 WebSocket API for FUTURES

**说明： websocket接口暂时未升级到v3，但已经支持使用v3 验证方式登录，此文档标注了部分字段与v3的对应，尽请期待后续升级。**

[开始使用](https://github.com/okcoin-okex/API-docs-OKEx.com#%E5%BC%80%E5%A7%8B%E4%BD%BF%E7%94%A8--)

WebSocket是HTML5一种新的协议(Protocol)。它实现了客户端与服务器全双工通信，使得数据可以快速地双向传播。通过一次简单的握手就可以建立客户端和服务器连接，服务器根据业务规则可以主动推送信息给客户端。其优点如下：

* 客户端和服务器进行数据传输时，请求头信息比较小，大概2个字节；
* 客户端和服务器皆可以主动地发送数据给对方；
* 不需要多次创建TCP请求和销毁，节约宽带和服务器的资源。

强烈建议开发者使用WebSocket API获取市场行情和买卖深度等信息。

[请求交互](https://github.com/okcoin-okex/API-docs-OKEx.com#%E8%AF%B7%E6%B1%82%E4%BA%A4%E4%BA%92--)

合约交易WebSocket服务连接地址：<wss://real.okex.com:10440/websocket/okexapi>

访问时需要科学上网

[**发送请求**](https://github.com/okcoin-okex/API-docs-OKEx.com#%E5%8F%91%E9%80%81%E8%AF%B7%E6%B1%82)

请求数据格式为：  
{'event':'addChannel','channel':'channelValue','parameters':{'api\_key':'value1','sign':'value2'，'timestamp':'value3','passphrase':'value4'}}  
其中  
event: addChannel(注册请求数据)/removeChannel(注销请求数据)  
channel: OKEx提供请求数据类型  
parameters: 参数为选填参数，其中api\_key为用户申请的APIKEY，sign为签名字符串，签名规则参照请求说明（API概述验证部分）,passphrase为申请v3 api时所填写，timestamp为时间戳（是unit时间戳，单位是秒，  时间戳30秒后会过期， 推荐使用 time endponit 查询 api 服务器的时间，如果你的服务器时间和api 服务器时间有偏差的话）

其中timestamp示例：

const timestamp = ‘’ + Date.now() / 1000

其中sign 示例 ：

sign=CryptoJS.enc.Base64.stringify(CryptoJS.HmacSHA256(timestamp + 'GET' + '/users/self/verify', secret))

method一律默认为'GET 。

requestPath 一律默认为'/users/self/verify'。

例如： websocket.send("{'event':'addChannel','channel':'ok\_sub\_spot\_usd\_btc\_ticker'}")  
websocket.send("[{'event':'addChannel','channel':'ok\_sub\_spot\_usd\_btc\_ticker'},{'event':'addChannel','channel':'ok\_sub\_spot\_usd\_btc\_depth'},{'event':'addChannel','channel':'ok\_sub\_spot\_usd\_btc\_trades'}]"),支持批量注册

[**服务器响应**](https://github.com/okcoin-okex/API-docs-OKEx.com#%E6%9C%8D%E5%8A%A1%E5%99%A8%E5%93%8D%E5%BA%94)

返回数据格式为： [{"channel":"channel","success":"","errorcode":"","data":{}}, {"channel":"channel","success":"","errorcode":1,"data":{}}]  
其中  
channel: 请求的数据类型  
result: true成功，false失败(用于WebSocket 交易API) data: 返回结果数据  
errorcode: 错误码(用于WebSocket 交易API)

[**推送过程说明**](https://github.com/okcoin-okex/API-docs-OKEx.com#%E6%8E%A8%E9%80%81%E8%BF%87%E7%A8%8B%E8%AF%B4%E6%98%8E)

为保证推送的及时性及减少流量，行情数据（ticker）和委托深度（depth）这两种数据类型只会在数据发生变化的情况下才会推送数据，交易记录（trades）是推送从上次推送到本次推送产生的增量数据。

[**如何判断连接是否断开**](https://github.com/okcoin-okex/API-docs-OKEx.com#%E5%A6%82%E4%BD%95%E5%88%A4%E6%96%AD%E8%BF%9E%E6%8E%A5%E6%98%AF%E5%90%A6%E6%96%AD%E5%BC%80)

OKEx通过心跳机制解决这个问题。客户端每30秒发送一次心跳数据：{"event":"ping"}，服务器会响应客户端：{"event":"pong"}以此来表明客户端和服务端保持正常连接。如果客户端未接到服务端响应的心跳数据则需要客户端重新建立连接。

[API参考](https://github.com/okcoin-okex/API-docs-OKEx.com#api%E5%8F%82%E8%80%83)

[**合约行情 API**](https://github.com/okcoin-okex/API-docs-OKEx.com#%E5%90%88%E7%BA%A6%E8%A1%8C%E6%83%85-api)

获取OKEx合约行情数据

1. **ok\_sub\_futureusd\_X\_ticker\_Y   订阅合约行情**

websocket.send("{'event':'addChannel','channel':'ok\_sub\_futureusd\_X\_ticker\_Y'}");

① X值为：btc, ltc, eth, etc, bch,eos,xrp,btg  
② Y值为：this\_week, next\_week, quarter 示例

# Request

{'event':'addChannel','channel':'ok\_sub\_futureusd\_btc\_ticker\_this\_week'}

# Response

[

{

"data": {

"limitHigh": "1030.3",

"vol": 276406,

"last": 998.05,

"sell": 998.05,

"buy": 997.61,

"unitAmount": 100,

"hold\_amount": 180178,

"contractId": 20170324034,

"high": 1049.18,

"low": 973.15,

"limitLow": "968.1"

},

"channel": "ok\_sub\_futureusd\_btc\_ticker\_this\_week"

}

]

返回值说明

limitHigh(string):最高买入限制价格

limitLow(string):最低卖出限制价格

vol(double):24小时成交量。 // 对应restapi v3 >合约api>获取ticke信息> volume\_24h （24小时成交量。按张数统计）

sell(double):卖一价格 // 对应restapi v3> 合约api>获取ticker信息> best\_ask ( 卖一价)

buy(double): 买一价格 // 对应restapi v3 > 合约api> 获取ticker信息> best\_bid (买一价)

unitAmount(double):合约价值

hold\_amount(double):当前持仓量 //对应restapi v3>合约api>获取平台总持仓量> amount （总持仓量）

contractId(long):合约ID // 对应restapi v3>合约api>获取ticker信息> instrument\_id ( 合约ID)。

high(double):24小时最高价格 // 对应restapi v3>合约api>获取ticker信息> high\_24h (24小时最高价)

low(double):24小时最低价格 // 对应restapi v3>合约api>获取ticker信息> low\_24h (24小时最低价)

1. **ok\_sub\_futureusd\_X\_kline\_Y\_Z   订阅合约K线数据**

websocket.send("{'event':'addChannel','channel':'ok\_sub\_futureusd\_X\_kline\_Y\_Z'}");

① X值为：btc, ltc, eth, etc, bch,eos,xrp,btg ② Y值为：this\_week, next\_week, quarter  
③ Z值为：1min, 3min, 5min, 15min, 30min, 1hour, 2hour, 4hour, 6hour, 12hour, day, 3day, week 示例

# Request

{'event':'addChannel','channel':'ok\_sub\_futureusd\_btc\_kline\_this\_week\_1min'}

# Response

[

{

"data":[

[

"1490340480000",

"998.96",

"998.96",

"998.96",

"998.96",

"12.0",

"1.201249299271"

]

],

"channel":"ok\_sub\_futureusd\_btc\_kline\_this\_week\_1min"

}

]

返回值说明

[时间 ,开盘价,最高价,最低价,收盘价,成交量(张),成交量(币)]

[string, string, string, string, string, string]

// 对应restapi v3> 获取k线数据>

time 开始时间 open 开盘价 high 最高价 low最低价格 close收盘价格 volume交易量(张) currency\_volume 按币种折算的交易量

1. ok\_sub\_futureusd\_X\_depth\_Y   订阅合约市场深度(200增量数据返回)

websocket.send("{'event':'addChannel','channel':'ok\_sub\_futureusd\_X\_depth\_Y'}");

① X值为：btc, ltc, eth, etc, bch,eos,xrp,btg ② Y值为：this\_week, next\_week, quarter

示例

# Request

{'event':'addChannel','channel':'ok\_sub\_futureusd\_btc\_depth\_this\_week'}

# Response

[

{

"data": {

"timestamp": 1490337551299,

"asks": [

[

"996.72",

"20.0",

"2.0065",

"85.654",

"852.0"

]

],

"bids": [

[

"991.67",

"6.0",

"0.605",

"0.605",

"6.0"

]

},

"channel": "ok\_sub\_futureusd\_btc\_depth\_this\_week"

}

]

返回值说明

timestamp(long): 服务器时间戳

asks(array):卖单深度 数组索引(string) 0 价格, 1 量(张), 2 量(币) 3, 累计量(币)

bids(array):买单深度 数组索引(string) 0 价格, 1 量(张), 2 量(币) 3, 累计量(币)

使用描述:

1，第一次返回全量数据

2，根据接下来数据对第一次返回数据进行，如下操作

删除（量为0时）

修改（价格相同量不同）

增加（价格不存在）

1. ok\_sub\_futureusd\_X\_depth\_Y\_Z   订阅合约市场深度(全量返回)

websocket.send("{'event':'addChannel','channel':'ok\_sub\_futureusd\_X\_depth\_Y\_Z'}");

① X值为：btc, ltc, eth, etc, bch,eos,xrp,btg ② Y值为：this\_week, next\_week, quarter  
③ Z值为：5, 10, 20(获取深度条数)

示例

# Request

{'event':'addChannel','channel':'ok\_sub\_futureusd\_btc\_depth\_this\_week\_20'}

# Response

[

{

"data": {

"timestamp": 1490337551299,

"asks": [

[

"996.72",

"20.0",

"2.0065",

"85.654",

"852.0"

]

],

"bids": [

[

"991.67",

"6.0",

"0.605",

"0.605",

"6.0"

]

},

"channel": "ok\_sub\_futureusd\_btc\_depth\_this\_week\_20"

}

]

返回值说明

timestamp(long): 服务器时间戳

asks(array):卖单深度 数组索引(string) [价格, 量(张), 量(币),累计量(币)]

bids(array):买单深度 数组索引(string) [价格, 量(张), 量(币),累计量(币)]

1. ok\_sub\_futureusd\_X\_trade\_Y   订阅合约交易信息

websocket.send("{'event':'addChannel','channel':'ok\_sub\_futureusd\_X\_trade\_Y'}");

① X值为：btc, ltc, eth, etc, bch,eos,xrp,btg ② Y值为：this\_week, next\_week, quarter

示例

# Request

{'event':'addChannel','channel':'ok\_sub\_futureusd\_btc\_trade\_this\_week'}

# Response

[

{

"data": [

[

"732916869",

"999.49",

"12.0",

"15:25:03",

"ask",

"1.2006"

],

[

"732916871",

"999.49",

"2.0",

"15:25:03",

"ask",

"0.2001"

],

[

"732916899",

"999.49",

"2.0",

"15:25:04",

"ask",

"0.2001"

]

"channel": "ok\_sub\_futureusd\_btc\_trade\_this\_week"

}

]

返回值说明

**[交易序号, 价格, 成交量(张), 时间, 买卖类型，成交量(币-新增)] [string, string, string, string, string, string]**

/// restapi v3> 获取成交数据>  trade\_id 成交id， price 成交价格，qty 成交数量 timestamp  成交时间 side成交方向

1. ok\_sub\_futureusd\_X\_index   订阅合约指数

websocket.send("{'event':'addChannel','channel':'ok\_sub\_futureusd\_X\_index'}");

① X值为：btc, ltc, eth, etc, bch,eos,xrp,btg

示例

# Request

{'event':'addChannel','channel':'ok\_sub\_futureusd\_btc\_index'}

# Response

[

{

"data": {

"timestamp": "1490341322021",

"futureIndex": "998.0"

},

"channel": "ok\_sub\_futureusd\_btc\_index"

}

]

返回值说明

futureIndex(string): 指数 // 对应rest pai v3> 获取指数信息 >index 指数价格

timestamp(string): 时间戳

1. X\_forecast\_price   合约预估交割价格

① X值为：btc, ltc, eth, etc, bch,eos,xrp,btg

示例

# Response

[{

"channel": "btc\_forecast\_price",

"timestamp":"1490341322021",

"data": "998.8"

}]

返回值说明

data(string): 预估交割价格 // 对应rest api v3>获取预估交割价>estimated\_price 预估价格

timestamp(string): 时间戳

操作说明

无需订阅，交割前一小时自动返回

[**合约交易 API**](https://github.com/okcoin-okex/API-docs-OKEx.com#%E5%90%88%E7%BA%A6%E4%BA%A4%E6%98%93-api)

获取OKEx合约交易数据

1. login   登录事件(个人信息推送) ：个人信息推送返回成功后，才可订阅其它频道。

示例

# Request

{"event":"login",

"parameters":{

"api\_key":"xxx",

"sign":"xxx"，

"passphrase":'"xxxx",

"timestamp":""

}

}

# Response

[

{

"channel": "login",

"data": {

"result": true

}

}

]

请求参数

|  |  |
| --- | --- |
| api\_key | 用户申请的apiKey |
| sign | 请求参数的签名，请参考api概述里面验证部分及此文档开端签名示例 |
| passphrase | 申请apikey时所填写 |
| timestamp | 时间戳 |
| **参数名** | **描述** |

示例：

{"event":"login","parameters":{"api\_key":"985d5b66-57ce-40fb-b714-afc0b9787083","passphrase":"123456","timestamp":"1538054050.975","sign":"7L+zFQ+CEgGu5rzCj4+BdV2/uUHGqddA9pI6ztsRRPs="}}

代码示例：

const CryptoJS = require('crypto-js')

const timestamp = ‘’ + Date.now() / 1000

const secret = CryptoJS.enc.Utf8.parse('0FD34e5AF3790366A36AA1128F97')

const sign = CryptoJS.enc.Base64.stringify(CryptoJS.HmacSHA256(timestamp + 'GET' + '/users/self/verify', secret))

console.log(sign)

const request = {

event: 'login',

parameters: {

'api\_key': '63221-af66-4509-941f-634b2c008541',

passphrase: '1234e890',

timestamp: timestamp,

sign: sign

}

}

console.log(JSON.stringify(request))

说明

个人信息推送，个人数据有变化时会自动推送，其它旧的个人数据订阅类型可不订阅，

如:ok\_sub\_futureusd\_trades,ok\_sub\_futureusd\_userinfo,ok\_sub\_futureusd\_positions

1. **ok\_sub\_futureusd\_trades   交易信息**

需登录

示例

# Response

[{

"data":{

amount:1

contract\_id:20170331115,

contract\_name:"LTC0331",

contract\_type:"this\_week",

create\_date:1490583736324,

create\_date\_str:"2017-03-27 11:02:16",

deal\_amount:0,

fee:0,

lever\_rate:20,

orderid:5058491146,

price:0.145,

price\_avg:0,

status:0,

system\_type:0,

type:1,

unit\_amount:10,

user\_id:101

},

"channel":"ok\_sub\_futureusd\_trades"

}

]

返回值说明

amount(double): 委托数量。 // 对应 restapi v3>获取订单信息> order\_qty 数量

contract\_name(string): 合约名称

created\_date(long): 委托时间 // 对应restapi v3> 获取订单信息 create\_at 委托时间

create\_date\_str(string):委托时间字符串

deal\_amount(double): 成交数量 // 对应 restapi v3> 获取订单信息 filled\_qty 成交数量

fee(double): 手续费 // 和 rest v3相同

order\_id(long): 订单ID // 和rest v3相同

price(double): 订单价格 // 和rest v3相同

price\_avg(double): 平均价格。// 和rest v3相同

status(int): 订单状态(0等待成交 1部分成交 2全部成交 -1撤单 4撤单处理中) //和rest v3相同

symbol(string): btc\_usd ltc\_usd eth\_usd etc\_usd bch\_usd

type(int): 订单类型 1：开多 2：开空 3：平多 4：平空。 // 和rest v3相同

unit\_amount(double):合约面值 // 对应rest api v3 > 获取订单信息> contract\_val 合约面值

lever\_rate(double):杠杆倍数 value:10/20 默认10 // 对应rest api v3 >获取订单信息 > leverage 杠杆倍数。

system\_type(int):订单类型 0:普通 1:交割 2:强平 4:全平 5:系统反单

请求参数

|  |  |
| --- | --- |
| api\_key | 用户申请的apiKey |
| sign | 请求参数的签名，请参考api概述里面验证部分及此文档开端签名示例 |
| passphrase | 申请apikey时所填写 |
| timestamp | 时间戳 |
| **参数名** | **描述** |

1. ok\_sub\_futureusd\_userinfo   合约账户信息

需登录

示例

# Response

逐仓信息

[{

"data":{

"balance":10.16491751,

"symbol":"ltc\_usd",

"contracts":[

{

"bond":0.50922987,

"balance":0.50922987,

"profit":0,

"freeze":1.72413792,

"contract\_id":20170331115,

"available":6.51526374

},

{

"bond":0,

"balance":0,

"profit":0,

"freeze":1.64942529,

"contract\_id":20170407135,

"available":6.51526374

},

{

"bond":0,

"balance":0,

"profit":0,

"freeze":0.27609056,

"contract\_id":20170630116,

"available":6.51526374

}

]

},

"channel":"ok\_sub\_futureusd\_userinfo"

}]

全仓信息

[{

"data":{

"balance":0.2567337,

"symbol":"btc\_usd",

"keep\_deposit":0.0025,

"profit\_real":-0.00139596,

"unit\_amount":100

},

"channel":"ok\_sub\_futureusd\_userinfo"

}]

返回值说明

全仓信息

balance(double): 账户余额

symbol(string)：币种

keep\_deposit(double)：保证金。 // 对应rest v3>合约账户信息 > margin 保证金

profit\_real(double)：已实现盈亏 // 对应rest v3>合约账户信息>realized\_pnl 已实现盈亏

unit\_amount(int)：合约价值

逐仓信息

balance(double):账户余额。 // 和rest v3相同

available(double):合约可用 //对应rest v3 >合约账户信息> available\_qty 合约可用

balance(double):合约余额。 //和rest v3相同

bond(double):固定保证金 // 对应rest v3>合约账户信息 >margin\_fixed 冻结的保证金 成交以后仓位所需要

contract\_id(long):合约ID // 对应rest v3>合约账户信息>instrument\_id 合约id

contract\_type(string):合约类别

freeze(double):冻结 //对应 rest v3>合约账户信息 > hold 未成交的冻结

profit(double):已实现盈亏 // 对应restv3>合约账户信息> realized\_pnl 已实现盈亏

unprofit(double):未实现盈亏。 // 对应rest v3>合约账户信息 >unrealized\_pnl 未实现盈亏

rights(double):账户权益。 // 对应 rest v3>合约账户信息 >equity 账户权益

说明

登录后无需订阅，兼容旧的订阅方式，旧的订阅方式相当于登录操作。

Response:

[{"data":{"result":"true"},"channel":"login"}]

1. ok\_sub\_futureusd\_positions   合约持仓信息

需登录

示例

# Response

全仓返回

[{

"data":{

"positions":[

{

"position":"1",

"contract\_name":"BTC0630",

"costprice":"994.89453079",

"bondfreez":"0.0025",

"avgprice":"994.89453079",

"contract\_id":20170630013,

"position\_id":27782857,

"eveningup":"0",

"hold\_amount":"0",

"margin":0,

"realized":0

},

{

"position":"2",

"contract\_name":"BTC0630",

"costprice":"1073.56",

"bondfreez":"0.0025",

"avgprice":"1073.56",

"contract\_id":20170630013,

"position\_id":27782857,

"eveningup":"0",

"hold\_amount":"0",

"margin":0,

"realized":0

}

],

"symbol":"btc\_usd",

"user\_id":101

},

"channel":"ok\_sub\_futureusd\_positions"

}]

逐仓返回

[{

"data":{

"positions":[

{

"position":"1",

"profitreal":"0.0",

"contract\_name":"LTC0407",

"costprice":"0.0",

"bondfreez":"1.64942529",

"forcedprice":"0.0",

"avgprice":"0.0",

"lever\_rate":10,

"fixmargin":0,

"contract\_id":20170407135,

"balance":"0.0",

"position\_id":27864057,

"eveningup":"0.0",

"hold\_amount":"0.0"

},

{

"position":"2",

"profitreal":"0.0",

"contract\_name":"LTC0407",

"costprice":"0.0",

"bondfreez":"1.64942529",

"forcedprice":"0.0",

"avgprice":"0.0",

"lever\_rate":10,

"fixmargin":0,

"contract\_id":20170407135,

"balance":"0.0",

"position\_id":27864057,

"eveningup":"0.0",

"hold\_amount":"0.0"

}

"symbol":"ltc\_usd",

"user\_id":101

}]

返回值说明

全仓说明

position(string): 仓位 1多仓 2空仓

contract\_name(string): 合约名称

costprice(string): 开仓价格

bondfreez(string): 当前合约冻结保证金

avgprice(string): 开仓均价 // 对应rest api v3 >合约持仓信息 > 当position 值是1时此字段对应 v3 中的 long \_avg\_cost ；值为 2 对应short\_avg\_cost

contract\_id(long): 合约id //rest v3>合约持仓信息 > instrument\_id 合约id

position\_id(long): 仓位id

hold\_amount(string): 持仓量。 //对应restapi v3>合约持仓信息>当position值为1时此字段对应v3中的long\_qty 多仓数量;值为2时对应 short\_qty空仓数量.

eveningup(string): 可平仓量。 //对应restapi v3>合约持仓信息>当position值为1时此字段对应v3中的long\_avail\_qty 多仓可平仓;值为2时对应 short\_avail\_qty空仓可平仓.

margin(double): 固定保证金 //对应restapi v3>合约持仓信息>当position值为1时此字段对应v3中的long\_margin 多仓已实现盈亏;值为2时对应 short\_margin

realized(double):已实现盈亏 / /对应restapi v3>合约持仓信息>当position值为1时此字段对应v3中的long\_realized\_pnl 多仓已实现盈亏;值为2时对应 short\_realized\_pnl

逐仓说明

contract\_id(long): 合约id // 对应 rest apiv3 >合约持仓信息> instrument\_id 合约id

contract\_name(string):

avgprice(string): 开仓均价 // 同 全仓说明

balance(string): 合约账户余额

bondfreez(string): 当前合约冻结保证金

costprice(string): 开仓价格

eveningup(string): 可平仓量。 //同全仓说明

forcedprice(string): 强平价格。//对应rest api v3>合约持仓信息>当position值为1时此字段对应v3中的long\_liqui\_price多仓已实现盈亏;值为2时对应 short\_liqui\_price

position(string): 仓位 1多仓 2空仓

profitreal(string): 已实现盈亏 //同全仓说明

fixmargin(double): 固定保证金 //同全仓说明

hold\_amount(string): 持仓量。 //同全仓说明

lever\_rate(double): 杠杆倍数。 // 对应rest api v3>合约持仓信息>当position值为1时此字段对应v3中的long\_lever\_rate多仓已实现盈亏;值为2时对应 short\_lever\_rate

position\_id(long): 仓位id

symbol(string): btc\_usd ltc\_usd eth\_usd etc\_usd bch\_usd eos\_usd xrp\_usd btg\_usd

user\_id(long):用户ID

说明

登录后无需订阅，兼容旧的订阅方式，旧的订阅方式相当于登录操作。

Response:

[{"data":{"result":"true"},"channel":"login"}]