

QWin行情接口QuoteClientApi使用指引

|  |  |  |  |
| --- | --- | --- | --- |
| 文  档  状  态 | [ ]初稿  [ ]评审通过  [√]修改  [ ]发布  [ ]作废 | 文档标识 | QWin行情接口使用指引 |
| 当前版本 | V1.0 |
| 文档作者 | 上海钱育信息科技有限公司 |
| 公司部门 | 开发部 |
| 更新日期 | 2017-11-16 |

目录

[1.QWin行情模块总体设计 3](#_Toc31665)

[1.1 运行环境： 4](#_Toc12468)

[1.2 设计框架： 4](#_Toc25379)

[2.QuoteClientApi.DLL行情源模块开发相关文件 5](#_Toc31582)

[3.QuoteClientApi.DLL行情源模块配置文件 6](#_Toc30791)

[4.QuoteClientApi.DLL行情源模块导出函数说明 7](#_Toc29742)

[5.QuoteClientApi.DLL行情模块控制类和行情回调类说明 8](#_Toc7749)

[6.QuoteClientApi.DLL行情模块调用启动顺序说明 12](#_Toc9615)

[7.QuoteClientApi.DLL调用Demo 13](#_Toc8756)

**1.QWin行情模块总体设计**

1.1 运行环境：

* 运行平台/开发语言：基础于Windows平台的C++程序。
* 开发环境： Visual Studio C++, (32位)

1.2 设计框架：

* 设计思路： 程序需要通过加载钱育QuoteClientApi.DLL，并从中获取实时行情。

# 2.QuoteClientApi.DLL行情源模块开发相关文件

2.1 QuoteClientApi.h: Dll接口导出函数 / 调用接口文件

2.2 QuoteCtlDef.h: 行情数据结构定义文件

2.3 QuoteClientApi.DLL 行情模块

**注： 以下为接口头文件嵌入**



# 3.QuoteClientApi.DLL行情源模块配置文件

3.1 QuoteClientApi.ini: 模块配置文件, 配置需要支持的市场数据源模块列表。

3.2 配置信息说明如下：

[ServerName]

;市场ID(marketid)不能配错,程序会校验

[sh]

marketid=0

marketchn=上海证券行情

keyfile=./sh/tran2ndsh.dll

sleepm=2000

[sz]

marketid=1

marketchn=深圳证券行情

keyfile=./sz/tran2ndsz.dll

其中：

[sh]节点为上海市场的数据源子模块加载路径及相关配置； [sz]节点为深圳市场的数据源子模块的加载路径及配置；其他市场也可以依照这个格式进行扩充。

这里只举例说明，具体的配置(./sh/tran2ndsh.dll和./sz/tran2ndsz.dll )由钱育提供。

# 4.QuoteClientApi.DLL行情源模块导出函数说明

4.1 行情模块控制对象创建函数

QuoteClientApi \* STDCALL CreateQuoteApi(const char \*pszDebugPath);

功能： 用于创建行情接口对象，用于初始化/释放/启动/数据查询等功能。

输入参数： pszDebugPath默认值为NULL，保留[调试接口], 用于创建行情模块控制类

返回值： 返回控制对象指针

4.2 模块版本号获取函数

const char \* STDCALL GetDllVersion(int &nMajorVersion, int &nMinorVersion);

功能： 获取版本号

输出参数: nMajorVersion输出大版本号；nMinorVersion输出次级版本号

返回值: 返回格式化的版本号字符串

4.3 获取行情配置的信息函数

int STDCALL GetSettingInfo(tagQuoteSettingInfo\* pArrMarket, int nCount);

功能： 获取行情模块各市场的配置信息

输出参数： 返回配置信息列表 (注：如果填NULL, 则只返回配置信息条数)

输入参数: 给定信息列表缓存可容纳数量

返回值： 返回实际市场个数

结构说明：

#pragma pack(1)

typedef struct

{

unsigned char cMarketID; //市场ID

char cMarketChn[63]; //描述文字

char cAddress[128]; //连接地址和端口

int nStatus; //当前状态//0不可用 //1未知//2未就绪//5可用

}tagQuoteSettingInfo;

#pragma pack()

4.4 获取补充信息查询接口对象函数

QuotePrimeApi \* STDCALL CreatePrimeApi();

返回值： 返回QuotePrimeApi \*对象指针，用于查询一些基础信息，如：市场时间、市场信息...

# 5.QuoteClientApi.DLL行情模块控制类和行情回调类说明

5.1 行情模块控制类QuoteClientApi说明

5.1.1 模块初始化的接口

virtual int STDCALL Init() = 0;

返回值： 返回>0,表示初始化成功，否则为失败。

5.1.2 模块释放的接口

virtual void STDCALL Release() = 0;

5.1.3 启动行情数据接收的接口

virtual int STDCALL BeginWork() = 0;

返回值： 返回>0,表示初始化成功，否则为失败。

注： 需要在调用Init()函数成功后再调用。

5.1.4 关闭行情数据接收的接口

virtual void STDCALL EndWork() = 0;

5.1.5 行情数据回调、市场事件回调对象注册接口

virtual void STDCALL RegisterSpi(QuoteClientSpi \* pspi) = 0;

输入参数： 输入QuoteClientSpi 对象的派生类的对象地址，用于处理行情/日志/市场状态的回调。

5.1.6 获取指定市场信息的接口(包括分类，放大倍数等)

virtual int STDCALL GetMarketInfo(unsigned char cMarket, char\* pszInBuf, int nInBytes) =0;

输入参数： cMarket用来指定针对的操作市场的编号 (参考QuoteCltDef.h文件的XDFMarket的定义)

输出参数： pszInBuf用来返回市场信息，格式：XDFAPI\_MsgHead + XDFAPI\_MarketKindHead + XDFAPI\_UniMsgHead + XDFAPI\_MarketKindInfo + XDFAPI\_MarketKindInfo + ...

输入参数： nInBytes指定pszInBuf的缓存长度

返回值： 返回>0成功，表示获取的数据的长度， <=0表示失败。

5.1.7 获取指定市场的码表数据的接口

virtual int STDCALL GetCodeTable(unsigned char cMarket, char\* pszInBuf, int nInBytes, int& nCount) = 0;

输入参数： cMarket用来指定针对的操作市场的编号 (参考QuoteCltDef.h文件的XDFMarket的定义)

输出参数： pszInBuf码表数据的缓存(注：若NULL== pszInBuf，则nCount代表商品数，且不返回具体的码表内容，只作一下统计): XDFAPI\_UniMsgHead + XDFAPI\_NameTableSh/XDFAPI\_NameTableSz + ...

输入参数： nInBytes码表缓存pszInBuf的最大可写长度

输出参数： 返回商品的个数

返回值： >0表示调用成功，否则表示失败

5.1.8 获取指定市场的快照数据的接口

virtual int STDCALL GetLastMarketDataAll(unsigned char cMarket, char\* pszInBuf, int nInBytes) = 0;

输入参数： cMarket用来指定针对的操作市场的编号 (参考QuoteCltDef.h文件的XDFMarket的定义)

输出参数： pszInBuf,数据缓存: XDFAPI\_UniMsgHead + XDFAPI\_IndexData + ... + XDFAPI\_UniMsgHead + XDFAPI\_StockData5 + ...

输入参数： nInBytes表示数据输出缓存pszInBuf的最大可写长度

返回值： >0表示调用成功，否则表示失败

5.1.9 获取指定市场的市场状态的接口

virtual int STDCALL GetMarketStatus(unsigned char cMarket,int& nStatus, unsigned int& ulTime, \_\_int64 \* pI64Send, \_\_int64 \* pI64Recv)=0;

输入参数： cMarket用来指定针对的操作市场的编号 (参考QuoteCltDef.h文件的XDFMarket的定义)

输出参数： nStatus, 当前市场状态(0:不可用; 1:未知状态; 2:初始化; 5:可服务)

输出参数： uiTime, 当前的市场时间

输出参数: pI64Send, 发送的字节数

输出参数: pI64Recv, 接收的字节数

5.2 行情模块回调类QuoteClientSpi说明

5.2.1 市场状态变化通知回调

virtual bool STDCALL XDF\_OnRspStatusChanged(unsigned char cMarket, int nStatus)=0;

输入参数： cMarket 用来指定针对的操作市场的编号 (参考QuoteCltDef.h文件的XDFMarket的定义)

输入参数： nStatus, 市场状态变化(0:不可用; 1:未知状态; 2:初始化; 5:可服务)

5.2.2 行情数据实时回调

virtual void STDCALL XDF\_OnRspRecvData(XDFAPI\_PkgHead \* pHead, const char \* pszBuf, int nBytes)=0;

输入参数： pHead, 数据包的包头, 用来描述市场编号和数据的长度(参考QuoteCltDef.h文件的XDFAPI\_PkgHead的定义)

输入参数： pszBuf, 行情数据体可能返回如下两种结构的数据：

1. 单个数据块: XDFAPI\_MsgHead (MsgType>0) + 结构体内容 (XDFAPI\_MsgHead 定义见QuoteCltDef.h)
2. 复合数据块:XDFAPI\_UniMsgHead (MsgType < 0) + 结构体1 + 结构体2 ... (XDFAPI\_UniMsgHead定义见QuoteCltDef.h)

5.2.3 日志输出回调

virtual void STDCALL XDF\_OnRspOutLog(unsigned char nLogType, unsigned char nLogLevel,const char \* pLogBuf)=0;

输入参数： nLogType, 0=流水日志、1=信息日志、2=警告日志、3=错误日志

输入参数： nLogLevel, (0-7)

输入参数： pLogBuf, 输入的日志的内容字符串

5.2.4 其他通知回调扩展

virtual int STDCALL XDF\_OnRspNotify(unsigned int nNotifyNo, void\* wParam, void\* lParam )=0;

注： 该接口暂时可以忽略

5.3 扩展功能类QuotePrimeApi说明

5.3.1 信息查询接口

virtual int STDCALL ReqFuncData(int FuncNo, void\* wParam, void\* lParam) =0;

输入参数： FuncNo, 功能号100表示以有可能阻塞的方式查询市场信息(因为部分市场可能未就绪，中间需要等待)；101表示以非阻塞方式查询某市场的基础信息

输出参数： wParam, 对于功能号100/101的情况，返回指定市场的时间。

输出参数： lParam, 对于功能号100/101的情况，返回XDFAPI\_MarketStatusInfo结构，用于描述市场基础状态信息

返回值: 返回>0表示成功，否则表示失败。

# 6.QuoteClientApi.DLL行情模块调用启动顺序说明

6.1 初始化Socket库环境

6.2 加载QuoteClientApi.DLL模块, 导获取导出函数CreateQuoteApi()

6.3 通过CreateQuoteApi()获取行情模块控制对象指针QuoteClientApi\*

6.4 初始化模块：

QuoteClientApi::Init()

6.5 注册行情/事件回调：

QuoteClientApi::RegisterSpi( QuoteClientSpi\* )

6.6 真正启动行情模块：

QuoteClientApi::BeginWork()

6.7 程序进入行情/事件接收循环...

# 7.QuoteClientApi.DLL调用Demo

#include "stdafx.h"

#include “QuoteCltDef.h”

#include "QuoteClientApi.h"

#pragma comment( lib, "Ws2\_32.lib" )

typedef QuoteClientApi\* (\_\_stdcall \*T\_Func\_CreateQuoteApi)( const char \*pszDebugPath );///< 导出函数定义

int \_tmain(int argc, \_TCHAR\* argv[])

{

WSADATA oWSAData = { 0 };

WORD wVerRequested = { 0 };

Dll m\_oQuoateClientApi; ///< 插件加载类

QuotationSpi m\_oCallBackQuotation; ///< 行情/事件回调对象，派生自QuoteClientSpi

QuoteClientApi\* m\_pQuoteClientApi = NULL; ///< 导出控制类

T\_Func\_CreateQuoteApi m\_funcCreateQuoteApi = NULL; ///< 控制类导出函数

int nErrorCode = 0; ///< 错误编号

wVerRequested = MAKEWORD( 2, 1 );

nErrorCode = WSAStartup( wVerRequested, &oWSAData );

if( 0 != nErrorCode )

{

DWORD dw = ::GetLastError();

::printf( "SOCKET环境初始化失败, errorcode=%d \n", nErrorCode );

return -1;

}

nErrorCode = m\_oQuoateClientApi.LoadDll( "./QuoteClientApi.dll" );

if( 0 != nErrorCode )

{

::printf( "加载钱育行情模块, errorcode=%d \n", nErrorCode );

return nErrorCode;

}

m\_funcCreateQuoteApi = (T\_Func\_CreateQuoteApi)m\_oQuoateClientApi.GetDllFunction( "CreateQuoteApi" );

if( NULL == m\_funcCreateQuoteApi )

{

::printf( "获取导出函数失败\n" );

return -2;

}

m\_pQuoteClientApi = m\_funcCreateQuoteApi( "" );

if( NULL == m\_pQuoteClientApi )

{

::printf( "获取行情对象控制指针失败\n" );

return -3;

}

if( m\_pQuoteClientApi->Init() <= 0 )

{

::printf( "行情模块控制对象初始化失败 \n" );

return -4;

}

m\_pQuoteClientApi->RegisterSpi( &m\_oCallBackQuotation );

if( m\_pQuoteClientApi->BeginWork() <=0 )

{

::printf( "启动行情模块失败\n" );

return -5;

}

::printf( "[OK] 服务启动成功 ................................. \n" );

::Sleep( 1000 \* 6 );

///< ---------------- 获取上海市场基本数据 ----------------------------------------

char tempbuf[8192] = { 0 };

nErrorCode = m\_pQuoteClientApi->GetMarketInfo( XDF\_SH, tempbuf, sizeof(tempbuf) );

if( nErrorCode <= 0 )

{

::printf( "error, GetMarketInfo()\n" );

}

XDFAPI\_MarketKindHead\* pKindHead = (XDFAPI\_MarketKindHead\*)(tempbuf+ sizeof(XDFAPI\_MsgHead));

::printf( "上海Lv1商品总数: %d\n", pKindHead->WareCount );

int m = sizeof(XDFAPI\_MsgHead)+sizeof(XDFAPI\_MarketKindHead);

for( int i = 0; m < nErrorCode; )

{

XDFAPI\_UniMsgHead\* pMsgHead = (XDFAPI\_UniMsgHead\*)(tempbuf + m);

char\* pbuf = tempbuf + m +sizeof(XDFAPI\_UniMsgHead);

int nMsgCount = pMsgHead->MsgCount;

::printf( "上海Lv1分类信息数量: %d\n", nMsgCount );

while( nMsgCount-- > 0 )

{

XDFAPI\_MarketKindInfo\* pInfo = (XDFAPI\_MarketKindInfo\*)pbuf;

::printf( "%d, 放大倍数：%f\n", ++i, pow(10.0, int(pInfo->PriceRate) ) );

pbuf += sizeof(XDFAPI\_MarketKindInfo);

}

m += sizeof(XDFAPI\_MsgHead) + pMsgHead->MsgLen;

}

while( true )

{

::Sleep( 1000 );

}

return 0;

}