最終更新日：2023-02-22

Twitch EventSub Response Bot (twitch-eventsub-response-py)

Twitch で配信中にレイドを受けたときに、それに応答して自動で「 `/shoutout レイド元のユーザー名` 」Twitch公式チャットコマンドの実行や、チャット欄に指定したメッセージを表示してくれる、ボットアプリです。

百聞は一見に如かず、 **本ボットの動作例 をご覧ください** 。

背景説明

Twitch配信のチャット欄に指定したメッセージを自動で表示してくれるボットサービスには Nightbot などがあり、例えば「 `!` 」で始まる『 **ユーザーチャットコマンド（以下：ユーザーコマンド）** 』を定義し、ある程度条件を指定して実行させることができます。 Streamlabs ないし StreamElements といったサービスと組み合わせれば、他配信者からのレイドや視聴者によるフォローなどのイベントが発生すると自動で応答してユーザーコマンドを実行させることもできます。

しかし、少なくとも **Nightbot に関しては、「 `/` 」 で始まる『Twitch公式チャットコマンド（以下：公式コマンド）』のうち、以下のものしか実行させることができないようです** 。

公式コマンド	実行内容
`/me メッセージ`	「 メッセージ 」をイタリックで表示させる（日本語は非対応）
`/announce メッセージ`	「 **お知らせ** （改行） メッセージ 」を表示させる

詳しくは未調査ですが、 Streamlabs や StreamElements も公式コマンドを実行させることができないと推測しています。

本ボットの機能

上記の背景を踏まえて、 **イベントに自動で応答し、かつ、上記以外の公式コマンドを実行させることのできるボット** を開発しました。本稿更新時点でサポートしている機能は以下です。

応答タイミング	コマンド	実行内容
レイドを受けたとき	`/shoutout レイド元のユーザー名`	レイド元のユーザーのチャンネルを応援し、フォローボタン付きでチャット内で紹介する
レイドを受けたとき	（任意のメッセージ）	「 （任意のメッセージ） 」 を表示させる（これを利用して、ユーザーコマンドも実行可能）

動作環境

- .exeファイル版：たぶん、本稿更新時点でサポートされている Windows（64bit版） で動作
- スクリプト版：たぶん、 Python 3.10 以降のPythonインタプリタで動作
    - 必要な外部パッケージは `./Venvs/requirements.txt` に記載

    - ただし `twitchio` は、本稿更新時点で最新のリリース版である `V2.5.0` ではなく最新の `master` ブランチを取ってくることが必須

ダウンロード方法

- .exeファイル版：右にある Releases → 最新版の `twitch-eventsub-response-py-vX.Y.zip` ファイルをダウンロードして展開
    - `X.Y` の部分は数字
- スクリプト版：右のReleasesからソースコードをダウンロードするなり、本リポジトリーをクローンするなりして、Pythonインタプリタを使って実行

事前設定

本ボットを起動する前にやるべきことは最大で3つあります。

ボットとして運用するユーザーにモデレーター権限を付与

ボットとして運用するユーザーを決めてください。

- 配信で使っているユーザーをボットとしても運用する場合：権限を付与する必要はなし
    - すでにモデレーター以上の権限を持っているため
- ボットとして運用するユーザーを別に用意する場合：そのユーザーにモデレーターの権限を与えること
    - **セキュリティーの観点から、こちらをお勧め**

すでに チャット翻訳ちゃん などでユーザーをボットとして使用している場合は、同じユーザーを本ボットに使用しても、お互い正常に動作するようです。

ボットとして運用するユーザーのユーザーアクセストークン文字列の取得

本ボットが正常に動作するには、 チャット翻訳ちゃん などと同様に、「ユーザーアクセストークン」文字列というものをTwitchから取得して使用しなければなりません。 **トークン文字列は、ユーザーによって、そして、本ボットを含むTwitch関係の外部アプリやサービスが何を行う権限を要求するかによって、異なるものになります** 。本ボットが要求する権限は以下のとおりです。

公式コマンド・メッセージ	実行に必要な権限名	権限の意味
`/shoutout`	`moderator:manage:shoutouts`	`/shoutout` 公式コマンドを実行できる
`/color`	`user:manage:chat_color`	チャット欄で表示されるボットユーザー名の色を設定できる
（任意のメッセージ） `/me`	`chat:edit`	チャット欄に投稿できる
（対応するものなし）	`chat:read`	チャット欄に接続できる

さて、トークン文字列を取得するのに外部サービスを利用すると、そのサービスはトークン文字列を知り得てしまうので、要求して承認された権限を悪用できてしまいます。なので、ここから先は **セキュリティー意識に応じてトークン文字列の取得方法を選んでください** 。

Twitch Chat OAuth Password Generator（Twitch Chat OAuth Token Generator）ウェブサービスが、 トークン文字列を悪用しないと信じる 場合

「Twitch Chat OAuth Password Generator（Twitch Chat OAuth Token Generator）」は、 チャット翻訳ちゃん の公式ページにて、トークン文字列を取得する方法として紹介されているウェブサービスです。ただし、本ボットは チャット翻訳ちゃん とは違う権限を要求するので、同じ方法ではなく、以下の手順でトークン文字列を取得してください。

まず、ブラウザーを開いて、**ボットとして運用するユーザーでTwitchにログインした状態にしてください** 。

- ボットとして運用するユーザーを別に用意する場合は、配信で使っているユーザーでいったんログアウトしてボットとして運用するユーザーでログインし直すか、ブラウザーのシークレットモードなどと呼ばれる機能を使ってボットとして運用するユーザーでログインしてください
    - Chrome：「シークレット ウィンドウ」
    - Edge：「InPrivate ウィンドウ」
    - Firefox：「新しいプライベートウィンドウ」

そして、以下のURLをコピーし、ブラウザーにペーストして、URLにアクセスしてください。

```
https://id.twitch.tv/oauth2/authorize?
response_type=token&client_id=q6batx0epp608isickayubi39itsckt&redirect_uri=https:/
/twitchapps.com/tmi/&scope=moderator:manage:shoutouts+user:manage:chat_color+chat:
edit+chat:read
```

「Twitch Chat OAuth Token Generatorアカウントにアクセスしようとしています」というページが表示されるので、「許可」を選んでください。すると、画面が遷移し、「`oauth:9y0urb0tuser0authacceesst0ken9` 」などという文字列が表示されます。このうち `oauth:` **より右の（おそらく30桁前後となる）文字列** （この例の場合、 `9y0urb0tuser0authacceesst0ken9` ） がトークン文字列になります。

Twitch Chat OAuth Password Generator（Twitch Chat OAuth Token Generator）ウェブサービスが、トークン文字列を悪用しないと信じない場合

自前でトークン文字列を取得してください。

- 例えば Twitch APIに必要なOAuth認証のアクセストークンを取得しよう に書かれている方法
    - 「トークンを取得してみる」
        - 「1. The OAuth implicit code flow」

どちらの場合でも、トークン文字列を取得できたら、ブラウザーは閉じてかまいませんが、その前にトークン文字列を一時的にどこかにコピペなどして忘れないようにしてください。

`config.json5` ファイルへの設定の記述

まず、 config.json5 ファイルを、テキストエディタ（メモ帳など）で開いてください。 config.json5 は、ダウンロード時点では以下の内容になっています。

```
// Twitch EventSub Response Bot - Config
{
    // メッセージ送信先となるチャンネルに関する設定たち
    "messageChannel": {
        // ユーザー名(チャンネル名)
        "userName": "YourChannelName",
    },
    //
    // メッセージ送信を行うボットに関する設定たち
    "bot": {
        // ボットとして運用するユーザーのOAuthアクセストークン
        //  (* 使う機能が要求する権限をボットとなるユーザーが持っていること)
        //  (* 使う機能が要求する権限をトークンが持っていること)
        "oAuthAccessToken": "9y0urb0tuser0authacceesst0ken9",
        //
        // 名前の色たち:
        //  (* トークンが "user:manage:chat_color" 権限を持っていること)
        //  Red, Blue, Green, Firebrick, Coral, YellowGreen, OrangeRed,
        //  SeaGreen, GoldenRod, Chocolate, CadetBlue,
        //  DodgerBlue, HotPink, BlueViolet,
        //  SpringGreen,
        //  #RRGGBB (* Turboユーザーのみ),
        //  DoNotChange (* 色を変えない)
        "nameColor": "Red",
    },
    //
    // イベントに対する応答に関する設定たち
    //  コマンドやメッセージの中で置換される文字列たち:
    //      {{raidBroadcasterUserName}} -> レイド元のユーザー名(チャンネル名)
    //      (* 置換される文字列は、要望があれば追加対応するかもしれません)
    "responses": {
        // イベントたち
        //  レイド
        "raid": {
            // [
            //  順序, 送信前の待機時間(秒),
            //  コマンド名, (* 必要あれば追加情報1, 追加情報2, ...)
            // ]の組たち
            "commands": [
                // (* ボットとなるユーザーが モデレーター 以上であること)
                // (* トークンが "moderator:manage:shoutouts" 権限を持っていること)
                [2, 10, "shoutout", ],
                //
                // (* ほかのコマンドは、要望があれば追加対応するかもしれません)
            ],
            //
            // [順序, 送信前の待機時間(秒), メッセージ]の組たち
            //  (* トークンが "chat:edit" 権限を持っていること)
            "messages": [
```

```
                [1,  3, "!raided {{raidBroadcasterUserName}}"],
                //
                //  (* ほかのメッセージを追加可能です)
            ],
        },
        //
        //  (* ほかのイベントは、要望があれば追加対応するかもしれません)
    },
}
```

そして、以下の箇所を、やりたいことに応じて変更して、上書き保存してください。

箇所	変更すべき部分	何に変更するか
`"userName": "YourChannelName"`	`YourChannelName`	配信を行うユーザー名（チャンネル名）
`"oAuthAccessToken": "9y0urb0tuser0authacceesst0ken9"`	`9y0urb0tuser0authacceesst0ken9`	上記の方法で取得したトークン文字列
`"nameColor": "DoNotChange"`	`DoNotChange`	直上コメントの「名前の色たち」で候補として挙げられている色を表す文字列たちから1つ
`[2, 10, "shoutout", ],`	`2`	レイドを受けて実行される全コマンド・メッセージの中で、`/shoutout` が実行されてほしい順番
	`10`	順序が1つ前のコマンド・メッセージが実行されてから `/shoutout` が実行されるまで待機してほしい時間（秒）
	`[2, 10, "shoutout", ],`	`/shoutout` 公式コマンドを実行したくない場合は行ごと削除
`[1, 3, "!raided {{raidBroadcasterUserName}}"],`	`1`	上記と同様の順番
	`3`	上記と同様の時間

箇所	変更すべき部分	何に変更するか
	`!raided {{raidBroadcasterUserName}}`	表示したいメッセージ（これを利用して、ユーザーコマンドも実行可能）
	`[1, 3, "!raided {{raidBroadcasterUserName}}"],`	このメッセージを表示したくない場合は行ごと削除

例として `[2, 10, "shoutout", ],` および `[1, 3, "!raided {{raidBroadcasterUserName}}"],` の行を全く変更しない場合、レイドを受けたときに本ボットは以下の動作をします。

- まず、3秒待機したのち、チャット欄に「 `!raided レイド元のユーザー名(チャンネル名)` 」というメッセージを表示
    - もし Twitchでレイドされたときに自動でお礼と宣伝をする方法 のとおりに Nightbot に `!raided` ユーザーコマンド表示したいメッセージを設定していた場合、チャット欄に「`【表示名】さんレイドありがとうございます！【表示名】さん(【ゲーム名】をプレイ中)のチャンネルはコチラ→【URL】`」と表示
        - 本ボットと Nightbot を設定すれば、 **Streamlabs ないし StreamElements といったサービス側の設定は不要**
- 次に、10秒待機したのち、 `/shoutout レイド元のユーザー名` 公式コマンドを実行

本ボットと チャット翻訳ちゃん を同時に使用する場合は、チャット翻訳ちゃん側で名前の色を指定し、こちらは `"nameColor": "DoNotChange"` のままにしてください。

`config.json5` の文字コードは、ダウンロード時点では `UTF-8（BOMなし）` ですが、上書き保存した際にほかの文字コードに変わってしまっても、問題なく動作するように作ったつもりです。

実行

さあ、本ボットを使いましょう！

なお、本ボットの起動や動作中であるかの確認が失敗する場合で、原因が上記で取得したトークン文字列であると推測される場合は、 ボットとして運用するユーザーでTwitchにログインし、「設定（アカウント設定）」の「 リンク 」の「その他のリンク」から、トークン文字列取得に使用したサービスをいったん「リンク解除」し、もう一度上記の方法でトークン文字列を取得すると、解決するかもしれません。

- 「Twitch Chat OAuth Password Generator（Twitch Chat OAuth Token Generator）ウェブサービスが、トークン文字列を悪用しないと信じる場合」を選択した場合、リンク解除するサービス名は「Twitch Chat OAuth Token Generator」

起動

- .exeファイル版： `twitch-eventsub-response-py.exe` を実行
- スクリプト版： Pythonで `./Code/main.py` を実行

.exeファイル版は、Windowsやセキュリティーソフトによりウイルスの疑いありと判定され、初回の起動が妨げられる可能性があります。その場合は、（もちろん本ボットはウイルスではないので）疑いを解除して

起動できるようにしてください。

- .exeファイルはスクリプト版に PyInstaller を適用して生成しているが、 PyInstaller を使用して生成した.exeファイルにはよくある現象

本ボットはネット通信を行うアプリであるため、初回起動時にファイアーウォールソフトが通信をブロックしようとする可能性があります。その場合は、 **通信を許可してください** 。

正常に起動すると、配信のチャット欄に「 YourBotUserName *yourbotusername bot for <ter>_ has joined.* 」と表示されます。

また、コンソール（黒い画面）に以下のようなメッセージが表示されます。

```
--- Twitch EventSub Response Bot (v0.1) ---

[Preprocess]
  JSON5 file path = C:\Users\youru\Desktop\twitch-eventsub-response-py-
vX.Y\config.json5
    parsing this file ... done.

[Activation of Bot]
  Initializing bot ...
    Bot token length = 99
    Test command = <ter>_test
    Message channel user name = YourChannelName
  done.

[Running of Bot]
  Joining channel ...
    Channel name = yourchannelname
    Bot name color = Red
  done.

  Making bot ready ...
    Bot user ID = 888888888
    Bot user name = yourbotusername
  done.

  Registering Cogs ...
    Cogs = [TERRaidCog]
  done.
```

本ボットが動作中であるかの確認

配信のチャット欄に「<ter>_test」と打つと、「YourBotUserName *yourbotusername bot for <ter>_ is alive.* 」と表示されます。

また、コンソール（黒い画面）に以下のようなメッセージが表示されます。

```
Testing bot ...
  Channel name = yourchannelname
  Bot user ID = 888888888
  Bot user name = yourbotusername
  Cogs = [TERRaidCog]
done.
```

この2つが表示されれば、本ボットは動作中です。

停止

- .exeファイル版：コンソール（黒い画面）を閉鎖
- スクリプト版：実行中の `./Code/main.py` スクリプトを停止

機能の制限

- 1分間の間に複数のレイドを受けた場合、最初のレイドに対してのみ `/shoutout` 公式コマンドが動作
    - Twitchの仕様で、1度 `/shoutout` を行った後、1分たたないと次の `/shoutout` ができないため

今後の展開

要望に応じて、本ボットで対応する公式コマンドやメッセージを増やしていければと思います。対応コマンドやメッセージが増えれば、例えば以下のようなことができるようになると予想します。

- 一定額以上のビッツをくれたユーザーに対して自動でVIP権限を付与
- アンフォローしたユーザーを自動でバン（笑）
- Nightbot を設定せずに本ボット単体で、 `!raided` ユーザーコマンドで表示したいメッセージを設定

また、セキュリティーの懸念が小さいような、ユーザーアクセストークン文字列を取得するアプリを、本ボットに付属させることができればよいと考えています。もしご協力いただける方がいればたいへんありがたいです。

バージョン履歴

2023/02/22：v0.2

- ボットとして運用するユーザーとして、配信で使っているユーザー以外も指定可能に
- 1分間の間に複数のレイドを受けた場合、ボットがエラー終了しないように

2023/02/22：v0.1（最初のリリース）

参考資料

- TwitchIO（documentation、GitHub）
- TwitchIOの実装例
- TwitchIOでTwitchのBotを作る
- Twitch APIに必要なOAuth認証のアクセストークンを取得しよう
    - 「トークンを取得してみる」
        - 「1. The OAuth implicit code flow」

- Twitchでレイドされたときに自動でお礼と宣伝をする方法
- チャット翻訳ちゃん

本ボット作者用の備忘録

本ボットの作者は忘れっぽいので、以下は自分用の備忘録です。

.exeファイルの生成方法

`Pyinstaller` フォルダーをカレントディレクトリにして、 `pyinstaller --clean --onefile --name twitch-eventsub-response-py ../Codes/main.py` をする。