易盛极速行情

API 使用手册

文档标识

文档名称	**易盛极速行情 API 使用手册**
版本号	**V1.1.5**
简称	**Esunny Extreme Quote（EEQ）**

文档修订历史

API 版本	日期	描述	修订者
EEQ V1.1.1.2	2019/09/20	第一版	蔡德宏
EEQ V1.1.1.3	2019/10/23	1. 增加修改密码功能； 2. 增加 UDP 收快照；	高理
EEQ V1.1.1.4	2019/11/08	1. 修正修改密码 bug； 2. 增加用户登出功能； 3. 增加强制退出功能；	张昌崇
EEQ V1.1.1.5	2020/03/11	1. 增加主备接入点； 2. 增加 LME 行情； 3. 增加交易所配置查询；	高理

一、API 简介

1、Api 介绍

EsunnyQuoteApi 是易盛基于 C++实现的用来为开发者提供高速行情的 API 接口，主要功能包括：实时行情推送、历史行情查询、合约查询、交易日查询等。行情种类包括：逐笔，切片，K 线，交易日数据等。

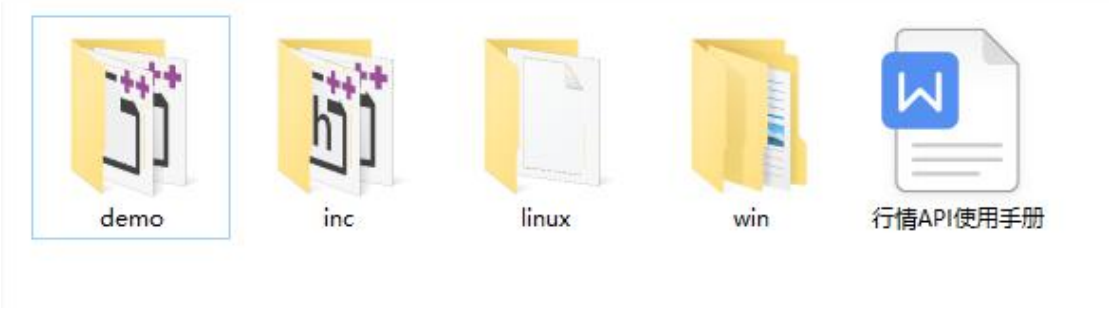
API 由调用接口类和回调接口类两部分组成，分别为 CEsunnyQuoteApi 和 CEsunnyQuoteSpi，CEsunnyQuoteApi 负责发送请求，CEsunnyQuoteSpi 负责响应应答。

目前提供 TCP 和 UDP，两种行情快照订阅模式，但不能同时使用，只能使用其中一种（通过 Init 的参数 serverType 控制），UDP 模式更快些，如果是数讯机房托管客户，建议使用 UDP 模式，详情请看范例。

2、适用平台

1）linux　　64 位
2）windows　32 位

3、目录结构



1）inc c++头文件
2）Linux 64 位动态库
3）Win 32 位动态库
4）Demo 示例程序

4、行情范围

1）内网用户目前支持 NYMEX、COMEX、CBOT 部分品种的主力合约高速低延时行情(提供逐笔快照)，品种如下，其他品种和交易所后续增加。

NYMEX ： CL、NG、HO

COMEX ： GC、HG、SI

CBOT ： ZS、ZC、ZM、ZL、ZW

2）互联网用户支持内盘(SHFE\CFFEX\DCE\CZCE\INE)和外盘(HKEX\NYMEX\COMEX\CBOT\CME)的全品种切片行情，频率为 4 笔/秒。

3）其他交易所和行情内容后续加入，如：LME,SGX,ICE 等。

5、接入方式

首先需要向易盛公司申请账号和权限，再通过以下 2 种方式接入：

1）易盛上海数讯机房内网接入点

接入点	接入类型	IP（数讯机房）	Port	备注
107	Front	192.168.99.107	6100	
108	Front	192.168.99.108	6100	

2）互联网接入点

可向易盛行情客服申请。

二、API 接口说明

1、连接登陆

1）创建 Api 实例对象

```
static CEsunnyQuoteApi *CreateEsunnyQuoteApi();
//释放 Api 对象
virtual void Release() = 0;

//获取当前 Api 版本信息
virtual const char* GetVersion () = 0;
//获取当前连接服务器类型：实时或历史
virtual QUOTE_SERVER_TYPE GetServerType() = 0;
```

2）注册服务器地址

```
//注册 Fens 前置地址（可注册多个），
通过 Fens 获取可用行情服务器集合，Fens 自动选择最优行情服务器
//@param ip  认证服务器 IP 地址
//@param port 认证服务器端口号
virtual void RegisterFens(const char* ip, int port) = 0;

//注册 Front 前置地址（可注册多个），通过 Front 直接接入行情服务器
//@param ip  服务器 IP 地址
//@param port 服务器端口号
virtual void RegisterFront(const char* ip, int port) = 0;
```

3）注册回调

```
void RegisterSpi(CEsunnyQuoteSpi *pSpi);
```

此方法进行回调注册以使数据用过用户创建的回调对象正确返回，参数即为用户回调对象。

4）初始化

```
//初始化,开始连接行情服务器
//@param serverType 服务器类型：实时，历史和逐笔快照
//@param waittime 等待超时时间：单位：秒
////////////////////////////////////////////////////////////////////////
///服务器类型(汇聚层)
////////////////////////////////////////////////////////////////////////
enum QUOTE_SERVER_TYPE
{
    // 提供实时服务（TCP，切片）
    QST_REALTIME = 0,
    // 提供历史服务（TCP，历史）
    QST_HISTORY = 1,
    // 提供 TCP 逐笔（只能内网接入使用）
```

```
    QST_TICKSNAP = 2,
    // 提供 UDP 逐笔（只能内网接入使用，不能跨机房）
    QST_UDPTICK = 3,
};
virtual void Init(QUOTE_SERVER_TYPE serverType = QST_REALTIME, int waittime = 1) = 0;
```

5）用户登陆登出请求

```
//登录请求
//@param login 登录对象
//@param requestID 请求 ID
//@返回值 0:成功
virtual int ReqQuoteLogin(ESunnyQuoteLogin *login, int requestID) = 0;
```

用户通过此方法登陆到服务器，参数一见 EsunnyQuoteApiStruct 对此结构定义，第二个参数表示一个请求 ID，这个 ID 和回调的应答 ID 一一对应。

```
//登出请求
//@param userid 用户id
//@param requestID 请求ID
//@返回值 0:成功
virtual int ReqQuoteLogout(TEsunnyQuoteID userid, int requestID) = 0;
```

6）修改密码

```
//修改密码
//@param req 修改密码对象
//@param requestID 请求 ID
//@返回值 0:成功
virtual int ReqChangePassword(ESunnyChangePassword *req, int requestID) = 0;
```

7）交易所信息

```
//取得交易所字符串
//@param exchangeEnum 交易所枚举值
//@返回值 返回空字符串则没有找到
virtual const char* GetExchangeString(const EXCHANGE_TYPE_ID exchangeEnum) = 0;

//取得交易所枚举值
//@param eti 交易所字符串（交易所列表从 OnRtnExchangeList 中获取）
//@返回值 返回 EXCHANGE_TYPE_ID::ETI_UNKNOWN 则没有找到
virtual EXCHANGE_TYPE_ID GetExchangeEnum(const char* exchangeID) = 0;
```

2、TCP 行情订阅

1）初始化连接 Init()；注：serverType 不为 QST_UDPTICK 即为 TCP

2）登录 ReqQuoteLogin;

3）订阅合约快照 SubscribeMarketData，**合约信息从** OnRspQryInstrument **中获取**

```
//订阅快照行情
//@param ppInstrumentID 合约对象集合（交易所+合约 ID）
//@param nCount 合约数量
//@返回值 0:成功
virtual int SubscribeMarketData(ESunnyQuoteSpecificInstrumentField *ppInstrumentID[], int
nCount) = 0;

//取消订阅快照行情
//@param ppInstrumentID 合约对象集合（交易所+合约 ID）
//@param nCount 合约数量
//@返回值 0:成功
virtual int UnSubscribeMarketData(ESunnyQuoteSpecificInstrumentField *ppInstrumentID[],
int nCount) = 0;

//订阅行情 KLine
//@param ppInstrumentID 合约对象集合（交易所+合约 ID）
//@param nCount 合约数量
//@返回值 0:成功
virtual int SubscribeKLineData(ESunnyQuoteSpecificInstrumentField *ppInstrumentID[], int
nCount, KLINE_TYPE_ID klineType = KLINE_TYPE_ID::KTI_MIN) = 0;

//取消订阅行情 KLine
//@param ppInstrumentID 合约对象集合（交易所+合约 ID）
//@param nCount 合约数量
//@返回值 0:成功
virtual int UnSubscribeKLineData(ESunnyQuoteSpecificInstrumentField *ppInstrumentID[],
int nCount, KLINE_TYPE_ID klineType = KLINE_TYPE_ID::KTI_MIN) = 0;
```

参数一含义见 EsunnyQuoteApiStruct 对此结构定义，参数二表示订阅合约的数量。

3、UDP 行情订阅

!!! 必须托管在易盛数讯机房内，才能收 udp 快照行情!!!

1）初始化连接 `Init(QUOTE_SERVER_TYPE::QST_UDPTICK);`
2）登录 `ReqQuoteLogin;`
3）先订阅 udp 通道信息 `SubscribeUdpInfo`，**udp 通道信息从** `OnRtnUdpInfoData` **中获取**
4）再订阅合约快照 `SubscribeMarketData`，**合约信息从** `OnRspQryInstrument` **中获取**
5）最后启动 UDP 工作 `StartUdpWork`

```
//订阅 UDP 信息，在 OnRtnUdpInfoData 中调用该函数
//客户端根据 udpinfo 中的 exchange 和 products 来判断，是否订阅
//@param udpinfo UDP 信息
//@返回值 0:成功
virtual int SubscribeUdpInfo(const ESunnyUdpInfoField& udpinfo) = 0;
```

```
//订阅快照行情
//@param ppInstrumentID 合约对象集合（交易所+合约 ID）
//@param nCount 合约数量
//@返回值 0:成功
virtual int SubscribeMarketData(ESunnyQuoteSpecificInstrumentField *ppInstrumentID[], int
nCount) = 0;

//取消订阅快照行情
//@param ppInstrumentID 合约对象集合（交易所+合约 ID）
//@param nCount 合约数量
//@返回值 0:成功
virtual int UnSubscribeMarketData(ESunnyQuoteSpecificInstrumentField *ppInstrumentID[],
int nCount) = 0;

//启动 UDP 工作
//@param interface_addr 本地接收 udp 网卡信息（网卡 ip 或网卡设备名，根据本地 ip 设置）
//@返回值 0:成功
virtual int StartUdpWork(const char* interface_addr = nullptr) = 0;
```

4、行情查询

```
//查询合约
//@param req 查询对象（根据参数值不同可查询所有合约，指定交易所合约，主力合约等）
//@param requestID 请求 ID
//@返回值 0:成功
virtual int ReqQryInstrument(ESunnyQuoteReqQryInstrument *req, int requestID) = 0;

//查询历史快照行情（根据客户权限，可查询不同交易所不同年份的行情）
//@param req 查询对象
//@param requestID 请求 ID
//@返回值 0:成功
virtual int ReqQrySnapshotQuote(ESunnyQuoteReqQrySnapshot *req, int requestID) = 0;

//查询行情 K 线请求（根据客户权限，可查询不同交易所的行情）
//@param req 查询对象
//@param requestID 请求 ID
//@返回值 0:成功
virtual int ReqQryKLineQuote(ESunnyQuoteReqQryKLine *req, int requestID) = 0;
```

这些方法的第一个参数具体见 EsunnyQuoteApiStruct 对此结构的定义,第二个参数表示一个请求 ID，这个 ID 和回调的应答 ID 一一对应。

5、回调函数

```
//当客户端与服务器建立连接时（还未登录前），该方法被调用。
//通常在这里进行登录请求
virtual void OnFrontConnected() {};

//当客户端与服务器连接断开时，该方法被调用。
//当发生这个情况后，API 会自动重新连接。
//@param  nReason 错误原因
//        0x1001 网络读失败
//        0x1002 网络写失败
//        0x2001 接收心跳超时
//        0x2002 发送心跳失败
//        0x2003 收到错误报文
virtual void OnFrontDisconnected(int nReason) {};

//登录应答
//@param  pRspUserLogin 应答对象
//@param  errorInfo 错误信息对象
//@param requestID 请求 ID
//@param  bIsLast 是否是最后一个应答
virtual void OnRspQuoteLogin(ESunnyQuoteRspUserLoginField *pRspUserLogin,
ESunnyQuoteErrorInfo* errorInfo, int requestID, bool bIsLast) {};

//登出应答
//@param  errorInfo 错误信息对象
//@param requestID 请求ID
virtual void OnRspQuoteLogout(ESunnyQuoteErrorInfo* errorInfo, int requestID) {};

//推送客户端退出消息，服务器主动踢出客户端
//@param  errorInfo 错误信息对象
virtual void OnRtnClientExit(ESunnyQuoteErrorInfo* errorInfo) {};

//推送交易所列表消息，登陆成功时推送
//@param  data 交易所信息（后台支持的交易所）
//@param  bIsLast 是否是最后一个信息
virtual void OnRtnExchangeList(ESunnyExchangeItemField* pData, bool isLast) {};

//推送 UDP 信息，如果 init 的参数 servertype=QST_UDPTICK，则登录成功后会收到一次推送
//@param  data UDP 信息（SubscribeUdpInfo 用到该数据）
//@param  bIsLast 是否是最后一个信息
virtual void OnRtnUdpInfoData(const ESunnyUdpInfoField& data, bool isLast) {};

//修改密码应答
```

```
//@param  errorInfo 错误信息对象
//@param requestID 请求 ID
virtual void OnRspChangePassword(ESunnyQuoteErrorInfo* errorInfo, int requestID) {};

//订阅行情响应
//@param  pSpecificInstrument 应答对象
//@param  errorInfo 错误信息对象
//@param requestID 请求 ID
//@param  bIsLast 是否是最后一个应答
virtual void OnRspSubscribeMarketData(ESunnyQuoteSpecificInstrumentField
*pSpecificInstrument, ESunnyQuoteErrorInfo *errorInfo, int nRequestID, bool bIsLast) {};

//取消订阅行情响应
//@param  pSpecificInstrument 应答对象
//@param  errorInfo 错误信息对象
//@param requestID 请求 ID
//@param  bIsLast 是否是最后一个应答
virtual void OnRspUnSubscribeMarketData(ESunnyQuoteSpecificInstrumentField
*pSpecificInstrument, ESunnyQuoteErrorInfo *errorInfo, int nRequestID, bool bIsLast) {};

//订阅 K 线行情响应
//@param  pSpecificInstrument 应答对象
//@param  errorInfo 错误信息对象
//@param requestID 请求 ID
//@param  bIsLast 是否是最后一个应答
virtual void OnRspSubscribeKLineData(ESunnyQuoteSpecificInstrumentField
*pSpecificInstrument, ESunnyQuoteErrorInfo *errorInfo, int nRequestID, bool bIsLast) {};

//取消订阅 K 线行情响应
//@param  pSpecificInstrument 应答对象
//@param  errorInfo 错误信息对象
//@param requestID 请求 ID
//@param  bIsLast 是否是最后一个应答
virtual void OnRspUnSubscribeKLineData(ESunnyQuoteSpecificInstrumentField
*pSpecificInstrument, ESunnyQuoteErrorInfo *errorInfo, int nRequestID, bool bIsLast) {};

//推送快照行情变动，对应订阅快照 SubscribeMarketData
//此快照推送包括：普通切片快照和逐笔快照
//@param  pData 应答对象
virtual void OnRtnDepthMarketData(ESunnyQuoteMarketField *pData) {};

//推送 K 线，对应订阅 K 线 SubscribeKLineData
//@param  pData 应答对象
virtual void OnRtnKLineQuoteData(ESunnyQuoteKLine *pData) {};
```

```
//查询合约应答
//@param  pData 应答对象
//@param  errorInfo 错误信息对象
//@param requestID 请求 ID
//@param  bIsLast 是否是最后一个应答
virtual void OnRspQryInstrument(ESunnyInstrumentField* pData, ESunnyQuoteErrorInfo*
errorInfo, int requestID, bool isLast) {};

//查询历史快照行情应答
//@param  pData 应答对象
//@param  errorInfo 错误信息对象
//@param requestID 请求 ID
//@param  bIsLast 是否是最后一个应答
virtual void OnRspQryHisSnapshotQuote(ESunnyQuoteMarketField* pData,
ESunnyQuoteErrorInfo* errorInfo, int requestID, bool isLast) {};

//查询行情 K 线应答
//@param  pData 应答对象
//@param  errorInfo 错误信息对象
//@param requestID 请求 ID
//@param  bIsLast 是否是最后一个应答
virtual void OnRspQryKLineQuote(ESunnyQuoteKLine* pData, ESunnyQuoteErrorInfo* errorInfo,
int requestID, bool isLast) {};
```

三、Demo 范例

API 文件中每个函数和对象有详细的注释，以下为示例代码演示 ，完整示例请参考 Api 附带的测试程序 EsunnyQuoteApiTest。

1、连接登陆

```
// 1.创建 API 实例
CEsunnyQuoteApi* pEQapi = CEsunnyQuoteApi::CreateEsunnyQuoteApi();

//获取当前 Api 版本信息
const char* sVersion = pEQapi->GetVersion();

// 2.注册回调对象
CTestEsunnyQuoteSpi spi;
spi.init(pEQapi);
pEQapi->RegisterSpi(&spi);

// 3.根据易盛提供的前置机地址类型进行注册
bool bFens = true;
if (bFens)
{
    //注册 Fens 前置地址（可注册多个），通过 Fens 获取行情服务器集合，自动选择最优服务器
    pEQapi->RegisterFens("192.168.99.251", 6400);
}
else
{
    //注册 Front 前置地址（可注册多个），通过 Front 直接接入行情服务器
    pEQapi->RegisterFront("192.168.99.251", 6100);
}

// 4.启动连接，默认连接实时推送行情服务器，可以根据实际情况修改默认参数
// 提供实时服务（TCP，切片）
QST_REALTIME = 0,
// 提供历史服务（TCP，历史）
QST_HISTORY = 1,
// 提供 TCP 逐笔（只能内网接入使用）
QST_TICKSNAP = 2,
// 提供 UDP 逐笔（只能内网接入使用，不能跨机房）
QST_UDPTICK = 3,
pEQapi->Init();

// 5.连接成功
// 在 OnFrontConnected 中进行登录
```

```
// 6.登录成功
// 在 OnRspQuoteLogin 中进行查询和订阅请求
```

2、TCP 行情订阅

连接登录成功后，直接订阅

合约信息从 OnRspQryInstrument 中获取

```
//登录应答中，订阅快照行情
ESunnyQuoteSpecificInstrumentField* Instruments[10];
ESunnyQuoteSpecificInstrumentField filed[10];
filed[0].ExchangeID = EXCHANGE_TYPE_ID::ETI_COMEX;
strcpy(filed[0].InstrumentID, "gc1912");
Instruments[0] = &filed[0];
m_pEQapi_->SubscribeMarketData(&Instruments[0], 1);
```

3、UDP 行情订阅

1）pEQapi->Init(QUOTE_SERVER_TYPE::QST_UDPTICK);

2）登录 ReqQuoteLogin

3）在 OnRtnUdpInfoData 中，订阅 udp 通道信息

```
//推送 UDP 信息，如果 init 的参数 servertype=QST_UDPTICK，则登录成功后会收到一次推送
//@param  data UDP 信息（SubscribeUdpInfo 用到该数据）
//@param  bIsLast 是否是最后一个信息
void CTestEsunnyQuoteSpi::OnRtnUdpInfoData(const ESunnyUdpInfoField& data, bool isLast)
{
    printf("OnRtnUdpInfoData channelno = %s, exchange = %s, products = %s\n",
            data.channelno, GetExchangeType(data.exchange).c_str(), data.products);
    std::set<std::string> setProduct;
    std::vector<std::string> vecProduct;
    if (stringsplit(data.products, sDivideSymbol, vecProduct) > 0)
    {
        setProduct.insert(vecProduct.begin(), vecProduct.end());
    }

    // 过滤 udp 通道
    if (data.exchange == EXCHANGE_TYPE_ID::ETI_NYMEX &&
        setProduct.find("CL") != setProduct.end())
    {
        //udp：订阅通道
        m_pEQapi_->SubscribeUdpInfo(data);
    }
}
```

4）订阅合约信息

合约信息从 OnRspQryInstrument 中获取

```
ESunnyQuoteSpecificInstrumentField* Instruments[1];
ESunnyQuoteSpecificInstrumentField filed[1];
filed[0].ExchangeID = m_pEQapi_->GetExchangeEnum(pData->exchangeid);
strcpy(filed[0].InstrumentID, pData->instrumentid);
Instruments[0] = &filed[0];
m_pEQapi_->SubscribeMarketData(&Instruments[0], 1);
```

5）启动 udp 工作

```
// 开始接受 udp 快照推送
// 本地接收 udp 网卡信息（网卡 ip 或网卡设备名，根据本地 ip 设置）
const char* interface_addr_ = "172.26.255.107";
m_pEQapi_->StartUdpWork(interface_addr_.c_str());
```

4、接收行情快照推送

所有行情快照数据（包括 TCP 和 UDP）都是通过下面回调函数推送。

```
//推送行情变动
void CTestEsunnyQuoteSpi::OnRtnDepthMarketData(ESunnyQuoteMarketField *pData)
{
    if (pData != NULL)
    {
        std::uint64_t timestamp = getTimeStamp();

        char content[2048] = { '\0' };
        sprintf(content, "%lu, %d, %s, %s"
            ", %lu, %s"
            ", %f, %d, %d"
            ", %f-%d, %f-%d\n",
            pData->msgseqnum, pData->rptseq, pData->exchangeID, pData->instrumentID,
            timestamp, pData->exchangeupdateTime,
            pData->lastPrice, pData->volume, pData->newvolume,
            pData->bidprice[0], pData->bidvolume[0],
            pData->askprice[0], pData->askvolume[0]);

        printf("%s", content);
    }
}
```

四、EsunnyQuoteApiTest

EsunnyQuoteApiTest 测试程序用来测试环境和数据，分别位于 linux 和 win 目录中，修改好配置，直接运行，结果会显示在命令行上同时也保存到当前 data 目录下。

config.ini 测试程序配置文件，如下：

config.ini

```
[comm]
ip=192.168.99.107
port=6100
userid=abc
password=123456
exchangeid=NYMEX
instrumentid=CL2004

// 连接服务类型
// 0:提供实时服务（TCP，切片）
// 1:提供历史服务（TCP，历史）
// 2:提供TCP逐笔（只能内网接入使用）
// 3:提供UDP逐笔（只能内网接入使用，不能跨机房）
servertype=0

// 操作类型
// 0:订阅推送
// 2:查询快照
// 3:查询K线
// 4:查询逐笔
operationtype=0

// 本地接收udp网卡信息（网卡ip或网卡设备名，根据本地ip设置）
interfaceaddr=172.26.255.107

//////////////////////////查询参数//////////////////////////////////////
// 合约过滤类型：0--不过滤， 2--指定交易所， 4--内盘主力， 8--外盘主力
instrumenttype=0
// 品种ID，为空不过滤，否则过滤指定品种（跨期：CMB-SP 跨品种：CMB-SPC）
productid=
// 品种类别：0--不过滤，49--期货，50--期权，51--组合，52--现货
productclass=0
// 期权类型：0--不过滤，49--看涨，50--看跌
optionstype=0

// 查询历史开始和结束日期时间
tradingday=20191210
```