# 行情接口设计

## 概述

### 需求背景

PC端原有的PLS编程接口中有关行情的接口只有两个，并且以JSON作为通信方式，对于做量化的用户来说：

1. 接口种类过少，不符合量化要求；
2. JSON通信方式对于用户来说使用麻烦；

所以需要改善编程接口方便用户使用。

### 需求规格描述

1. 提供多种接口，满足量化需求
2. 在PLS接口上封装python接口，方便用户使用
3. 接口分为两类：订阅接口和拉取接口，支持高频和低频调用，不会对后端服务器造成压力。

### 系统限制

本接口依附于PC客户端的PLS接口。

## 设计思路

接口分为两类：订阅接口和拉取接口。

1. 订阅接口满足高频调用需求，允许用户实时获取股票报价数据。数据来源于PC客户端订阅服务器推送的数据，PC客户端通过缓存向接口提供数据，这样可以避免对服务器造成冲击
2. 拉去接口满足低频调用需求，满足用户一次性获取大量股票数据的需求。数据来源于PC客户端向服务器请求拉取的数据，服务器端将这类数据打包为文件，客户端定时从服务器下载。这样避免在客户端按照通常方式访问批量数据时重复请求服务器。

## 总体设计



有颜色的部分是需要实现的部分；蓝色部分为拉取接口服务，红色部分是订阅接口部分。

拉取接口（对用户来说叫常规接口）

订阅接口（（对用户来说叫热点接口）

对于获取K线部分来说，客户端会有下载历史K线的勾选项，选中后客户端会每天从历史K线服务器下载数据，并和本地缓存以及订阅部分的数据拼接后返回给用户。

## 接口设计

蓝色表示客户端内部接口信息

Python接口是对PLS接口的封装，PLS一次请求回应中只使用单个JSON，不允许多个JSON。Python API在调用时建立socket连接，请求回应后再关闭，即短连接方式。

市场标识market：

HK = 港股 （1 = 港股）

US= 美股 （2 = 美股）

SH= 沪股 （3 = 沪股）

SZ= 深股 （4 = 深股）

HK\_FUTURE = 香港期货 （6 = 新期货）

证券类型stock\_type：

STOCK：正股 1

IDX：指数 2

ETF：ETF基金 3

WARRANT：涡轮牛熊 4

BOND：债券 5

复权类型：

前复权：qfq 1

后复权：hfq 2

不复权： None 3

K线类型：

K\_1M 1分K

K\_5M 5分K

K\_15M 15分K

K\_30M 30分K

K\_60M 60分K

K\_DAY 日K

K\_WEEK 周K

K\_MON 月K

股票代码模式为: 市场+原始代码

“HK.00700”, “SZ.000001”, “US.AAPL”

对于用户来说接口会返回两个值

ret\_code：调用执行返回状态，0为成功，其它为失败

ret\_data：

ret\_code 为成功，ret­­­\_data为实际数据

ret\_code 为失败，ret\_data为错误描述

用户接口分为**常规接口**和**热点接口**：

**常规接口**：

**get\_trading\_days(market, start\_date=None, end\_date=None)，获取交易日期列表**

**get\_stock\_basicinfo(market, stock\_type='STOCK')，获取指定条件股票的基本信息表**

**get\_history\_kline(code, start='', end='', ktype=' K\_DAY, autype='qfq')，获取历史K线**

**get\_autype\_list(code\_list)** **获取复权因子，数据来源为复权因子服务器**

**get\_market\_snapshot(code\_list): 获取指定股票列表中的各个股票的快照信息，最多一次取200只股票，频率限制，数据来源为市场快照服务器**

这些接口主要用来获取股票市场静态和全局的数据，让用户得到股票的基本信息，不允许高频调用。客户端限制频率为5s一次

**热点接口**：用来获取实时数据，可以针对小范围内的股票频繁调用；在调用之前需要将频繁获取的数据订阅注册，

订阅接口：

**subscribe(stock\_code, data\_type)** 订阅注册热点信息，指定股票和订阅的数据类型即可：

**unsubscribe(stock\_code, data\_type)** 退订热点信息

**query\_subscription()** 查询已订阅的热点信息

具体数据类型分为，

逐笔：TICKER

报价：QUOTE

摆盘：ORDER\_BOOK

K线： K\_1M 1分K

K\_5M 5分K

K\_15M 15分K

K\_30M 30分K

K\_60M 60分K

K\_DAY 日K

K\_WEEK 周K

K\_MON 月K

获取数据接口：

**get\_stock\_quote(code\_list)，获取实时报价数据**

**get\_rt\_ticker(code, num)，获取实时逐笔**

**get\_cur\_kline(code, num, ktype='D', autype='qfq') 获取实时K线数据**

**get\_order\_book(code)，获取实时摆盘**

例如，

subscribe(‘HK.00700’, ‘K\_1M’) 订阅腾讯的1分K线

subscribe(‘HK.00700’, ’TICKER’) 订阅腾讯的逐笔

ret, ret\_data = get\_cur\_kline(‘HK.00700’, 5, ‘K\_DAY’) 最近5根分K

ret, ticker = get\_rt\_ticker(‘HK.00700’,10) 持续获取腾讯最近10个逐笔

如果数据不订阅，直接调用热点接口则会返回失败。

订阅的热点数据有额度限制：

**订阅额度表**

|  |  |
| --- | --- |
| 订阅数据 | 订阅额度 |
| K线 | 150 |
| 逐笔 | 5 |
| 报价 | 50 |
| 摆盘 | 50 |

用户需要自己管理额度占用情况，如果需要的数据没有订阅或订阅失败，而去调用热点接口，则会返回失败；用户可以通过**query\_subscription()**来获得当前订阅情况，如果确定某些数据不再需要访问，则调用

**unsubscribe(stock\_code, data\_type)**，来取消订阅，空出订阅额度。例如，不再使用腾讯的逐笔则调用，unsubscribe(‘HK.00700’, ’TICKER’)即可。

**用户在订阅某个数据60s后，才允许退订，否则退订会失败。**

**ret\_code, ret\_data = get\_trading\_days(market, start\_date=None, end\_date=None)**

**功能：取指定市场，某个日期时间段的交易日列表**

**参数：**

market: 市场标识

start\_date: 起始日期; string类型，格式YYYY-MM-DD，仅指定到日级别即可，默认值None表示取去年今日

end\_date: 结束日期; string类型，格式YYYY-MM-DD，仅指定到日级别即可，取默认值None表示取当前日期

**返回：**

ret\_code失败时，ret\_data返回为错误描述字符串；

客户端无数据时，ret\_code为成功，ret\_data返回None

正常情况下，ret\_data为日期列表（每个日期是string类型），如果指定时间段中无交易日，则ret\_data为空列表。

失败情况：

1. 市场标识不合法
2. 起止日期输入不合法
3. PLS接口返回错误

客户端行为：

客户端可以从从节假日列表，生成交易日列表

客户端从交易日列表数据中筛选出符合查询条件的并返回，客户端应该在本地缓存交易日数据。

**ret\_code, ret\_data = get\_stock\_basicinfo(market, stock\_type='STOCK')**

**功能：取符合市场和股票类型条件的股票简要信息**

**参数：**

market: 市场标识, string，例如，HK，US

stock\_type: 证券类型, string, 例如，STOCK，ETF

**返回：**

ret\_code失败时，ret\_data返回为错误描述字符串；

客户端无符合条件数据时，ret\_code为成功，ret\_data返回None

正常情况下，ret\_data为一个dataframe，其中包括：

code：股票代码，string，例如： ”HK.00700”，“US.AAPL”

name：股票名称，string

lot\_size：每手股数，int

stock\_type：股票类型，string，例如： ETF，STOCK

失败情况：

1. 市场或股票类型不合法
2. PLS接口返回错误
3. 无法处理PLS返回的数据

客户端行为：

PLS接口会返回很大的JSON，然后由python API处理

**ret\_code, ret\_data = get\_autype\_list(code\_list)**

**功能：获取复权因子数据**

**参数：**

code\_list: 股票代码列表，例如，HK.00700，US.AAPL

**返回：**

ret\_code失败时，ret\_data返回为错误描述字符串；

客户端无符合条件数据时，ret\_code为成功，ret\_data返回None

正常情况下，ret\_data为一个dataframe，其中包括：

code：股票代码，string，例如： ”HK.00700”，“US.AAPL”

ex\_div\_date：除权除息日，string类型，格式YYYY-MM-DD

split\_ratio拆合股比例 double，例如，对于5股合1股为1/5，对于1股拆5股为5/1

per\_cash\_div：每股派现；double

per\_share\_div\_ratio：每股送股比例； double

per\_share\_trans\_ratio：每股转增股比例； double

allotment\_ratio： 每股配股比例；double

allotment\_price：配股价；double

stk\_spo\_ratio： 增发比例：double

stk\_spo\_price 增发价格：double

forward\_adj\_factorA：前复权因子A；double

forward\_adj\_factorB：前复权因子B；double

backward\_adj\_factorA：后复权因子A；double

backward\_adj\_factorB：后复权因子B；double

返回数据中不一定包含所有codelist中的代码，调用方自己需要检查，哪些股票代码是没有返回复权数据的，未返回复权数据的股票说明客户端没有找到相关信息。

复权价格 = 复权因子A \* 价格 + 复权因子B

失败情况：

1. Codelist中股票代码不合法
2. PLS接口返回错误

客户端行为：

客户端收到对应PLS请求后，到本地缓存文件中找对应数据，对于某只股票找不到数据则跳过即缓存文件中有多少数据就返回多少数据。客户端只用定时更新本地缓存数据即可。

**ret\_code, ret\_data = get\_history\_kline(code, start='', end='', ktype='K\_DAY', autype='qfq')**

**功能： 获取指定股票K线历史数据**

**参数：**

code 股票代码

start 开始时间, string; YYYY-MM-DD 为空时取去当前时间;

end 结束时间, string; YYYY-MM-DD为空时取当前时间;

ktype k线类型，K线类型；string；

K\_1M 1分K

K\_5M 5分K

K\_15M 15分K

K\_30M 30分K

K\_60M 60分K

K\_DAY 日K

K\_WEEK 周K

K\_MON 月K

，默认为K\_DAY

autype 复权类型，string；qfq-前复权，hfq-后复权，None-不复权，默认为qfq

开始结束时间按照闭区间查询，时间查询以k线时间time­\_key作为比较标准。即满足

start<=Time\_key<=end条件的k线作为返回内容，k线时间time\_key的设定标准在返回值中说明

**返回：**

ret\_code失败时，ret\_data返回为错误描述字符串；

客户端无符合条件数据时，ret\_code为成功，返回None

正常情况下返回K线数据为一个DataFrame包含:

code 股票代码；string

time\_key K线时间 string “YYYY-MM-DD HH:mm:ss”

open 开盘价；double

high 最高价；double

close 收盘价；double

low 最低价：double

volume 成交量；long

turnover 成交额；double

对于日K线，time\_key为当日时间具体到日，比如说2016-12-23日的日K，K线时间为” 2016-12-23 00:00:00”

对于分K线，time\_key为当日时间具体到分，例如，

1分K，覆盖时间为9：35：00到9：35：59的分K，K线时间time\_key为”2016-12-23 09:36:00”

5分K，覆盖时间为10：05：00到10：09：59的5分K，K线时间time\_key为”2016-12-23 10:10:00”

15分K，覆盖时间为10：00：00到10：14：59的15分K，K线时间time\_key为”2016-12-23 10:15:00”

30分K，覆盖时间为10：00：00到10：29：59的30分K，K线时间time\_key为” 2016-12-23 10:30:00”

60分K，覆盖时间为10：30：00到11：29：59的60分K，K线时间time\_key为” 2016-12-23 11:30:00”

对于周K线，12月19日到12月25日的周K线，K线时间time\_key为” 2016-12-19 00:00:00”

对于月K线，12月的月K线时间time\_key为” 2016-12-01 00:00:00”，即为当月1日时间

**失败情况:**

1. 股票代码不合法
2. PLS接口返回错误

客户端负责复权计算

客户端下载历史K线文件，分为定时和用户触发两种：

1. 服务器端保证每天17：30，8：00后数据上传更新，用户选择定时下载则默认17：30，8：00下载，用户可以指定每天下载的时间，但是客户端予以时间提示
2. 用户手动要求下载，则立即下载；

**ret\_code, ret\_data = get\_market\_snapshot(code\_list):**

**功能：**一次性获取最近当前股票列表的快照数据（每日变化的信息），该接口对股票个数有限制，一次最多传入200只股票

**参数：**

code\_list: 股票代码列表，例如，HK.00700，US.AAPL

传入的codelist只允许包含1种股票类型。

**返回：**

ret\_code失败时，ret\_data返回为错误描述字符串；

客户端无符合条件数据时，ret\_code为成功，ret\_data返回None

正常情况下，ret\_data为一个dataframe，其中包括：

code 股票代码；string

stock\_type 股票类型 string

data\_date 交易日期(yyyy-MM-dd)；string

data\_time 交易时间'HH:mm:ss'；string

last\_price 最新价格；double

open\_price 今日开盘价；double

high\_price 最高价格；double

low\_price 最低价格；double

prev\_close\_price 昨收盘价格；double

volume 成交数量； long

turnover 成交金额；double

turnover\_rate 换手率；double

amplitude 振幅；double //目前协议中没有直接返回

suspension 是否停牌(True表示停牌)；bool

xs\_listing\_date 上市日期 (yyyy-MM-dd)；string

返回DataFrame，包含上述字段

返回数据量不一定与codelist长度相等， 用户需要自己判断

**调用频率限制：**

5s一次

**失败情况:**

1. Codelist中股票代码不合法
2. Codelist长度超过规定数量
3. PLS接口返回错误，包括频率超限
4. 传入的codelist包含多种股票类型

客户端行为：

客户端收到PLS请求后，判断访问频率是否超过限制，如果超过则返回失败，并带上对应错误码；如果没有超限，则转换为对市场快照服务器的请求，发送给市场快照服务器。

一下就是有关热点接口的说明：

**ret\_code,ret\_data= subscribe(stock\_code, data\_type)**

**功能：**订阅注册热点信息，指定股票和订阅的数据类型即可：

**参数：**

stock\_code: 需要订阅的股票代码

data\_type: 需要订阅的数据类型

逐笔：TICKER

报价：QUOTE

摆盘：ORDER\_BOOK

K线： K\_1M 1分K

K\_5M 5分K

K\_15M 15分K

K\_30M 30分K

K\_60M 60分K

K\_DAY 日K

K\_WEEK 周K

K\_MON 月K

**返回：**

ret\_code失败时，ret\_data返回为错误描述字符串；

ret\_code为成功，ret\_data返回None

如果指定内容已订阅，则直接返回成功

**失败情况:**

1. 股票代码不合法，不存在
2. 数据类型不合法
3. 订阅额度已满，参考**订阅额度表**
4. PLS返回错误

**ret\_code,ret\_data = unsubscribe(stock\_code, data\_type)** 退订热点信息

**功能：**退订注册热点信息，指定股票和订阅的数据类型即可：

参数：

stock\_code: 需要退订的股票代码

data\_type: 需要退订的数据类型

**返回：**

ret\_code失败时，ret\_data返回为错误描述字符串；

ret\_code为成功，ret\_data返回None

如果指定内容已退订，则直接返回成功

**失败情况:**

1. 股票代码不合法，不存在
2. 数据类型不合法
3. 内容订阅后不超过60s，就退订
4. PLS返回错误

**ret\_data = query\_subscription()** 查询已订阅的热点信息

**返回：**

字典类型，已订阅类型为主键，值为订阅该类型的股票，例如

{ ‘QUOTE’: [‘HK.00700’, ‘SZ.000001’]

‘TICKER’: [‘HK.00700’]

‘K\_1M’: [‘HK.00700’]

//无股票订阅摆盘和其它类型分K

}

**失败情况:**

PLS返回错误

**ret\_code, ret\_data = get\_stock\_quote(code\_list)**

**功能：**获取订阅股票报价的实时数据，有订阅要求限制

**参数：**

code\_list: 股票代码列表，例如，HK.00700，US.AAPL

传入的codelist只允许包含1种股票类型的股票。

必须确保code\_list中的股票均订阅成功后才能够执行

**返回：**

ret\_code失败时，ret\_data返回为错误描述字符串；

客户端无符合条件数据时，ret\_code为成功，返回None

正常情况下，ret\_data为一个dataframe，其结构和**get\_market\_snapshot**一样

**失败情况:**

1. Codelist中股票代码不合法
2. Codelist包含未对QUOTE类型订阅的股票
3. PLS接口返回错误
4. 传入的codelist包含多种股票类型

**ret\_code, ret\_data = get\_rt\_ticker(code, , num=500)**

**功能： 获取指定股票的实时逐笔。取最近num个逐笔，**

**参数：**

code: 股票代码，例如，HK.00700，US.AAPL

num: 最近ticker个数(有最大个数限制，最近500个）

**返回：**

ret\_code失败时，ret\_data为错误描述字符串；

客户端无符合条件数据时，ret为成功，ret\_data返回None

通常情况下，返回DataFrame，DataFrame每一行是一个逐笔记录，包含：

Stock\_code

Sequence 逐笔序号

date 成交时间；string

price 成交价格；double

volume 成交数量（股数）；int

turnover 成交金额；double

deal\_type 逐笔的买卖类型；int

deal\_type:

TT\_BUY 0x0001 // 订单买

TT\_ASK 0x0002 // 订单卖

TT\_N 0x0100 // Normal - blank

TT\_X 0x0200 // 碎股成交 + 人工成交 + 经纪内部直接成交

TT\_D 0x0400 // 碎股成交

TT\_R 0x0800 // reject

TT\_M 0x1000 // 碎股成交 + 人工成交 + 跨经纪成交

TT\_Y 0x2000 // 自动对盘成交 + 经纪内部直接成交

TT\_P 0x4000 // 开盘前成交 + 跨经纪成交

TT\_U 0x8000 // 竞价成交

返回的逐笔记录个数不一定与num指定的相等，客户端只会返回自己有的数据。

失败情况：

1. Code不合法
2. Code是未对TICKER类型订阅的股票
3. PLS返回错误

客户端行为：

客户端收到PLS请求后，判断code是否合法并且已经订阅过TICKER类型数据，如果没有则返回失败，如果有则返回数据。

客户端将会缓存订阅后推送的逐笔，拉取500个逐笔，按照用户请求返回。

**ret\_code, ret\_data = get\_cur\_kline(code, num, ktype='K\_DAY', autype='qfq')**

**功能： 实时获取指定股票最近num个K线数据，最多1000根**

**参数：**

Code 股票代码

ktype k线类型，和get\_history\_kline一样

autype 复权类型，string；qfq-前复权，hfq-后复权，None-不复权，默认为qfq

**返回：**

ret\_code失败时，ret\_data为错误描述字符串；

客户端无符合条件数据时，ret为成功，ret\_data返回None

通常情况下，返回DataFrame，DataFrame内容和get\_history\_kline一样

失败情况：

1. Code不合法
2. 该股票未对指定K线类型订阅
3. PLS返回错误

客户端行为：

客户端收到PLS请求后，判断code是否合法并且已经订阅过K线类型数据，如果没有则返回失败，如果有则返回数据。

订阅后会触发拉取一次K线

对于实时K线数据最多取最近1000根

**ret\_code, ret\_data = get\_order\_book(code)**

**功能：**获取实时摆盘数据

**参数：**

code: 股票代码，例如，HK.00700，US.AAPL

**返回：**

ret\_code失败时，ret\_data为错误描述字符串；

客户端无符合条件数据时，ret为成功，ret\_data返回None

通常情况下，返回字典

{‘stock\_code’: stock\_code

‘Ask’:[ (ask\_price1, ask\_volume1), (ask\_price2, ask\_volume2),…]

‘bid’: [ (bid\_price1, bid\_volume1), (bid\_price2, bid\_volume2),…]

}

失败情况：

1. Code不合法
2. 该股票未对ORDER\_BOOK类型订阅
3. 客户端内部或网络错误

客户端行为：

客户端收到PLS请求后，判断code是否合法并且已经订阅过ORDER\_BOOK数据，如果没有则返回失败，如果有则返回数据。

## 后端服务

### 历史K线数据服务

使用腾讯云对象存储。每天盘后上传当天k线文件，客户端下载。

总体采用增量更新方式，即每天将当天产生的新K线数据打包为文件上传。

#### 文件索引

服务端提供一个索引列表，告诉客户端服务器上的文件信息，客户端先下载索引文件，确定内容后下载。

索引文件内容结构：

版本号

K线文件名1，文件url，文件大小，校验码，更新时间

K线文件名2，文件url，文件大小，校验码，更新时间

K线文件名3，文件url，文件大小，校验码，更新时间

K线文件名4，文件url，文件大小，校验码，更新时间

K线文件名5，文件url，文件大小，校验码，更新时间

...

具体格式采用JSON：

{

“Version”: 1

“MetaList”: [{"UpdateTime": "2017-02-07 11:19:31", "CheckCode": “1173728886”, "FileSize": 58452875, "FileName": "us\_1min\_2017\_02\_07.zip"},{"UpdateTime": "2017-02-07 11:14:09", "CheckCode": “1533185716”, "FileSize": 391166, "FileName": "us\_day\_2017\_02\_07.zip"} ]

}

客户端根据索引内容，做以下判断：

1. 哪些已有文件有更新（根据当前校验码和之前的索引文件中对应校验码）；
2. 哪些文件为新增文件（根据本地文件列表、之前的索引文件以及最新的索引文件）
3. 哪些为不再存在的文件

确定更新文件和新增文件后，根据对应K线文件的url下载。同时根据下载情况更新本地元数据。

#### K线数据的组织



由于服务器是按日期做数据聚合形成增量文件，而从用户使用角度客户端会按照股票为主键聚合数据，所以客户端在下载后会需要将增量文件中每一只股票的数据提取出，再分别归类到各个股票中。

增量下载方式难以应对客户端指定股票全量下载数据

目前K线总量有200G，按照增量更新，客户端第一次下载时会有较大流量要求。客户端在执行下载时应该有优化策略：

1. 让客户指定下载的时间段，K线类型，和市场；
2. 由于月K、周K、日K、分K，数据大小依次增大，可以下载可以分为两个通道，一个通道下较小的数据，另外一个下较大的数据。
3. 提示用户磁盘占用空间。
4. 客户端应该维持一个已下载内容列表，如果用户停止下载后，又开始下载，默认只下载本地没有的文件；当然可以允许用户强制全量刷新本地数据，但是必须要用户勾选并二次确认。

对于增量下载的数据需要分类命名打包：

格式为“市场\_K线类型\_日期”，市场：us、cn、hk

目前只提供最近三个月分K，2015年开始的日K

|  |  |  |
| --- | --- | --- |
|  | 2015年起始文件增量  以2016-12-23为例 | 文件包 |
| 日K | us\_day\_2016\_12\_23 | us \_day\_2016\_12\_23.zip |
| 1分 | us \_1min \_2016\_12\_23 | us \_1min \_arc.zip  us \_1min \_2016\_12\_23.zip |
| 5分 | us \_5min \_2016\_12\_23 | us \_xmin \_2016\_12\_23.zip |
| 15分 | us \_15min \_2016\_12\_23 |
| 30分 | us \_30min \_2016\_12\_23 |
| 60分 | us \_60min \_2016\_12\_23 |
| 周K | 2016-12-23当周的周一为文件名  us\_week \_2016\_12\_19 | 打包为  us\_week \_2016\_12\_19.zip |
| 月K | 2016-12-23当月的1号为文件名  us\_month \_2016\_12\_1 | 打包为  us\_month\_2016\_12\_1.zip |

从对于日K和分K每天都会新增一个文件包；

对于周K和月K每天都会更新一个文件包，每周或每月产生新增一个文件包，由于月K一个月只会产生并多次更新一个包，所以客户端是利用校验码和更新时间来判断本地数据是否是最新的。

对于每个K线文件采用二级存储格式：

1. 文件包含多个K线数据组
2. 一组K线数据记录有多个K线记录构成
3. 组长度为K线数据组PB二进制结构的长度



版本号为uint16

message KlineItem

{

optional uint64 stock\_id = 1;

optional uint32 time\_key = 2;

optional uint64 open\_price = 3; //价格\*1000000000

optional uint64 close\_price = 4; //价格\*1000000000

optional uint64 highest\_price = 5; //价格\*1000000000

optional uint64 lowest\_price = 6; //价格\*1000000000

optional uint64 volume = 7;

optional uint64 turnover = 8;

optional uint32 pe = 9;

optional uint32 turnover\_rate = 10;

}

message KlineGroup

{

optional uint32 kline\_type = 1;

repeated KlineItem kline\_record = 2;

}

对于服务端：

1．将K线记录分组，并用PB打包为K线记录组；

3. 写入版本号和K线记录组个数后，然后将K线记录组的长度和PB二进制写入文件

对于客户端：

1. 从文件头中读出版本号和组个数，设定计数器为0
2. 如果计数器数值小于组个数，继续下一步；否则退出
3. 读出K线组长度，依据组长度读出K线组数据，用PB解包读出K线记录，计数器加1，重复第2步

对K线文件来说解码错误的情况有：

1. 版本号异常
2. 索引中的包校验码和下载包的校验码对不上
3. 文件解包错误，解包密码错误
4. K线文件中组个数为0
5. Pb解包错误
6. 实际组个数比文件头中的组个数小

客户端应该对上述错误做检查处理并记录日志。

#### K线文件包URL路径结构

K线文件包的部分URL路径结构为：

/市场/k线包类型/k线文件包

市场：us,cn,hk

k线包类型: day, week ,month ,1min,xmin

#### 文件下载行为验证

客户端下载索引文件的URL以及索引文件中的URL有可能暴露给用户，用户就有可能依据URL大量下载，而不受控制。风险如下：

1. 用户绕过客户端下载K线数据并尝试解码；
2. 即使用户无法解码，用户扩散导致大量下载，产生大量付费流量，提高成本；

对于风险1，可以对下载的文件包加密（zip和tar两种包方式，linux都有方式加密，但是客户端不一定都方便，客户端需要确定下）

对于风险2，可以使用带认证的URL：索引中URL加上认证信息才能够得到下载K线文件的权限。（调研腾讯云cos的认证方式，腾讯云COS会提供有security\_id和security\_key,这两个信息，由这两个信息生成签名，从而保证认证过程）

### 复权因子服务

每天定时上传复权因子文件，客户端定时下载，更新本地数据。使用腾讯云对象存储。

服务端每天上传全量的复权因子文件，覆盖原来的文件。

和K线一样，复权因子下载也需要索引和复权因子文件

索引文件：

版本号

复权因子文件文件名1，文件url，文件大小，校验码，更新时间

复权因子文件文件名1，文件url，文件大小，校验码，更新时间

复权文件，即实际复权数据：

1. 复权因子文件按月存储，每月公司行动放在一个文件；
2. 当月文件每天更新；
3. 加密和验证方式与k线一致。

文件名，举例

cn\_ertfile\_2016\_12打包为cn\_ertfile\_2016\_12.zip

hk\_ertfile\_2016\_12打包为hk\_ertfile\_2016\_12.zip

us\_ertfile\_2016\_12打包为us\_ertfile\_2016\_12.zip

url路径

K线文件包的部分URL路径结构为：

/市场/复权文件包

市场：us,cn,hk

 版本号为uint16

和K线一样采用二级存储结构：

1. 文件由多个除权记录组组成；
2. 每个除权记录组含有多个除权记录

文件打包和解包过程和K线一样

单个除权记录如下：

目前沿用除权信息协议中除权记录的定义：

公司行动标记：

enum ErtFlag

{

NONE = 0x00;

SPLIT = 0x01; //拆股

JOIN = 0x02; //合股

BONUS\_STK = 0x04; //送股

INTOSHARES = 0x08; //转增股

ALLOT = 0x10; //配股

ADD = 0x20; //增发股

DIVIDEND = 0x40; //有现金分红

SPECIALDIVIDEND = 0x80; //有现金分红

}

除权记录：

message ExRight\_Item

{

optional uint64 stock\_id = 1;

optional uint32 ex\_date = 2; // 除权除息日期, 例如20160615

optional uint32 ert\_flag = 3; // 公司行动类型组合，ErtFlag

//拆股(eg. 1拆5，Base为1，ERT为5)

optional uint32 split\_base = 4;

optional uint32 split\_ert = 5;

//合股(eg. 50合1，Base为50，ERT为1)

optional uint32 join\_base = 6;

optional uint32 join\_ert = 7;

//送股(eg. 10送3, Base为10,ERT为3)

optional uint32 bonus\_stk\_base = 8;

optional uint32 bonus\_stk\_ert = 9;

//配股(eg. 10送2, 配股价为6.3元, Base为10, ERT为2, Price为6300)

optional uint32 allot\_base = 10;

optional uint32 allot\_ert = 11;

optional uint64 allot\_price = 12;

//转增股(跟送股类似)

optional uint32 into\_shr\_base = 13;

optional uint32 into\_ert = 14;

//增发(跟配股类似)

optional uint32 stk\_add\_base = 15;

optional uint32 stk\_add\_ert = 16;

optional uint64 stk\_add\_price = 17;

// 现金分红(eg. 每10股派现0.5元，Base为10, Amount为500)

optional uint32 dividend\_base = 18;

optional uint64 dividend\_amount = 19;

// 特别股息

optional uint32 dividend\_special\_base = 20;

optional uint32 dividend\_special\_amount = 21;

// result\_self

optional sint64 fwd\_factor\_a = 22;

optional sint64 fwd\_factor\_b = 23;

optional sint64 bwd\_factor\_a = 24;

optional sint64 bwd\_factor\_b = 25;

optional string sc\_txt = 26; // 简体中文文本描述

optional string tc\_txt = 27; // 繁体中文文本描述

}

message ExRightGroup

{

repeated ExRight\_Item exright\_record = 1;

}

客户端下载复权也主要是覆盖型操作，所以一定要验证下载的文件合法、可用后才可以覆盖原来的文件，即新下载的文件确认正确后覆盖先前下载的文件：

1．下载文件，解压得到摘要文件和数据文件

2. 检查索引校验码和数据文件实际校验码是否相同

3. 读出数据文件中，组个数，组校验标识，并PB解码正确

4. 得到缓存文件与前一版本对比。对于全量的除权信息来说：

1） 每版本比前一版本文件大小或数据记录数，应该是非严格递增的，

2）而且历史除权信息应该有很多相同的记录，

客户端可以对以上做检查，发现server端上传数据过程或下载出现错误，导致原先正确的文件被新的错误文件覆盖。

#### 市场快照服务

让客户端拉取市场快照服务。

一只股票的快照记录结构如下：

message StockSnapshot

{

optional uint64 stock\_id = 1; // 股票id

optional uint32 stock\_code = 2; // 股票代码

optional uint32 instrument\_type = 3; // 股票类型

optional uint32 market\_code = 4; // 属于哪个市场

// 价格相关

optional uint64 nominal\_price = 5; // 当前价格 价格\*1000000000

optional uint64 last\_close\_price = 6; // 昨收价格 价格\*1000000000

optional uint64 open\_price; = 7 ; //今日开盘价 价格\*1000000000

optional uint64 update\_time = 8; //当前数据时间

optional uint64 suspend\_flag = 9; // 是否停牌。0表示没停，1表示停

optional uint64 listing\_status = 10; // 上市状态

optional uint64 listing\_date = 11; // 上市日期

// 成交统计信息

optional uint64 shares\_traded = 12; // 成交量

optional uint64 turnover = 13; // 成交额 成交额 \* 1000

optional int64 highest\_ price = 14; // 最高价 价格\*1000000000

optional int64 lowest\_ price = 15; // 最低价 价格\*1000000000

optional int32 turnover\_ratio = 16; // 换手率

}

**请求回应协议**：

message SnapshotDataReq

{

repeated uint64 stock\_id\_list = 1; // 待查询的股票ID列表

}

// 返回股票信息

Message StockItem

{

Optional int32 ret = 1; //是否找到快照记录，0为成功找到，-1为stock\_id不存在，1为确实没有数据

Optional uint64 stock \_id = 2;

optional StockSnapshot record = 3; //快照记录

}

message SnapshotDataRsp

{

Optional int32 result = 1; // 查询结果，0:查询成功，其它:失败

repeated StockItem StockItem = 2; // 股票详细信息列表

}

客户端通过SnapshotData \_Req发送需要查询的股票列表，服务器查询后返回股票快照的列表

如果服务端数据正常，无论有没有查询到指定股票的数据，返回时result均为0；

如果出现某些股票确实没有数据则在Stock\_Item中的ret标明未查到数据；

服务端出问题时，result不为0，此时仍然可以在Stock\_Res返回确定有效的数数据。客户端此时应该记录服务器出错的信息。

客户端向服务器请求的股票数量限制为200，由客户端实施，服务端做好对用户请求的监控上报