BratanREF API 1.0 DOCS

Внимание! Используйте API V1, если не знаете зачем нужен V2! API V1 более стабилен и безопасен для исполнения!

Документация ниже соответствует API версии 1.

Все запросы отправляются на https://api.tgcryptoref.com

API ключ, вставляется в headers.

```
headers = {
   "Authorization": "Bearer {api_key}",
   "Content-type": 'application/json'
}
```

Ограничения API – 2 запроса в секунду

Методы API:

1) **Получить информацию о текущем балансе:**

/balance - GET

Успешный ответ (200):

{‘balance’: 500.00}

Типы данных:

balance – float

Не успешный ответ (400):

{"error": "string"}

2 - **Получить ID-категорий:**

/services – GET

Успешный ответ (200):

[{"name": "string","price": 0,"category_id": 1, ‘url’: ‘string’}]

Типы данных:

name – string

price – float

category_id – int

url - string

Не успешный ответ (400):

{"error": "string"}

3 – Создать заказ:

/tasks – POST, отправить json

{"category_id": 1, "quantity": 10,

"url": "https://t.me/notpixel/app?startapp=f396019363_s594995"}

Успешный ответ (200):

{"status": "OK", "order_id": 1}

Не успешный ответ (400):

{"status": "BAD", "error_code": 100, ‘detail’: ‘string’}

Типы данных ответа:

status – string (Может быть - OK и BAD)

order_id – int

Если статус BAD – вернёт:

error_code – int

detail – доп описание ошибки

Коды error_code:

100 – Ошибка парсинга реферального кода

101 – Недостаток баланса

102 – Неопознанная ошибка

Типы данных для отправки запроса:

category_id – int (ID категории из метода /services)

quantity – int (Количество для заказа)

url – string (Ссылка для заказа)

4 – Получить информацию по заказу:

/task_info – POST, отправить json

{"order_id": 1}

order_id – int

Успешный ответ (200):

{

"order_id": 0,

"category_id": 0,

"name": "string",

"quantity": 0,

"quantity_executed": 0,

"price_one": 0,

"cost": 0,

"url": "string",

"refferal_code": "string",

"status": 0,

"cancel_reason": "string",

"date": timestamp

}

cancel_reason – Optional if status is 2

status:

0 – в работе

1 – выполнен

2 – отменён

Не успешный ответ (400):

{"error": "string"}

Python Example. AioHttp.

PasteBin Link: https://pastebin.com/Vcbmbt7p

Password: grYn4XcHVQ

```python
import aiohttp
import asyncio
import json

class APITestClient:
    def __init__(self, base_url, api_key):
        """
        Initializes the API Test Client.

        :param base_url: The base URL of the API (https://tgcryptoref.com)
        :param api_key: The API key for authentication
        """
        self.base_url = base_url
        self.api_key = api_key
        self.headers = {
            "Authorization": f"Bearer {self.api_key}",
            "Content-Type": "application/json"
        }
        self.session = None
        self.order_id = None

    async def __aenter__(self):
        """
        Enter the asynchronous context manager.
        """
        self.session = aiohttp.ClientSession(headers=self.headers)
        return self

    async def __aexit__(self, exc_type, exc, tb):
        """
        Exit the asynchronous context manager.
        """
        await self.session.close()

    async def get_balance(self):
        """
```

```
        Retrieves the current balance.

        :return: The balance as a float
        """
        url = f"{self.base_url}/balance"
        async with self.session.get(url) as response:
            if response.status == 200:
                data = await response.json()
                balance = data.get('balance')
                print(f"Balance: {balance}")
                return balance
            else:
                error = await response.json()
                print(f"Error fetching balance: {error}")
                return None

    async def get_services(self):
        """
        Retrieves the list of available services.

        :return: A list of services
        """
        url = f"{self.base_url}/services"
        async with self.session.get(url) as response:
            if response.status == 200:
                services = await response.json()
                return services
            else:
                error = await response.json()
                print(f"Error fetching services: {error}")
                return None

    async def create_task(self, category_id, quantity, url):
        """
        Creates a new task/order.

        :param category_id: The category ID for the service
        :param quantity: The quantity for the order
        :param url: The URL associated with the order
        :return: The order ID if successful
        """
        endpoint = "/tasks"
        full_url = f"{self.base_url}{endpoint}"
        payload = {
            "category_id": category_id,
            "quantity": quantity,
            "url": url
```

```
        }
        async with self.session.post(full_url, json=payload) as response:
            if response.status == 200:
                data = await response.json()
                self.order_id = data.get('order_id')
                print(f"Task Created: {data}")
                return self.order_id
            else:
                error = await response.json()
                print(f"Error creating task: {error}")
                return None

    async def get_task_info(self, order_id):
        """
        Retrieves information about a specific task/order.

        :param order_id: The ID of the order
        :return: The order information as a dictionary
        """
        endpoint = "/task_info"
        full_url = f"{self.base_url}{endpoint}"
        payload = {
            "order_id": order_id
        }
        async with self.session.post(full_url, json=payload) as response:
            if response.status == 200:
                task_info = await response.json()
                print(f"Task Info: {json.dumps(task_info, indent=2)}")
                return task_info
            else:
                error = await response.json()
                print(f"Error fetching task info: {error}")
                return None

    async def run_tests(self):
        """
        Runs all the test methods sequentially.
        """
        print("Starting API Tests...\n")
        await self.get_balance()
        print("\n")
        await asyncio.sleep(0.5)
        await self.get_services()
        await asyncio.sleep(0.5)
        print("\n")
        # Parameters for creating a task as per the user's request
        test_category_id = 21
```

```
        test_quantity = 1
        test_url = "https://t.me/wcoin_tapbot/wcoin_app?startapp=NTQ2NTMQ=="
        await self.create_task(test_category_id, test_quantity, test_url)
        print("\n")
        if self.order_id:
            await asyncio.sleep(1)  # Wait a moment before fetching task info. Do
not fetch if your order just started please :)
            await self.get_task_info(self.order_id)
        print("\nAPI Tests Completed.")

# Entry point for the script
async def main():
    base_url = "https://api.tgcryptoref.com"
    api_key = "YOUR_API_KEY"

    async with APITestClient(base_url, api_key) as client:
        await client.run_tests()

if __name__ == "__main__":
    asyncio.run(main())
```

Документация для API V2 (ДЛЯ ПАНЕЛЕЙ)

Для API V2 – соответствие API панелей, запросы отправляются на https://api.tgcryptoref.com/apiv2 - POST {‘key’: api_key, ‘action’: action}

Доступные action – balance, services, add, status

Сверяйте типы данных для V2, они у панелей не соответствуют действительности, поэтому пришлось сделать такое же не соответствие (выдают string вместо float or integer) ☺

В случае успешного ответа сервис вернет статус 200, в случае ошибки 400, 503 с описанием ошибки или другой без ее описания. (неопознанный)

Получение баланса:

action: balance

key: api_key

Пример ответа:

{'balance': '390.0', 'currency': 'RUB'}

Получение статуса заказа:

action: status

order: order_id (int)

key: api_key

Пример ответа:

{'category_id': 8, 'charge': '24.0', 'currency': 'RUB', 'date': 1728848582, 'name': 'Notpixel', 'order_id': 315, 'price_one': '20.0', 'quantity': '3', 'quantity_executed': '0', 'refferal_code': 'f546534121_t', 'remains': '3', 'status': 2, 'url': 'https://t.me/notpixel/app?startapp='f546534121_t'}

Получение списка сервисов:

action: services

key: api_key

Пример ответа:

[{'cancel': False, 'max': 2500, 'min': 1, 'name': 'Yescoin', 'rate': 8.0, 'refill': False, 'service': 1, 'url': 'https://t.me/theYescoin_bot'},…]

Создание заказа:

action: add

service: category_id (int)

quantity: quantity (int)

link: ссылка (str)

Пример ответа:

{'order': 397}