**米筐实时行情推送**

行情数据在金融数据中占据重要位置，具备高实时性、高稳定性、高存储和高数据吞吐量的特质，因此对任何行情系统来说都有严格的技术要求。

米筐能提供准确、高效、稳定的实时行情数据订阅功能，为上层应用准备了坚实的基础。

# 简介

Ricequant 实时数据的推送方案分为两种：WebSocket接口和Redis部署。

* Redis 部署：需要本地部署，数据库将同步全部数据，用户通过本地redis订阅合约进行推送
* WebSocket 接口：无需部署，用户通过接口订阅合约进行推送

**适用场景**

1. 驱动实盘交易或者模拟交易
2. 若客户已有实时行情，米筐可以作为备份

# 数据种类

米筐提供丰富的行情数据品种，引入在国内金融业务数据的多年治理经验，对多市场金融品种的合约代码进行统一整理，返回统一的行情数据结构。

Ricequant 实时数据的种类包括：

* 中国 A 股：来源交易所
* 场内基金：包括 ETF、LOF，来源交易所
* 可转债：来源交易所
* 中国期货：包括股指、国债、商品期货，来源交易所
* 中国期权：包括 ETF、股指、商品期权，来源交易所
* 国债逆回购：来源交易所
* 现货：包括黄金、铂金、白银等，来源上金所

Ricequant 实时数据的频率包括：

* 提供Level1 tick五档深度行情
* 提供1 分钟级别数据

# 数据推送的优点

相较于rqdata请求数据：

1. 推送会比拉取型api返回实时行情更及时，效率更高
2. 提供 python sdk 和 websocket 网络接口，用户可以使用任意语言接入，语言中性
3. 基于 ricequant 的数据能力和 rqdata 的基础设施，数据准确快速，可靠性高

## websocket优点

1. 比拉取型api更省资源和流量，减轻了服务器的压力
2. 无需部署，通过接口订阅即可（redis需要部署）

适用场景：普通需求均可满足

延时：主要取决于地理位置和网络情况，约为2-300ms

## redis优点

1. 本地部署功能更丰富，同时支持推送和拉取，例如实时和日内tick行情的拉取
2. 可以支持用户本地更高的负载，
3. 不受rqdata流量和连接数的限制
4. 如果客户网络发生中断，由于Redis本身处理了数据同步的相关问题，降低了运维的所需能力

适用场景：需要全量数据

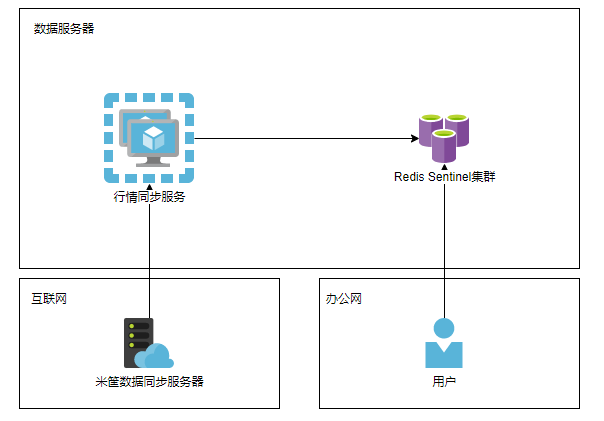
延时：主要取决于地理位置和网络情况，从交易所到客户本机，约为200ms；本地redis耗时约为20ms

# Redis方案介绍

本方案需要部署Redis数据库，客户通过本地Redis获取数据。

## 部署方案

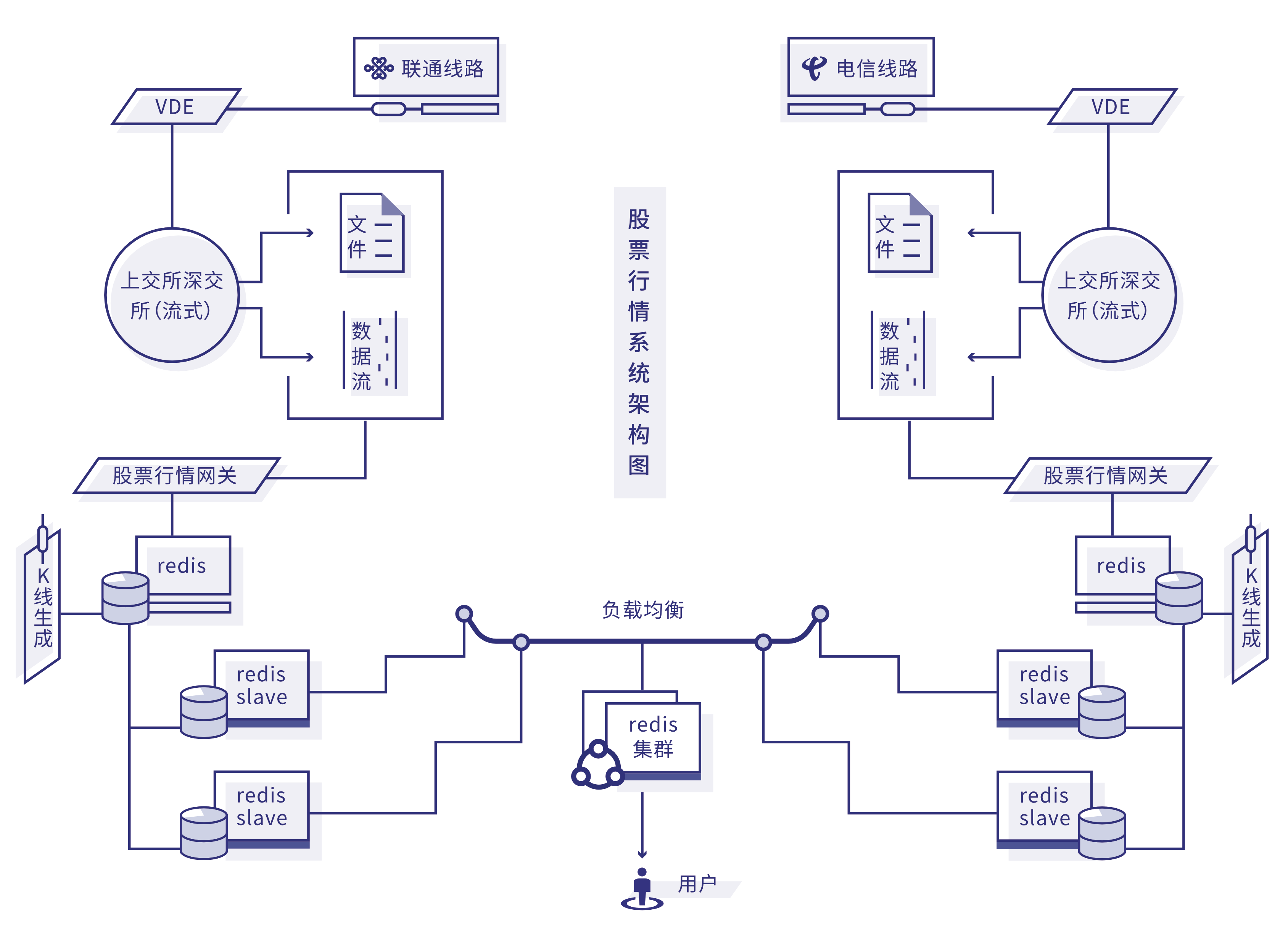
用户可以通过订阅的方式获取实时行情，米筐需要在客户服务器上部署Redis，并通过互联网将实时tick数据同步至客户Redis内，客户可通过订阅Redis的方式获取实时数据。



redis同步架构图

## 技术架构

米筐科技采用多服务器集群分布式处理，使用电信+联通交易所双线路、双路备份处理行情源、双路分发源等方式，可降低行情系统内部延迟至毫秒级别，保障行情系统稳定可靠。



米筐实时行情技术架构

## 示例代码

用户可将以下代码保存为sub\_tick.py，并参照调用范例进行订阅行情。如果需要订阅tick行情，则使用第十三行代码；如果需要订阅分钟行情，则使用第十五行代码，具体见注释。

|  |
| --- |
| import sys  import redis  # 订阅tick或者分钟行情  def subscribe\_tick\_event(redis\_url, order\_book\_ids):      redis\_instance = redis.from\_url(redis\_url)      ps = redis\_instance.pubsub(ignore\_subscribe\_messages=True)      if isinstance(order\_book\_ids, str):          order\_book\_ids = [order\_book\_ids]      # tick行情      obid\_keys = ['tick\_{}'.format(order\_book\_id) for order\_book\_id in order\_book\_ids]      # 分钟行情      # obid\_keys = ['bar\_{}'.format(order\_book\_id) for order\_book\_id in order\_book\_ids]      ps.subscribe(obid\_keys)      for msg in ps.listen():          # 监听redis，有新行情就打印          print("redis msg {}".format(msg))          print("\*" \* 50)  # 从命令行中返回 redis地址以及需要订阅的标的列表  def \_get\_redis\_url\_and\_obids():      args = sys.argv      redis\_url = args[1]      obids = args[2:]      return redis\_url, obids  def main():      redis\_url, obids = \_get\_redis\_url\_and\_obids()      subscribe\_tick\_event(redis\_url, obids)  if \_\_name\_\_ == "\_\_main\_\_":      main() |

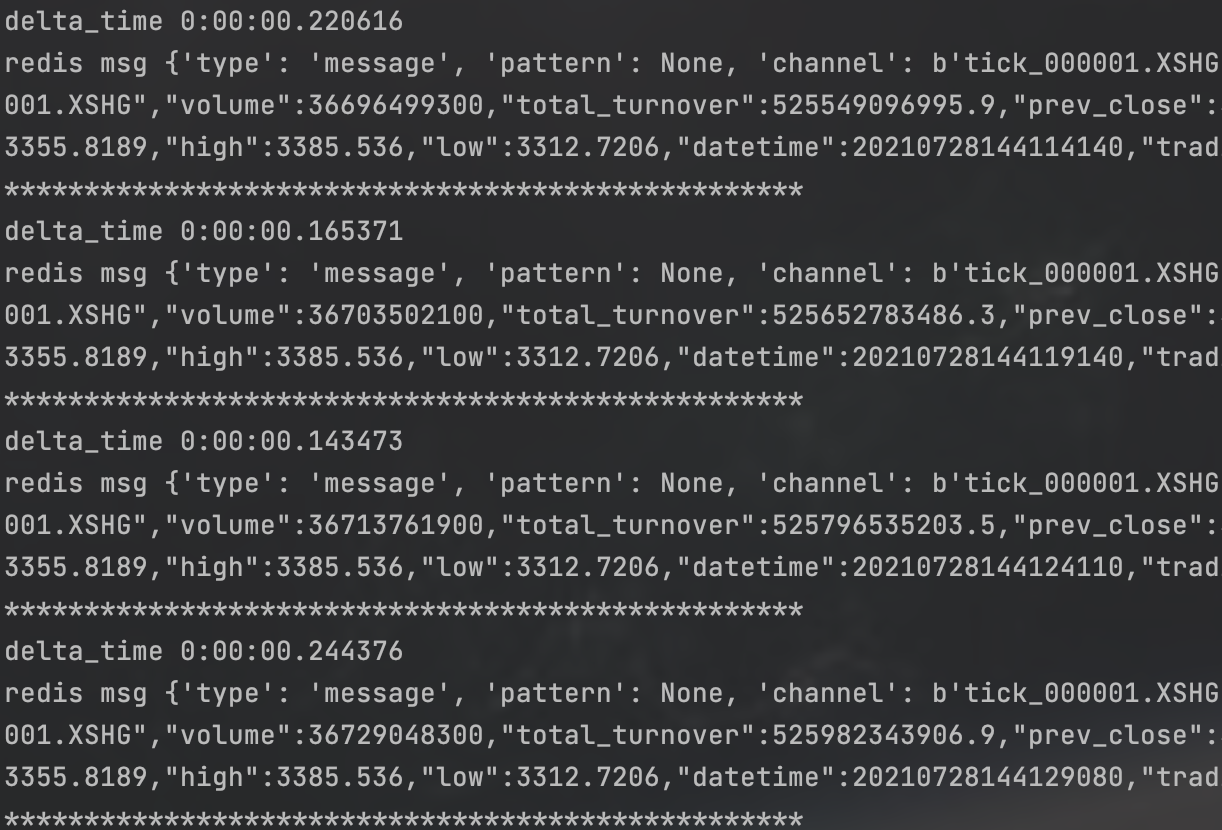
**调用范例：**

在命令行中运行下方代码，需将redis地址替换为客户本地redis的地址，并添加待推送的合约id，即可完成订阅。

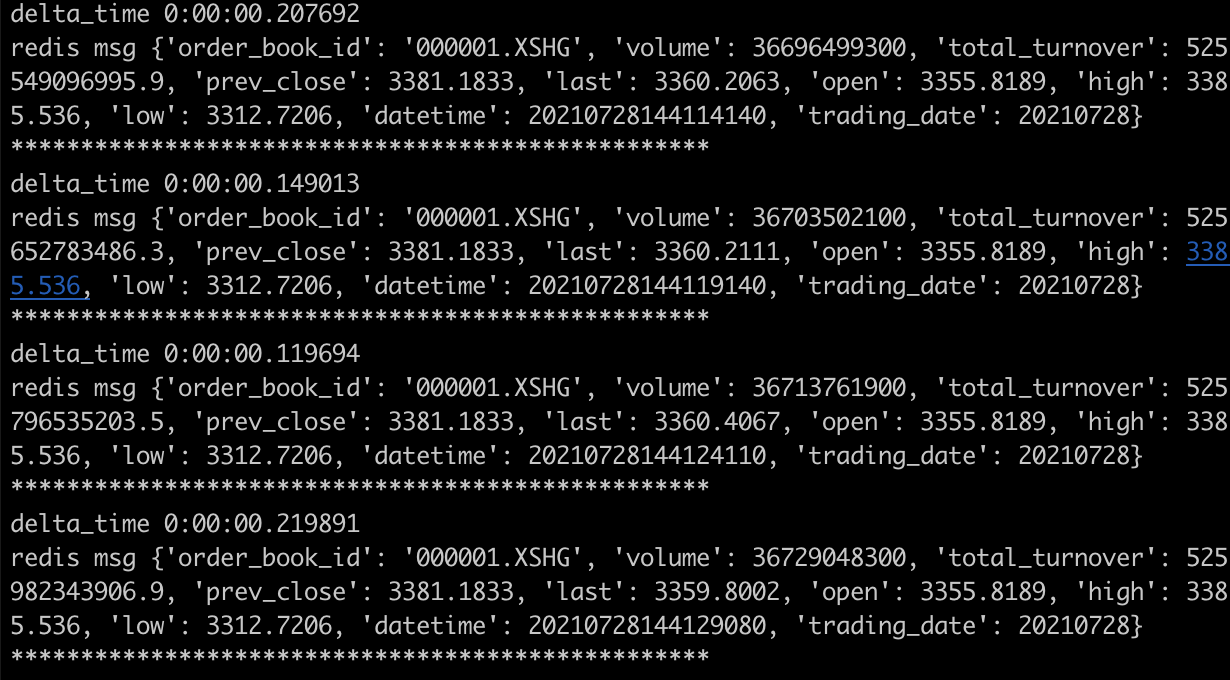
|  |
| --- |
| python sub\_tick.py  redis://192.168.10.69:6379/0 000001.XSHE  python sub\_tick.py  redis://192.168.10.69:6379/0 000001.XSHE 000002.XSHE |

## 性能测试

下图中的delta\_time为交易所时间戳和本地时间戳差距，可以理解为tick行情的时间到客户本机时间。可见大约为200ms左右。



下图中的delta\_time为交易所时间戳和米筐数据库时间戳差距，将上图和下图相减，可得redis同步数据的延时，大约为20ms。



## 机器配置

不同数据种类的CPU和内存的推荐配置如下图，并都需要50GB系统盘、50GB数据盘、CentOS7.6操作系统。若为双机方案，则为两台上述相同的设备。

|  |  |  |
| --- | --- | --- |
| **实时数据** | **CPU** | **内存** |
| 股票 | 2C | 16G |
| 期货 | 1C | 4G |
| 可转债 | 1C | 1G |
| 期权 | 2C | 16G |
| 现货 | 1C | 1G |

同步tick数据需要客户开通机器防火墙访问米筐的同步服务器，地址为：

212.64.46.200:443  
121.5.96.28:36379  
121.5.96.28:36380

# WebSocket方案介绍

米筐提供基于 WebSocket 的实时行情接口，用户可使用任意一种编程语言连接到 Websocket 接口进行使用。

**数据获取流程**

1. 登录米筐 rqdata
2. 订阅合约

## 通过python订阅

米筐提供python包的形式，用户可方便地进行订阅实时行情。

### 安装

输入下面的命令即可安装：

|  |
| --- |
| pip install websocket-client |

### 登录

|  |
| --- |
| import rqdatac  from rqdatac import LiveMarketDataClient  rqdatac.init()  client = LiveMarketDataClient() |

### 订阅

支持订阅多个 tick 和分钟 bar，在米筐 order\_book\_id 前面加上tick\_或bar\_即可。

|  |
| --- |
| client.subscribe(bar或tick\_order\_book\_id) |

示例：

|  |
| --- |
| **client.subscribe('tick\_000001.XSHE') # 订阅一支tick标的**  **client.subscribe('bar\_000001.XSHE') # 订阅分钟行情**  **client.subscribe(['tick\_000001.XSHE', 'tick\_000002.XSHE']) # 订阅多支tick标的** |

### 取消订阅

**与订阅类似，传入已订阅的合约id即可**。

|  |
| --- |
| client.unsubscribe(bar或tick\_order\_book\_id) |

示例：

|  |
| --- |
| **client.unsubscribe('tick\_000002.XSHE')** |

### 监听行情

订阅成功后，用户可监听合约的实时数据。

|  |
| --- |
| client.listen() |

如果订阅了多个合约，可使用循环进行打印：

|  |
| --- |
| for market in client.listen():  print(market) |

### 请求示例

以下是调用 websocket 接口的示例代码：

|  |
| --- |
| import rqdatac  from rqdatac import LiveMarketDataClient    if \_\_name\_\_ == "\_\_main\_\_":  rqdatac.init()  client = LiveMarketDataClient()  # 订阅一支tick标的  client.subscribe('tick\_000001.XSHE')  # 订阅分钟行情  client.subscribe('bar\_000001.XSHE')  # 订阅多支tick标的  client.subscribe(['tick\_000001.XSHE', 'tick\_000002.XSHE'])  # 取消订阅tick标的  client.unsubscribe('tick\_000002.XSHE')  # 检听行情  for market in client.listen():  print(market) |

## 以Websocket接口方式订阅

对于非python的语言，可使用本方法进行订阅。

请求地址：wss://rqdata.ricequant.com/live\_md

### 登录

**输入参数**

| **字段** | **说明** |
| --- | --- |
| action | 操作，auth为登录 |
| license | 米筐数据 rqdata 许可证 |

示例：

{

"action": "auth",

"license": "5tyGjIiAWfs6azz40Ad9TOc1E8SF..."

}

**返回值**

| **字段** | **说明** |
| --- | --- |
| action | 操作响应 |
| permissions | 实时数据权限 |
| quota | 流量限制 |
| request\_id | 请求 ID，用于区分多个订阅 |

示例：

{

    "action":"auth\_reply",

    "permissions":["tick\_futures","bar\_equity","bar\_futures","tick\_equity"],

    "quota":1073741824.0,

    "request\_id":null

}

### 订阅

支持订阅多个 tick 和分钟 bar，在米筐 order\_book\_id 前面加上tick\_或bar\_即可。

**输入参数**

| **字段** | **类型** | **说明** |
| --- | --- | --- |
| action | str | 操作 |
| channels | list | 订阅的合约。以tick\_开头为 tick 数据，以bar\_开头为 bar 数据 |

示例：

{

"action": "subscribe",

"channels": ["bar\_000001.XSHE", "tick\_000002.XSHE"]

}

**返回值**

| **字段** | **说明** |
| --- | --- |
| action | 操作，subsubscribe\_reply为订阅响应 |
| subscribe | 订阅成功的合约数据类型 |
| request\_id | 请求 ID，用于区分多个订阅 |

示例：

{  
    "action": "subscribe\_reply",  
    "subscribed": [  
        "bar\_000001.XSHE"  
    ],  
    "request\_id": null  
}

### 取消订阅

**参数**

| **字段** | **说明** |
| --- | --- |
| action | 操作，unsubscribe为取消订阅 |
| channels | 取消订阅的合约，以tick\_开头为 tick 数据，以bar\_开头为 bar 数据 |

示例：

{

"action": "unsubscribe",

"channels": ["tick\_000001.XSHE"]

}

**返回值**

| **字段** | **说明** |
| --- | --- |
| action | 操作，unsubsubscribe\_reply为取消订阅响应 |
| unsubscribe\_reply | 取消订阅的 channel |
| request\_id | 请求 ID，用于区分多个订阅 |

示例：

{  
    "action": "unsubscribe\_reply",  
    "unsubscribed": [  
        "tick\_000001.XSHE"  
    ],  
    "request\_id": null  
}

## 行情数据样例

**tick 数据：**

{

"datetime": 20210816150003000,

"order\_book\_id": "000001.XSHE",

"prev\_close": 19.89,

"num\_trades": 76258,

"volume": 62705022.0,

"total\_turnover": 1247681171,

"trading\_phase\_code": "T",

"last": 19.95,

"open": 20.09,

"high": 20.2,

"low": 19.57,

"limit\_up": 21.88,

"limit\_down": 17.9,

"ask": [

19.95,

19.96,

19.97,

19.98,

19.99

],

"bid": [

19.94,

19.93,

19.92,

19.91,

19.9

],

"ask\_vol": [

34255.0,

80100.0,

46800.0,

95400.0,

130525.0

],

"bid\_vol": [

35900.0,

7100.0,

6000.0,

29600.0,

28700.0

],

"trading\_date": 20210816,

"channel": "tick\_000001.XSHE",

"action": "feed"

}

**分钟 bar 数据：**

{

"order\_book\_id": "000001.XSHE",

"trading\_date": 20210818,

"open": 20.11,

"high": 20.11,

"low": 20.07,

"close": 20.09,

"volume": 598200.0,

"total\_turnover": 12025148,

"limit\_up": 21.64,

"limit\_down": 17.7,

"datetime": 20210818095900,

"channel": "bar\_000001.XSHE",

"action": "feed"

}