya dikirim oleh saluran budak sehingga pengguna dapat merespon pesan tertentu yang memiliki tindakan khusus yang diperlukan.

Mungkin kasus-kasus ketika perintah pesan dapat berguna:

- Tambahkan sebagai teman ketika kontak kartu diterima.
- Menerima atau menolak ketika permintaan teman diterima.
- Vote to a voting message.

Sebuah pesan anda dapat melampirkan daftar perintah, di mana masing-masing dari mereka memiliki:

- manusia-nama,
- a callable name,
- a list of positional arguments (\*args), and
- a dict of keyword arguments (\*\*kwargs)

When the User clicked the button, the corresponding method of your channel will be called with provided arguments.

Note that all such methods **MUST** return a `str` as a respond to the action from user, and they **MUST NOT** raise any exception to its caller. Any exceptions occurred within **MUST** be expected and processed.

### 8.1.3 Pengiriman pesan

Slave channels **SHOULD** deliver all messages that the IM provides, including what the User sent outside of this channel. But it **SHOULD NOT** deliver message sent from the master channel again back to the master channel as a new message.

### 8.1.4 Implementation details

See *SlaveChannel*.

## 8.2 Saluran inti

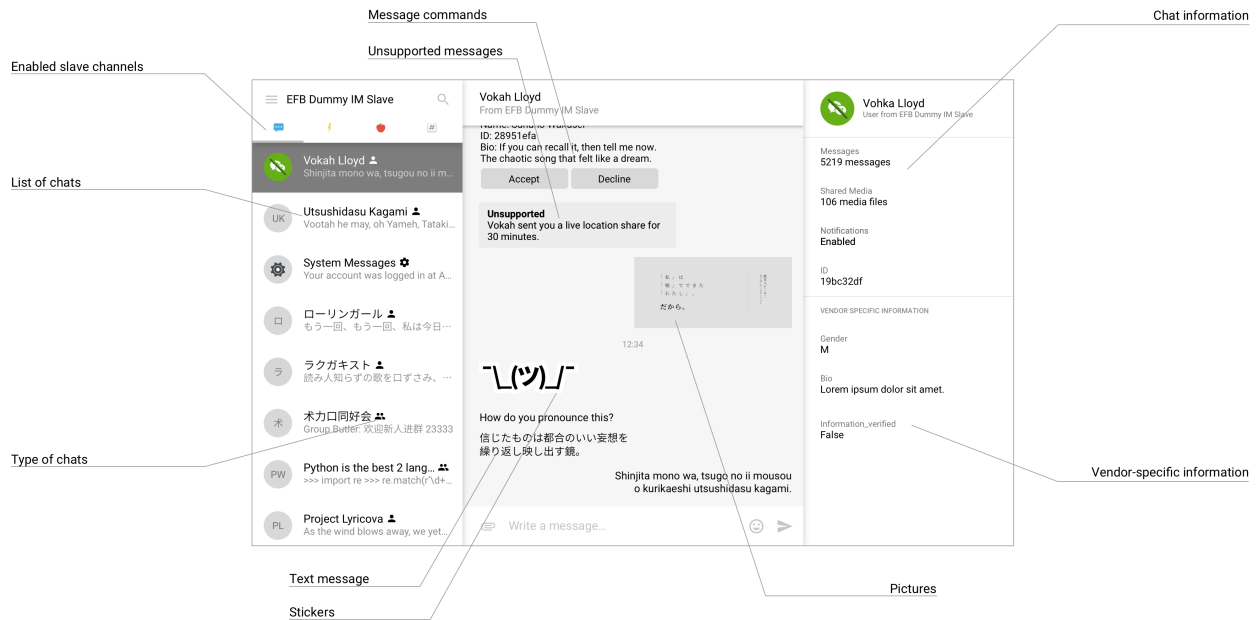
Saluran utama adalah antarmuka yang secara langsung atau tidak langsung berinteraksi dengan pengguna. Meskipun saluran inti pertama dari EFB (EFB Telegram Master) ditulis dalam sebuah bentuk Telegram Bot, saluran inti dapat dituliskan dalam banyak bentuk, seperti:

- Sebuah aplikasi web
- A server that expose APIs to dedicated desktop and mobile clients
- Sebuah bot obrolan pada sebuah IM yang ada
- Server yang mengumpulkan dengan Protokol IM yang umum
- Sebuah klien CLI
- Ada yang lain yang bisa anda pikirkan...

### 8.2.1 Pedoman desain

When the master channel is implemented on an existing protocol or platform, as far as possible, while considering the user experience, a master channel **SHOULD**:

- maintain one conversation thread per chat, indicating its name, source channel and type;
- mendukung semua, atau paling tidak sebagian besar, jenis pesan yang ditetapkan dalam kerangka kerja, memproses dan menyampaikan pesan antara pengguna dan saluran slave;
- support all, if not most, features of messages, including: targeted message reply, chat substitution in text (usually used in @ references), commands, etc. Master channel **SHOULD** be able to process incoming messages with such features, and send messages with such features to slave channels if applicable;
- dapat memanggil dan memproses "fungsi tambahan" yang ditawarkan oleh saluran slave.



Gambar. 1: Sebuah contoh desain yang bagus dari saluran utama, terinspirasi oleh Telegram Desktop

Selain itu, saluran utama juga dapat mendukung / mengidentifikasi informasi yang disampaikan oleh vendor dari saluran slave tertentu.

Bergantung pada apa yang anda terapkan, saluran utama mungkin perlu untuk menyimpan daftar obrolan dan pesan, untuk presentasi atau tujuan lainnya.

### 8.2.2 Pengiriman pesan

Note that sometimes the User may send messages outside of this EFB session, so that slave channels MAY provide a message with its author in the “self” type.

### 8.2.3 Implementation details

See *MasterChannel*.

## 8.3 Middlewares

Middleware bekerja di antara saluran induk dan saluran slave, mereka terlihat melalui pesan dan status yang dikirimkan di antara saluran, melewatkannya, membuat perubahan atau membuangnya, satu demi satu.

Seperti saluran, masing-masing middlewares juga memiliki tingkatan per sesi EFB, yang dikelola oleh koordinator. Akan tetapi, mereka tidak memiliki urutan pemungutan suara terpusat, yang berarti jika middleware ingin memiliki urutan pemungutan atau yang serupa berjalan di latar belakang, ia harus menghentikan urutan menggunakan Python `Atexit` atau sebaliknya.

### 8.3.1 Pemrosesan Pesan dan Status

Each middleware by default has 2 methods, `process_message()` which processes message objects, and `process_status()` which processes status objects. If they are not overridden, they will not touch on the object and pass it on as is.

To modify an object, just override the relative method and make changes to it. To discard an object, simply return `None`. When an object is discarded, it will not be passed further to other middlewares or channels, which means a middleware or a channel will never receive a `None` message or status.

### 8.3.2 Penggunaan lainnya

Mempunyai batasan yang agak sedikit dibandingkan dengan saluran, middlewares lebih mudah untuk ditulis, yang memungkinkannya melakukan lebih dari sekedar mencegat pesan dan status.

Beberapa gagasan:

- Penyiaran berulang-ulang ke saluran utama / slave
- Penggabungan dengan bot saluran
- Operasi otomatis pada perintah khusus vendor / fitur tambahan
- Membagikan sesi pengguna dari saluran slave dengan program lainnya
- dll...

### 8.3.3 Implementation details

See *Middleware*.

## 8.4 Lifecycle

This section talks about the lifecycle of an EFB instance, and that of a message / status.

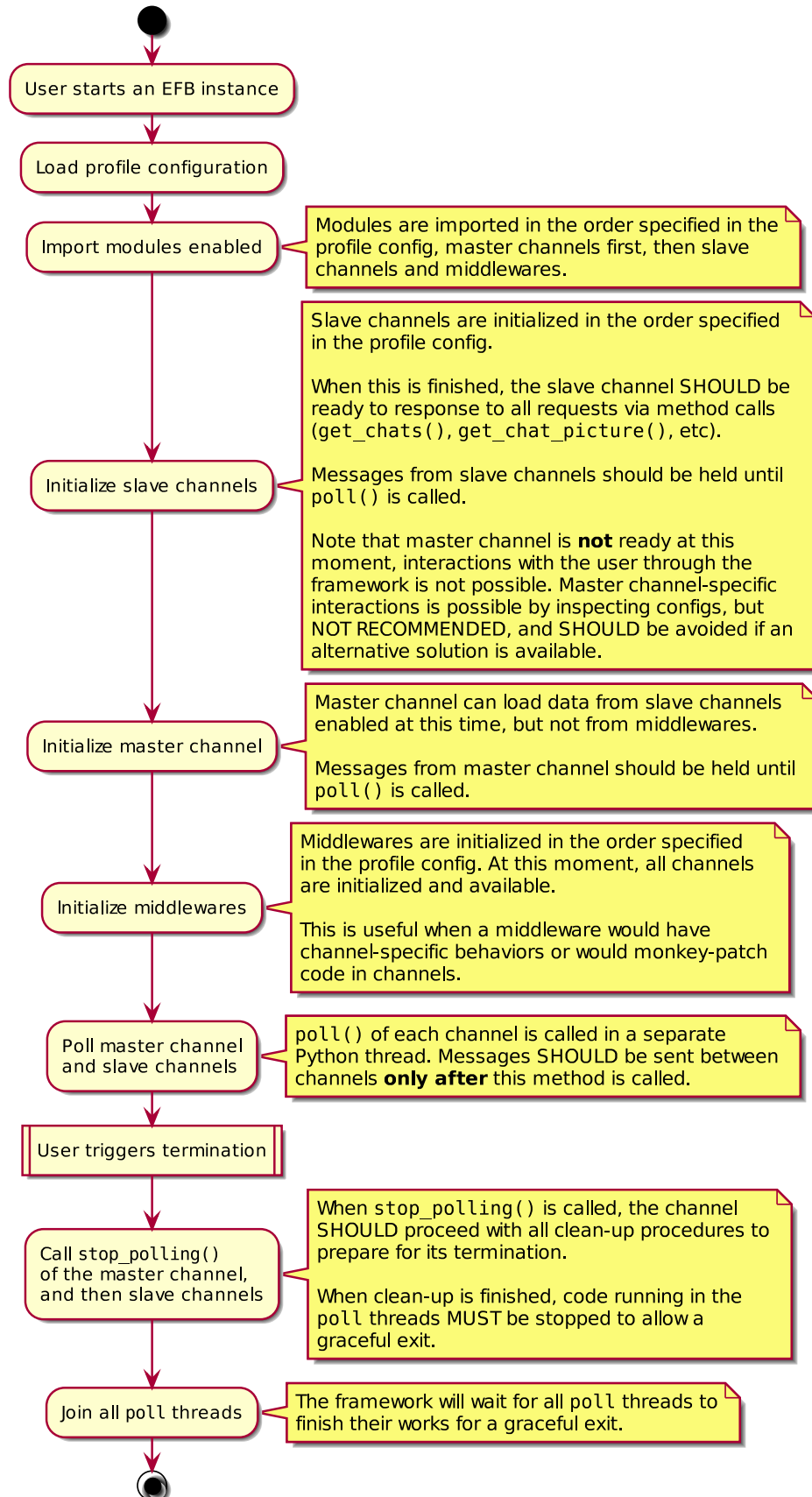
### 8.4.1 Lifecycle of an EFB instance

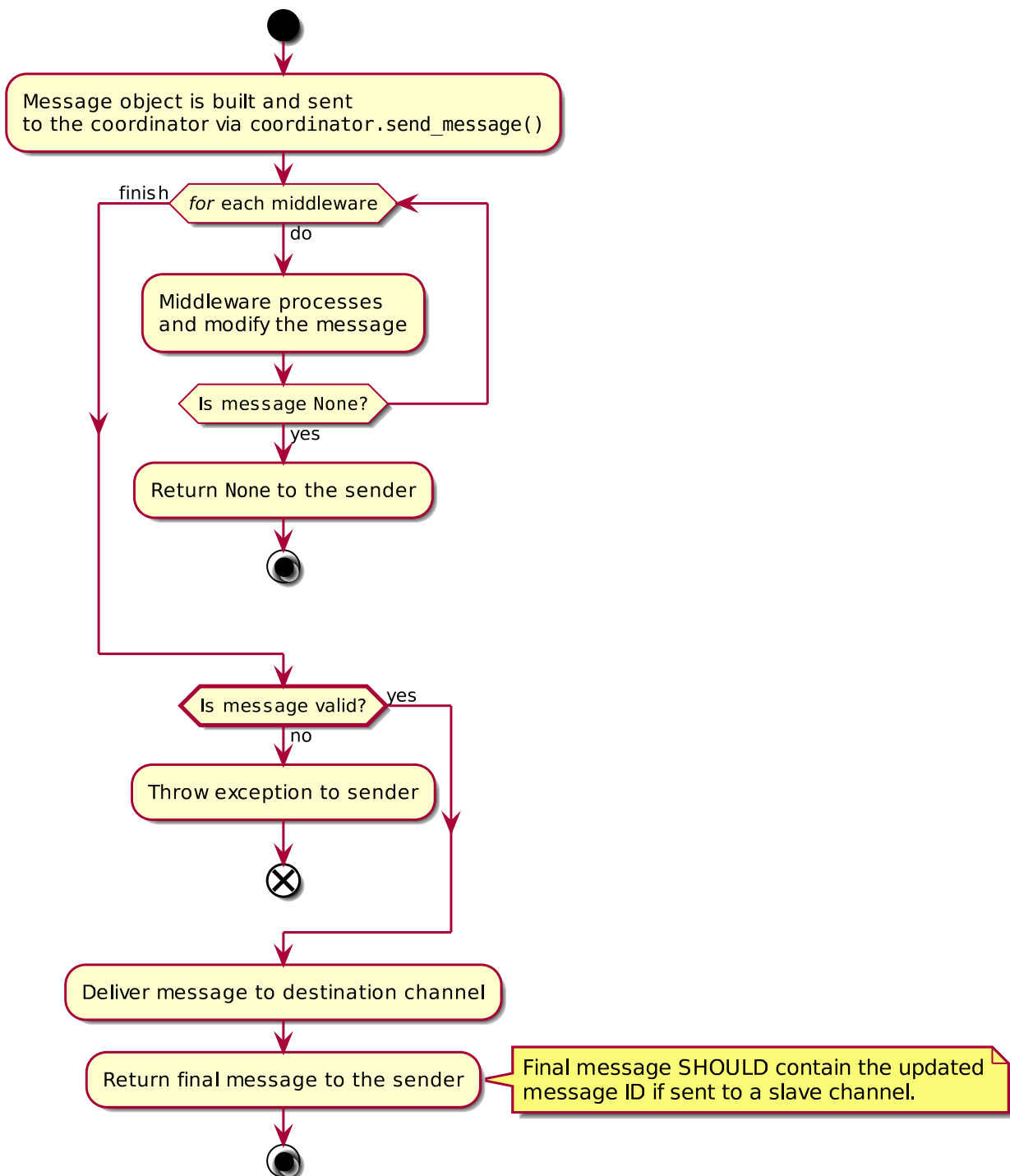
The diagram below outlines the lifecycle of an EFB instance, and how channels and middlewares are involved in it.

### 8.4.2 Lifecycle of a message

The diagram below outlines the lifecycle of a message sending from a channel, going through all middlewares, sent to the destination channel, and returned back to the sending channel.

Status objects processed in the same way.





Gambar. 3: Lifecycle of a message

## 8.5 Media processing

### 8.5.1 Memilih format media

Saluran Induk dan Slave dapat mengambil dakwaan mereka untuk mengkonversi berkas media yang mereka kirim atau terima. Secara umum: **apabila berkas media yang diterima dari server jarak jauh bukan merupakan format yang umum, konversikan terlebih dahulu sebelum mengirimkannya; apabila berkas media yang dikirim ke server jarak jauh mengharuskan dalam format tertentu, maka berkas tersebut harus dikonversi sebelum dikirim.** Walaupun demikian, ini hanya sebuah pedoman tanggung jawab saluran mengenai pengolahan media, dan setiap orang memiliki pendapat mereka sendiri mengenai format umum / pengkodean. Oleh karena itu kami hanya menyarankan perilaku ini, tapi jangan memaksakan kode kami. Ini merupakan untuk mengatakan bahwa, anda masih harus memperhatikan jenis pengkodean media yang diterima dari metode penyajian yang cocok, dan mengubah dan/atau mengganti jenis representasi yang berbeda apabila diperlukan. Bagaimanapun juga, penyampaian informasi itu lebih penting.

### 8.5.2 Media encoders

Begitu pula, kita juga tidak akan membatasi hal ini juga, tapi hanya sebuah saran. Seperti yang mungkin sudah anda ketahui, ada beberapa pengolahan perpustakaan Python yang murni, kebanyakan dari mereka akan sedikit banyak membutuhkan terhadap internal atau eksternal.

Kami mencoba untuk menggunakan beberapa pustaka yang berbeda sebisa kami, ketika lebih banyak pustaka yang dipasang berarti akan lebih banyak ruang, waktu pasang, dan kompleksitas. Saat memproses berkas media, kami menyarankan untuk menggunakan pustaka berikut jika memungkinkan:

- Pillow
- FFmpeg

### 8.5.3 File dalam pesan

When a file sent out from a channel, it **MUST** be open, and sought back to 0 ( `file.seek(0)` ) before sending.

Files sent **MUST** be able to be located somewhere in the file system, and **SHOULD** with a appropriate extension name, but not required. All files **MUST** also have its MIME type specified in the message object. If the channel is not sure about the correct MIME type, it can try to guess with `libmagic`, or fallback to `application/octet-stream`. Always try the best to provide the most suitable MIME type when sending.

Untuk file yang demikian, kami menggunakan `tutup` untuk menandakan akhir siklus hidupnya. Apabila file tidak dibutuhkan oleh saluran pengirim lagi, itu dapat dibuang dengan aman.

Biasanya, `tempfile.NamedTemporaryFile` harus bekerja untuk kasus-kasus biasa.

## 8.6 Konfigurasi dan penyimpanan

### 8.6.1 Konfigurasi dan Penyimpanan Permanen

Seperti yang dijelaskan di *Direktori*, masing-masing modul telah dialokasikan dengan sebuah folder per profil untuk konfigurasi dan penyimpanan lainnya. Path dapat diperoleh dengan menggunakan `get_data_path()` dengan ID modul anda. Semua penyimpanan seperti itu dikhususkan hanya untuk satu profil.

Untuk konfigurasi, kami sarankan untuk menggunakan `` <module\_data\_path> / config.yaml``. Demikian pula, kami menyiapkan `:meth: ~.ehforwarderbot.utils.get_config_path` untuk mendapatkan path untuk file konfigurasi default. Sekali lagi, Anda tidak dipaksa untuk menggunakan nama ini atau YAML sebagai format file konfigurasi Anda.



Biasanya di folder penyimpanan tinggal:

- File konfigurasi
- Kredensial pengguna / penyimpanan sesi
- Database

## 8.6.2 Penyimpanan Sementara

Saat memproses pesan multimedia, kita pasti perlu menyimpan file tertentu untuk sementara, baik di dalam saluran maupun di saluran. Biasanya, file sementara bisa ditangani dengan perpustakaan *tempfile* Python.

## 8.6.3 Wizard

If your module requires relatively complicated configuration, it would be helpful to provide users with a wizard to *check prerequisites of your module* and *guide them to setup your module for use*.

From version 2, EFB introduced a centralised wizard program to allow users to enable or disable modules in a text-based user interface (TUI). If you want to include your wizard program as a part of the wizard, you can include a new entry point in your `setup.py` with [Setuptools' Entry Point feature](#).

The group for wizard program is `ehforwarderbot.wizard`, and the entry point function **MUST** accept 2 positional arguments: profile ID and instance ID.

### Example

`setup.py` script

```
setup(
    # ...
    entry_points={
        "ehforwarderbot.wizard": ['alice.irc = efb_irc_slave.wizard:main']
    },
    # ...
)
```

`.egg-info/entry_points.txt`

```
[ehforwarderbot.wizard]
alice.irc = efb_irc_slave.wizard:main
```

`efb_irc_slave/wizard.py`

```
# ...

def main(profile, instance):
    print("Welcome to the setup wizard of my channel.")
    print("You are setting up this channel in profile "
          "'{0}' and instance '{1}'.".format(profile, instance))
    print("Press ENTER/RETURN to continue.")
    input()

    step1()

# ...
```

## 8.7 Mengemas dan mempublikasi

### 8.7.1 Terbitkan modul anda pada PyPI

Mencantumkan modul pada PyPI adalah salah satu cara paling mudah bagi pengguna untuk memasang paket anda. Silakan melihat ke dokumentasi dan tutorial yang terkait tentang PyPI dan pip untuk paket penerbitan.

For EFB modules, the package is RECOMMENDED to have a name starts with `efb-`, or in the format of `efb-platform-type`, e.g. `efb-irc-slave` or `efb-wechat-mp-filter-middleware`. If there is a collision of name, you MAY adjust the package name accordingly while keeping the package name starting with `efb-`.

When you are ready, you may also want to add your module to the [Modules Repository](#) of EFB.

### 8.7.2 Modul penemuan

EH Forwarder Bot menggunakan [Setuptools' Entry Point](#) fitur untuk menemukan dan mengelola saluran dan middlewares. *Setup.py* script atau *.egg-info/entry\_points.txt*, menentukan kelompok dan objek sebagai berikut:

- Grup untuk saluran utama: `ehforwarderbot.master`
- Grup untuk saluran slave: `ehforwarderbot.slave`
- Grup untuk middlewares: `ehforwarderbot.middleware`

Convention for object names is `<author>.<platform>`, e.g. `alice.irc`. This **MUST** also be your module's ID.

Object reference **MUST** point to your module's class, which is a subclass of either [Channel](#) or [Middleware](#).

### 8.7.3 Contoh

`setup.py` script

```
setup(  
    # ...  
    entry_points={  
        "ehforwarderbot.slave": ['alice.irc = efb_irc_slave:IRCChannel']  
    },  
    # ...  
)
```

`.egg-info/entry_points.txt`

```
[ehforwarderbot.slave]  
alice.irc = efb_irc_slave:IRCChannel
```

### 8.7.4 Modul pribadi

Jika anda ingin mengembangkan dari, atau membuat perubahan pada modul yang ada untuk digunakan secara pribadi, anda dapat memiliki modulnya di modul pribadi *direktori*.

For such modules, your channel ID MUST be the fully-qualified name of the class. For example, if your class is located at `<EFB_BASE_PATH>/modules/bob_irc_mod/__init__.py:IRCChannel`, the channel MUST have ID `bob_irc_mod.IRCChannel` for the framework to recognise it.

## 8.8 Miscellaneous

### 8.8.1 Logging

In complex modules, you should have detailed logs in DEBUG level and optionally INFO level. All your log handlers SHOULD follow that of the root logger, which is controlled by the framework. This could be helpful when for you to locate issues reported by users.

### 8.8.2 Vendor-specifics

Jika anda memuat informasi khusus untuk vendor dalam pesan dan/atau obrolan, tolong buat upaya anda untuk mengumpulkan di README atau dokumentasi, sehingga pengembang lain dapat mengambilnya ketika menyesuaikan dengan modul anda.

### 8.8.3 Threading

All channels are RECOMMENDED a separate thread while processing a new message, so as to prevent unexpectedly long thread blocking.

We are also considering to move completely to asynchronous programming when most channels are ready for the change.

### 8.8.4 Static type checking

EH Forwarder Bot is fully labeled in the Python 3 type hint notations. Since sometimes maintaining a module with high complexity could be difficult, we RECOMMEND you to type your module too and use tools like `mypy` to check your code statically.



---

## Bagaimana cara berkontribusi

---

Pertama-tama, terima kasih karena telah meluangkan waktu anda untuk berkontribusi!

Please note that only questions on the framework will be answered here. For issue related with any channels, please contact their respective authors or post in their corresponding repositories.

Inilah panduan sederhana tentang bagaimana anda dapat mengajukan suatu masalah, atau mengajukan pull request yang berguna dan efektif.

If you need help, or want to talk to the authors, feel free to visit our [GitHub Discussions](#) forum, or chat with us at our [Telegram support group](#).

Before you ask a question, please read and follow [this guide](#) as far as possible. Without doing so might lead to unfriendly or no response from the community, although we try to refrain from doing so.

### 9.1 Melaporkan bug

#### 9.1.1 Sebelum mengajukan laporan bug

- Please ensure if your issue is about the framework itself, not about any module. Reports about modules should go to their respective issue trackers.
- Baca melalui dokumentasi untuk mengetahui jika itu telah mencakup pertanyaan anda.
- Periksa daftar ``masalah baru-baru ini`__` untuk mengetahui jika telah dilaporkan.

### 9.1.2 Bagaimana saya mengajukan laporan bug yang (baik)?

- **Gunakan judul yang jelas dan deskriptif** untuk masalah tersebut untuk mengidentifikasi masalah.
- **Deskripsikan langkah-langkah yang tepat yang mereproduksi masalah** dalam rincian sebanyak mungkin.
- **Berikan contoh yang spesifik untuk menunjukkan langkah-langkahnya.**
- **Deskripsikan perilaku yang telah anda amati setelah mengikuti langkah-langkah tersebut** dan tunjukkan masalah apa sebenarnya dengan perilaku itu.
- **Deskripsikan perilaku mana yang ingin anda ketahui dan mengapa.**
- **Jika masalah tersebut tidak dicetus oleh tindakan tertentu**, deskripsikan apa yang anda lakukan sebelum masalah itu terjadi dan bagikan informasi lebih banyak dengan menggunakan panduan di bawah ini.
- **Berikan log yang terkait dengan masalah ini.** Gunakan bendera verbose untuk memulai proses masuk, dan ajukan seluruh log dari langkah pertama yang anda lakukan.

Berikan konteks lebih banyak dengan menjawab pertanyaan-pertanyaan ini:

- **Apakah masalah tersebut mulai terjadi baru-baru ini** (misalnya setelah memperbarui ke versi terbaru) atau apakah ini selalu menjadi masalah?
- **Bisakah anda dapat dipercaya mereproduksi masalah ini?** Jika tidak, berikan rincian tentang seberapa sering masalah tersebut terjadi dan dalam kondisi mana biasanya itu terjadi.

Masukkan rincian tentang konfigurasi dan lingkungan anda:

- **What version of EFB are you using?** You can get the version by using the flag `--version`.
- **Apakah nama dan versi OS yang anda gunakan?**

<b>Pewhatian:</b> Saat mengajukan log anda, harap ingat untuk menyembunyikan informasi pribadi anda.
--

## 9.2 Menyarankan perbaikan

Jika anda memiliki saran, jangan ragu untuk mengajukannya dalam daftar masalah. harap coba untuk memberikan rincian sebanyak mungkin yang anda bisa, yang mana termasuk:

- **\*\* Gunakan judul yang jelas dan deskriptif \*\*** untuk masalah ini untuk mengidentifikasi sarannya.
- **Berikan rincian tentang bagaimana perbaikan berperilaku.**
- **Provide specific examples to demonstrate the abstraction.**
- The enhancement to the framework must be applicable to considerably many IM platforms, not just for a single IM. Suggestions for a specific IM should be made to their relative channel.

Diadaptasi dari 'Panduan kontribusi Atom`\_\_ oleh GitHub Inc.

## 9.3 Tarik permintaan

Saat anda sudah melakukan beberapa perubahan dan ingin mengajukannya kepada kami, salin itu ke akun anda dan ajukan pull request Github. harap tulis deskripsi yang terperinci untuk pull request anda pada:

- **Perubahan apa yang sudah anda lakukan?**
- **Masalah apa yang sudah anda selesaikan?**
- **Masalah yang mana yang anda tujukan** jika berlaku.

Selalu tulis pesan log yang jelas untuk komit anda. Pesan satu baris baik untuk perubahan kecil, namun perubahan yang lebih besar memerlukan penjelasan rinci setelah satu baris.

Adapted from **`OpenGovernment contribution guide`** \_\_ by Participatory Politics Foundation





Bagian ini berisi pengumpulan untuk API Bot EH Forwarder saat ini.

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “NOT RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [BCP 14 \[RFC 2119\] \[RFC 8174\]](#) when, and only when, they appear in all capitals, as shown here.

## 10.1 Channel

**class** `ehforwarderbot.channel.Channel` (*instance\_id=None*)

The abstract channel class.

**channel\_name**

A human-friendly name of the channel.

**Type** `str`

**channel\_emoji**

Emoji icon of the channel. Recommended to use a visually-length-one (i.e. a single [grapheme cluster](#)) emoji or other symbol that represents the channel best.

**Type** `str`

**channel\_id**

Unique identifier of the channel. Convention of IDs is specified in [Mengemas dan mempublikasi](#). This ID will be appended with its instance ID when available.

**Type** `ModuleID` (`str`)

**instance\_id**

The instance ID if available.

**Type** `str`

**\_\_init\_\_** (*instance\_id=None*)

Initialize the channel. Inherited initializer MUST call the "super init" method at the beginning.

**Parameter** `instance_id` (`Optional[NewType() (InstanceID, str)]`) -- Instance ID of the channel.

**get\_message\_by\_id** (`chat, msg_id`)

Get message entity by its ID. Applicable to both master channels and slave channels. Return `None` when message not found.

Override this method and raise `EFBOperationNotSupported` if it is not feasible to perform this for your platform.

**Parameter**

- **chat** (`Chat`) -- Chat in slave channel / middleware.
- **msg\_id** (`NewType() (MessageID, str)`) -- ID of message from the chat in slave channel / middleware.

**Return type** `Optional[Message]`

**abstract poll** ()

Method to poll for messages. This method is called when the framework is initialized. This method SHOULD be blocking.

**abstract send\_message** (`msg`)

Process a message that is sent to, or edited in this channel.

---

#### Catatan

Master channel MUST take care of the returned object that contains the updated message ID. Depends on the implementation of slave channels, the message ID MAY change even after being edited. The old message ID MAY be disregarded for the new one.

---

**Parameter** `msg` (`Message`) -- Message object to be processed.

**Kembali** The same message object. Message ID of the object MAY be changed by the slave channel once sent. This can happen even when the message sent is an edited message.

**Return type** `Message`

**Raises**

- `EFBChatNotFound` -- Raised when a chat required is not found.
- `EFBMessageTypeNotSupported` -- Raised when the message type sent is not supported by the channel.
- `EFBOperationNotSupported` -- Raised when an message edit request is sent, but not supported by the channel.
- `EFBMessageNotFound` -- Raised when an existing message indicated is not found. E.g.: The message to be edited, the message referred in the `msg.target` attribute.
- `EFBMessageError` -- Raised when other error occurred while sending or editing the message.

**abstract send\_status** (`status`)

Process a status that is sent to this channel.

**Parameter** `status` (`Status`) -- the status object.

**Raises**

- ***EFBChatNotFound*** -- Raised when a chat required is not found.
- ***EFBMessageNotFound*** -- Raised when an existing message indicated is not found. E.g.: The message to be removed.
- ***EFBOperationNotSupported*** -- Raised when the channel does not support message removal.
- ***EFBMessageError*** -- Raised when other error occurred while removing the message.

---

**Catatan:** Exceptions SHOULD NOT be raised from this method by master channels as it would be hard for a slave channel to process the exception.

This method is not applicable to Slave Channels.

---

#### **stop\_polling()**

When EFB framework is asked to stop gracefully, this method is called to each channel object to stop all processes in the channel, save all status if necessary, and terminate polling.

When the channel is ready to stop, the polling function MUST stop blocking. EFB framework will quit completely when all polling threads end.

**class** ehforwarderbot.channel.**MasterChannel** (*instance\_id=None*)  
The abstract master channel class. All master channels MUST inherit this class.

**class** ehforwarderbot.channel.**SlaveChannel** (*instance\_id=None*)  
The abstract slave channel class. All slave channels MUST inherit this class.

#### **supported\_message\_types**

Types of messages that the slave channel accepts as incoming messages. Master channels may use this value to decide what type of messages to send to your slave channel.

Leaving this empty may cause the master channel to refuse sending anything to your slave channel.

**Type** Set[*MsgType*]

#### **suggested\_reactions**

A list of suggested reactions to be applied to messages.

Reactions SHOULD be ordered in a meaningful way, e.g., the order used by the IM platform, or frequency of usage. Note that it is not necessary to list all suggested reactions if that is too long, or not feasible.

Set to None when it is known that no reaction is supported to ANY message in the channel. Set to empty list when it is not feasible to provide a list of suggested reactions, for example, the list of reactions is different for each chat or message.

**Type** Optional[Sequence[*str*]]

#### **abstract get\_chat(chat\_uid)**

Get the chat object from a slave channel.

**Parameter** **chat\_uid** (NewType() (ChatID, *str*)) -- ID of the chat.

**Kembali** The chat found.

**Return type** .Chat

**Raises** ***EFBChatNotFound*** -- Raised when a chat required is not found.

#### **abstract get\_chat\_picture(chat)**

Get the profile picture of a chat. Profile picture is also referred as profile photo, avatar, "head image" sometimes.

**Parameter** `chat` (*.Chat*) -- Chat to get picture from.

**Kembali**

Opened temporary file object. The file object MUST have appropriate extension name that matches to the format of picture sent, and seek to position 0.

It MAY be deleted or discarded once closed, if not needed otherwise.

**Return type** BinaryIO

**Raises**

- *EFBChatNotFound* -- Raised when a chat required is not found.
- *EFBOperationNotSupported* -- Raised when the chat does not offer a profile picture.

### Contoh-contoh

```
if chat.channel_uid != self.channel_uid:
    raise EFBChannelNotFound()
file = tempfile.NamedTemporaryFile(suffix=".png")
response = requests.post("https://api.example.com/get_profile_picture/png",
                        data={"uid": chat.uid})
if response.status_code == 404:
    raise EFBChatNotFound()
file.write(response.content)
file.seek(0)
return file
```

**abstract** `get_chats()`

Return a list of available chats in the channel.

**Kembali** a list of available chats in the channel.

**Return type** Collection[*Chat*]

**get\_extra\_functions()**

Get a list of additional features

**Return type** Dict[NewType() (ExtraCommandName, str), Callable]

**Kembali** A dict of methods marked as additional features. Method can be called with `get_extra_functions()["methodName"]()`.

## 10.1.1 Operasi umum

### Mengirim pesan dan status

Sending messages and statuses to other channels is the most common operation of a channel. When enough information is gathered from external sources, the channel would then further process and pack them into the relevant objects, i.e. *Message* and *Status*.

When the object is built, the channel should sent it to the coordinator for following steps.

For now, both *Message* and *Status* has an attribute that indicates that where this object would be delivered to (`deliver_to` and `destination_channel`). This is used by the coordinator when delivering the message.

Messages MUST be sent using `coordinator.send_message()`. Statuses MUST be sent using `coordinator.send_status()`.

When the object is passed onto the coordinator, it will be further processed by the middleware and then to its destination.

For example, to send a message to the master channel

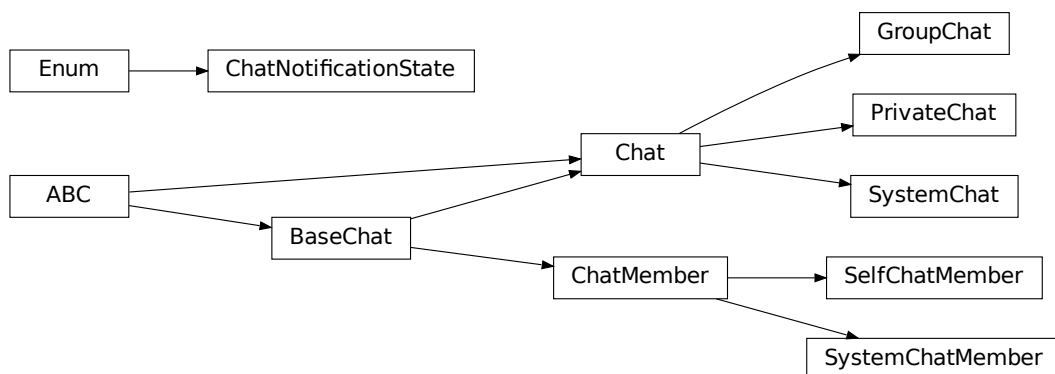
```
def on_message(self, data: Dict[str, Any]):
    """Callback when a message is received by the slave channel from
    the IM platform.
    """
    # Prepare message content ...
    message = coordinator.send_message(Message(
        chat=chat,
        author=author,
        type=message_type,
        text=text,
        # more details ...
        uid=data['uid'],
        deliver_to=coordinator.master
    ))
    # Post-processing ...
```

### 10.1.2 Tentang ID Saluran

Dengan diperkenalkannya contoh ID, diperlukan untuk menggunakan `self.channel_id` atau setara dengan ID kode keras atau konstanta saat mengacu pada peranti tengah (misalnya saat mengambil jalur pada berkas konfigurasi, membuat objek obrolan dan pesan, dll).

## 10.2 Chat and Chat Members

### Inheritance diagram



## Summary

<code>PrivateChat(*[, channel, middleware, ...])</code>	A private chat, where usually only the User Themselves and the other participant are in the chat.
<code>SystemChat(*[, channel, middleware, ...])</code>	A system chat, where usually only the User Themselves and the other participant (system chat member) are in the chat.
<code>GroupChat(*[, channel, middleware, ...])</code>	A group chat, where there are usually multiple members present.
<code>ChatMember(chat, *[, name, alias, uid, id, ...])</code>	Member of a chat.
<code>SelfChatMember(chat, *[, name, alias, id, ...])</code>	The User Themselves as member of a chat.
<code>SystemChatMember(chat, *[, name, alias, id, ...])</code>	A system account/prompt as member of a chat.
<code>ChatNotificationState(value)</code>	Indicates the notifications settings of a chat in its slave channel or middleware.

## Classes

```
class ehforwarderbot.chat.BaseChat (*, channel=None, middleware=None, module_name="",
                                   channel_emoji="", module_id="", name="", alias=None, uid="",
                                   id="", vendor_specific=None, description="")
```

Basis: `abc.ABC`

Base chat class, this is an abstract class sharing properties among all chats and members. No instance can be created directly from this class.

---

**Catatan:** `BaseChat` objects are picklable, thus it is RECOMMENDED to keep any object of its subclass also picklable.

---

### **module\_id**

Unique ID of the module.

**Type** `ModuleID` (str)

### **channel\_emoji**

Emoji of the channel, empty string if the chat is from a middleware.

**Type** `str`

### **module\_name**

Name of the module.

**Type** `ModuleID` (str)

### **name**

Name of the chat.

**Type** `str`

### **alias**

Alternative name of the chat, usually set by user.

**Type** `Optional[str]`

### **uid**

Unique ID of the chat. This MUST be unique within the channel.

**Type** `ChatID` (str)

**description**

A text description of the chat, usually known as “bio”, “description”, “purpose”, or “topic” of the chat.

Type `str`

**vendor\_specific**

Any vendor specific attributes.

Type `Dict[str, Any]`

```
__init__ (*, channel=None, middleware=None, module_name="", channel_emoji="", module_id="", name="",
          alias=None, uid="", id="", vendor_specific=None, description="")
```

**Parameter**

- **channel** (Optional[*SlaveChannel*]) -- Provide the channel object to fill *module\_name*, *channel\_emoji*, and *module\_id* automatically.
- **middleware** (Optional[*Middleware*]) -- Provide the middleware object to fill *module\_name*, and *module\_id* automatically.
- **module\_id** (NewType() (ModuleID, `str`)) -- Unique ID of the module.
- **channel\_emoji** (`str`) -- Emoji of the channel, empty string if the chat is from a middleware.
- **module\_name** (`str`) -- Name of the module.
- **name** (`str`) -- Name of the chat.
- **alias** (Optional[`str`]) -- Alternative name of the chat, usually set by user.
- **uid** (NewType() (ChatID, `str`)) -- Unique ID of the chat. This MUST be unique within the channel.
- **description** (`str`) -- A text description of the chat, usually known as “bio”, “description”, “purpose”, or “topic” of the chat.
- **vendor\_specific** (Dict[`str`, Any]) -- Any vendor specific attributes.

**copy()**

Return a shallow copy of the object.

Return type `TypeVar(_BaseChatSelf, bound=BaseChat, covariant=True)`

**property display\_name: str**

Shortcut property, equivalent to `alias` or `name`

Return type `str`

**property long\_name: str**

Shortcut property, if `alias` exists, this will provide the alias with name in parenthesis. Otherwise, this will return the name

Return type `str`

**abstract verify()**

Verify the completeness of the data.

Raises `AssertionError` -- When this chat is invalid.

```
class ehforwarderbot.chat.Chat (*, channel=None, middleware=None, module_name="",
                                channel_emoji="", module_id="", name="", alias=None, id="", uid="",
                                vendor_specific=None, description="", members=None,
                                notification=ChatNotificationState.ALL, with_self=True)
```

Basis: `ehforwarderbot.chat.BaseChat`, `abc.ABC`

A chat object, indicates a user, a group, or a system chat. This class is abstract. No instance can be created directly from this class.

If your IM platform is providing an ID of the User Themselves, and it is using this ID to indicate the author of a message, you SHOULD update `Chat.self.uid` accordingly.

```
>>> channel.my_chat_id
"david_divad"
>>> chat = Chat(channel=channel, name="Alice", uid=ChatID("alice123"))
>>> chat.self.uid = channel.my_chat_id
```

By doing so, you can get the author in one step:

```
author = chat.get_member(author_id)
```

... instead of using a condition check:

```
if author_id == channel.my_chat_id:
    author = chat.self
else:
    author = chat.get_member(author_id)
```

---

**Catatan:** Chat objects are picklable, thus it is RECOMMENDED to keep any object of its subclass also picklable.

---

### **module\_id**

Unique ID of the module.

**Type** `ModuleID` (str)

### **channel\_emoji**

Emoji of the channel, empty string if the chat is from a middleware.

**Type** str

### **module\_name**

Name of the module.

**Type** str

### **name**

Name of the chat.

**Type** str

### **alias**

Alternative name of the chat, usually set by user.

**Type** Optional[str]

### **uid**

Unique ID of the chat. This MUST be unique within the channel.

**Type** `ChatID` (str)



**description**

A text description of the chat, usually known as “bio”, “description”, “purpose”, or “topic” of the chat.

Type `str`

**notification**

Indicate the notification settings of the chat in its slave channel (or middleware), defaulted to `ALL`.

Type `ChatNotificationState`

**members**

Provide a list of members in the chat. Defaulted to an empty `list`.

You can extend this object and implement a `@property` method set for loading members on demand.

Note that this list may include members created by middlewares when the object is a part of a message, and these members MAY not appear when trying to retrieve from the slave channel directly. These members would have a different `module_id` specified from the chat.

Type list of `ChatMember`

**vendor\_specific**

Any vendor specific attributes.

Type `Dict[str, Any]`

**self**

the User as a member of the chat (if available).

Type `Optional[SelfChatMember]`

**\_\_init\_\_** (\*, `channel=None`, `middleware=None`, `module_name=""`, `channel_emoji=""`, `module_id=""`, `name=""`, `alias=None`, `id=""`, `uid=""`, `vendor_specific=None`, `description=""`, `members=None`, `notification=ChatNotificationState.ALL`, `with_self=True`)

**Argumen Kata Kunci**

- **module\_id** (`str`) -- Unique ID of the module.
- **channel\_emoji** (`str`) -- Emoji of the channel, empty string if the chat is from a middleware.
- **module\_name** -- Name of the module.
- **name** (`str`) -- Name of the chat.
- **alias** (`Optional[str]`) -- Alternative name of the chat, usually set by user.
- **id** -- Unique ID of the chat. This MUST be unique within the channel.
- **description** (`str`) -- A text description of the chat, usually known as “bio”, “description”, “purpose”, or “topic” of the chat.
- **notification** (`ChatNotificationState`) -- Indicate the notification settings of the chat in its slave channel (or middleware), defaulted to `ALL`.
- **members** (`MutableSequence[ChatMember]`) -- Provide a list of members of the chat. Defaulted to an empty `list`.
- **vendor\_specific** (`Dict[str, Any]`) -- Any vendor specific attributes.
- **with\_self** (`bool`) -- Initialize the chat with the User Themselves as a member.

**add\_member** (*name*, *uid*, *alias=None*, *id=""*, *vendor\_specific=None*, *description=""*, *middleware=None*)  
Add a member to the chat.

---

**Tip:** This method does not check for duplicates. Only add members with this method if you are sure that they are not added yet. To check if the member is already added before adding, you can do something like this:

```
with contextlib.suppress(KeyError):  
    return chat.get_member(uid)  
return chat.add_member(name, uid, alias=..., vendor_specific=...)
```

---

#### Parameter

- **name** (*str*) -- Name of the member.
- **uid** (*NewType() (Chat ID, str)*) -- ID of the member.

#### Argumen Kata Kunci

- **alias** (*Optional[str]*) -- Alias of the member.
- **vendor\_specific** (*Dict[str, Any]*) -- Any vendor specific attributes.
- **description** (*str*) -- A text description of the chat, usually known as “bio”, “description”, “purpose”, or “topic” of the chat.
- **middleware** (*Optional[Middleware]*) -- Initialize this chat as a part of a middleware.

**Return type** *ChatMember*

**add\_self** ()  
Add self to the list of members.

**Raises** **AssertionError** -- When there is already a self in the list of members.

**Return type** *SelfChatMember*

**add\_system\_member** (*name=""*, *alias=None*, *id=""*, *uid=""*, *vendor\_specific=None*, *description=""*, *middleware=None*)  
Add a system member to the chat.

Useful for slave channels and middlewares to create an author of a message from a system member when the “system” member is intended to become a member of the chat.

---

**Tip:** This method does not check for duplicates. Only add members with this method if you are sure that they are not added yet.

---

#### Argumen Kata Kunci

- **name** (*str*) -- Name of the member.
- **uid** -- ID of the member.
- **alias** (*Optional[str]*) -- Alias of the member.
- **vendor\_specific** (*Dict[str, Any]*) -- Any vendor specific attributes.
- **description** (*str*) -- A text description of the chat, usually known as “bio”, “description”, “purpose”, or “topic” of the chat.

- **middleware** (Optional[*Middleware*]) -- Initialize this chat as a part of a middleware.

**Return type** *SystemChatMember*

**get\_member** (*member\_id*)

Find a member of chat by its ID.

**Parameter** *member\_id* (NewType () (ChatID, *str*)) -- ID of the chat member.

**Return type** *ChatMember*

**Kembali** the chat member.

**Raises** *KeyError* -- when the ID provided is not found.

**property has\_self:** *bool*

Indicate if this chat has yourself.

**Return type** *bool*

**make\_system\_member** (*name=""*, *alias=None*, *id=""*, *uid=""*, *vendor\_specific=None*, *description=""*,  
*middleware=None*)

Make a system member for this chat.

Useful for slave channels and middlewares to create an author of a message from a system member when the “system” member is NOT intended to become a member of the chat.

#### Argumen Kata Kunci

- **name** (*str*) -- Name of the member.
- **uid** -- ID of the member.
- **alias** (Optional[*str*]) -- Alias of the member.
- **vendor\_specific** (Dict[*str*, Any]) -- Any vendor specific attributes.
- **description** (*str*) -- A text description of the chat, usually known as “bio”, “description”, “purpose”, or “topic” of the chat.
- **middleware** (Optional[*Middleware*]) -- Initialize this chat as a part of a middleware.

**Return type** *SystemChatMember*

**self:** Optional[*ehforwarderbot.chat.SelfChatMember*]

The user as a member of the chat (if available).

**class** *ehforwarderbot.chat.ChatMember* (*chat*, \*, *name=""*, *alias=None*, *uid=""*, *id=""*,  
*vendor\_specific=None*, *description=""*, *middleware=None*)

Basis: *ehforwarderbot.chat.BaseChat*

Member of a chat. Usually indicates a member in a group, or the other participant in a private chat. Chat bots created by the users of the IM platform is also considered as a plain *ChatMember*.

To represent the User Themselves, use *SelfChatMember*.

To represent a chat member that is a part of the system, the slave channel, or a middleware, use *SystemChatMember*.

*ChatMembers* MUST be created with reference of the chat it belongs to. Different objects MUST be created even when the same person appears in different groups or in a private chat.

*ChatMembers* are RECOMMENDED to be created using *Chat.add\_member()* method.

---

**Catatan:** `ChatMember` objects are pickleable, thus it is RECOMMENDED to keep any object of its subclass also pickleable.

---

```
__init__(chat, *, name="", alias=None, uid="", id="", vendor_specific=None, description="",
         middleware=None)
```

**Parameter** `chat` (*Chat*) -- Chat associated with this member.

#### Argumen Kata Kunci

- **name** (*str*) -- Name of the member.
- **alias** (*Optional[str]*) -- Alternative name of the member, usually set by user.
- **uid** (*ChatID* (*str*)) -- Unique ID of the member. This MUST be unique within the channel. This ID can be the same with a private chat of the same person.
- **description** (*str*) -- A text description of the member, usually known as “bio”, “description”, “summary” or “introduction” of the member.
- **middleware** (*Middleware*) -- Initialize this chat as a part of a middleware.

**verify** ()

Verify the completeness of the data.

**Raises** `AssertionError` -- When this chat is invalid.

**class** `ehforwarderbot.chat.ChatNotificationState` (*value*)

Basis: `enum.Enum`

Indicates the notifications settings of a chat in its slave channel or middleware. If an exact match is not available, choose the most similar one.

**ALL** = -1

All messages in the chat triggers notifications.

**MENTIONS** = 1

Notifications are sent only when the User is mentioned in the message, in the form of @-references or quote-reply (message target).

**NONE** = 0

No notification is sent to slave IM channel at all.

```
class ehforwarderbot.chat.GroupChat (*, channel=None, middleware=None, module_name="",
                                     channel_emoji="", module_id="", name="", alias=None, id="",
                                     uid="", vendor_specific=None, description="",
                                     notification=ChatNotificationState.ALL, with_self=True)
```

Basis: `ehforwarderbot.chat.Chat`

A group chat, where there are usually multiple members present.

Members can be added with the `add_member()` method.

If the `with_self` argument is `True` (which is the default setting), the User Themselves would also be initialized as a member of the chat.

## Contoh-contoh

```
>>> group = GroupChat(channel=slave_channel, name="Wonderland", uid=ChatID(
↳ "wonderland001"))
>>> group.add_member(name="Alice", uid=ChatID("alice"))
ChatMember(chat=<GroupChat: Wonderland (wonderland001) @ Example slave channel>,
↳ name='Alice', alias=None, uid='alice', vendor_specific={}, description='')
>>> group.add_member(name="bob", alias="Bob James", uid=ChatID("bob"))
ChatMember(chat=<GroupChat: Wonderland (wonderland001) @ Example slave channel>,
↳ name='bob', alias='Bob James', uid='bob', vendor_specific={}, description='')
>>> from pprint import pprint
>>> pprint(group.members)
[SelfChatMember(chat=<GroupChat: Wonderland (wonderland001) @ Example slave
↳ channel>, name='You', alias=None, uid='__self__', vendor_specific={},
↳ description=''),
ChatMember(chat=<GroupChat: Wonderland (wonderland001) @ Example slave channel>,
↳ name='Alice', alias=None, uid='alice', vendor_specific={}, description=''),
ChatMember(chat=<GroupChat: Wonderland (wonderland001) @ Example slave channel>,
↳ name='bob', alias='Bob James', uid='bob', vendor_specific={}, description='')]

```

**Catatan:** GroupChat objects are picklable, thus it is RECOMMENDED to keep any object of its subclass also picklable.

### verify()

Verify the completeness of the data.

Raises **AssertionError** -- When this chat is invalid.

```
class ehforwarderbot.chat.PrivateChat(*, channel=None, middleware=None, module_name="",
channel_emoji="", module_id="", name="", alias=None, id="",
uid="", vendor_specific=None, description="",
notification=ChatNotificationState.ALL, with_self=True,
other_is_self=False)

```

Basis: `ehforwarderbot.chat.Chat`

A private chat, where usually only the User Themselves and the other participant are in the chat. Chat bots SHOULD also be categorized under this type.

There SHOULD only be at most one non-system member of the chat apart from the User Themselves, otherwise it might lead to unintended behavior.

This object is by default initialized with the other participant as its member.

If the `with_self` argument is `True` (which is the default setting), the User Themselves would also be initialized as a member of the chat.

**Parameter other** -- the other participant of the chat as a member

**Catatan:** PrivateChat objects are picklable, thus it is RECOMMENDED to keep any object of its subclass also picklable.

### verify()

Verify the completeness of the data.

Raises **AssertionError** -- When this chat is invalid.

```
class ehforwarderbot.chat.SelfChatMember (chat, *, name="", alias=None, id="", uid="",
                                         vendor_specific=None, description="",
                                         middleware=None)
```

Basis: `ehforwarderbot.chat.ChatMember`

The User Themselves as member of a chat.

`SelfChatMembers` are RECOMMENDED to be created together with a chat object by setting `with_self` value to `True`. The created object is accessible at `Chat.self`.

The default ID of a `SelfChatMember` object is `SelfChatMember.SELF_ID`, and the default name is a translated version of the word “You”.

You are RECOMMENDED to change the ID of this object if provided by your IM platform, and you MAY change the name or alias of this object depending on your needs.

---

**Catatan:** `SelfChatMember` objects are picklable, thus it is RECOMMENDED to keep any object of its subclass also picklable.

---

#### **SELF\_ID**

The default ID of a `SelfChatMember`.

```
__init__ (chat, *, name="", alias=None, id="", uid="", vendor_specific=None, description="",
          middleware=None)
```

**Parameter** `chat` (`Chat`) -- Chat associated with this member.

#### **Argumen Kata Kunci**

- **name** (`str`) -- Name of the member.
- **alias** (`Optional[str]`) -- Alternative name of the member, usually set by user.
- **uid** (`ChatID` (`str`)) -- Unique ID of the member. This MUST be unique within the channel. This ID can be the same with a private chat of the same person.
- **description** (`str`) -- A text description of the member, usually known as “bio”, “description”, “summary” or “introduction” of the member.
- **middleware** (`Middleware`) -- Initialize this chat as a part of a middleware.

```
class ehforwarderbot.chat.SystemChat (*, channel=None, middleware=None, module_name="",
                                       channel_emoji="", module_id="", name="", alias=None, id="",
                                       uid="", vendor_specific=None, description="",
                                       notification=ChatNotificationState.ALL, with_self=True)
```

Basis: `ehforwarderbot.chat.Chat`

A system chat, where usually only the User Themselves and the other participant (system chat member) are in the chat. This object is used to represent system chat where the other participant is neither a user nor a chat bot of the remote IM.

Middlewares are RECOMMENDED to create chats with this type when they want to send messages in this type.

This object is by default initialized with the system participant as its member.

If the `with_self` argument is `True` (which is the default setting), the User Themselves would also be initialized as a member of the chat.

**Parameter** `other` -- the other participant of the chat as a member

---

**Catatan:** `SystemChat` objects are picklable, thus it is RECOMMENDED to keep any object of its subclass also picklable.

---

**verify()**

Verify the completeness of the data.

**Raises** `AssertionError` -- When this chat is invalid.

```
class ehforwarderbot.chat.SystemChatMember(chat, *, name="", alias=None, id="", uid="",
                                           vendor_specific=None, description="",
                                           middleware=None)
```

Basis: `ehforwarderbot.chat.ChatMember`

A system account/prompt as member of a chat.

Use this chat to send messages that is not from any specific member. Middlewares are RECOMMENDED to use this member type to communicate with the User in an existing chat.

Chat bots created by the users of the IM platform SHOULD NOT be created as a `SystemChatMember`, but a plain `ChatMember` instead.

`SystemChatMembers` are RECOMMENDED to be created using `Chat.add_system_member()` or `Chat.make_system_member()` method.

---

**Catatan:** `SystemChatMember` objects are picklable, thus it is RECOMMENDED to keep any object of its subclass also picklable.

---

**SYSTEM\_ID**

The default ID of a `SystemChatMember`.

```
__init__(chat, *, name="", alias=None, id="", uid="", vendor_specific=None, description="",
         middleware=None)
```

**Parameter** `chat` (`Chat`) -- Chat associated with this member.

**Argumen Kata Kunci**

- **name** (`str`) -- Name of the member.
- **alias** (`Optional[str]`) -- Alternative name of the member, usually set by user.
- **uid** (`ChatID`(`str`)) -- Unique ID of the member. This MUST be unique within the channel. This ID can be the same with a private chat of the same person.
- **description** (`str`) -- A text description of the member, usually known as “bio”, “description”, “summary” or “introduction” of the member.
- **middleware** (`Middleware`) -- Initialize this chat as a part of a middleware.

## 10.3 Konstan

**class** ehforwarderbot.constants.**MsgType** (*value*)

Sebuah enumerasi.

**Animation** = 'Animation'

Message with an animation, usually in the form of GIF or soundless video.

**Audio** = 'Voice'

Audio messages (deprecated).

Ditinggalkan sejak versi Use: *Voice* if the message has a voice message (usually recorded). Use *File* if the message has a music file (usually uploaded).

**File** = 'File'

File message.

**Image** = 'Image'

Image (picture) message.

---

### Catatan

Animated GIF images must use *Animation* type instead.

---

**Link** = 'Link'

Pesan yang terutama salah satu tautan tertentu, atau sebuah pesan teks dengan satu peninjauan tautan.

**Location** = 'Location'

Lokasi pesan.

**Status** = 'Status'

Status dari pengguna dalam obrolan, biasanya mengetik dan mengunggah.

**Sticker** = 'Sticker'

Gambar dikirim dengan beberapa keterangan teks, biasanya berlatar belakang transparan, dan sejumlah pilihan yang terbatas yang biasanya bukan berasal dari galeri foto pengguna.

**Text** = 'Text'

Pesan teks

**Unsupported** = 'Unsupported'

Apa pun jenis pesan yang tidak tercantum di atas. Sebuah teks representasi diperlukan.

**Video** = 'Video'

Pesan video

**Voice** = 'Voice'

Voice messages, usually recorded right before sending.



## 10.4 Koordinator

Coordinator among channels.

`ehforwarderbot.coordinator.profile`  
Name of current profile..

**Type** `str`

`ehforwarderbot.coordinator.mutex`  
Global interaction thread lock.

**Type** `threading.Lock`

`ehforwarderbot.coordinator.master`  
The running master channel object.

**Type** `Channel`

`ehforwarderbot.coordinator.slaves`  
Dictionary of running slave channel object. Keys are the unique identifier of the channel.

**Type** `Dict[str, EFBCChannel]`

`ehforwarderbot.coordinator.middlewares`  
List of middlewares

**Type** `List[Middleware]`

`ehforwarderbot.coordinator.add_channel(channel)`  
Register the channel with the coordinator.

**Parameter channel** (`Channel`) -- Channel to register

`ehforwarderbot.coordinator.add_middleware(middleware)`  
Register a middleware with the coordinator.

**Parameter middleware** (`Middleware`) -- Middleware to register

`ehforwarderbot.coordinator.get_module_by_id(module_id)`  
Return the module instance of a provided module ID

**Parameter module\_id** (`NewType() (ModuleID, str)`) -- Module ID, with instance ID if available.

**Return type** `Union[Channel, Middleware]`

**Kembali** Module instance requested.

**Raises** `NameError` -- When the module is not found.

`ehforwarderbot.coordinator.master: ehforwarderbot.channel.MasterChannel`  
The instance of the master channel.

`ehforwarderbot.coordinator.master_thread: Optional[threading.Thread] = None`  
The thread running poll() of the master channel.

`ehforwarderbot.coordinator.middlewares:`  
`List[ehforwarderbot.middleware.Middleware] = []`  
Instances of middlewares. Sorted in the order of execution.

`ehforwarderbot.coordinator.mutex: _thread.allocate_lock = <unlocked  
_thread.lock object>`  
Mutual exclusive lock for user interaction through CLI interface

`ehforwarderbot.coordinator.profile: str = 'default'`

Current running profile name

`ehforwarderbot.coordinator.send_message(msg)`

Deliver a new message or edited message to the destination channel.

**Parameter** `msg` (`Message`) -- The message

**Return type** `Optional[Message]`

**Kembali** The message processed and delivered by the destination channel, includes the updated message ID if sent to a slave channel. Returns `None` if the message is not sent.

`ehforwarderbot.coordinator.send_status(status)`

Deliver a status to the destination channel.

**Parameter** `status` (`Status`) -- The status

`ehforwarderbot.coordinator.slave_threads: Dict[ModuleID, threading.Thread] = {}`

Threads running `poll()` from slave channels. Keys are the channel IDs.

`ehforwarderbot.coordinator.slaves: Dict[ModuleID, ehforwarderbot.channel.SlaveChannel] = {}`

Instances of slave channels. Keys are the channel IDs.

`ehforwarderbot.coordinator.translator: gettext.NullTranslations = <gettext.NullTranslations object>`

Internal GNU gettext translator.

## 10.5 Pengecualian

**exception** `ehforwarderbot.exceptions.EFBException`

Basis: `Exception`

A general class to indicate that the exception is from EFB framework.

**exception** `ehforwarderbot.exceptions.EFBChatNotFound`

Basis: `ehforwarderbot.exceptions.EFBException`

Raised by a slave channel when a chat indicated is not found.

Can be raised by any method that involves a chat or a message.

**exception** `ehforwarderbot.exceptions.EFBChannelNotFound`

Basis: `ehforwarderbot.exceptions.EFBException`

Raised by the coordinator when the message sent delivers to a missing channel.

**exception** `ehforwarderbot.exceptions.EFBMessageError`

Basis: `ehforwarderbot.exceptions.EFBException`

Raised by slave channel for any other error occurred when sending a message or a status.

Can be raised in `Channel.send_message()` and `Channel.send_status()`.

**exception** `ehforwarderbot.exceptions.EFBMessageNotFound`

Basis: `ehforwarderbot.exceptions.EFBMessageError`

Raised by a slave channel when a message indicated is not found.

Can be raised in `Channel.send_message()` (edited message / target message not found) and in `Channel.send_status()` (message to delete is not found).

**exception** `ehforwarderbot.exceptions.EFBMessageTypeNotSupported`

Basis: `ehforwarderbot.exceptions.EFBMessageError`

Raised by a slave channel when the indicated message type is not supported.

Can be raised in `Channel.send_message()`.

**exception** `ehforwarderbot.exceptions.EFBOperationNotSupported`

Basis: `ehforwarderbot.exceptions.EFBMessageError`

Raised by slave channels when a chat operation is not supported. E.g.: cannot edit message, cannot delete message.

Can be raised in `Channel.send_message()` and `Channel.send_status()`.

**exception** `ehforwarderbot.exceptions.EFBMessageReactionNotPossible`

Basis: `ehforwarderbot.exceptions.EFBException`

Raised by slave channel when a message reaction request from master channel is not possible to be processed.

Can be raised in `Channel.send_status()`.

## 10.6 Message

### Summary

<code>Message(*[, attributes, chat, author, ...])</code>	A message.
<code>LinkAttribute(title[, description, image, url])</code>	Attributes for link messages.
<code>LocationAttribute(latitude, longitude)</code>	Attributes for location messages.
<code>StatusAttribute(status_type[, timeout])</code>	Attributes for status messages.
<code>MessageCommands(commands)</code>	Message commands.
<code>MessageCommand(name, callable_name[, args, ...])</code>	A message command.
<code>Substitutions(substitutions)</code>	Message text substitutions, or “@-references”.

### Classes

**class** `ehforwarderbot.message.Message(*, attributes=None, chat=None, author=None, commands=None, deliver_to=None, edit=False, edit_media=False, file=None, filename=None, is_system=False, mime=None, path=None, reactions=None, substitutions=None, target=None, text="", type=MsgType.Unsupported, uid=None, vendor_specific=None)`

A message.

---

**Catatan:** Message objects are picklable, thus it is strongly RECOMMENDED to keep any object of its subclass also picklable.

---

### Argumen Kata Kunci

- **attributes** (Optional[`MessageAttribute`]) -- Attributes used for a specific message type. Only specific message type requires this attribute. Defaulted to `None`.
  - Link: `LinkAttribute`

- Location: *LocationAttribute*
- Status: Typing/Sending files/etc.: *StatusAttribute*

---

**Catatan:** Do NOT use object of the abstract class *MessageAttribute* for attributes, but object of specific class instead.

---

- **chat** (*Chat*) -- Sender of the message.
- **author** (*ChatMember*) -- Author of this message. Author of the message MUST be indicated as a part of the same chat this message is from. If the message is sent from the User Themselves, this MUST be an object of *SelfChatMember*.

Note that the author MAY not be inside *members* of the chat of this message. The author MAY have a different *module\_id* from the chat, and could be un retrievable otherwise.

- **commands** (Optional[*MessageCommands*]) -- Commands attached to the message

This attribute will be ignored in *\_Status\_* messages.

- **deliver\_to** (*Channel*) -- The channel that the message is to be delivered to.
- **edit** (*bool*) -- Flag this up if the message is edited. Flag only this if no multimedia file is modified, otherwise flag up both this one and *edit\_media* as well.

If no media file is modified, the edited message MAY carry no information about the file.

This attribute will be ignored in *\_Status\_* messages.

- **edit\_media** (*bool*) -- Flag this up if any file attached to the message is modified. If this value is true, *edit* MUST also be True. This attribute is ignored if the message type is not supposed to contain any media file, e.g. Text, Location, etc.

This attribute will be ignored in *\_Status\_* messages.

- **file** (Optional[*BinaryIO*]) -- File object to multimedia file, type "rb". None if N/A. Recommended to use *NamedTemporaryFile*. The file SHOULD be able to be safely deleted (or otherwise discarded) once closed. All file object MUST be sought back to 0 (*file.seek(0)*) before sending.

- **filename** (Optional[*str*]) -- File name of the multimedia file. None if N/A

- **is\_system** (*bool*) -- Mark as true if this message is a system message.

- **mime** (Optional[*str*]) -- MIME type of the file. None if N/A

- **path** (Optional[*Path*]) -- Local path of multimedia file. None if N/A

- **reactions** (Dict[str, Collection[*Chat*]]) -- Indicate reactions to the message. Dictionary key is the canonical name of reaction, usually an emoji. Value is a collection of users who reacted to the message with that certain emoji. All *Chat* objects in this dict MUST be members in the chat of this message.

This attribute will be ignored in *\_Status\_* messages.

- **substitutions** (Optional[*Substitutions*]) -- Substitutions of messages, usually used when the some parts of the text of the message refers to another user or chat.

This attribute will be ignored in *\_Status\_* messages.

- **target** (Optional[*Message*]) -- Target message (usually for messages that "replies to" another message).

This attribute will be ignored in *\_Status\_* messages.

---

**Catatan:** This message MAY be a "minimum message", with only required fields:

- `Message.chat`
  - `Message.author`
  - `Message.text`
  - `Message.type`
  - `Message.uid`
- 

- **text** (*str*) -- Text of the message.  
This attribute will be ignored in `_Status_` messages.
- **type** (*MsgType*) -- Type of message
- **uid** (*str*) -- Unique ID of message. Usually stores the message ID from slave channel. This ID MUST be unique among all chats in the same channel.

---

**Catatan:** Some channels may not support message editing. Some channels may issue a new uid for edited message.

---

- **vendor\_specific** (*Dict[str, Any]*) -- A series of vendor specific attributes attached. This can be used by any other channels or middlewares that is compatible with such information. Note that no guarantee is provided for information in this section.

**property link:** `Optional[ehforwarderbot.message.LinkAttribute]`

Get the link attributes of the current message, if available.

**Return type** `Optional[LinkAttribute]`

**property location:** `Optional[ehforwarderbot.message.LocationAttribute]`

Get the location attributes of the current message, if available.

**Return type** `Optional[LocationAttribute]`

**property status:** `Optional[ehforwarderbot.message.StatusAttribute]`

Get the status attributes of the current message, if available.

**Return type** `Optional[StatusAttribute]`

**verify()**

Verify the validity of message.

**Raises** `AssertionError` -- when the message is not valid

**class** `ehforwarderbot.message.MessageAttribute`

Basis: `abc.ABC`

Abstract class of a message attribute.

**class** `ehforwarderbot.message.LinkAttribute` (*title, description=None, image=None, url=""*)

Basis: `ehforwarderbot.message.MessageAttribute`

Attributes for link messages.

**title**

Title of the link.

**Type** `str`

**description**

Description of the link.

Type `str`, optional

**image**

Image/thumbnail URL of the link.

Type `str`, optional

**url**

URL of the link.

Type `str`

**\_\_init\_\_** (*title, description=None, image=None, url=""*)

**Parameter**

- **title** (*str*) -- Title of the link.
- **description** (*str, optional*) -- Description of the link.
- **image** (*str, optional*) -- Image/thumbnail URL of the link.
- **url** (*str*) -- URL of the link.

**class** ehforwarderbot.message.**LocationAttribute** (*latitude, longitude*)

Basis: `ehforwarderbot.message.MessageAttribute`

Attributes for location messages.

**latitude**

Latitude of the location.

Type `float`

**longitude**

Longitude of the location.

Type `float`

**\_\_init\_\_** (*latitude, longitude*)

**Parameter**

- **latitude** (*float*) -- Latitude of the location.
- **longitude** (*float*) -- Longitude of the location.

**class** ehforwarderbot.message.**MessageCommand** (*name, callable\_name, args=None, kwargs=None*)

Basis: `object`

A message command.

This object records a way to call a method in the module object. In case where the message has an `author` from a different module from the `chat`, this function **MUST** be called on the `author`'s module.

The method specified **MUST** return either a `str` as result or `None` if this message will be edited or deleted for further interactions.

**name**

Human-friendly name of the command.

Type `str`

**callable\_name**

Callable name of the command.

**Type** `str`

**args**

Arguments passed to the function.

**Type** `Collection[Any]`

**kwargs**

Keyword arguments passed to the function.

**Type** `Mapping[str, Any]`

**\_\_init\_\_** (*name*, *callable\_name*, *args=None*, *kwargs=None*)

**Parameter**

- **name** (*str*) -- Human-friendly name of the command.
- **callable\_name** (*str*) -- Callable name of the command.
- **args** (*Optional[Collection[Any]]*) -- Arguments passed to the function. Defaulted to empty list;
- **kwargs** (*Optional[Mapping[str, Any]]*) -- Keyword arguments passed to the function. Defaulted to empty dict.

**class** `ehforwarderbot.message.MessageCommands` (*commands*)

Basis: `List[ehforwarderbot.message.MessageCommand]`

Message commands.

Message commands allow user to take action to a specific message, including vote, add friends, etc.

**commands**

Commands for the message.

**Type** list of `MessageCommand`

**\_\_init\_\_** (*commands*)

**Parameter** **commands** (list of `MessageCommand`) -- Commands for the message.

**class** `ehforwarderbot.message.StatusAttribute` (*status\_type*, *timeout=5000*)

Basis: `ehforwarderbot.message.MessageAttribute`

Attributes for status messages.

Message with type `Status` notifies the other end to update a chat-specific status, such as typing, send files, etc.

**status\_type**

Type of status, possible values are defined in the `StatusAttribute`.

**timeout**

Number of milliseconds for this status to expire. Default to 5 seconds.

**Type** `Optional[int]`

**Types**

List of status types supported

**class** **Types** (*value*)

Basis: `enum.Enum`

**TYPING**

Used in `status_type`, represent the status of typing.

**UPLOADING\_FILE**

Used in `status_type`, represent the status of uploading file.

**UPLOADING\_IMAGE**

Used in `status_type`, represent the status of uploading image.

**UPLOADING\_VOICE**

Used in `status_type`, represent the status of uploading voice.

**UPLOADING\_VIDEO**

Used in `status_type`, represent the status of uploading video.

`__init__` (`status_type`, `timeout=5000`)

**Parameter**

- **status\_type** (*Types*) -- Type of status.
- **timeout** (*Optional[int]*) -- Number of milliseconds for this status to expire. Default to 5 seconds.

**class** ehforwarderbot.message.Substitutions (*substitutions*)

Basis: `Dict[Tuple[int, int], Union[ehforwarderbot.chat.Chat, ehforwarderbot.chat.ChatMember]]`

Message text substitutions, or “@-references”.

This is for the case when user “@-referred” a list of users in the message. Substitutions here is a dict of correspondence between the index of substring used to refer to a user/chat in the message and the chat object it referred to.

Values of the dictionary MUST be either a member of the chat (`self` or the other for private chats, group members for group chats) or another chat of the slave channel.

A key in this dictionary is a tuple of two `ints`, where first of it is the starting position in the string, and the second is the ending position defined similar to Python's substring. A tuple of `(3, 15)` corresponds to `msg.text[3:15]`. The value of the tuple `(a, b)` MUST satisfy  $0 \leq a < b \leq l$ , where  $l$  is the length of the message text.

**Type:** `Dict[Tuple[int, int], Chat]`

**property is\_mentioned:** `bool`

Returns `True` if you are mentioned in this message.

In the case where a chat (private or group) is mentioned in this message instead of a group member, you will also be considered mentioned if you are a member of the chat.

**Return type** `bool`



## 10.6.1 Contoh

### Pendahuluan: Mendefinisikan obrolan terkait

```
master: MasterChannel = coordinator.master
slave: SlaveChannel = coordinator.slave['demo.slave']
alice: PrivateChat = slave.get_chat("alice101")
bob: PrivateChat = slave.get_chat("bobrocks")
wonderland: GroupChat = slave.get_chat("thewonderlandgroup")
wonderland_alice: ChatMember = wonderland.get_member("alice101")
```

### Inisialisasi dan menandai obrolan

1. A message delivered from slave channel to master channel

```
message = Message(
    deliver_to=master,
    chat=wonderland,
    author=wonderland_alice,
    # More attributes go here...
)
```

2. A message delivered from master channel to slave channel

```
message = Message(
    deliver_to=slave,
    chat=alice,
    author=alice.self,
    # More attributes go here...
)
```

### Quoting a previous message (targeted message)

Data of the quoted message **SHOULD** be retrieved from recorded historical data. `Message.deliver_to` is not required for quoted message, and complete data is not required here. For details, see `Message.target`.

You **MAY** use the `Channel.get_message()` method to get the message object from the sending channel, but this might not always be possible depending on the implementation of the channel.

```
message.target = Message(
    chat=alice,
    author=alice.other,
    text="Hello, world.",
    type=MsgType.Text,
    uid=MessageID("100000002")
)
```

### Ubah pesan yang sebelumnya terkirim

Message ID MUST be the ID from the slave channel regardless of where the message is delivered to.

```
message.edit = True
message.uid = MessageID("100000003")
```

### Informasi berjenis khusus

#### 1. Pesan teks

```
message.type = MessageType.Text
message.text = "Hello, Wonderland."
```

#### 2. Pesan media

Informasi yang berkaitan dengan pengolahan media dijelaskan di *Media processing*.

Contoh di bawah ini adalah untuk gambar (foto) pesan. Audio, berkas, video, stiker bekerja dengan cara yang sama.

In non-text messages, the text attribute MAY be an empty string.

```
message.type = MessageType.Image
message.text = "Image caption"
message.file = NamedTemporaryFile(suffix=".png")
message.file.write(binary_data)
message.file.seek(0)
message.filename = "holiday photo.png"
message.mime = "image/png"
```

#### 3. Lokasi pesan

In non-text messages, the text attribute MAY be an empty string.

```
message.type = MessageType.Location
message.text = "I'm here! Come and find me!"
message.attributes = LocationAttribute(51.4826, -0.0077)
```

#### 4. Tautan pesan

In non-text messages, the text attribute MAY be an empty string.

```
message.type = MessageType.Link
message.text = "Check it out!"
message.attributes = LinkAttribute(
    title="Example Domain",
    description="This domain is established to be used for illustrative_
examples in documents.",
    image="https://example.com/thumbnail.png",
    url="https://example.com"
)
```

#### 5. Status

In status messages, the text attribute is disregarded.

```
message.type = MessageType.Status
message.attributes = StatusAttribute(StatusAttribute.TYPING)
```

## 6. Pesan tidak didukung

Perlengkapan teks diperlukan untuk jenis pesan ini.

```
message.type = MsgType.Unsupported
message.text = "Alice requested USD 10.00 from you. "
               "Please continue with your Bazinga App."
```

## Informasi tambahan

### 1. Pengganti

@-reference the User Themselves, another member in the same chat, and the entire chat in the message text.

```
message.text = "Hey @david, @bob, and @all. Attention!"
message.substitutions = Substitutions({
    # text[4:10] == "@david", here David is the user.
    (4, 10): wonderland.self,
    # text[12:16] == "@bob", Bob is another member of the chat.
    (12, 16): wonderland.get_member("bob"),
    # text[22:26] == "@all", this calls the entire group chat, hence the
    # chat object is set as the following value instead.
    (22, 26): wonderland
})
```

### 2. Perintah

```
message.text = "Carol sent you a friend request."
message.commands = MessageCommands([
    EFBCCommand(name="Accept", callable_name="accept_friend_request",
        kwargs={"username": "carol_jhonos", "hash": "2a9329bd93f"}
    ↵),
    EFBCCommand(name="Decline", callable_name="decline_friend_request",
        kwargs={"username": "carol_jhonos", "hash": "2a9329bd93f"})
])
```

## 10.7 Middleware

**class** ehforwarderbot.**Middleware** (*instance\_id=None*)

Middleware class.

**middleware\_id**

Unique ID of the middleware. Convention of IDs is specified in *Mengemas dan mempublikasi*. This ID will be appended with its instance ID when available.

**Type** str

**middleware\_name**

Human-readable name of the middleware.

**Type** str

**instance\_id**

The instance ID if available.

**Type** str

**\_\_init\_\_** (*instance\_id=None*)

Initialize the middleware. Inherited initializer MUST call the "super init" method at the beginning.

**Parameter** **instance\_id** (`Optional[NewType() (InstanceID, str)]`) -- Instance ID of the middleware.

**get\_extra\_functions** ()

Get a list of additional features

**Kembali** A dict of methods marked as additional features. Method can be called with `get_extra_functions() ["methodName"]()`.

**Return type** Dict[str, Callable]

**process\_message** (*message*)

Process a message with middleware

**Parameter** **message** (`Message`) -- Message object to process

**Kembali** Processed message or None if discarded.

**Return type** Optional[`Message`]

**process\_status** (*status*)

Process a status update with middleware

**Parameter** **status** (`Status`) -- Message object to process

**Kembali** Processed status or None if discarded.

**Return type** Optional[`Status`]

### 10.7.1 Tentang Middleware ID

With the introduction of instance IDs, it is required to use the `self.middleware_id` or equivalent instead of any hard-coded ID or constants while referring to the middleware ID (e.g. while retrieving the path to the configuration files, etc).

### 10.7.2 Accept commands from user through Master Channel

Despite we do not limit how the User interact with your middleware, there are 2 common ways to do it through a master channel.

#### Capture messages

If the action is chat-specific, you can capture messages with a specific pattern. Try to make the pattern easy to type but unique enough so that you don't accidentally catch messages that were meant to sent to the chat.

You may also construct a virtual chat or chat member of type "System" to give responses to the User.

## “Additional features”

If the action is not specific to any chat, but to the system as a whole, we have provided the same command line-like interface as in slave channels to middlewares as well. Details are available at *Fitur tambahan*.

### 10.7.3 Chat-specific interactions

Middlewares can have chat-specific interactions through capturing messages and reply to them with a chat member created by the middleware.

The following code is an example of a middleware that interact with the user by capturing messages.

When the master channel sends a message with a text starts with `time``, the middleware captures this message and reply with the name of the chat and current time on the server. The message captured is not delivered to any following middlewares or the slave channel.

```
def process_message(self: Middleware, message: Message) -> Optional[Message]:
    if message.deliver_to != coordinator.master and \ # sent from master channel
        text.startswith('time`'):

        # Make a system chat object.
        # For difference between `make_system_member()` and `add_system_member()`,
        # see their descriptions above.
        author = message.chat.make_system_member(
            uid="__middleware_example_time_reporter__",
            name="Time reporter",
            middleware=self
        )

        # Make a reply message
        reply = Message(
            uid=f"__middleware_example_{uuid.uuid4()}__",
            text=f"Greetings from chat {message.chat.name} on {datetime.now()}.
↳strftime('%c')}.",
            chat=chat,
            author=author, # Using the new chat we created before
            type=MsgType.Text,
            target=message, # Quoting the incoming message
            deliver_to=coordinator.master # message is to be delivered to master
        )

        # Send the message back to master channel
        coordinator.send_message(reply)

        # Capture the message to prevent it from being delivered to following
↳middlewares
        # and the slave channel.
        return None

    # Continue to deliver messages not matching the pattern above.
    return message
```

## 10.8 Status

**class** ehforwarderbot.status.**Status**

Abstract class of a status

**destination\_channel**

The channel that this status is sent to, usually the master channel.

Type *Channel*

**class** ehforwarderbot.status.**ChatUpdates** (*channel*, *new\_chats=()*, *removed\_chats=()*,  
*modified\_chats=()*)

Inform the master channel on updates of slave chats.

**channel**

Slave channel that issues the update

Type *SlaveChannel*

**new\_chats**

Unique ID of new chats

Type *Optional[Collection[str]]*

**removed\_chats**

Unique ID of removed chats

Type *Optional[Collection[str]]*

**modified\_chats**

Unique ID of modified chats

Type *Optional[Collection[str]]*

**\_\_init\_\_** (*channel*, *new\_chats=()*, *removed\_chats=()*, *modified\_chats=()*)

Parameter

- **channel** (*SlaveChannel*) -- Slave channel that issues the update
- **new\_chats** (*Optional[Collection[str]]*) -- Unique ID of new chats
- **removed\_chats** (*Optional[Collection[str]]*) -- Unique ID of removed chats
- **modified\_chats** (*Optional[Collection[str]]*) -- Unique ID of modified chats

**class** ehforwarderbot.status.**MemberUpdates** (*channel*, *chat\_id*, *new\_members=()*,  
*removed\_members=()*, *modified\_members=()*)

Inform the master channel on updates of members in a slave chat.

**channel**

Slave channel that issues the update

Type *SlaveChannel*

**chat\_id**

Unique ID of the chat.

Type *str*

**new\_members**

Unique ID of new members

Type Optional[Collection[str]]

**removed\_members**

Unique ID of removed members

Type Optional[Collection[str]]

**modified\_members**

Unique ID of modified members

Type Optional[Collection[str]]

**\_\_init\_\_** (*channel*, *chat\_id*, *new\_members*=(), *removed\_members*=(), *modified\_members*=())

#### Parameter

- **channel** (*SlaveChannel*) -- Slave channel that issues the update
- **chat\_id** (*str*) -- Unique ID of the chat.
- **new\_members** (*Optional[Collection[str]]*) -- Unique ID of new members
- **removed\_members** (*Optional[Collection[str]]*) -- Unique ID of removed members
- **modified\_members** (*Optional[Collection[str]]*) -- Unique ID of modified members

**class** ehforwarderbot.status.**MessageReactionsUpdate** (*chat*, *msg\_id*, *reactions*)

Update reacts of a message, issued from slave channel to master channel.

#### Parameter

- **chat** (*Chat*) -- The chat where message is sent
- **msg\_id** (*str*) -- ID of the message for the reacts
- **reactions** (*Mapping[NewType() (ReactionName, str), Collection[ChatMember]]*) -- Indicate reactions to the message. Dictionary key represents the reaction name, usually an emoji. Value is a collection of users who reacted to the message with that certain emoji. All Chat objects in this dict MUST be members in the chat of the message.
- **destination\_channel** (*MasterChannel*) -- Channel the status is issued to, which is always the master channel.

**\_\_init\_\_** (*chat*, *msg\_id*, *reactions*)

#### Parameter

- **chat** (*Chat*) -- The chat where message is sent
- **msg\_id** (*str*) -- ID of the message for the reacts
- **reactions** (*Mapping[NewType() (ReactionName, str), Collection[ChatMember]]*) -- Indicate reactions to the message. Dictionary key represents the reaction name, usually an emoji. Value is a collection of users who reacted to the message with that certain emoji. All Chat objects in this dict MUST be members in the chat of the message.

**class** ehforwarderbot.status.**MessageRemoval** (*source\_channel*, *destination\_channel*, *message*)

Inform a channel to remove a certain message.

This is usually known as “delete from everyone”, “delete from recipient”, “recall a message”, “unsend”, or “revoke a message” as well, depends on the IM platform.

Some channels MAY not support removal of messages, and raises a `exceptions.EFBOperationNotSupported` exception.

Feedback by sending another `MessageRemoval` back is not required when this object is sent from a master channel. Master channels SHOULD treat a successful delivery of this status as a successful removal.

**source\_channel**

Channel issued the status

Type `Channel`

**destination\_channel**

Channel the status is issued to

Type `Channel`

**message**

Message to remove. This MAY not be a complete `message.Message` object.

Type `Message`

**Raises** `.exceptions.EFBOperationNotSupported` -- When message removal is not supported in the channel.

**\_\_init\_\_** (*source\_channel, destination\_channel, message*)

Create a message removal status

Try to provided as much as you can, if not, provide a minimum information in the channel:

- Slave channel ID and chat ID (`message.chat.module_id` and `message.chat.uid`)
- Message unique ID from the slave channel (`message.uid`)

**Parameter**

- **source\_channel** (`Channel`) -- Channel issued the status
- **destination\_channel** (`Channel`) -- Channel the status is issued to
- **message** (`Message`) -- Message to remove.

**class** `ehforwarderbot.status.ReactToMessage` (*chat, msg\_id, reaction*)

Created when user react to a message, issued from master channel.

When this status is sent, a `status.MessageReactionsUpdate` is RECOMMENDED to be issued back to master channel.

**Parameter**

- **chat** (`Chat`) -- The chat where message is sent
- **msg\_id** (`str`) -- ID of the message to react to
- **reaction** (`Optional[str]`) -- The reaction name to be sent, usually an emoji. Set to `None` to remove reaction.
- **destination\_channel** (`SlaveChannel`) -- Channel the status is issued to, extracted from the chat object.

**Raises**



- **.exceptions.EFBMessageReactionNotPossible** -- Raised when the reaction is not valid (e.g. the specific reaction is not in the list of possible reactions).
- **.exceptions.EFBOperationNotSupported** -- Raised when reaction in any form is not supported to the message at all.

`__init__(chat, msg_id, reaction)`

#### Parameter

- **chat** (*Chat*) -- The chat where message is sent
- **msg\_id** (*str*) -- ID of the message to react to
- **reaction** (*Optional[NewType() (ReactionName, str)]*) -- The reaction name to be sent, usually an emoji

## 10.9 Custom Type Hints

A list of type aliases when no separate class is defined for some types of values. Types for user-facing values (display names, descriptions, message text, etc.) are not otherwise defined.

Most of types listed here are defined under the "NewType" syntax in order to clarify some ambiguous values not covered by simple type checking. This is only useful if you are using static type checking in your development. If you are not using type checking of any kind, you can simply ignore values in this module.

`ehforwarderbot.types.ChatID`

Chat ID from slave channel or middleware, applicable to both chat and chat members.

alias of `str`

`ehforwarderbot.types.ExtraCommandName`

Command name of additional features, in the format of `^[A-Za-z][A-Za-z0-9_]{0,19}$`.

alias of `str`

`ehforwarderbot.types.InstanceID`

Instance ID of a module.

alias of `str`

`ehforwarderbot.types.MessageID`

Message ID from slave channel or middleware.

alias of `str`

`ehforwarderbot.types.ModuleID`

Module ID, including instance ID after # if available.

alias of `str`

`ehforwarderbot.types.ReactionName`

Canonical representation of a reaction, usually an emoji.

alias of `str`

`ehforwarderbot.types.Reactions`

Reactions to a message.

alias of `Mapping[ReactionName, Collection[ChatMember]]`

## 10.10 Utilitas

`ehforwarderbot.utils.extra(name, desc)`

Decorator for slave channel's "additional features" interface.

**Parameter**

- **name** (`str`) -- A human readable name for the function.
- **desc** (`str`) -- A short description and usage of it. Use `{function_name}` in place of the function name in the description.

**Return type** `Callable[...Optional[str]]`

**Kembali** The decorated method.

### Contoh

```
@extra(name="Echo", desc="Return the text entered.\n\nUsage:\n    {function_name} <text>")
def echo(self, text: str) -> Optional[str]:
    return text
```

`ehforwarderbot.utils.get_base_path()`

Get the base data path for EFB. This can be defined by the environment variable `EFB_DATA_PATH`.

If `EFB_DATA_PATH` is not defined, this gives `~/ .ehforwarderbot`.

This method creates the queried path if not existing.

**Return type** `Path`

**Kembali** The base path.

`ehforwarderbot.utils.get_config_path(module_id=None, ext='yaml')`

Get path for configuration file. Defaulted to `~/ .ehforwarderbot/profiles/profile_name/module_id/config.yaml`.

This method creates the queried path if not existing. The config file will not be created, however.

**Parameter**

- **module\_id** (`Optional[NewType() (ModuleID, str)]`) -- Module ID.
- **ext** (`str`) -- Extension name of the config file. Defaulted to `"yaml"`.

**Return type** `Path`

**Kembali** The path to the configuration file.

`ehforwarderbot.utils.get_custom_modules_path()`

Get the path to custom channels

**Return type** `Path`

**Kembali** The path for custom channels.

`ehforwarderbot.utils.get_data_path(module_id)`

Get the path for permanent storage of a module.

This method creates the queried path if not existing.

**Parameter** **module\_id** (`NewType() (ModuleID, str)`) -- Module ID

**Return type** `Path`

**Kembali** The data path of indicated module.

`ehforwarderbot.utils.locate_module(module_id, module_type=None)`

Locate module by module ID

**Parameter**

- **module\_id** (`NewType() (ModuleID, str)`) -- Module ID
- **module\_type** (`Optional[str]`) -- Type of module, one of 'master', 'slave' and 'middleware'



# BAB 11

---

## Indeks dan tabel

---

- genindex
- modindex
- search



## BAB 12

---

### Merasa seperti berkontribusi?

---

Setiap orang disambut untuk mengajukan masalah atau mengajukan pull request, ingatlah untuk terus membaca dan mengikuti panduan kontribusi sebelum anda melakukannya.





## BAB 13

---

### Artikel terkait

---

- [Idea: Group Chat Tunneling \(Sync\) with EH Forwarder Bot](#)
- [What's so new in EH Forwarder Bot 2 \(and its modules\)](#)

Untuk tip, trik dan saran artikel komunitas, lihat [project wiki](#).



EFB framework is licensed under [GNU Affero General Public License 3.0](#) or later versions.

EH Forwarder Bot: An extensible message tunneling chat bot framework.  
Copyright (C) 2016 - 2020 Eana Hufwe, and the EH Forwarder Bot contributors  
All rights reserved.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU Affero General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Donate.



### e

- `ehforwarderbot.channel`, 37
- `ehforwarderbot.chat`, 42
- `ehforwarderbot.constants`, 52
- `ehforwarderbot.coordinator`, 53
- `ehforwarderbot.exceptions`, 54
- `ehforwarderbot.message`, 57
- `ehforwarderbot.status`, 66
- `ehforwarderbot.types`, 69
- `ehforwarderbot.utils`, 70



## Simbol

[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.Middleware*), 64  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.channel.Channel*), 37  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.chat.BaseChat*), 43  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.chat.Chat*), 45  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.chat.ChatMember*), 48  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.chat.SelfChatMember*), 50  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.chat.SystemChatMember*), 51  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.message.LinkAttribute*), 58  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.message.LocationAttribute*), 58  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.message.MessageCommand*), 59  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.message.MessageCommands*), 59  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.message.StatusAttribute*), 60  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.status.ChatUpdates*), 66  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.status.MemberUpdates*), 67  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.status.MessageReactionsUpdate*), 67  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.status.MessageRemoval*), 68  
[\\_\\_init\\_\\_\(\)](#) (method *ehforwarderbot.status.ReactToMessage*), 69

## A

[add\\_channel\(\)](#) (di modul *ehforwarderbot.coordinator*), 53  
[add\\_member\(\)](#) (method *ehforwarderbot.chat.Chat*), 45  
[add\\_middleware\(\)](#) (di modul *ehforwarder-*

*bot.coordinator*), 53

[add\\_self\(\)](#) (method *ehforwarderbot.chat.Chat*), 46  
[add\\_system\\_member\(\)](#) (method *ehforwarderbot.chat.Chat*), 46  
[alias](#) (atribut *ehforwarderbot.chat.BaseChat*), 42  
[alias](#) (atribut *ehforwarderbot.chat.Chat*), 44  
[ALL](#) (atribut *ehforwarderbot.chat.ChatNotificationState*), 48  
[Animation](#) (atribut *ehforwarderbot.constants.MsgType*), 52  
[args](#) (atribut *ehforwarderbot.message.MessageCommand*), 59  
[Audio](#) (atribut *ehforwarderbot.constants.MsgType*), 52

## B

[BaseChat](#) (class di *ehforwarderbot.chat*), 42

## C

[callable\\_name](#) (atribut *ehforwarderbot.message.MessageCommand*), 58  
[channel](#) (atribut *ehforwarderbot.status.ChatUpdates*), 66  
[channel](#) (atribut *ehforwarderbot.status.MemberUpdates*), 66  
[Channel](#) (class di *ehforwarderbot.channel*), 37  
[channel\\_emoji](#) (atribut *ehforwarderbot.channel.Channel*), 37  
[channel\\_emoji](#) (atribut *ehforwarderbot.chat.BaseChat*), 42  
[channel\\_emoji](#) (atribut *ehforwarderbot.chat.Chat*), 44  
[channel\\_id](#) (atribut *ehforwarderbot.channel.Channel*), 37  
[channel\\_name](#) (atribut *ehforwarderbot.channel.Channel*), 37  
[chat](#), 18  
[Chat](#) (class di *ehforwarderbot.chat*), 43  
[chat member](#), 19  
[chat\\_id](#) (atribut *ehforwarderbot.status.MemberUpdates*), 66  
[Chat ID](#) (di modul *ehforwarderbot.types*), 69

ChatMember (class di ehforwarderbot.chat), 47  
ChatNotificationState (class di ehforwarderbot.chat), 48  
ChatUpdates (class di ehforwarderbot.status), 66  
commands (atribut ehforwarderbot.message.MessageCommands), 59  
coordinator, 18  
copy() (method ehforwarderbot.chat.BaseChat), 43

## D

description (atribut ehforwarderbot.chat.BaseChat), 42  
description (atribut ehforwarderbot.chat.Chat), 44  
description (atribut ehforwarderbot.message.LinkAttribute), 57  
destination\_channel (atribut ehforwarderbot.status.MessageRemoval), 68  
destination\_channel (atribut ehforwarderbot.status.Status), 66  
display\_name (ehforwarderbot.chat.BaseChat property), 43

## E

EFBChannelNotFound, 54  
EFBChatNotFound, 54  
EFBException, 54  
EFBMessageError, 54  
EFBMessageNotFound, 54  
EFBMessageReactionNotPossible, 55  
EFBMessageTypeNotSupported, 55  
EFBOperationNotSupported, 55  
ehforwarderbot.channel  
  modul, 37  
ehforwarderbot.chat  
  modul, 42  
ehforwarderbot.constants  
  modul, 52  
ehforwarderbot.coordinator  
  modul, 53  
ehforwarderbot.exceptions  
  modul, 54  
ehforwarderbot.message  
  modul, 57  
ehforwarderbot.status  
  modul, 66  
ehforwarderbot.types  
  modul, 69  
ehforwarderbot.utils  
  modul, 70  
extra() (di modul ehforwarderbot.utils), 70  
ExtraCommandName (di modul ehforwarderbot.types), 69

## F

File (atribut ehforwarderbot.constants.MsgType), 52

## G

get\_base\_path() (di modul ehforwarderbot.utils), 70  
get\_chat() (method ehforwarderbot.channel.SlaveChannel), 39  
get\_chat\_picture() (method ehforwarderbot.channel.SlaveChannel), 39  
get\_chats() (method ehforwarderbot.channel.SlaveChannel), 40  
get\_config\_path() (di modul ehforwarderbot.utils), 70  
get\_custom\_modules\_path() (di modul ehforwarderbot.utils), 70  
get\_data\_path() (di modul ehforwarderbot.utils), 70  
get\_extra\_functions() (method ehforwarderbot.channel.SlaveChannel), 40  
get\_extra\_functions() (method ehforwarderbot.Middleware), 64  
get\_member() (method ehforwarderbot.chat.Chat), 47  
get\_message\_by\_id() (method ehforwarderbot.channel.Channel), 38  
get\_module\_by\_id() (di modul ehforwarderbot.coordinator), 53  
group chat, 19  
GroupChat (class di ehforwarderbot.chat), 48

## H

has\_self (ehforwarderbot.chat.Chat property), 47

## I

Image (atribut ehforwarderbot.constants.MsgType), 52  
image (atribut ehforwarderbot.message.LinkAttribute), 58  
instance\_id (atribut ehforwarderbot.channel.Channel), 37  
instance\_id (atribut ehforwarderbot.Middleware), 63  
InstanceID (di modul ehforwarderbot.types), 69  
is\_mentioned (ehforwarderbot.message.Substitutions property), 60

## K

kwargs (atribut ehforwarderbot.message.MessageCommand), 59

## L

latitude (atribut ehforwarderbot.message.LocationAttribute), 58  
Link (atribut ehforwarderbot.constants.MsgType), 52  
link (ehforwarderbot.message.Message property), 57  
LinkAttribute (class di ehforwarderbot.message), 57  
locate\_module() (di modul ehforwarderbot.utils), 71



Location (*atribut ehforwarderbot.constants.MsgType*),  
52  
location (*ehforwarderbot.message.Message property*),  
57  
LocationAttribute (*class di ehforwarder-  
bot.message*), 58  
long\_name (*ehforwarderbot.chat.BaseChat property*), 43  
longitude (*atribut ehforwarder-  
bot.message.LocationAttribute*), 58

## M

make\_system\_member() (*method ehforwarder-  
bot.chat.Chat*), 47  
master (*di modul ehforwarderbot.coordinator*), 53  
master channel, 18  
master\_thread (*di modul ehforwarder-  
bot.coordinator*), 53  
MasterChannel (*class di ehforwarderbot.channel*), 39  
members (*atribut ehforwarderbot.chat.Chat*), 45  
MemberUpdates (*class di ehforwarderbot.status*), 66  
MENTIONS (*atribut ehforwarder-  
bot.chat.ChatNotificationState*), 48  
message, 19  
message (*atribut ehforwarder-  
bot.status.MessageRemoval*), 68  
Message (*class di ehforwarderbot.message*), 55  
MessageAttribute (*class di ehforwarder-  
bot.message*), 57  
MessageCommand (*class di ehforwarderbot.message*), 58  
MessageCommands (*class di ehforwarderbot.message*),  
59  
MessageID (*di modul ehforwarderbot.types*), 69  
MessageReactionsUpdate (*class di ehforwarder-  
bot.status*), 67  
MessageRemoval (*class di ehforwarderbot.status*), 67  
middleware, 18  
Middleware (*class di ehforwarderbot*), 63  
middleware\_id (*atribut ehforwarderbot.Middleware*),  
63  
middleware\_name (*atribut ehforwarder-  
bot.Middleware*), 63  
middlewares (*di modul ehforwarderbot.coordinator*),  
53  
modified\_chats (*atribut ehforwarder-  
bot.status.ChatUpdates*), 66  
modified\_members (*atribut ehforwarder-  
bot.status.MemberUpdates*), 67  
modul  
ehforwarderbot.channel, 37  
ehforwarderbot.chat, 42  
ehforwarderbot.constants, 52  
ehforwarderbot.coordinator, 53  
ehforwarderbot.exceptions, 54  
ehforwarderbot.message, 57

ehforwarderbot.status, 66  
ehforwarderbot.types, 69  
ehforwarderbot.utils, 70  
module, 18  
module\_id (*atribut ehforwarderbot.chat.BaseChat*), 42  
module\_id (*atribut ehforwarderbot.chat.Chat*), 44  
module\_name (*atribut ehforwarderbot.chat.BaseChat*),  
42  
module\_name (*atribut ehforwarderbot.chat.Chat*), 44  
ModuleID (*di modul ehforwarderbot.types*), 69  
MsgType (*class di ehforwarderbot.constants*), 52  
mutex (*di modul ehforwarderbot.coordinator*), 53

## N

name (*atribut ehforwarderbot.chat.BaseChat*), 42  
name (*atribut ehforwarderbot.chat.Chat*), 44  
name (*atribut ehforwarder-  
bot.message.MessageCommand*), 58  
new\_chats (*atribut ehforwarderbot.status.ChatUpdates*),  
66  
new\_members (*atribut ehforwarder-  
bot.status.MemberUpdates*), 66  
NONE (*atribut ehforwarderbot.chat.ChatNotificationState*),  
48  
notification (*atribut ehforwarderbot.chat.Chat*), 45

## P

poll() (*method ehforwarderbot.channel.Channel*), 38  
private chat, 19  
PrivateChat (*class di ehforwarderbot.chat*), 49  
process\_message() (*method ehforwarder-  
bot.Middleware*), 64  
process\_status() (*method ehforwarder-  
bot.Middleware*), 64  
profile (*di modul ehforwarderbot.coordinator*), 53

## R

ReactionName (*di modul ehforwarderbot.types*), 69  
Reactions (*di modul ehforwarderbot.types*), 69  
ReactToMessage (*class di ehforwarderbot.status*), 68  
removed\_chats (*atribut ehforwarder-  
bot.status.ChatUpdates*), 66  
removed\_members (*atribut ehforwarder-  
bot.status.MemberUpdates*), 67  
RFC  
RFC 2119, 21, 37  
RFC 8174, 21, 37

## S

self (*atribut ehforwarderbot.chat.Chat*), 45, 47  
SELF\_ID (*atribut ehforwarderbot.chat.SelfChatMember*),  
50  
SelfChatMember (*class di ehforwarderbot.chat*), 49

`send_message()` (di modul *ehforwarder-bot.coordinator*), 54  
`send_message()` (method *ehforwarder-bot.channel.Channel*), 38  
`send_status()` (di modul *ehforwarder-bot.coordinator*), 54  
`send_status()` (method *ehforwarder-bot.channel.Channel*), 38  
`slave channel`, 18  
`slave_threads` (di modul *ehforwarder-bot.coordinator*), 54  
`SlaveChannel` (class di *ehforwarderbot.channel*), 39  
`slaves` (di modul *ehforwarderbot.coordinator*), 53, 54  
`source_channel` (atribut *ehforwarder-bot.status.MessageRemoval*), 68  
`status`, 19  
`Status` (atribut *ehforwarderbot.constants.MsgType*), 52  
`Status` (class di *ehforwarderbot.status*), 66  
`status` (*ehforwarderbot.message.Message* property), 57  
`status_type` (atribut *ehforwarder-bot.message.StatusAttribute*), 59  
`StatusAttribute` (class di *ehforwarderbot.message*), 59  
`StatusAttribute.Types` (class di *ehforwarder-bot.message*), 59  
`Sticker` (atribut *ehforwarderbot.constants.MsgType*), 52  
`stop_polling()` (method *ehforwarder-bot.channel.Channel*), 39  
`Substitutions` (class di *ehforwarderbot.message*), 60  
`suggested_reactions` (atribut *ehforwarder-bot.channel.SlaveChannel*), 39  
`supported_message_types` (atribut *ehforwarder-bot.channel.SlaveChannel*), 39  
`system chat`, 19  
`SYSTEM_ID` (atribut *ehforwarder-bot.chat.SystemChatMember*), 51  
`SystemChat` (class di *ehforwarderbot.chat*), 50  
`SystemChatMember` (class di *ehforwarderbot.chat*), 51

## T

`Text` (atribut *ehforwarderbot.constants.MsgType*), 52  
`the User`, 18  
`the User Themselves`, 18  
`timeout` (atribut *ehforwarder-bot.message.StatusAttribute*), 59  
`title` (atribut *ehforwarderbot.message.LinkAttribute*), 57  
`translator` (di modul *ehforwarderbot.coordinator*), 54  
`Types` (atribut *ehforwarderbot.message.StatusAttribute*), 59  
`TYPING` (atribut *ehforwarder-bot.message.StatusAttribute.Types*), 59

## U

`uid` (atribut *ehforwarderbot.chat.BaseChat*), 42

`uid` (atribut *ehforwarderbot.chat.Chat*), 44  
`Unsupported` (atribut *ehforwarder-bot.constants.MsgType*), 52  
`UPLOADING_FILE` (atribut *ehforwarder-bot.message.StatusAttribute.Types*), 60  
`UPLOADING_IMAGE` (atribut *ehforwarder-bot.message.StatusAttribute.Types*), 60  
`UPLOADING_VIDEO` (atribut *ehforwarder-bot.message.StatusAttribute.Types*), 60  
`UPLOADING_VOICE` (atribut *ehforwarder-bot.message.StatusAttribute.Types*), 60  
`url` (atribut *ehforwarderbot.message.LinkAttribute*), 58

## V

`vendor_specific` (atribut *ehforwarder-bot.chat.BaseChat*), 43  
`vendor_specific` (atribut *ehforwarderbot.chat.Chat*), 45  
`verify()` (method *ehforwarderbot.chat.BaseChat*), 43  
`verify()` (method *ehforwarderbot.chat.ChatMember*), 48  
`verify()` (method *ehforwarderbot.chat.GroupChat*), 49  
`verify()` (method *ehforwarderbot.chat.PrivateChat*), 49  
`verify()` (method *ehforwarderbot.chat.SystemChat*), 51  
`verify()` (method *ehforwarderbot.message.Message*), 57  
`Video` (atribut *ehforwarderbot.constants.MsgType*), 52  
`Voice` (atribut *ehforwarderbot.constants.MsgType*), 52