第三方接口（websocket部分）简要说明

提供基于请求协议http 和 webSocket的两种请求方式。

现对webSocket方式进行说明，如下：

1.客户端简要说明

客户需要编写webSocket客户端，向我方提供的地址发送请求，客户需要按规则传递签名，我方会在通信握手过程中，校验用户的身份。

上述中的编写客户端，以及签名可以参考DEMO的 JS部分。

```
const socket = new WebSocket('ws://open.coinceres.com:19001?
api_key=bGppXpZqNrzVOptf&sign=2bb24f69ca812c766a7bd73e3c2a331b55373a122bba8
ede47ced0b4a07139c0');
```

websocket打开链接，同服务端进行tcp握手，需要传入api_key 和sign，用于用户识别绑定和身份验证。sign即签名后的字符串。

```
签名说明：
签名需要api_key和secret，关于api_key和secret的获取参考文档-第三方接口（http部分）。
使用SHA256对原始字符串加密得到的字符串即称为sign

比如，原始字符串：
api_key=bGppXpZqNrzVOptf&secret=bcBFnn3GbFz1cB73f5TkJa8P5gbrFdgt
签名后：2bb24f69ca812c766a7bd73e3c2a331b55373a122bba8ede47ced0b4a07139c0
```

JS客户端代码示例如下，仅供参考：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>WEBSOCKET-DEMO</title>
    </head>
    <body>
        <div class="footer">
            <input id="messageText" type="text" placeholder="说点什么吧..."
>
            <span id="btn" onclick="sendMsg()">发送</span>
        </div>
        <script type="text/javascript">
                // Create WebSocket connection.
```

```
                const socket = new
WebSocket('ws://open.coinceres.com:19001?
api_key=bGppXpZqNrzVOptf&sign=2bb24f69ca812c766a7bd73e3c2a331b55373a122bba8
ede47ced0b4a07139c0');

                // Connection opened
                socket.addEventListener('open', function (event) {
                    socket.send('ping');
                });

                // Listen for messages
                socket.addEventListener('message', function (event) {
                  if(typeof event.data === String) {
                    console.log("Received data string>>>>: ",event.data);
                  }

                  if(event.data instanceof ArrayBuffer){
                    var buffer = event.data;
                    console.log("Received arraybuffer>>>>: ", buffer);
                  }
                });

                socket.addEventListener('close',function(event){
                    console.log("connecton closed>>>>")
                });

                socket.addEventListener("error", function(event) {
                    console.log("handle error event>>>>")
                });

                const sendMsg = function() {
                    const msg = document.getElementById('messageText');
                    socket.send(msg.value);
                    msg.value="";
                }
                //回车事件绑定
                document.onkeydown = function(e){
                    var ev = document.all ? window.event : e;
                    if(ev.keyCode==13) {
                        sendMsg();
                    }
                }
                const autoHeart = function () {
                // log("发心跳了")
                    socket.send('ping')
                }
                window.setInterval(autoHeart,1000*50)
            </script>
    </body>
```

```
</html>
```

2.调用过程简要示例说明

上述的websocket握手通过后，客户可以按约定规则向服务端发送字符串调用接口，现约定调用规则，如下：

方便数据的交互，约定数据结构采用Json格式表示。

如下单请求，客户只需要向服务端传入下列json字符串即可

```
{
    "exchange":"TEST",
    "contract":"ADA_BTC",
    "price_type":"limit",
    "entrust_price":"0.000011",
    "profit_value":"0.00000997",
    "entrust_vol":"10",
    "entrust_bs":"buy",
    "future_dir":"open",
    "lever":"10",
    "client_oid":"12345",
    "biz_type":11
}
```

得到的响应如下：

```
{
    biz_type: 11
    code: "200"
    data:
        {
            client_oid: "12345"
            system_oid: "126014167605456896"
        }
    message: "SUCESS"
}
```

对比http方式，可以发现请求和相应的响应都多出一个键值对"biz_type":11, 该键值对用来表示请求的方式为下单。

biz_type的约定值有：

```
11 表示下单
12 表示撤单
13 表示获取交易对
14 表示获取账户信息
15 表示获取订单信息
16 表示获取合约持仓信息
17 表示获取交易记录
```

webSocket调用接口的详细文档，参考文档websocket接口说明。