QQ聊天机器人开发文档

**目录**

[QQ聊天机器人开发文档 1](#_Toc66118893)

[一、版本信息： 3](#_Toc66118894)

[二、开发环境及架构： 4](#_Toc66118895)

[1、程序架构： 5](#_Toc66118896)

[三、CQ HTTP API（已失效）： 5](#_Toc66118897)

[1、插件使用方法： 5](#_Toc66118898)

[2、插件配置： 5](#_Toc66118899)

[3、插件配置代码： 9](#_Toc66118900)

[4、插件API调用： 9](#_Toc66118901)

[（1）API请求方式： 9](#_Toc66118902)

[（2）API响应： 10](#_Toc66118903)

[（3）异步调用： 11](#_Toc66118904)

[（4）限速调用： 11](#_Toc66118905)

[（5）API列表： 11](#_Toc66118906)

[/send\_private\_msg 发送私聊消息 11](#_Toc66118907)

[/send\_group\_msg 发送群消息 12](#_Toc66118908)

[/send\_discuss\_msg 发送讨论组消息 12](#_Toc66118909)

[/send\_msg 发送消息 12](#_Toc66118910)

[/delete\_msg 撤回消息 13](#_Toc66118911)

[/set\_group\_ban 群组单人禁言 13](#_Toc66118912)

[/set\_group\_whole\_ban 群组全员禁言 14](#_Toc66118913)

[/set\_friend\_add\_request 处理加好友请求 14](#_Toc66118914)

[/set\_group\_add\_request 处理加群请求／邀请 14](#_Toc66118915)

[/get\_login\_info 获取登录号信息 15](#_Toc66118916)

[/get\_friend\_list 获取好友列表 15](#_Toc66118917)

[/get\_group\_list 获取群列表 15](#_Toc66118918)

[/get\_cookies 获取 Cookies 16](#_Toc66118919)

[/get\_csrf\_token 获取 CSRF Token 16](#_Toc66118920)

[/get\_credentials 获取 QQ 相关接口凭证 16](#_Toc66118921)

[/get\_status 获取插件运行状态 17](#_Toc66118922)

[/get\_version\_info 获取酷Q及CQHTTP插件的版本信息 18](#_Toc66118923)

[/set\_restart\_plugin 重启 CQHTTP 插件 18](#_Toc66118924)

[/clean\_data\_dir 清理数据目录 18](#_Toc66118925)

[/clean\_plugin\_log 清理插件日志 19](#_Toc66118926)

[四、Warframe API： 19](#_Toc66118927)

[1、API调用地址： 19](#_Toc66118928)

[2、内建接口： 19](#_Toc66118929)

[warframe\_info.py 20](#_Toc66118930)

[（1）get\_alerts() 20](#_Toc66118931)

[（2）get\_arbitration() 20](#_Toc66118932)

[（3）get\_all\_cycle() 21](#_Toc66118933)

[（4）get\_earth\_cycle() 21](#_Toc66118934)

[（5）get\_cetus\_cycle() 21](#_Toc66118935)

[（6）get\_orb\_cycle() 22](#_Toc66118936)

[（7）get\_cetus\_mission() 22](#_Toc66118937)

[（8）get\_conclave\_challenge() （不再开发） 23](#_Toc66118938)

[（9）get\_darvo\_daily\_deal() 24](#_Toc66118939)

[（10）get\_orb\_mission() 24](#_Toc66118940)

[（11）get\_events() 25](#_Toc66118941)

[（12）get\_fissures() 26](#_Toc66118942)

[（13）get\_flash\_sales() （不再开发） 27](#_Toc66118943)

[（14）get\_global\_upgrades() （未开发） 27](#_Toc66118944)

[（15）get\_invasions() 28](#_Toc66118945)

[（16）get\_construction\_progress() 29](#_Toc66118946)

[（17）get\_kuva() （不稳定） 29](#_Toc66118947)

[（18）get\_news() 30](#_Toc66118948)

[（19）get\_nightwave() 31](#_Toc66118949)

[（20）get\_persistent\_enemy() （未开发） 32](#_Toc66118950)

[（21）get\_riven\_statistic() （不再开发） 32](#_Toc66118951)

[（22）get\_sentient\_outpost() 33](#_Toc66118952)

[（23）get\_sanctuary() （不再开发） 34](#_Toc66118953)

[（24）get\_sortie() 34](#_Toc66118954)

[（25）get\_syndicate\_mission() 35](#_Toc66118955)

[（26）get\_void\_trader() 35](#_Toc66118956)

[（27）get\_arcane\_enhancement() （不再开发） 36](#_Toc66118957)

[（28）get\_rest\_time(expiry) 37](#_Toc66118958)

[五、Warframe Market API： 37](#_Toc66118959)

[1、API调用地址： 37](#_Toc66118960)

[2、内建接口： 37](#_Toc66118961)

[warframe\_market.py 37](#_Toc66118962)

[（1）get\_market(message\_from\_user) 37](#_Toc66118963)

[六、小栗子HTTP API： 38](#_Toc66118964)

[七、数据库结构设计： 38](#_Toc66118965)

[1、数据库查询： 39](#_Toc66118966)

[2、数据库添加： 39](#_Toc66118967)

[3、数据库删除：（未验证） 40](#_Toc66118968)

[4、数据库更改：（未验证） 40](#_Toc66118969)

[5、数据库架构设计： 40](#_Toc66118970)

[（1）设计目标： 40](#_Toc66118971)

[（2）数据分布策略： 40](#_Toc66118972)

[（3）分布式数据目标的实现： 40](#_Toc66118973)

[八、管理员控制： 41](#_Toc66118974)

[九、腾讯云腾讯智能对话平台TBP： 41](#_Toc66118975)

[1、通信协议及密钥处理： 42](#_Toc66118976)

[2、实现代码： 42](#_Toc66118977)

[十、腾讯AI开放平台智能闲聊NLP： 42](#_Toc66118978)

[1、通信协议及密钥处理： 42](#_Toc66118979)

[2、实现代码： 43](#_Toc66118980)

[十一、使用Git上传代码至仓库： 44](#_Toc66118981)

[1、提交commit： 44](#_Toc66118982)

[2、推送到远程仓库： 44](#_Toc66118983)

[3、强制推送到仓库： 45](#_Toc66118984)

[4、添加版本标签： 45](#_Toc66118985)

[5、发布版本标签： 45](#_Toc66118986)

[十二、网页爬虫： 45](#_Toc66118987)

[1、必应搜索背景图片抓取： 45](#_Toc66118988)

[十三、其它工具： 46](#_Toc66118989)

[1、文字转图片工具： 46](#_Toc66118990)

# 一、版本信息：

**该程序由开发者（QQ：465152177）开发提供，未经允许禁止传播本文档，程序代码仅供学习使用，禁止传播、抄袭、发布。**

**开发者Github地址：**[**https://github.com/Effend-Wang**](https://github.com/Effend-Wang)

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 发布日期 | 改动信息 | 额外信息 |
| v0.1 | 2020.7.26 | / | 初版机器人，具备部分warframe游戏数据的查询功能 |
| v0.2 | 2020.8.9 | 改动部分warframe游戏数据查询输出信息 | 新增warframe游戏数据的查询功能、warframe market物品交易查询功能、warframe游戏内容翻译功能 |
| v0.3 | 2020.8.12 | / | 优化程序结构 |
| v0.4 | 2020.8.16 | 改动程序整体框架；引入数据库存储与操作功能；添加管理员操作数据库功能 | / |
| v0.5 | 2020.8.30 | 改动数据库读取结构，采取数据库加载形式进行读取，并通过函数接口获取数据，避免反复操作数据库导致的问题，并细化数据库 | 提供了管理员数据库加载功能 |
| v0.6 | 2020.10.06 | / | 新增腾讯AI聊天功能，包括两种不同平台来源的AI聊天，腾讯AI开放平台智能闲聊NLP及腾讯云智能对话平台TBP |
| v0.7 | 2021.02.01 | 改动部分框架；细化消息内容以提高代码的可扩展性；细分出获取回复消息模块；删除从文件读取机器人ID的方式，改用从监听到的消息中提取 | 新增必应网页背景图片的爬虫抓取，新增文字转图片工具模块 |

**已提供和未来将提供的功能有：**

（1）Warframe国际服：

1、查询游戏实时信息；

2、查询Warframe Market实时交易信息；

3、游戏内容翻译；

4、查询Warframe Wiki内容（暂未开发）；

（2）崩坏三国服（暂未开发）：

（3）机器人管理员：

（4）腾讯云服务：

1、腾讯智能对话平台TBP；

2、腾讯AI开放平台智能闲聊NLP；

（5）阿里云服务：

1、阿里智能语音；（暂未开发）

# 二、开发环境及架构：

该机器人使用酷Q软件的插件CQHTTP为信息传输平台，通过CQHTTP API插件完成数据的发送及接收，使用Python脚本完成数据处理。

CQHTTP插件的Github链接：<https://github.com/richardchien/coolq-http-api/releases>

CQHTTP插件文档链接：<https://richardchien.gitee.io/coolq-http-api/docs/4.15/#/>

Python开发版本：3.8.6

Python使用的第三方库：flask、requests、json、pytz、re、time、datetime、pypyodbc、tencentcloud-sdk-python、urllib、bs4、os、PIL、textwrap、random

**注意：**酷Q已经停止提供服务，无法提供未来之更新等支持。已改用小栗子框架。

**程序打包：**

pyinstaller -F .\qq\_robot.py -i .\image\icon\_32X32.ico

## 1、程序架构：

程序采取分布式架构进行编写，通过各个模块将程序功能相互连接。

（1）监听模块：qq\_robot.py 🡪 def server():

监听到消息data后，在模块中将消息内容传递给message\_operate 🡪 get\_message(data)进行消息处理；

（2）消息处理模块：message\_operate 🡪 get\_message(data):

在消息处理模块中，将获得的消息数据整理到字典变量message中。获得消息类型并区分信息内容，然后将处理后得到的message传回。

（3）创建回复数据模块：response 🡪 create(message):

在该模块中，将发送的消息内容初始化为字典变量res\_msg，然后获得消息鉴权，通过判断权限再获得消息回复的内容，将回复的内容存储在字典变量res\_msg中，最后返回该字典变量为回复数据。

（4）发送数据模块：send\_message 🡪 send(res\_msg):

当res\_msg中的have\_text、have\_pic、have\_audio均为False时，则直接返回空值结束当前会话，不回复消息。在发送消息中，首先判断消息类型为私密消息或群消息，然后判断具体消息类型并从res\_msg中获取相关数据组成数据包data，再经过API发送数据包。

# 三、CQ HTTP API（已失效）：

## 1、插件使用方法：

启用后插件将开启一个HTTP服务器来接收请求，监听服务器某端口，首次启用会生成一个默认配置文件，在酷Q的data\app\io.github.richardchien.coolqhttpapi\config文件夹中，文件名为 <user\_id>.json（<user\_id>为登录的QQ号），使用JSON格式填写。此时通过http://<host>:<port>/即可调用酷Q的函数。

## 2、插件配置：

|  |  |  |
| --- | --- | --- |
| **配置项名称** | **默认值** | **说明** |
| host | [::] | HTTP 服务器监听的 IP |
| port | 5700 | HTTP 服务器监听的端口 |
| use\_http | true | 是否开启HTTP接口，即通过HTTP调用API |
| ws\_host | [::] | WebSocket服务器监听的 IP |
| ws\_port | 6700 | WebSocket服务器监听的端口 |
| use\_ws | false | 是否开启WebSocket服务器，可用于调用 API 和推送事件 |
| ws\_reverse\_url | 空 | 反向WebSocket Event和事件上报的共用地址 |
| ws\_reverse\_api\_url | 空 | 反向 WebSocket API 地址，如果为空，则使用ws\_reverse\_url指定的值 |
| ws\_reverse\_event\_url | 空 | 反向 WebSocket 事件上报地址，如果为空，则使用 ws\_reverse\_url 指定的值 |
| ws\_reverse\_reconnect\_interval | 3000 | 反向 WebSocket 客户端断线重连间隔，单位毫秒 |
| ws\_reverse\_reconnect\_on\_code\_1000 | true | 是否在关闭状态码为1000的时候重连 |
| ws\_reverse\_use\_universal\_client | false | 是否使用Universal客户端 |
| use\_ws\_reverse | false | 是否使用反向WebSocket服务，即插件作为WebSocket客户端主动连接指定的API和事件上报地址 |
| post\_url | 空 | 消息和事件的上报地址，通过POST方式请求，数据以JSON格式发送 |
| post\_timeout | 0 | HTTP上报（即访问post\_url）的超时时间，单位秒，0表示不设置超时 |
| access\_token | 空 | API访问token，如果不为空，则会在接收到请求时验证Authorization请求头是否为Bearer xxxxxxxx，xxxxxxxx为access token |
| secret | 空 | 上报数据签名密钥，如果不为空，则会在 HTTP 上报时对 HTTP 正文进行 HMAC SHA1 哈希，使用 secret 的值作为密钥，计算出的哈希值放在上报的 X-Signature 请求头，例如 X-Signature: sha1=f9ddd4863ace61e64f462d41ca311e3d2c1176e2 |
| post\_message\_format | string | 上报消息格式，string 为字符串格式，array 为数组格式 |
| serve\_data\_files | false | 是否提供请求 data 目录的文件的功能 |
| update\_source | v4.12.3 之前 github v4.12.3 及之后 china | 更新源，可选 global/github 和 china/coding |
| update\_channel | stable | 更新通道，目前有 stable、beta、alpha 三个 |
| auto\_check\_update | false | 是否自动检查更新（每次启用插件时检查），不启用的情况下，仍然可以在 酷Q 应用菜单中手动检查更新 |
| auto\_perform\_update | false | 是否自动执行更新，仅在 auto\_check\_update 启用时有效，若启用，则插件将在自动检查到更新后，自动下载新版本（需要手动重启 酷Q 以生效） |
| thread\_pool\_size | 4 | 工作线程池大小，用于异步发送消息和一些其它小的异步任务，应根据计算机性能和实际需求适当调节，若设为 0，则使用 CPU 核心数 \* 2 + 1 |
| server\_thread\_pool\_size | 4 | API 服务器线程池大小，用于异步处理请求，应根据计算机性能和实际需求适当调节，若设为 0，则使用 CPU 核心数 \* 2 + 1 |
| convert\_unicode\_emoji | true | 是否在 CQ:emoji 和实际的 Unicode 之间进行转换，转换可能耗更多时间，但日常情况下影响不大，如果你的机器人需要处理非常大段的消息（上千字），且对性能有要求，可以考虑关闭转换 |
| event\_filter | 空 | 指定事件过滤规则文件，留空将不开启事件过滤器 |
| enable\_backward\_compatibility | false | 是否启用旧版兼容性，启用时事件上报的数据将和 3.x 版本保持兼容 |
| show\_log\_console | true | 是否显示日志输出控制台 |
| max\_log\_file\_size | 6291456 | 最大单日志文件大小，单位字节，默认 6 MB |
| max\_log\_files | 1 | 最大日志文件备份数量（采用日志轮替机制） |
| log\_level | info | 日志文件和日志控制台的日志等级，可选 debug、info、warning、error、fatal |
| use\_extension | false | 是否启用扩展机制 |
| disable\_coolq\_log | true | 是否禁用酷Q原生日志，由于使用 酷Q 原生日志可能会导致快速重启时插件卡死，所以默认禁用，如果你不在乎重启时卡死，并且需要在酷Q原生日志窗口查看插件的日志，可以将此项设为false |
| online\_status\_detection\_method | get\_stranger\_info | QQ在线状态检测方式，默认（get\_stranger\_info）通过陌生人查询接口判断，设为log\_db可切换成从 酷Q的日志数据库判断 |
| enable\_heartbeat | false | 是否启用心跳机制，启用时会产生类型为heartbeat的元事件 |
| heartbeat\_interval | 15000 | 产生心跳元事件的时间间隔，单位毫秒 |
| enable\_rate\_limited\_actions | false | 是否启用限速API调用的支持 |
| rate\_limit\_interval | 500 | 限速API调用的排队间隔时间，单位毫秒 |

## 3、插件配置代码：

酷Q路径下：data\app\io.github.richardchien.coolqhttpapi\config\<user\_id>.json



其中，host使用本地回传地址127.0.0.1，端口号8080，数据post URL地址<http://127.0.0.1:5701/api/message>，数据post格式为array数组，设置响应延时500ms，日志记录等级为warning。

## 4、插件API调用：

### （1）API请求方式：

所有API都同时支持GET和POST两种请求方式（除获取data目录中的文件），参数可通过URL参数、表单或JSON传入，后两者分别对应Content-Type application/x-www-form-urlencoded和application/json。如果使用JSON传入，参数要放在根级别的JSON对象中，且参数的数据类型必须符合API列表中给出的每个参数所要求的数据类型。为避免可能的问题，建议使用JSON方式传入数据。

如果配置文件中填写了access\_token，则需要在每次请求时在请求头中加入验证头。如：

GET /send\_private\_msg?access\_token=<access\_token>&user\_id=123456&message=hello HTTP/1.1

**数据包示例：**

data={"user\_id": 123456, "message": "hello"}

以JSON格式，加入API列表中的参数及对应数据。

### （2）API响应：

对于所有请求：

1. 如果access token未提供，状态码为401；
2. 如果access token不符合，状态码为403；
3. 如果POST请求的Content-Type不支持，状态码为406；
4. 如果POST请求的正文格式不正确，状态码为400；
5. 如果API不存在，状态码为404；
6. 其余情况下，状态码均为200；

响应内容为JSON格式，结构如下：

{

"status": "ok",

"retcode": 0,

"data": {

"id": 123456,

"nickname": "滑稽"

}

}

响应返回码retcode说明如下表：

|  |  |
| --- | --- |
| **retcode** | **说明** |
| 0 | 同时status为ok，表示操作成功 |
| 1 | 同时status为async，表示操作已进入异步执行，具体结果未知 |
| 100 | 参数缺失或参数无效，通常是因为没有传入必要参数，某些接口中也可能因为参数明显无效（比如传入的QQ号小于等于0，此时无需调用酷Q函数即可确定失败），此项和以下的status均为failed |
| 102 | 酷Q函数返回的数据无效，一般是因为传入参数有效但没有权限，比如试图获取没有加入的群组的成员列表 |
| 103 | 操作失败，一般是因为用户权限不足，或文件系统异常、不符合预期 |
| 104 | 由于酷Q提供的凭证（Cookie和CSRF Token）失效导致请求QQ相关接口失败，可尝试清除酷Q缓存来解决 |
| 201 | 工作线程池未正确初始化（无法执行异步任务） |

### （3）异步调用：

所有API都可以通过附加后缀 \_async 进行异步调用。当进行异步调用时，以 get\_ 开头的接口无法获得调用结果，不存在调用意义；某些接口本身即为异步调用（status=async）时，使用后缀不会出现区别。

### （4）限速调用：

设置配置项 enable\_rate\_limited\_actions 为true可开启限速调用（默认为false）。

API接口可通过附加后缀 \_rate\_limited 来进行限速调用，主要用于发送接口，以避免消息发送频率过快。

所有限速调用将以指定速度排队执行，速度由配置项rate\_limit\_interval控制，单位：毫秒。

限速调用时，status=async。

### （5）API列表：

#### /send\_private\_msg 发送私聊消息

**参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **默认值** | **说明** |
| user\_id | number | - | 对方QQ号 |
| message | message | - | 要发送的内容 |
| auto\_escape | boolean | false | 消息内容是否作为纯文本发送（即不解析CQ码），只在message字段是字符串时有效 |

**响应数据：**

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| message\_id | number (int32) | 消息ID |

#### /send\_group\_msg 发送群消息

**参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **默认值** | **说明** |
| group\_id | number | - | 群号 |
| message | message | - | 要发送的内容 |
| auto\_escape | boolean | false | 消息内容是否作为纯文本发送（即不解析CQ码），只在message字段是字符串时有效 |

**响应数据：**

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| message\_id | number (int32) | 消息ID |

#### /send\_discuss\_msg 发送讨论组消息

**参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **默认值** | **说明** |
| discuss\_id | number | - | 讨论组ID（正常情况下看不到，需要从讨论组消息上报的数据中获得） |
| message | message | - | 要发送的内容 |
| auto\_escape | boolean | false | 消息内容是否作为纯文本发送（即不解析CQ码），只在message字段是字符串时有效 |

**响应数据：**

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| message\_id | number (int32) | 消息ID |

#### /send\_msg 发送消息

**参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **默认值** | **说明** |
| message\_type | string | - | 消息类型，支持private、group、discuss，分别对应私聊、群组、讨论组，如不传入，则根据传入的\*\_id参数判断 |
| user\_id | number | - | 对方QQ号（消息类型为private时需要） |
| group\_id | number | - | 群号（消息类型为group时需要） |
| discuss\_id | number | - | 讨论组ID（消息类型为discuss时需要） |
| message | message | - | 要发送的内容 |
| auto\_escape | boolean | false | 消息内容是否作为纯文本发送（即不解析CQ码），只在message字段是字符串时有效 |

**响应数据：**

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| message\_id | number (int32) | 消息ID |

#### /delete\_msg 撤回消息

**参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **默认值** | **说明** |
| message\_id | number (int32) | - | 消息 ID |

**响应数据：**

无

#### /set\_group\_ban 群组单人禁言

**参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **默认值** | **说明** |
| group\_id | number | - | 群号 |
| user\_id | number | - | 要禁言的QQ号 |
| duration | number | 30\*60 | 禁言时长，单位秒，0表示取消禁言 |

**响应数据：**

无

#### /set\_group\_whole\_ban 群组全员禁言

**参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **默认值** | **说明** |
| group\_id | number | - | 群号 |
| enable | boolean | true | 是否禁言 |

**响应数据：**

无

#### /set\_friend\_add\_request 处理加好友请求

**参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **默认值** | **说明** |
| flag | string | - | 加好友请求的flag（需从上报的数据中获得） |
| approve | boolean | true | 是否同意请求 |
| remark | string | 空 | 添加后的好友备注（仅在同意时有效） |

**响应数据：**

无

#### /set\_group\_add\_request 处理加群请求／邀请

**参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **默认值** | **说明** |
| flag | string | - | 加群请求的flag（需从上报的数据中获得） |
| sub\_type或type | string | - | add或invite，请求类型（需要和上报消息中的sub\_type字段相符） |
| approve | boolean | true | 是否同意请求／邀请 |
| reason | string | 空 | 拒绝理由（仅在拒绝时有效） |

**响应数据：**

无

#### /get\_login\_info 获取登录号信息

**参数：**

无

**响应数据：**

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| user\_id | number (int64) | QQ号 |
| nickname | string | QQ昵称 |

#### /get\_friend\_list 获取好友列表

**参数：**

无

**响应数据：**

响应内容为JSON数组，每个元素如下：

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| user\_id | number (int64) | QQ号 |
| nickname | string | 昵称 |
| remark | string | 备注名 |

#### /get\_group\_list 获取群列表

**参数：**

无

**响应数据：**

响应内容为JSON数组，每个元素如下：

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| group\_id | number (int64) | 群号 |
| group\_name | string | 群名称 |

#### /get\_cookies 获取 Cookies

**参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **默认值** | **说明** |
| domain | string | 空 | 需要获取cookies的域名 |

**响应数据：**

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| cookies | string | Cookies |

#### /get\_csrf\_token 获取 CSRF Token

**参数：**

无

**响应数据：**

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| token | number (int32) | CSRF Token |

#### /get\_credentials 获取 QQ 相关接口凭证

即上面两个接口的合并。

**参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **默认值** | **说明** |
| domain | string | 空 | 需要获取cookies的域名 |

**响应数据：**

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| cookies | string | Cookies |
| csrf\_token | number (int32) | CSRF Token |

#### /get\_status 获取插件运行状态

**参数：**

无

**响应数据：**

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| app\_initialized | boolean | CQHTTP插件已初始化 |
| app\_enabled | boolean | CQHTTP插件已启用 |
| plugins\_good | object | CQHTTP的各内部插件是否正常运行 |
| app\_good | boolean | CQHTTP插件正常运行（已初始化、已启用、各内部插件正常运行） |
| online | boolean | 当前QQ在线，null表示无法查询到在线状态 |
| good | boolean | CQHTTP插件状态符合预期，意味着插件已初始化，内部插件都在正常运行，且QQ在线 |

通常情况下建议只使用online和good这两个字段来判断运行状态，因为随着插件的更新，其它字段有可能频繁变化。

其中，online字段的在线状态检测有两种方式，可通过online\_status\_detection\_method配置项切换，默认通过读取酷Q日志数据库实现，可切换为get\_stranger\_info以通过测试陌生人查询接口的可用性来检测。具体区别如下：

|  |  |  |
| --- | --- | --- |
| **在线检测方式** | **优点** | **缺点** |
| get\_stranger\_info（默认） | 正常情况下比log\_db准确，但请求频率过高时可能变得不准确（在线被认为不在线）；需要发送网络请求 | （几乎不可能）会因为酷Q更新而失效 |
| log\_db | 查询速度较快；无需网络请求（不会触发腾讯风控）；不会因为请求频率过高而不准确 | 可能因为酷Q修改数据库表名、文件名而失效；月尾掉线，月初无法检测到 |

#### /get\_version\_info 获取酷Q及CQHTTP插件的版本信息

**参数：**

无

**响应数据：**

|  |  |  |
| --- | --- | --- |
| **字段名** | **数据类型** | **说明** |
| coolq\_directory | string | 酷Q根目录路径 |
| coolq\_edition | string | 酷Q版本，air或pro |
| plugin\_version | string | CQHTTP插件版本，例如2.1.3 |
| plugin\_build\_number | number | CQHTTP 插件build号 |
| plugin\_build\_configuration | string | CQHTTP插件编译配置，debug或release |

#### /set\_restart\_plugin 重启 CQHTTP 插件

由于重启插件同时需要重启API服务，这意味着当前的API请求会被中断，因此需在异步地重启插件，接口返回的status是async。

**参数：**

|  |  |  |  |
| --- | --- | --- | --- |
| **字段名** | **数据类型** | **默认值** | **说明** |
| delay | number | 0 | 要延迟的毫秒数，如果默认情况下无法重启，可以尝试设置延迟为2000左右 |

**响应数据：**

无

#### /clean\_data\_dir 清理数据目录

用于清理积攒了太多旧文件的数据目录，如image。

**参数：**

| **字段名** | **数据类型** | **默认值** | **说明** |
| --- | --- | --- | --- |
| data\_dir | string | - | 收到清理的目录名，支持image、record、show、bface |

**响应数据：**

无

#### /clean\_plugin\_log 清理插件日志

用于清空插件的日志文件。

**参数：**

无

**响应数据：**

无

# 四、Warframe API：

## 1、API调用地址：

Warframe API链接：<https://api.warframe.us/>

Warframe API Doc链接：<https://docs.warframestat.us/>

API链接末尾可附加平台，包括pc、swi、ps4、xb1。

Warframe相关开发Github项目查看：

Warframe Community Developers：<https://github.com/WFCD>

请求头参数设置仅支持接受语言。请求头格式示例：



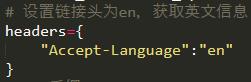
响应数据均为JSON格式。通过from json import loads，使用data=loads(data)将响应数据转换为dict类型。

## 2、内建接口：

相关代码文件均放于warframe文件夹下。

### warframe\_info.py

该文件代码用于通过Warframe API查询游戏内容，其中，设定链接请求头为：



获取英文内容，以避免翻译问题导致的错误。

下列函数返回的数据均为已经过处理的数据，返回的内容为要回复的消息。格式均为string。

#### （1）get\_alerts()

获取警报内容。请求链接为：

<https://api.warframstat.us/pc/alerts>

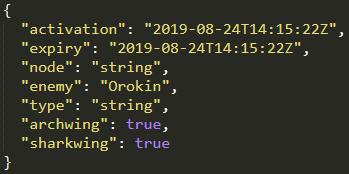
由于游戏不再提供任何警报任务，因此该请求链接回复为空数组，即[]。

#### （2）get\_arbitration()

获取仲裁内容。请求链接为：

<https://api.warframstat.us/pc/arbitration>

**数据示例：**



**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| activation | string | 开始时间 |
| expiry | string | 结束时间 |
| node | string | 节点 |
| enemy | string | 敌人派系 |
| type | string | 任务类型 |
| archwing | bollean | 是否需要archwing |
| sharkwing | bollean | 是否需要sharkwing |

#### （3）get\_all\_cycle()

获取所有的循环状态，该模块需要获得其它3个模块返回的数据，包括：

get\_earth\_cycle() ：获取地球循环状态

get\_cetus\_cycle() ：获取希图斯循环状态

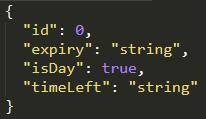
get\_orb\_cycle() ：获取奥布山谷循环状态

#### （4）get\_earth\_cycle()

获取地球循环状态。API链接为：

<https://api.warframstat.us/pc/earthCycle>

**数据示例：**



**数据参数：**

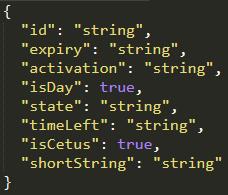
|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| id | number | 事件ID |
| expiry | string | 结束时间 |
| isDay | boolean | 是否为白天 |
| timeLeft | string | 剩余时间 |

#### （5）get\_cetus\_cycle()

获取希图斯循环状态，API链接为：

<https://api.warframestat.us/pc/cetusCycle>

**数据示例：**



**数据参数：**

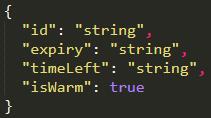
|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| id | string | 事件ID |
| expiry | string | 结束时间 |
| isDay | boolean | 是否为白天 |
| timeLeft | string | 剩余时间 |
| isCetus | boolean | 是否为希图斯相关数据 |
| shortString | string | 剩余循环时间 |

#### （6）get\_orb\_cycle()

获取福尔图娜循环状态，API链接为：

<https://api.warframestat.us/pc/vallisCycle>

**数据示例：**



**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| id | string | 事件ID |
| expiry | string | 结束时间 |
| timeLeft | string | 剩余时间 |
| isWarm | boolean | 是否为温暖状态 |

#### （7）get\_cetus\_mission()

获取希图斯赏金内容，API链接为：

<https://api.warframestat.us/pc/syndicateMissions>

希图斯赏金内容来源于集团任务API，从中提取出含有”syndicate”:“Ostrons”字段的即为希图斯赏金。

**数据示例：**



**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| nodes | array | 节点 |
| eta | string | 预计结束时间 |
| jobs | array | 任务列表 |
| syndicate | string | 集团名称 |
| id | string | 事件ID |
| expiry | string | 事件结束时间 |
| activation | string | 事件开始时间 |

**“jobs”字段内参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| activation | string | 任务开始时间 |
| expiry | string | 任务结束时间 |
| rewardPool | array | 奖励池 |
| type | string | 任务类型 |
| enemyLevels | array | 敌人等级 |
| standingStages | array | （未知） |
| minMR | number | （未知） |

#### （8）get\_conclave\_challenge() （不再开发）

获取PVP任务内容，API链接为：

<https://api.warframestat.us/pc/conclaveChallenges>

**数据示例：**

**数据参数：**

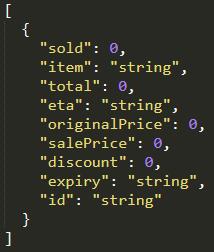
|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
|  |  |  |
|  |  |  |
|  |  |  |

#### （9）get\_darvo\_daily\_deal()

获取Darvo的每日特惠内容，API链接为：

<https://api.warframestat.us/pc/dailyDeals>

**数据示例：**



**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| sold | number | 已出售数量 |
| item | string | 出售的物品名称 |
| total | number | 出售总数 |
| eta | string | 预计结束时间 |
| originalPrice | number | 原价 |
| salePrice | number | 售价 |
| discount | number | 折扣百分比 |
| expiry | string | 售卖结束时间 |
| id | string | 事件ID |

#### （10）get\_orb\_mission()

获取奥布山谷赏金内容，API链接为：

<https://api.warframestat.us/pc/syndicateMissions>

奥布山谷赏金内容来源于集团任务API，从中提取出含有”syndicate”:“Solaris United”字段的即为奥布山谷赏金。

数据示例及数据参数参考：[（7）get\_cetus\_mission()](#_（7）get_cetus_mission())

#### （11）get\_events()

获取事件活动内容，API链接为：

<https://api.warframestat.us/pc/events>

**数据示例：**

**数据参数：**

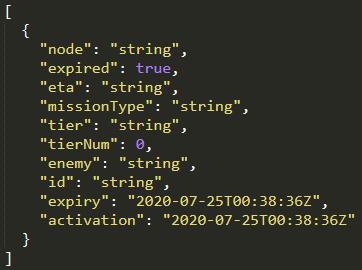
|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| activation | string | 事件开始时间 |
| expiry | string | 事件结束时间 |
| maximumScore | number | 分数上限 |
| currentScore | number | 目前分数 |
| smallInterval | number | （未知） |
| largeInterval | number | （未知） |
| faction | string | 敌人派系 |
| description | string | 事件描述 |
| tolltip | string | 事件副描述 |
| node | string | 节点 |
| concurrentNodes | array | 其它节点列表 |
| victimNode | string | 牺牲节点 |
| scoreLocTag | string | 分数标签 |
| rewards | array | 主要奖励表 |
| expired | boolean | 事件是否到期 |
| health | number | 剩余生命值 |
| affiliatedWith | string | 事件陪同集团 |
| jobs | array | 事件任务 |
| interimSteps | array | 阶段目标内容 |
| progressSteps | dict | 阶段进度 |
| progressTotal | number | 总进度 |
| showTotalAtEndOfMission | boolean | 任务结束时是否显示总数 |
| isPersonal | boolean | 是否为个人任务 |
| isCommunity | boolean | 是否为团队任务 |
| regionDrops | array | 地区掉落列表 |
| archwingDrops | array | 空战掉落列表 |
| asString | string | 掉落总表 |
| metadata | dict | （未知） |
| completionBonuses | array | 事件完成额外奖励 |
| scoreVar | string | 分数 |
| altExpiry | string | 事件切换结束时间 |
| altActivation | string | 事件切换开始时间 |
| nextAlt | dict | 下一事件切换时间表 |

#### （12）get\_fissures()

获取虚空裂缝任务内容，API链接为：

<https://api.warframestat.us/pc/fissures>

**数据示例：**



**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| node | string | 节点 |
| expired | boolean | 是否到期 |
| eta | string | 预计结束时间 |
| missionType | string | 任务类型 |
| tier | string | 关联遗物级别名称 |
| tierNum | number | 关联遗物级别 |
| enemy | string | 敌人类型 |
| id | string | 事件ID |
| expiry | string | 结束时间 |
| activation | string | 开始时间 |

#### （13）get\_flash\_sales() （不再开发）

获取商店热卖内容，API链接为：

<https://api.warframestat.us/pc/flashSales>

**数据示例：**

**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
|  |  |  |
|  |  |  |
|  |  |  |

#### （14）get\_global\_upgrades() （未开发）

获取当前全服加成内容，API链接为：

<https://api.warframestat.us/pc/globalUpgrades>

**数据示例：**

**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
|  |  |  |
|  |  |  |
|  |  |  |

#### （15）get\_invasions()

获取入侵任务内容，API链接为：

<https://api.warframestat.us/pc/invasions>

**数据示例：**

**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| defenderReward | dict | 防守方奖励 |
| attakingFaction | string | 攻击方派系 |
| completion | number | 完成百分比 |
| attakerReward | dict | 攻击方奖励 |
| count | number | 攻击次数计数 |
| completed | boolean | 是否已完成 |
| requiredRuns | number | 要求参与次数 |
| vsInfestation | boolean | 是否为对抗I系怪物 |
| node | string | 节点 |
| eta | string | 预计结束时间 |
| defendingFaction | string | 防守方派系 |
| id | string | 事件ID |
| activation | string | 开始时间 |
| rewardTypes | array | 奖励类型 |
| desc | string | 任务描述 |

其中，字段 count 、 requiredRuns 、 completion有如下关系：

(requiredRuns+count)/2/requiredRuns=completion

可得出：

attaker\_rest\_time=requiredRuns-count

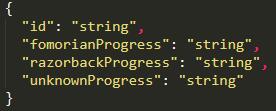
defender\_rest\_time=requiredRuns+count

#### （16）get\_construction\_progress()

获取入侵舰队建造进度，API链接为：

<https://api.warframestat.us/pc/constructionProgress>

**数据示例：**



**数据参数：**

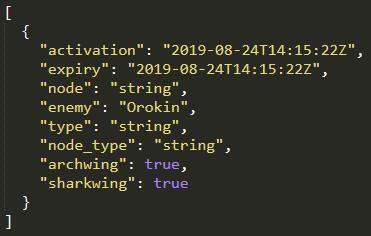
|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| id | string | 事件ID |
| fomorianProgress | string | 利刃豺狼建造进度百分比 |
| razorbackProgress | string | 巨人舰队建造百分比 |
| unknownProgress | string | 未知内容 |

#### （17）get\_kuva() （不稳定）

获取赤毒虹吸器任务内容，API链接为：

<https://api.warframestat.us/pc/kuva>

**数据示例：**



**数据参数：**

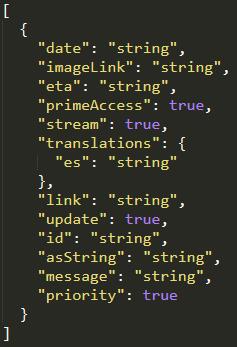
|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| activation | string | 开始时间 |
| expiry | string | 结束时间 |
| node | string | 节点 |
| enemy | string | 敌人派系 |
| type | string | 任务类型 |
| node\_type | string | 节点类型 |
| archwing | boolean | 是否需要archwing |
| sharkwing | boolean | 是否需要sharkwing |

#### （18）get\_news()

获取游戏新闻内容，API链接为：

<https://api.warframestat.us/pc/news>

**数据示例：**



**数据参数：**

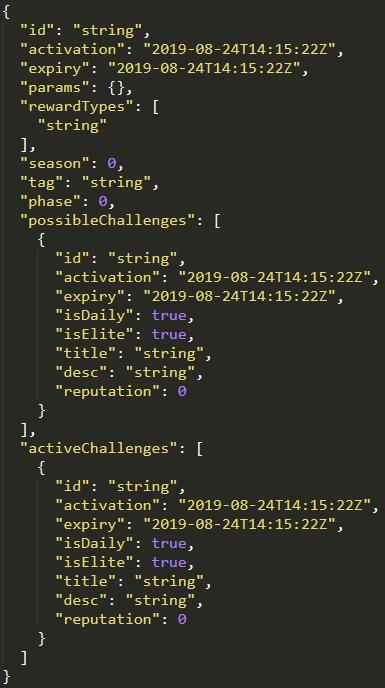
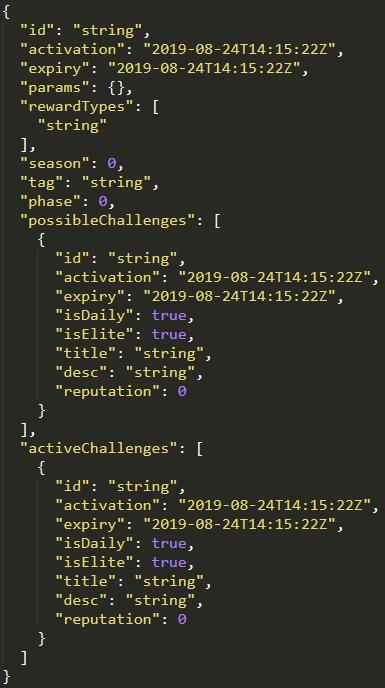
|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| date | string | 新闻日期 |
| imageLink | string | 图片链接 |
| eta | string | 预计结束时间 |
| primeAccess | boolean | 是否为PA新闻 |
| stream | boolean | 是否为平台新闻 |
| translations | dict | 新闻信息各语言翻译 |
| link | string | 新闻链接 |
| update | boolean | 是否更新 |
| id | string | 新闻ID |
| asString | string | 新闻字段 |
| message | string | 新闻信息 |
| priority | boolean | 是否优先显示 |

#### （19）get\_nightwave()

获取午夜电波任务内容，API链接为：

<https://api.warframestat.us/pc/nightwave>

**数据示例：**

****

**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| id | string | 事件ID |
| activation | string | 开始时间 |
| expiry | string | 结束时间 |
| params | list | 参数表 |
| rewardTypes | array | 奖励类型 |
| season | number | 电波季度 |
| tag | string | 电波活动标签 |
| phase | number | 电波阶段 |
| possibleChallenges | list | 可能的挑战内容 |
| activeChallenges | list | 正在进行的挑战内容 |

**“activeChallenges”字段内参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| id | string | 事件ID |
| activation | string | 开始时间 |
| expiry | string | 结束时间 |
| isDaily | boolean | 是否为每日任务，若不是，则该项不存在 |
| isElite | boolean | 是否为精英任务 |
| title | string | 任务名称 |
| desc | string | 任务描述 |
| reputation | number | 任务奖励声望 |

#### （20）get\_persistent\_enemy() （未开发）

获取追随者信息，API链接为：

<https://api.warframestat.us/pc/persistentEnemies>

**数据示例：**

**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
|  |  |  |
|  |  |  |
|  |  |  |

#### （21）get\_riven\_statistic() （不再开发）

获取指定紫卡信息，API链接为：

[https://api.warframestat.us/pc/rivens/search/{query}](https://api.warframestat.us/pc/rivens/search/%7bquery%7d)

链接后需要加上武器名称才可查询到具体的紫卡信息。

**数据示例：**

**数据参数：**

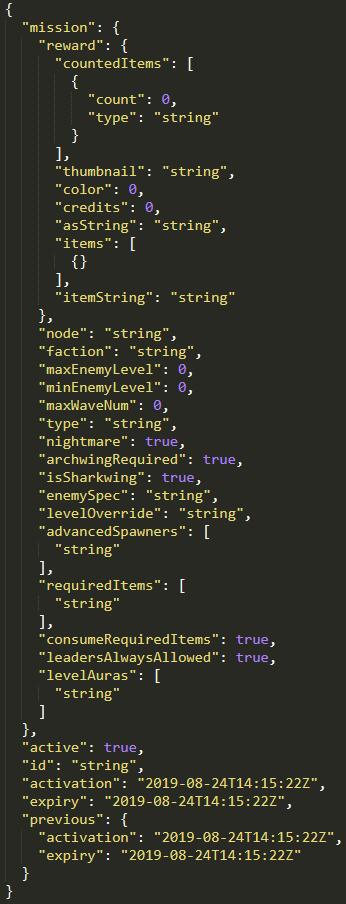
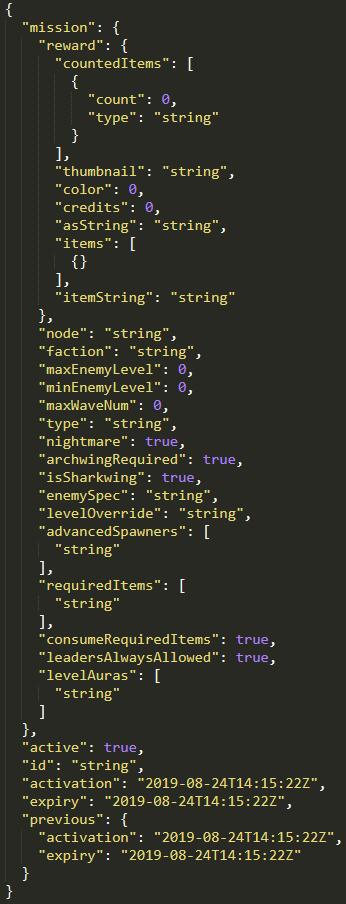
|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
|  |  |  |
|  |  |  |
|  |  |  |

#### （22）get\_sentient\_outpost()

获取航道星舰S船出没信息，API链接为：

<https://api.warframestat.us/pc/sentientOutposts>

**数据示例：**



**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| mission | list | 任务信息 |
| active | boolean | 是否进行中 |
| id | string | 事件ID |
| activation | string | 开始时间 |
| expiry | string | 结束时间 |
| previous | list | 上一任务信息 |

**“mission”字段内参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| node | string | 任务节点 |
| faction | string | 敌人派系 |
| type | string | 任务类型 |

#### （23）get\_sanctuary() （不再开发）

获取圣殿任务内容，API链接为：

<https://api.warframestat.us/pc/simaris>

**数据示例：**

**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
|  |  |  |
|  |  |  |
|  |  |  |

#### （24）get\_sortie()

获取突击任务内容，API链接为：

<https://api.warframestat.us/pc/sortie>

**数据示例：**



**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| id | string | 事件ID |
| activation | string | 开始时间 |
| expiry | string | 结束时间 |
| rewardPool | string | 奖励池 |
| variants | list | 任务详细内容 |
| boss | string | BOSS名称 |
| faction | string | 敌人派系 |
| expired | boolean | 是否已结束 |
| eta | string | 结束剩余时间 |

**“variants”字段内参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| node | string | 任务节点 |
| boss | string | BOSS名称 |
| missionType | string | 任务类型 |
| planet | string | 星球 |
| modifier | string | 任务状态 |
| modifierDescription | string | 任务状态描述 |

#### （25）get\_syndicate\_mission()

获取集团任务内容，API链接为：

<https://api.warframestat.us/pc/syndicateMissions>

集团任务内容来源于集团任务API，从中提取出含有下列集团特定字段的即为集团任务。

|  |  |  |
| --- | --- | --- |
| 字段名 | **字段值** | **对应集团** |
| syndicate | Steel Meridian | 钢铁防线 |
| Arbiters of Hexis | 均衡仲裁者 |
| Cephalon Suda | 中枢苏达 |
| Perrin Sequence | 佩兰数列 |
| Red Veil | 血色面纱 |
| New Loka | 新世间 |

数据示例及数据参数参考：[（7）get\_cetus\_mission()](#_（7）get_cetus_mission())

#### （26）get\_void\_trader()

获取虚空商人内容，API链接为：

<https://api.warframestat.us/pc/voidTrader>

**数据示例：**



**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| id | string | 事件ID |
| activation | string | 开始时间 |
| expiry | string | 结束时间 |
| character | string | 到访角色名 |
| location | string | 到访位置 |
| inventory | list | 携带物品 |
| psID | string | 事件副ID |
| active | boolean | 是否已到访 |
| startString | string | 开始剩余时间 |
| endString | string | 结束剩余时间 |

**“inventory”字段内参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
| item | string | 物品名称 |
| ducats | number | 需求杜卡德金币数 |
| credits | number | 需求星币数 |

#### （27）get\_arcane\_enhancement() （不再开发）

获取指定附能信息，API链接为：

[https://api.warframestat.us/arcanes/search/{query}](https://api.warframestat.us/arcanes/search/%7bquery%7d)

查询指定附能，链接后需要指定附能名称，且链接不能指定平台。

**数据示例：**

**数据参数：**

|  |  |  |
| --- | --- | --- |
| 字段名 | **数据类型** | **说明** |
|  |  |  |
|  |  |  |
|  |  |  |

#### （28）get\_rest\_time(expiry)

传入GMT时区结束时间，获取当前GMT时区时间并计算获得时间差，返回时间差字符串。

# 五、Warframe Market API：

## 1、API调用地址：

Warframe Market API链接：<https://api.warframe.market/v1>

Warframe Market网站链接：<https://warframe.market>

添加 **/items/{item\_name}** 可查询物品信息；

添加 **/items/{item\_name}/orders** 可查询物品交易信息。

其它API链接地址查看：<https://github.com/WFCD/market-api-spec>

Warframe相关开发Github项目查看：

Warframe Community Developers：<https://github.com/WFCD>

## 2、内建接口：

相关代码文件均放于warframe文件夹下。

### warframe\_market.py

#### （1）get\_market(message\_from\_user)

获取物品交易信息，提取数据并整理，返回要发送的消息内容。

**1) 提取API返回的交易订单：**

data.get(“payload”).get(“orders”)

**2) 筛选交易订单：**

data[i].get("user").get("status")=="ingame"

and

data[i].get("order\_type")=="sell"

and

data[i].get("platform")=="pc"

and

data[i].get("region")=="en"

**3) 对筛选后的交易订单进行排序（按价格从低到高）：**

for i in range(online\_length-1):

ex\_flag=False

for j in range(online\_length-i-1):

if (online\_list[j].get("platinum")>online\_list[j+1].get("platinum")): online\_list[j],online\_list[j+1]=online\_list[j+1],online\_list[j]

ex\_flag=True

if not ex\_flag:

break

**4）建立返回的内容：**

返回的内容最多显示10名在线卖家。

**5）从物品信息中获取物品的英文全名：**

根据API查询物品信息，并提取出全名。

data=data.get("payload").get("item").get("items\_in\_set")

items\_length=len(data)

for n in range(items\_length):

if (data[n].get("url\_name")==url\_name):

item\_name="（"+data[n].get("en").get("item\_name")+"）"

break

# 六、小栗子HTTP API：

该API文档位于”All Examples/xzfg API Examples/HTTP API.txt”。

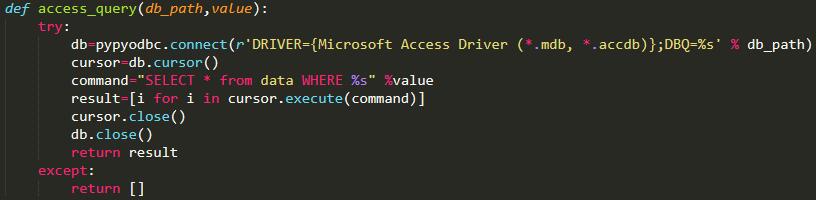
# 七、数据库结构设计：

为了便于修改程序需要查询和使用的数据，以及为了提供管理员控制功能，引入数据库结构进行增删改查操作。所有数据均存放为.mdb文件中，位于data文件夹下，所有数据均存放于一张表中，表名为data。数据库操作代码在database\_operate.py文件中，包括增删改查四种功能。

数据库的数据在程序运行期间通过一次性读取的方式存放在内存中，因此，在程序开启时需首先读取数据库，管理员控制可以通过命令使程序对非程序配置数据库进行重载。从数据库读取出的数据一律放在对应的\*\_globalvar.py文件代码中作为global变量存储，通过预设的接口对数据进行读取。此种考虑建立在以下两点上，一是避免多次访问数据库源文件导致的可能的数据库驱动崩溃问题，这是由于Microsoft Access驱动在密集访问下可能崩溃，二是通过内存查询的方式可以提高查询效率。

**注意：**对数据库进行操作时，需预设必要前提避免随意更改，即在管理员权限下使用，即只有拥有预设在程序配置文件中的拥有管理员权限的帐号才可使用。

## 1、数据库查询：

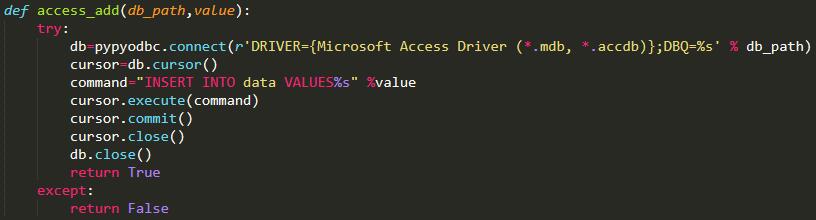


其中，db\_path为数据库路径，value为xxx=xxx and xxx=xxx格式。

使用try-except结构以避免出现数据库操作出错的问题。

当查询到数据时，返回所查询到的所有数据。当查询出错时，返回空数组。

## 2、数据库添加：



其中，db\_path为数据库路径，value为'' '' ...格式。

使用try-except结构以避免出现数据库操作出错的问题。

当添加成功时返回True布尔值。当添加出错时，返回False布尔值。

## 3、数据库删除：（未验证）

## 4、数据库更改：（未验证）

## 5、数据库架构设计：

为了保证程序所用数据的高可维护性、高可优化性、高安全性等要求，有必要对数据库架构进行分析和设计。在本程序中，以大规模数据库架构设计为目标进行设计，形成分布式数据库系统，即所用数据在物理上分散、逻辑上集中的数据库系统。

### （1）设计目标：

分布式数据库的特征是本地自治、非集中式管理以及高可用性，其分布透明性由位置独立性、数据分片独立性和数据复制独立性构成。总体目标有：本地自治、非集中式管理、高可用性、位置独立性、数据分片独立性、数据复制独立性、分布式查询处理、分布式事务管理、硬件独立性、操作系统独立性、网络独立性、数据库管理系统独立性。

### （2）数据分布策略：

数据分布策略从数据分片和数据分配两个角度考虑，分片是对关系的操作，分配是对分片结果的操作。

数据分片是对某一个关系划分为多个片断，而这些片断包含足够的信息可以重构关系。

数据分配是分布式数据库的特征，包括集中式方法、分割式方法、全复制式方法、混合式方法。

### （3）分布式数据目标的实现：

1）位置独立性：程序所用的数据均存放在/data下，依据各个功能分割，而对各个功能所需数据采取进一步的分割，细化查询和功能，物理上整体采取分割式方法进行数据分配。

2）数据分片独立性：在代码中，对各个功能使用\*\_globalvar.py读取各功能所需数据，并将数据载入内存存储，各个功能的globalvar是分开存储的，由此完成数据的局部汇总，但每个数据库由单一变量存储，形成局部统一下的数据分割，从分割式分配模式到功能分片模式。

3）高可用性：在此模块中的全局数据均按固定数据存储，不可修改，只能通过对数据库的重载命令进行更新，以此保障当数据库被错误修改时不会立即对程序造成影响，并且在一定程度上通过内存存储方式提高数据读取速度。

4）数据复制独立性：在globalvar下，建立数据读取接口以供程序对模块中全局变量数据的使用，每次读取时对数据完全复制，不进行任何修改操作。

5）分布式查询处理：在对数据进行访问时，直接通过对复制过来的数据进行查询，查询方法包括字典式查询、遍历查询等，由于数据本身的存储方式具备分布的特征，因此对数据的查询操作也是分布查询。

6）非集中性管理：在admin功能中，程序管理员可通过特定命令对数据库文件进行直接的增删改查操作，在命令中，需要指定操作的数据库文件，每次只能完成对一个文件的一个命令操作。

7）数据库管理系统独立性：数据库与程序中的数据是相互独立的，程序读取数据库后在查询操作中不会对数据库采取操作，因此在程序运行期间，数据库的状态不会影响到程序运行，对数据库的管理也是可独立的。

# 八、管理员控制：

管理员控制相关代码均位于admin文件夹内。由于管理员具备对程序所使用的数据的操作权限，因此该功能不可用于普通用户，只可用于开发者授权之用户。权限的控制文件位于data/config/aurhority.mdb文件中，通过自行设置该数据库文件来控制聊天权限。

# 九、腾讯云腾讯智能对话平台TBP：

腾讯云腾讯智能对话平台TBP为腾讯云产品的AI平台服务内容之一，其功能侧重于机器人问答，类似于简单应答型机器人，可自设语义模型添加问答意图和词典来丰富应答能力。

在使用前，需建立Bot，Bot建立完成后会获得唯一的BotId，通过BotId访问该机器人。

在本程序中，使用该平台应用接入中的API接入完成对代码的编写。另外借助腾讯云自建的python SDK进行通讯和密钥的处理，该python第三方库为tencentcloud-sdk-python。

## 1、通信协议及密钥处理：

TBP使用HTTPS通信协议，使用签名方法v3（TC3-HMAC-SHA256）。密钥则需首先生成API密钥，每一个API密钥只能提供一个Bot使用，生成API密钥后将会有唯一的APPID、SecretId和SecretKey，分别表示API密钥ID、密钥ID和密钥，任何访问Bot的请求都需要进行验证，由于通过此密钥可操作所有腾讯云资源，等同于登录密码，因此需进行高度保密。在本程序中，与密钥相关的数据均以数据库文件存储。由于SDK中已经包含对密钥的处理，具体的通信方式和密钥算法可查看相关腾讯云文档。

腾讯云TBP文档：<https://cloud.tencent.com/document/product/1060/37428>

## 2、实现代码：

实现代码借用腾讯云文档中TextProcess页面代码调试中代码生成的内容，通过填写必要的信息，可直接获得代码块。

# 十、腾讯AI开放平台智能闲聊NLP：

腾讯AI开放平台链接：<https://ai.qq.com/v1/>

此功能使用腾讯AI开放平台中的智能闲聊功能，与TBP不同的是，该功能不用提供语义词库，其本身具备一定的智能性，还可通过自定义配置闲聊画像对AI设置属性。

在使用前，需创建一个实际应用，将会获得APPID和APPKEY，分别为密钥ID和密钥。在程序中，采用API接入的方式完成代码的编写。

API地址：<https://api.ai.qq.com/fcgi-bin/nlp/nlp-textchat>

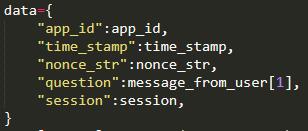
API文档：<https://ai.qq.com/doc/nlpchat.shtml>

## 1、通信协议及密钥处理：

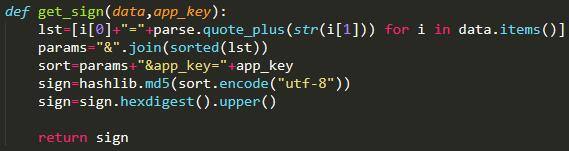
API接口采取的是HTTPS协议，通过GET或POST请求，以UTF-8字符编码，JSON响应格式，签名接口鉴权的方式完成通信。

在本代码中使用GET方法请求。与TBP类似，密钥相关数据均以数据库文件的方式存放，以保证其保密性。由于其API文档对密钥的说明存在不明之处，因此做参考用即可，实际情况以下述代码为准。

密钥需通过特殊的算法进行加密处理，在算法中需要传入的数据包括：



密钥加密算法为：



## 2、实现代码：

在通信请求代码中，较为重要的有两点，一个是传入接口的数据，一个是请求头。具体实现代码如下：



# 十一、使用Git上传代码至仓库：

## 1、提交commit：

git commit -m “<message>”

## 2、推送到远程仓库：

git push origin master

## 3、强制推送到仓库：

git push --force origin master

## 4、添加版本标签：

git tag -a <tag> -m “<message>”

## 5、发布版本标签：

git push origin <tag>

# 十二、网页爬虫：

该模块所有代码均存放在/webscrawler下。开发网页爬虫而实现抓取网页文本、网页图片等功能，通过机器人将内容提供给用户。

## 1、必应搜索背景图片抓取：

必应搜索：<http://cn.bing.com/>

该爬虫用于从必应搜索网页中抓取出背景图片链接。

使用的第三方库：requests、urllib、bs4、os。

**实现过程：**

代码从res=requests.get()链接网站url获得页面数据，通过from bs4 import BeautifulSoup导入BeautifulSoup，使用html.parser方法获得页面内容BeautifulSoup(res.text,’html.parser’)。在必应网页中，背景图片链接在第一个link中可获得，通过select(‘link’)方法选择出所有的包含link分支的内容，并选择第一个作为获得的link分支。在该link分支中的标签href的值就是背景图片的尾链接，通过soup.get(‘href’)获得该标签的值，与必应搜索的链接头合并即为背景图片链接。还可通过urllib.request.urlretrieve(url,path)将图片保存在本地路径下。

# 十三、其它工具：

工具性的代码均存放于/tools中。

## 1、文字转图片工具：

该工具是为了通过将文字转换为图片，发送图片来避免文字消息被屏蔽。

使用的第三方库有：PIL、os、textwrap

**实现过程：**

为了避免文字消息过长导致的图片长度过大，通过textwrap.wrap(text,width=<number>)来控制每行的文本长度，其当文本长度超过width时，会自动添加\n在最大长度处。

通过PIL库来制作图片，from PIL import Image,ImageFont,ImageDraw。通过Image.new(type,(width,height),(red,Green,Blue))创建一个背景图片，并通过ImageDraw.Draw定义绘图，通过ImageFont.truetype(fontpath,size)确定文字字体和大小，其中文字字体需要提供一个.ttf格式的字体文件路径，该文件存放于/data/fonts下。然后使用<ImageDraw>.text((left,up),text,font=font,fill=”#000000”)方法在图片中绘制文本。最后保存图片在本地路径下，图片保存为/buffer/w2p.png。函数返回该图片的本地路径。