**WebSocket API简介**

WebSocket协议是基于TCP的一种新的网络协议。它实现了客户端与服务器之间在单个 tcp 连接上的全双工通信，由服务器主动发送信息给客户端，减少了频繁的身份验证等不必要的开销。其最大优点有两个：

* 两方请求的 header 数据很小，大概只有2 Bytes。
* 服务器不再是被动的接到客户端的请求后才返回数据，而是有了新数据后主动推送给客户端。

以上 WebSocket 协议带来的优点使得其十分适用于数字货币行情和交易这种实时性强的接口。

WebSocket 只支持行情查询，交易接口将在后续提供。在使用中如果遇到问题，请加技术讨论 QQ 群: 火币网API交流群(5) 778160723，我们将尽力帮您答疑解惑（加群时请注明uid（账户管理中）和编程语言）。

# 请求与订阅说明

### 1. 访问地址

* Pro 站行情请求地址为：wss://api.huobipro.com/ws
* HADAX 站行情请求地址为：wss://api.hadax.com/ws

### 2. 数据压缩

WebSocket API 返回的所有数据都进行了 GZIP 压缩，需要 client 在收到数据之后解压，推荐使用pako。（[【pako】](https://github.com/nodeca/pako) 是一个支持压缩和解压 GZIP 的库）

### 3. WebSocket库

[【ws】](https://github.com/websockets/ws) 是 Node.js 下的 WebSocket 库。

### 4. 心跳

WebSocket API 支持双向心跳，无论是 Server 还是 Client 都可以发起 ping message，对方返回 pong message。

WebSocket Server 发送心跳：

{"ping": 18212558000}

WebSocket Client 应该返回：

{"pong": 18212558000}

注：返回的数据里面的 "pong" 的值为收到的 "ping" 的值 注：WebSocket Client 和 WebSocket Server 建立连接之后，WebSocket Server 每隔 5s（这个频率可能会变化） 会向 WebSocket Client 发起一次心跳，WebSocket Client 忽略心跳2次后，WebSocket Server 将会主动断开连接。

WebSocket Client 发送心跳：

{"ping": 18212553000}

注：发送的 message 里面，"ping" 的值必须为 Long 类型，否则返回错误信息：

{

"ts": 1492420473027,

"status": "error",

"err-code": "bad-request",

"err-msg": "invalid ping"

}

WebSocket Server 会返回：

{"pong": 18212553000}

注：返回的数据里面的 "pong" 的值为收到的 "ping" 的值

### 5. topic格式

订阅数据和请求数据都要使用 topic，topic 的语法如下：

| **topic 类型** | **topic 语法** | **描述** |
| --- | --- | --- |
| KLine | market.$symbol.kline.$period | $period 可选值：{ 1min, 5min, 15min, 30min, 60min, 1day, 1mon, 1week, 1year } |
| Market Depth | market.$symbol.depth.$type | $type 可选值：{ step0, step1, step2, step3, step4, step5 } （合并深度0-5）；step0时，不合并深度 |
| Trade Detail | market.$symbol.trade.detail |  |
| Market Detail | market.$symbol.detail |  |

* $symbol 为币种，可选值： { ethbtc, ltcbtc, etcbtc, bchbtc...... }
* 用户选择“合并深度”时，一定报价精度内的市场挂单将予以合并显示。合并深度仅改变显示方式，不改变实际成交价格。

### 6. 请求数据(req/rep)

请求数据，仅返回一次数据

#### 请求数据的格式

{

"req": "topic to req",

"id": "id generate by client"

}

* "req" 的值为 **topic** ，请参考 "5. topic格式" 的 **topic 格式**

**正确请求数据的例子**

{

"req": "market.btcusdt.kline.1min",

"id": "id10"

}

返回数据的例子：

{

"status": "ok",

"rep": "market.btcusdt.kline.1min",

"tick": [

{

"amount": 1.6206,

"count": 3,

"id": 1494465840,

"open": 9887.00,

"close": 9885.00,

"low": 9885.00,

"high": 9887.00,

"vol": 16021.632026

},

{

"amount": 2.2124,

"count": 6,

"id": 1494465900,

"open": 9885.00,

"close": 9880.00,

"low": 9880.00,

"high": 9885.00,

"vol": 21859.023500

}

]

}

**错误请求数据的例子**

{

"req": "market.invalidsymbo.kline.1min",

"id": "id10"

}

返回的错误信息的例子：

{

"status": "error",

"id": "id10",

"err-code": "bad-request",

"err-msg": "invalid topic market.invalidsymbol.trade.detail",

"ts": 1494483996521

}

### 7. 订阅数据(sub/pub)

#### 订阅数据的格式

成功建立和 WebSocket API 的连接之后，向 Server 发送如下格式的数据来订阅数据：

{

"sub": "topic to sub",

"id": "id generate by client"

}

**正确订阅的例子**

正确订阅：

{

"sub": "market.btcusdt.kline.1min",

"id": "id1"

}

* "sub" 的值为 **topic** ，请参考 "5. topic格式" 的 **topic 格式**

订阅成功返回数据的例子：

{

"id": "id1",

"status": "ok",

"subbed": "market.btcusdt.kline.1min",

"ts": 1489474081631

}

之后每当 KLine 有更新时，client 会收到数据，例子：

{

"ch": "market.btcusdt.kline.1min",

"ts": 1489474082831,

"tick": {

"id": 1489464480,

"amount": 0.0,

"count": 0,

"open": 7962.62,

"close": 7962.62,

"low": 7962.62,

"high": 7962.62,

"vol": 0.0

}

}

tick 说明:

"tick": {

"id": K线id,

"amount": 成交量,

"count": 成交笔数,

"open": 开盘价,

"close": 收盘价,当K线为最晚的一根时，是最新成交价

"low": 最低价,

"high": 最高价,

"vol": 成交额, 即 sum(每一笔成交价 \* 该笔的成交量)

}

**错误订阅的例子**

错误订阅（错误的 symbol）：

{

"sub": "market.invalidsymbol.kline.1min",

"id": "id2"

}

订阅失败返回数据的例子：

{

"id": "id2",

"status": "error",

"err-code": "bad-request",

"err-msg": "invalid topic market.invalidsymbol.kline.1min",

"ts": 1494301904959

}

错误订阅（错误的 topic）：

{

"sub": "market.btcusdt.kline.3min",

"id": "id3"

}

订阅失败返回数据的例子：

{

"id": "id3",

"status": "error",

"err-code": "bad-request",

"err-msg": "invalid topic market.btcusdt.kline.3min",

"ts": 1494310283622

}

### 8. 取消订阅(unsub)

#### 取消订阅的格式

WebSocket Client 订阅数据之后，可以取消订阅，取消订阅之后 WebSocket Server 将不会再发送该 topic 的数据，取消订阅的格式如下：

{

"unsub": "topic to unsub",

"id": "id generate by client"

}

**正确取消订阅的例子**

正确取消订阅的例子：

{

"unsub": "market.btcusdt.trade.detail",

"id": "id4"

}

取消订阅成功返回信息的例子：

{

"id": "id4",

"status": "ok",

"unsubbed": "market.btcusdt.trade.detail",

"ts" 1494326028889

}

**错误取消订阅的例子**

错误取消订阅的例子（取消订阅一个尚未订阅的 topic）：

{

"unsub": "market.btcusdt.trade.detail",

"id": "id5"

}

返回的错误信息的例子

{

"id": "id5",

"status": "error",

"err-code": "bad-request",

"err-msg": "unsub with not subbed topic market.btcusdt.trade.detail",

"ts": 1494326217428

}

错误取消订阅的例子（取消订阅一个不存在的 topic）：

{

"unsub": "not-exists-topic",

"id": "id5"

}

返回的错误信息的例子：

{

"id": "id5",

"status": "error",

"err-code": "bad-request",

"err-msg": "unsub with not subbed topic not-exists-topic",

"ts": 1494326318809

}

# WebSocket API Reference

## 订阅 KLine 数据 market.$symbol.kline.$period

成功建立和 WebSocket API 的连接之后，向 Server 发送如下格式的数据来订阅数据：

{

"sub": "market.$symbol.kline.$period",

"id": "id generate by client"

}

| **参数名称** | **是否必须** | **类型** | **描述** | **默认值** | **取值范围** |
| --- | --- | --- | --- | --- | --- |
| symbol | true | string | 交易对 |  | ethbtc, ltcbtc, etcbtc, bchbtc...... |
| period | true | string | K线周期 |  | 1min, 5min, 15min, 30min, 60min, 1day, 1mon, 1week, 1year |

**正确订阅的例子**

正确订阅

{

"sub": "market.btcusdt.kline.1min",

"id": "id1"

}

订阅成功返回数据的例子

{

"id": "id1",

"status": "ok",

"subbed": "market.btcusdt.kline.1min",

"ts": 1489474081631

}

之后每当 KLine 有更新时，client 会收到数据，例子

{

"ch": "market.btcusdt.kline.1min",

"ts": 1489474082831,

"tick": {

"id": 1489464480,

"amount": 0.0,

"count": 0,

"open": 7962.62,

"close": 7962.62,

"low": 7962.62,

"high": 7962.62,

"vol": 0.0

}

}

tick 说明

"tick": {

"id": K线id,

"amount": 成交量,

"count": 成交笔数,

"open": 开盘价,

"close": 收盘价,当K线为最晚的一根时，是最新成交价

"low": 最低价,

"high": 最高价,

"vol": 成交额, 即 sum(每一笔成交价 \* 该笔的成交量)

}

**错误订阅的例子**

错误订阅(错误的 symbol)

{

"sub": "market.invalidsymbol.kline.1min",

"id": "id2"

}

订阅失败返回数据的例子

{

"id": "id2",

"status": "error",

"err-code": "bad-request",

"err-msg": "invalid topic market.invalidsymbol.kline.1min",

"ts": 1494301904959

}

错误订阅(错误的 topic)

{

"sub": "market.btcusdt.kline.3min",

"id": "id3"

}

订阅失败返回数据的例子

{

"id": "id3",

"status": "error",

"err-code": "bad-request",

"err-msg": "invalid topic market.btcusdt.kline.3min",

"ts": 1494310283622

}

## 请求 KLine 数据 market.$symbol.kline.$period

{

"req": "market.$symbol.kline.$period",

"id": "id generated by client",

"from": "optional, type: long, 2017-07-28T00:00:00+08:00 至 2050-01-01T00:00:00+08:00 之间的时间点，单位：秒",

"to": "optional, type: long, 2017-07-28T00:00:00+08:00 至 2050-01-01T00:00:00+08:00 之间的时间点，单位：秒，必须比 from 大"

}

| **参数名称** | **是否必须** | **类型** | **描述** | **默认值** | **取值范围** |
| --- | --- | --- | --- | --- | --- |
| symbol | true | string | 交易对 |  | btcusdt, ethusdt, ltcusdt, etcusdt, bchusdt, ethbtc, ltcbtc, etcbtc, bchbtc... |
| period | true | string | K线周期 |  | 1min, 5min, 15min, 30min, 60min, 1day, 1mon, 1week, 1year |

[t1, t5] 假设有 t1 ~ t5 的K线.

* from: t1, to: t5, return [t1, t5].
* from: t5, to: t1, which t5 > t1, return [].
* from: t5, return [t5].
* from: t3, return [t3, t5].
* to: t5, return [t1, t5].
* from: t which t3 < t <t4, return [t4, t5].
* to: t which t3 < t <t4, return [t1, t3].
* from: t1 and to: t2, should satisfy 1501171200 < t1 < t2 < 2524579200.

一次最多300条

#### 请求 KLine 数据的例子

{

"req": "market.btcusdt.kline.1min",

"id": "id10"

}

返回数据的例子

{

"rep": "market.btcusdt.kline.1min",

"status": "ok",

"id": "id10",

"tick": [

{

"amount": 17.4805,

"count": 27,

"id": 1494478080,

"open": 10050.00,

"close": 10058.00,

"low": 10050.00,

"high": 10058.00,

"vol": 175798.757708

},

{

"amount": 15.7389,

"count": 28,

"id": 1494478140,

"open": 10058.00,

"close": 10060.00,

"low": 10056.00,

"high": 10065.00,

"vol": 158331.348600

},

// more KLine data here

]

}

## 订阅 Market Depth 数据 market.$symbol.depth.$type

成功建立和 WebSocket API 的连接之后，向 Server 发送如下格式的数据来订阅数据：

{

"sub": "market.$symbol.depth.$type",

"id": "id generated by client"

}

| **参数名称** | **是否必须** | **类型** | **描述** | **默认值** | **取值范围** |
| --- | --- | --- | --- | --- | --- |
| symbol | true | string | 交易对 |  | btcusdt, ethusdt, ltcusdt, etcusdt, bchusdt, ethbtc, ltcbtc, etcbtc, bchbtc... |
| type | true | string | Depth 类型 |  | step0, step1, step2, step3, step4, step5（合并深度0-5）；step0时，不合并深度 |

* 用户选择“合并深度”时，一定报价精度内的市场挂单将予以合并显示。合并深度仅改变显示方式，不改变实际成交价格。

正确订阅例子

{

"sub": "market.btcusdt.depth.step0",

"id": "id1"

}

订阅成功返回数据的例子

{

"id": "id1",

"status": "ok",

"subbed": "market.btcusdt.depth.step0",

"ts": 1489474081631

}

之后每当 depth 有更新时，client 会收到数据，例子

{

"ch": "market.btcusdt.depth.step0",

"ts": 1489474082831,

"tick": {

"bids": [

[9999.3900,0.0098], // [price, amount]

[9992.5947,0.0560],

// more Market Depth data here

]

"asks": [

[10010.9800,0.0099]

[10011.3900,2.0000]

//more data here

]

}

}

tick 说明

"tick": {

"bids": [

[买1价,买1量]

[买2价,买2量]

//more data here

]

"asks": [

[卖1价,卖1量]

[卖2价,卖2量]

//more data here

]

}

## 请求 Market Depth 数据 market.$symbol.depth.$type

{

"req": "market.$symbol.depth.$type",

"id": "id generated by client"

}

#### 请求 Market Depth 数据的例子

{

"req": "market.btcusdt.depth.step0",

"id": "id10"

}

返回数据的例子：

{

"rep": "market.btcusdt.depth.step0",

"status": "ok",

"id": "id10",

"tick": {

"bids": [

[9999.3900,0.0098], // [price, amount]

[9992.5947,0.0560],

// more Market Depth data here

]

"asks": [

[10010.9800,0.0099]

[10011.3900,2.0000]

//more data here

]

}

}

## 订阅 Trade Detail 数据 market.$symbol.trade.detail

{

"sub": "market.$symbol.trade.detail",

"id": "id generated by client"

}

| **参数名称** | **是否必须** | **类型** | **描述** | **默认值** | **取值范围** |
| --- | --- | --- | --- | --- | --- |
| symbol | true | string | 交易对 |  | btcusdt, ethusdt, ltcusdt, etcusdt, bchusdt, ethbtc, ltcbtc, etcbtc, bchbtc... |

正确订阅例子：

{

"sub": "market.btcusdt.trade.detail",

"id": "id1"

}

订阅成功返回数据的例子：

{

"id": "id1",

"status": "ok",

"subbed": "market.btcusdt.trade.detail",

"ts": 1489474081631

}

之后每当 Trade Detail 有更新时，client 会收到数据，例子：

{

"ch": "market.btcusdt.trade.detail",

"ts": 1489474082831,

"data": [

{

"id": 601595424,

"price": 10195.64,

"time": 1494495766,

"amount": 0.2943,

"direction": "buy",

"tradeId": 601595424,

"ts": 1494495766000

},

{

"id": 601595423,

"price": 10195.64,

"time": 1494495711,

"amount": 0.2430,

"direction": "buy",

"tradeId": 601595423,

"ts": 1494495711000

},

// more Trade Detail data here

]

}

}

data 说明：

"data": [

{

"id": 消息ID,

"price": 成交价,

"time": 成交时间,

"amount": 成交量,

"direction": 成交方向,

"tradeId": 成交ID,

"ts": 时间戳

}

]

## 请求 Trade Detail 数据 market.$symbol.trade.detail

{

"req": "market.$symbol.trade.detail",

"id": "id generated by client"

}

* 仅能获取最近 300 个 Trade Detail 数据

#### 请求 Trade Detail 数据的例子

{

"req": "market.btcusdt.trade.detail",

"id": "id11"

}

返回数据的例子：

{

"rep": "market.btcusdt.trade.detail",

"status": "ok",

"id": "id11",

"data": [

{

"id": 601595424,

"price": 10195.64,

"time": 1494495766,

"amount": 0.2943,

"direction": "buy",

"tradeId": 601595424,

"ts": 1494495766000

},

{

"id": 601595423,

"price": 10195.64,

"time": 1494495711,

"amount": 0.2430,

"direction": "buy",

"tradeId": 601595423,

"ts": 1494495711000

},

// more Trade Detail data here

]

}

## 请求 Market Detail 数据 market.$symbol.detail

{

"req": "market.$symbol.detail",

"id": "id generated by client"

}

* 仅返回当前 Market Detail

#### 请求 Market Detail 数据的例子

{

"req": "market.btcusdt.detail",

"id": "id12"

}

返回数据的例子：

{

"rep": "market.btcusdt.detail",

"status": "ok",

"id": "id12",

"tick": {

"amount": 12224.2922,

"open": 9790.52,

"close": 10195.00,

"high": 10300.00,

"ts": 1494496390000,

"id": 1494496390,

"count": 15195,

"low": 9657.00,

"vol": 121906001.754751

}

}

# 错误码

### 错误信息返回格式

{

"id": "id generate by client",

"status": "error",

"err-code": "err-code",

"err-msg": "err-message",

"ts": 1487152091345

}

注："ts" 为错误信息生成的时间戳，单位：毫秒

| **err-msg** | **描述** |
| --- | --- |
| invalid topic market.invalidsymbol.kline.1min | 错误的 symbol |
| invalid topic market.btccny.kline.3min | 错误的 topic |
| unsub with not subbed topic market.btccny.trade.detail | 取消订阅一个尚未订阅的 topic |
| unsub with not subbed topic not-exists-topic | 取消订阅一个不存在的 topic |
| invalid topic market.invalidsymbol.trade.detail | 错误的 symbol |