EOSPark WebSocket 接口文档

连接说明

1. API KEY可在这里 自助申请
2. 同一个API KEY只能有一个连接，只有头一个连接有效，后来的连接会失败
3. 订阅账号/合约数分别最多5个
4. 消息类型

类型	描述
heartbeat	心跳
auth	鉴权信息
subscribe	订阅信息
data	订阅数据
unsubscribe	取消订阅

使用方式及接口说明

1. 使用方式

WebSocket 地址：

```
wss://ws.eospark.com/v1/ws?apikey=YOUR_API_KEY
```

或将API KEY 放在请求header中，可参照后面的使用示例 连接成功后服务器主动10s一次心跳包，包内容：

```
{
  "errno":0,
  "msg_type":"heartbeat",
  "errmsg":"",
  "data":{
    "heart_beat":"2018-11-02 15:30:02"
  }
}
```

参数	类型	说明
errno	Number	状态码
msg_type	String	消息类型
errmsg	String	状态描述
data	Object	数据内容
heart_beat	String	心跳时间

2. 接口

订阅

请求

```
{
    "msg_type": "subscribe",
    "account": "iameoschatio",
    "contract": "eosio.token",
    "action_name": "transfer"
}
```

参数	类型	说明
msg_type	String	消息类型
account	String	订阅账号数组，逗号分隔，不能为空
contract	String	订阅合约数组，逗号分隔，不能为空
action_name	String	动作名称，传空为全部

请求的返回内容

```
{
  "errno":0,
  "msg_type":"subscribe",
  "errmsg":"",
  "data":{
    "subscribe":true,
    "msg":"ok"
  }
}
```

参数	类型	说明
errno	Number	状态码
msg_type	String	消息类型
errmsg	String	状态描述
data	Object	消息内容
subscribe	Bool	订阅状态 成功:true 失败:false
msg	String	订阅状态描述

数据的推送内容

```
{
    "errno": 0,
    "msg_type": "data",
    "errmsg": "",
    "data": {
        "trx_id":
"689b818fa45704e3c92d4c8b6c10bcea4fb0b24e0009c2d6273282eea7d1bec6",
        "block_num": 25477790,
        "trx_timestamp": "2018-11-06T06:30:50.000",
        "actions": [
            {
                "account": "eosio.token",
                "authorization": [
                    {
                        "actor": "luckymedice1",
                        "permission": "active"
                    }
                ],
                "data": {
                    "from": "luckymedice1",
                    "memo": "Bet 12553721178981466931 refer reward! -
www.eosluckyme.com",
                    "quantity": "0.0050 EOS",
```

```
                    "to": "vvvvvvvvvvva"
                },
                "hex_data":
"10147249490f918e60f6de7befbdf7de320000000000000004454f53000000003b4265
74203132353533373231313738393831343636393331207265666572207265776172642
1202d207777772e656f736c75636b796d652e636f6d",
                "name": "transfer"
            }
        ]
    }
}
```

取消订阅

请求

```
{
    "msg_type": "unsubscribe",
    "account": "iameoschatio",
    "contract": "eosio.token",
    "action_name": "transfer"
}
```

参数	类型	说明
msg_type	String	消息类型
account	String	取消订阅的账号(不能为空,暂时只支持一个)
contract	String	取消订阅合约(不能为空,暂时只支持一个)
action_name	String	动作名称(如果action_name为空，就取消订阅账户和合约,如果不为空就取消相应的action)

请求的返回内容

```
{
  "errno":0,
  "msg_type":"unsubscribe",
  "errmsg":"",
  "data":{
    "subscribe":true,
    "msg":"ok"
  }
}
```

参数	类型	说明
errno	Number	状态码
msg_type	String	消息类型
errmsg	String	状态描述
data	Object	消息内容
unsubscribe	Bool	取消订阅状态 成功:true 失败:false
msg	String	取消订阅状态描述

3. 错误码说明

1. 鉴权失败(服务器直接拒绝连接)

```
{
  "errno":403,
  "msg_type":"auth",
  "errmsg":"Forbidden"
}
```

参数	类型	说明
errno	Number	状态码
msg_type	String	消息类型
errmsg	String	状态描述

2. 重复连接(API KEY 正在使用中)

```
{
    "errno":406,
    "msg_type":"auth",
    "errmsg":"No repeat connection."
}
```

参数	类型	说明
errno	Number	状态码
msg_type	String	消息类型
errmsg	String	状态描述

客户端连接示例

客户端代码示例（GO）

```go
//导入包 go get github.com/gorilla/websocket

package main

import (
    "github.com/gorilla/websocket"
    "log"
    "os"
    "os/signal"
    "time"
)

func main() {
    createClient()
}

func createClient() {
    // 中断通知chan
    interrupt := make(chan os.Signal, 1)
    signal.Notify(interrupt, os.Interrupt)
    url := "wss://ws.eospark.com/v1/ws"
    c, _, err := websocket.DefaultDialer.Dial(url, map[string][]string{
        "apikey": {"a9564ebc3289b7a14551baf8ad5ec60a"},
    })
    if err != nil {
        log.Fatal("dial:", err)
    }else {
        log.Printf("connect to %s successfully, waiting for heartbeat\n", url)
    }
    defer c.Close()
    done := make(chan struct{})
    go func() {
        defer close(done)
        for {
            _, message, err := c.ReadMessage()
            if err != nil {
                log.Println("read:", err)
                return
            }
            //收到服务器推送消息
            log.Printf("recv: %s", message)
```

```
        }
    }()
    // 模拟订阅 20s一次
    ticker := time.NewTicker(time.Second * 20)
    defer ticker.Stop()
    for {
        select {
        case <-done:
            return
        case <-ticker.C:
            // 发送订阅信息
            json :=
`{"msg_type":"subscribe","account":"eosio.token","contract":"eosio.token","action_name": "transfer"}`
            err := c.WriteMessage(websocket.TextMessage, []byte(json))
            log.Println("写数据:", json, err)
            if err != nil {
                log.Println("write:", err)
                return
            }
        case <-interrupt:
            log.Println("interrupt")
            // Cleanly close the connection by sending a close message and then
            // waiting (with timeout) for the server to close the connection.
            err := c.WriteMessage(websocket.CloseMessage, websocket.FormatCloseMessage(websocket.CloseNormalClosure, "bay bay"))
            if err != nil {
                log.Println("write close:", err)
                return
            }
            select {
            case <-done:
            case <-time.After(time.Second):
            }
            return
        }
    }
}
```

网页版WebSocket测试

大家也可以在这里 直接体验

服务器配置 状态: 连接成功

服务地址 wss://ws.eospark.com/v1/ws?apik 关闭连接

发包设置

每隔 1 秒发送内容 PING 开始发送

需要发送到服务端的内容

发包清空输入

发送到服务端

调试消息

15:34:20 => 初始化完成

15:34:24 => OPENED => ws.eospark.com/v1/ws?apikey=[redacted]

消息记录

收包清空记录 收包JSON解码 暂停接收

收到消息 15:37:14


```
{
  "errno":0,
  "msg_type":"heartbeat",
  "errmsg":"",
  "data":{
    "heart_beat":"2018-11-09 15:37:14"
  }
}
```