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

1. 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。

2. 网络通信方式

uniQuot 使用基于 socket 套接字的 TCP 通信方式。

二. 业务流程

1. 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

2. 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

3. 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

4. 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

5. 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

6. 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

(2) 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
(3) 每次查询返回的数据量有限制。具体限制请咨询运营部门

三.　数据包结构

1. 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

2. 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

四. 业务通讯协议

1. 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
};
```

(2) 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

2. 心跳

(1) 心跳请求

PkgHead	cmdid	Cmd_uniQuot_HEARTBEAT
PkgBody		空

(2) 心跳请求返回

PkgHead	cmdid	Cmd_uniQuot_HEARTBEATRSP
	seq	请求包的 seq
PkgBody		空

3. 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

(2) 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

4. 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

(2) 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

5. 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

(2) 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

6. 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

		3. 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

(2) 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

7. 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
};
```

(2) 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
};
```

(3) 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；

8. 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
};
```

(2) 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
} ;
```

9. 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)

(2) 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
};
```

10. 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
};
```

(2) 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
};
```

五.　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)
```

六.　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)
```