**uniQuot行情服务器通信接口说明**

V1.0.0.0

中期信息

交易软件开发部

2014.12.30

**版本历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 发布日期 | 备注 |
| V1.0.0.0 |  | 2014/12/31 | 初稿   1. 第一版 |

目录

[一． 概述 4](#_Toc409111364)

[1. 使用范围 4](#_Toc409111365)

[2. 网络通信方式 4](#_Toc409111366)

[二． 业务流程 5](#_Toc409111367)

[1. 客户端连接并登录uniQuot 5](#_Toc409111368)

[2. 客户端发送心跳 5](#_Toc409111369)

[3. 客户端查询主力合约ID和全部合约ID 5](#_Toc409111370)

[4. 客户端订阅/退订实时行情 5](#_Toc409111371)

[5. 客户端接收实时行情 5](#_Toc409111372)

[6. 客户端查询历史数据 5](#_Toc409111373)

[三． 数据包结构 6](#_Toc409111374)

[1. 数据包遵循统一格式 6](#_Toc409111375)

[2. 数据包结构 6](#_Toc409111376)

[四． 业务通讯协议 7](#_Toc409111377)

[1. 登录 7](#_Toc409111378)

[2. 心跳 7](#_Toc409111379)

[3. 查询全部合约列表 7](#_Toc409111380)

[4. 查询主力合约列表 8](#_Toc409111381)

[5. 实时行情订阅 8](#_Toc409111382)

[6. 实时行情退订请求 8](#_Toc409111383)

[7. 实时行情推送 9](#_Toc409111384)

[8. K线数据查询接口 10](#_Toc409111385)

[9. 分时线数据查询请求 11](#_Toc409111386)

[10. 波动率查询接口 12](#_Toc409111387)

[五． 命令字定义 13](#_Toc409111388)

[六． 错误类型 15](#_Toc409111389)

# 概述

## 使用范围

uniQuot统一提供期货、期货期权、证券、股票期权、期权波动率的行情数据。包括实时行情和历史行情。可应用于PC端和移动端行情/交易客户端。

本文档描述了uniQuot的通信接口。

## 网络通信方式

uniQuot使用基于socket 套接字的TCP通信方式。

# 业务流程

## 客户端连接并登录uniQuot

1. 客户端以TCP方式连接uniQuot
2. 客户端发送登录请求包，指定登录用户名、密码、客户端名称、客户端序列号及客户端版本号
3. uniQuot返回登录请求结果

## 客户端发送心跳

1. 客户端每8秒一次向uniQuot发送心跳包
2. uniQuot返回心跳返回包
3. uniQuot在24秒内未收到客户端的心跳包，即断开与该客户端的连接

## 客户端查询主力合约ID和全部合约ID

1. 客户端可以查询主力合约ID列表和全部合约ID列表

## 客户端订阅/退订实时行情

1. 客户端发送订阅实时行情请求包，在保内指定要订阅的合约ID。可以一次订阅多个合约的实时行情，不同合约ID用分号分隔。
2. 客户端发送退订实时行情请求包，退订保内指定的合约ID
3. 每个客户端同时可以订阅的合约数量是有限制的，具体限制数量请咨询运营部门

## 客户端接收实时行情

1. uniQuot实时向客户端推送所订阅的合约的最新的实时行情信息

## 客户端查询历史数据

1. 客户端可以查询历史数据，包括：

* 日内分时数据
* 历史K线数据，含1分钟、5分钟、15分钟、60分钟、日线、周线、月线数据
* 期权标的物合约的波动率

1. 每种历史数据可回溯查询的时间不一样，具体限制请咨询运营部门
2. 每次查询返回的数据量有限制。具体限制请咨询运营部门

# 数据包结构

## 数据包遵循统一格式

1. 包头 + 数据 + 包尾
2. 数据包结构按4字节对齐

## 数据包结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 数据项 | | 数据类型 | 字节长度 | 备注 |
| 包头 | PkgHead | headid | unsigned int | 4 | 恒为0x7cab5caa |
| len | int | 4 | 包体长度，不含包头包尾 |
| compressflag | int | 4 | 包体压缩标识，目前无效 |
| prioritylevel | int | 4 | 优先级，目前无效 |
| cmdid | unsigned int | 4 | 命令字 |
| moduleid | unsigned int | 4 | 模块标志，目前无效 |
| seq | unsigned int | 4 | 包的序列号 |
| subseq | unsigned int | 4 | 包的子序列号 |
| userdata1 | unsigned int | 4 | 自定义字 |
| userdata2 | unsigned int | 4 | 自定义字 |
| userdata3 | unsigned int | 4 | 自定义字 |
| userdata4 | unsigned int | 4 | 自定义字 |
| crc32 | unsigned int | 4 | CRC校验字 |
| 包体 | PkgBody |  |  | len | 包内容，可以没有 |
| 包尾 | PkgTail | tailid | unsigned int | 4 | 恒为0xba55ca81 |

# 业务通讯协议

## 登录

1. 登录请求

|  |  |  |
| --- | --- | --- |
| PkgHead | cmdid | Cmd\_uniQuot\_Login\_Req |
| PkgBody |  | Stru\_uniQuotLogin数据 |

struct Stru\_uniQuotLoginReq

{

char UserName[32]; //用户名

char PassWord[32]; //密码

char ClientName[16]; //客户端名称

char ClientLicense[16]; //客户端许可证号

char ClientVersion[16]; //客户端版本号

};

1. 登录返回

|  |  |  |
| --- | --- | --- |
| PkgHead | cmdid | Cmd\_uniQuot\_Login\_Rsp |
| seq | 请求包的seq |
| userdata1 | 成功：0  失败：-1 |
| PkgBody | 字符串 | 成功：空  失败：错误信息 |

## 心跳

1. 心跳请求

|  |  |  |
| --- | --- | --- |
| PkgHead | cmdid | Cmd\_uniQuot\_HEARTBEAT |
| PkgBody |  | 空 |

1. 心跳请求返回

|  |  |  |
| --- | --- | --- |
| PkgHead | cmdid | Cmd\_uniQuot\_HEARTBEATRSP |
| seq | 请求包的seq |
| PkgBody |  | 空 |

## 查询全部合约列表

1. 查询全部合约列表请求

|  |  |  |  |
| --- | --- | --- | --- |
| PkgHead | cmdid | Cmd\_uniQuot\_QueryAllInstrumentID\_Req | |
| PkgBody |  | 空 |

1. 查询全部合约列表返回

|  |  |  |
| --- | --- | --- |
| PkgHead | cmdid | Cmd\_uniQuot\_QueryAllInstrumentID\_Rsp |
| seq | 请求包的seq |
| PkgBody | 字符串 | 全部合约ID组成的字符串，不同合约ID之间用半角分号【;】分隔 |

## 查询主力合约列表

1. 查询主力合约列表请求

|  |  |  |
| --- | --- | --- |
| PkgHead | cmdid | Cmd\_uniQuot\_QueryMainInstrumentID\_Req |
| PkgBody |  | 空 |

1. 查询主力合约列表返回

|  |  |  |
| --- | --- | --- |
| PkgHead | cmdid | Cmd\_uniQuot\_QueryMainInstrumentID\_Rsp |
| seq | 请求包的seq |
| PkgBody | 字符串 | 主力合约ID组成的字符串，不同合约ID之间用半角分号【;】分隔 |

## 实时行情订阅

1. 实时行情订阅请求

|  |  |  |
| --- | --- | --- |
| PkgHead | cmdid | Cmd\_uniQuot\_Subscribe\_Req |
| PkgBody |  | 合约ID字符串   1. 可以一次订阅多个合约，不同合约之间用【;】分隔 2. 每个合约ID=【交易所代码:合约代码】，如上交所的000342为sse:000342；深交所的000342为szse:000342；中金所IF1505为CFFEX:IF1505 |

1. 实时行情订阅返回

|  |  |  |
| --- | --- | --- |
| PkgHead | cmdid | Cmd\_uniQuot\_Subscribe\_Rsp |
| seq | 请求包的seq |
| userdata1 | 成功：0  失败：错误信息 |
| PkgBody |  | 空 |

## 实时行情退订请求

1. 实时行情退订请求

|  |  |  |
| --- | --- | --- |
| PkgHead | cmdid | Cmd\_uniQuot\_Unsubscribe\_Req |
| PkgBody |  | 合约ID字符串   1. 可以一次订阅多个合约，不同合约之间用【;】分隔   每个合约ID=【交易所代码:合约代码】，如上交所的000342为sse:000342；深交所的000342为szse:000342；中金所IF1505为CFFEX:IF1505 |

1. 实时行情退订返回

|  |  |  |
| --- | --- | --- |
| PkgHead | cmdid | Cmd\_uniQuot\_Unsubscribe\_Rsp |
| seq | 请求包的seq |
| userdata1 | 成功：0  失败：错误信息 |
| PkgBody |  | 空 |

## 实时行情推送

1. 股票实时行情推送

|  |  |  |
| --- | --- | --- |
| PkgHead | cmdid | Cmd\_uniQuot\_RealStock\_Push |
| PkgBody |  | Stru\_uniQuotRealStockData |

struct Stru\_uniQuotRealStockData

{

char ExchangeID[8]; //交易所代码

char InstrumentID[16]; //合约代码

double LastPrice; //最新价

int Volume; //成交量

double OpenPrice; //开盘价

double BidPrice1; //申买价一

int BidVolume1; //申买量一

double AskPrice1; //申卖价一

int AskVolume1; //申卖量一

double BidPrice2; //申买价二

int BidVolume2; //申买量二

double AskPrice2; //申卖价二

int AskVolume2; //申卖量二

double BidPrice3; //申买价三

int BidVolume3; //申买量三

double AskPrice3; //申卖价三

int AskVolume3; //申卖量三

double BidPrice4; //申买价四

int BidVolume4; //申买量四

double AskPrice4; //申卖价四

int AskVolume4; //申卖量四

double BidPrice5; //申买价五

int BidVolume5; //申买量五

double AskPrice5; //申卖价五

int AskVolume5; //申卖量五

unsigned int UpdateTime; //更新时间，utc秒数

int updateTimeMS; //更新时间的毫秒数

};

1. 期货实时行情推送

|  |  |  |
| --- | --- | --- |
| PkgHead | cmdid | Cmd\_uniQuot\_RealFuture\_Push |
| PkgBody |  | Stru\_uniQuotRealFutureData |

struct Stru\_uniQuotRealFutureData

{

char ExchangeID[8]; //交易所代码

char InstrumentID[16]; //合约代码

double LastPrice; //最新价

int Volume; //成交量

double OpenPrice; //开盘价

double HighestPrice; //最高价

double LowestPrice; //最低价

int OpenInterest; //持仓量

double Turnover; //成交金额

double BidPrice1; //申买价一

int BidVolume1; //申买量一

double AskPrice1; //申卖价一

int AskVolume1; //申卖量一

unsigned int UpdateTime; //更新时间，utc秒数

int updateTimeMS; //更新时间的毫秒数

};

1. 备注：

* 开盘价、收盘价、结算价、昨收盘、昨结算请查询日K线数据；
* 最高价、最低价请先查询日K线数据然后用实时最新价自行更新；

## K线数据查询接口

1. K线数据请求

|  |  |  |
| --- | --- | --- |
|  | cmdid | Cmd\_uniQuot\_QueryKLine\_Req |
| PkgBody |  | Stru\_KLineQuery |

enum EnumPhrase

{

PHRASE\_1MIN = 1,

PHRASE\_5MIN = 5,

PHRASE\_15MIN = 15,

PHRASE\_30MIN = 30,

PHRASE\_60MIN = 60,

PHRASE\_DAY = 24\*60,

PHRASE\_WEEK = 7\*24\*60,

PHRASE\_MONTH = 31\*24\*60,

PHRASE\_USERDEFINE = 0

};

struct Stru\_KLineQuery

{

char ExchangeID[8]; //交易所代码

char InstrumentID[16]; //合约代码

int StartTime; //请求的开始时间utc秒数

int EndTime; //请求的结束时间utc秒数

int Cycle; //K线周期分钟数。参考EnumPhrase定义

int DataType; //1-请求K线数据，2-请求分时线数据

};

1. K线数据返回

|  |  |  |
| --- | --- | --- |
|  | cmdid | Cmd\_uniQuot\_QueryKLine\_Rsp |
| seq | 请求包的seq |
| userdata1 | 成功：0  失败：<0, 错误信息 |
| PkgBody |  | 成功：N\*sizeof(Stru\_KLineBase)  失败：NULL |

struct Stru\_KLineBase //k线数据结构，分钟到月线都是这个数据结构

{

unsigned int Time; //时间，utc秒数

double OpenPrice; //开盘价

double HighPrice; //最高价

double LowPrice; //最低价

double ClosePrice; //收盘价

double Turnover; //成交金额

int Volume; //成交量

int OpenInterest; //持仓量

} ;

## 分时线数据查询请求

1. 分时数据查询请求

|  |  |  |
| --- | --- | --- |
|  | cmdid | Cmd\_uniQuot\_QueryTimeLine\_Req |
| PkgBody |  | Stru\_KLineQuery数据(参考上一节) |

1. 分时线数据返回

|  |  |  |
| --- | --- | --- |
|  | cmdid | Cmd\_uniQuot\_QueryTimeLine\_Rsp |
| seq | 请求包的seq |
| userdata1 | 成功：0  失败：<0,错误信息 |
| PkgBody |  | 成功：N\*sizeof(Stru\_TLineBase)  失败：NULL |

struct Stru\_TLineBase //分时数据结构

{

unsigned int Time; //时间，utc秒数

double ClosePrice; //收盘价

double Turnover; //成交金额

int Volume; //成交量

int OpenInterest; //持仓量

};

## 波动率查询接口

1. 波动率请求

|  |  |  |
| --- | --- | --- |
|  | cmdid | Cmd\_uniQuot\_QueryVolatilityData\_Req |
| PkgBody |  | Stru\_VolatilityQuery结构体 |

struct Stru\_VolatilityQuery

{

char ExchangeID[8]; //交易所代码

char InstrumentID[16]; //合约代码

int TradingDay; //交易日utc秒数

};

1. 波动率数据返回

|  |  |  |
| --- | --- | --- |
|  | cmdid | Cmd\_uniQuot\_QueryVolatilityData\_Rsp |
| seq | 请求包的seq |
| userdata1 | 成功：0  失败：<0, 错误信息 |
| PkgBody |  | 成功：Stru\_Volatility  失败：NULL |

struct Stru\_Volatility

{

char ExchangeID[8]; //交易所代码

char InstrumentID[16]; //合约代码

int TradingDay; //交易日utc秒数

double Volatility20; //20天波动率

double Volatility60; //60天波动率

};

# 命令字定义

Cmd\_uniQuot\_Login\_Req

Cmd\_uniQuot\_Login\_Rsp

Cmd\_uniQuot\_HEARTBEAT

Cmd\_uniQuot\_HEARTBEATRSP

Cmd\_uniQuot\_QueryAllInstrumentID\_Req

Cmd\_uniQuot\_QueryAllInstrumentID\_Rsp

Cmd\_uniQuot\_QueryMainInstrumentID\_Req

Cmd\_uniQuot\_QueryMainInstrumentID\_Rsp

Cmd\_uniQuot\_Subscribe\_Req

Cmd\_uniQuot\_Subscribe\_Rsp

Cmd\_uniQuot\_Unsubscribe\_Req

Cmd\_uniQuot\_Unsubscribe\_Rsp

Cmd\_uniQuot\_RealStock\_Push

Cmd\_uniQuot\_RealFuture\_Push

Cmd\_uniQuot\_QueryKLine\_Req

Cmd\_uniQuot\_QueryKLine\_Rsp

Cmd\_uniQuot\_QueryTimeLine\_Req

Cmd\_uniQuot\_QueryTimeLine\_Rsp

Cmd\_uniQuot\_QueryVolatilityData\_Req

Cmd\_uniQuot\_QueryVolatilityData\_Rsp

#define Cmd\_uniQuot\_HEARTBEAT      0x7fff0001      //心跳命令  
#define Cmd\_uniQuot\_HEARTBEATRSP      0x7fff0002      //心跳答复

#define CMDID\_BASE\_HIWORD\_QUOT 0x003F //行情高位CMDID

#define Cmd\_uniQuot\_Login\_Req MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x000C)

#define Cmd\_uniQuot\_Login\_Req MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x000D)

#define Cmd\_uniQuot\_QueryAllInstrumentID\_Req MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0010)

#define Cmd\_uniQuot\_QueryAllInstrumentID\_Rsp MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0011)

#define Cmd\_uniQuot\_QueryMainInstrumentID\_Req MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0012)

#define Cmd\_uniQuot\_QueryMainInstrumentID\_Rsp MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0013F)

#define Cmd\_uniQuot\_Subscribe\_Req MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0020)

#define Cmd\_uniQuot\_Subscribe\_Rsp MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0021)

#define Cmd\_uniQuot\_Unsubscribe\_Req MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0022)

#define Cmd\_uniQuot\_Unsubscribe\_Rsp MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0023)

#define Cmd\_uniQuot\_RealStock\_Push MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0030)

#define Cmd\_uniQuot\_RealFuture\_Push MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0031)

#define Cmd\_uniQuot\_QueryKLine\_Req MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0040)

#define Cmd\_uniQuot\_QueryKLine\_Rsp MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0041)

#define Cmd\_uniQuot\_QueryTimeLine\_Req MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0042)

#define Cmd\_uniQuot\_QueryTimeLine\_Rsp MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0043)

#define Cmd\_uniQuot\_QueryVolatilityData\_Req MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0044)

#define Cmd\_uniQuot\_QueryVolatilityData\_Rsp MAKE\_CMDID(CMDID\_BASE\_HIWORD\_QUOT,0x0045)

# 错误类型

//公用故障

#define CF\_ERROR\_COMMON\_BASE 0x0000

//登陆错误

#define CF\_ERROR\_LOGIN\_BASE 0x0050

//合约错误

#define CF\_ERROR\_INSTRUMENT\_BASE 0x0060

//输入错误

#define CF\_ERROR\_COMMON\_INPUT\_PARAM MAKE\_ERROR\_CODE(CF\_ERROR\_COMMON\_BASE,0x0001)

//密码错误

#define CF\_ERROR\_LOGIN\_PASS\_INVALIDATE MAKE\_ERROR\_CODE(CF\_ERROR\_LOGIN\_BASE,0x0001)

//不存在用户

#define CF\_ERROR\_LOGIN\_USER\_NOT\_EXIST MAKE\_ERROR\_CODE(CF\_ERROR\_LOGIN\_BASE,0x0003)

//用户未登录

#define CF\_ERROR\_LOGIN\_QUOTSERVER\_ERROR MAKE\_ERROR\_CODE(CF\_ERROR\_LOGIN\_BASE,0x0010)

//没有合约

#define CF\_ERROR\_NO\_INSTRUMENTS MAKE\_ERROR\_CODE(CF\_ERROR\_INSTRUMENT\_BASE,0x0001)