

用户手册

User Manual

实时行情系统 v2.6

发布日期：2017 年 02 月 09 日

内容

1. [概述](#)

介绍实时行情系统的基本信息。

1.1 [开发包目录介绍](#)

1.2 [头文件说明](#)

1.3 [系统整体架构](#)

1.4 [基础接口概述](#)

2. [支持的环境及安装方法](#)

声明本系统运行所需环境版本、支持的编程语言及安装方法。

3. [接口使用指引](#)

此部分提供了基础接口的使用指引,如操作步骤、部分函数的调用方法等等。

3.1 [基础接口使用指引](#)

3.2 [订阅代码数限制权限](#)

4. [行情接口](#)

此部分对 QTS2.6 所有的接口进行解释说明。

4.1 [基础接口 GTAQTSInterfaceBase.h](#)

[4.2 数据类接口 QTSDataFieldDefine.h](#)

[5. 数据定义](#)

此部分针对用户在使用 QTS2.6 时需要用到的概念及相关数据进行解释，主要包含以下几点：

[5.1 消息类型 MsgType](#)

[5.2 市场列表 MarketTag](#)

[5.3 网络状态含义 ConnectState](#)

[5.4 返回码含义列表 RetCode](#)

[5.5 消息结构体的字段类型 FIELD_TYPE](#)

[5.6 消息结构体（基础 API 适用）](#)

[6. 接口使用注意事项](#)

此部分包含用户在使用本系统须知的注意事项。如：

[6.1 用户权限](#)

[6.2 订阅/取消订阅](#)

[6.3 消息类型连接状态](#)

[6.4 浮点数精度](#)

[6.5 断线重连](#)

[6.6 系统延时](#)

[6.7 查询数据超时](#)

[7. 附录](#)

此部分主要是对数据字段的补充解释。

[7.1 消息类型结构体与基础接口的对应](#)

[7.2 当前品种交易状态](#)

[7.3 深圳市场统计指标定义表](#)

[7.4 QTS1.X 与 QTS2.X 中金所字段对比](#)

[7.5 QTS1.X 与 QTS2.X 上交所 L1 字段对比](#)

[7.6 QTS1.X 与 QTS2.X 上期所 L1 字段对比](#)

[7.7 QTS1.X 与 QTS2.X 郑商所 L1 字段对比](#)

[7.8 QTS1.X 与 QTS2.X 大商所 L1 字段对比](#)

[7.9 QTS1.X 与 QTS2.X 大商所 L2 字段对比](#)

[7.10 业务 FAQ](#)

[8. 实例](#)

[8.1 例一（C++ 示例代码）](#)

[8.2 例二（C++ 示例代码）](#)

[8.3 Java 示例代码](#)

8.4 C#示例代码

1. 概述

QTS2.6 提供基础接口用于获取行情数据。**基础接口**是一套面向基础业务用户的接口，一个接口支持单个市场单个消息的查询、订阅和取消订阅，订阅结果通过回调函数返回，返回消息类型结构体。

1.1 [开发包目录介绍](#)

简单介绍了 QTS2.6 接口目录的结构及包含的内容。

1.2 [头文件说明](#)

对目录中每个头文件的作用及包含的内容进行描述。

1.3 [系统整体架构](#)

介绍 QTS 的系统架构

1.4 [基础接口概述](#)

概述基础行情接口所支持的功能及使用场景。

1.1 开发包目录介绍

- **头文件目录**
- **ReleaseLib 库文件目录**
 - Win32 32 位库文件目录
 - Win64 64 位库文件目录
- **Release 64 位动态链接库 (DLL) 目录**
 - Linux64 CentOS 6.5 64 位库文件及系统配置文件
 - Win32 32 位动态链接库 (DLL) 目录
 - Win64 64 位动态链接库 (DLL) 目录
- **Doc 文档目录**
 - 实时行情系统 v2.6 用户手册.chm
- **Example 示例代码目录**
 - Example_1 用基础接口获取代码列表和权限消息列表的示例代码
 - Example_2 用基础接口实现不同方式的订阅、取消订阅和查询快照的示例代码
 - Example.java 利用 Java 版本 API 实现从创建 API 实例，到注册、登录、订阅数据、释放实例等一系列完整的操作。
 - ExampleDotNet 利用 C#版本 API 实现从创建 API 实例，到注册、登录、订阅数据、释放

实例等一系列完整的操作。

1.2 头文件说明

表 1.1 编程所需头文件列表

序号	文件名	说明
1	GTAQTSInterfaceBase.h	使用基础接口时使用（C++接口）
2	GTAQTSInterfaceBaseC.h	使用基础接口时使用（C 接口）
3	QTSDataFieldDefine.h	数据类定义文件，包括数据缓存模板类，按字段访问类，
4	QTSDataType.h	QTS 数据格式定义
5	QTSStruct.h	数据消息结构体定义文件

1.3 系统整体架构

QTS2.6 采用双消息源的设计方式，通过高效的通信方式实现本地和异地采集的互联，确保了整个系统的高可用、高效以及数据的完整、准确。整体系统架构如下图所示。

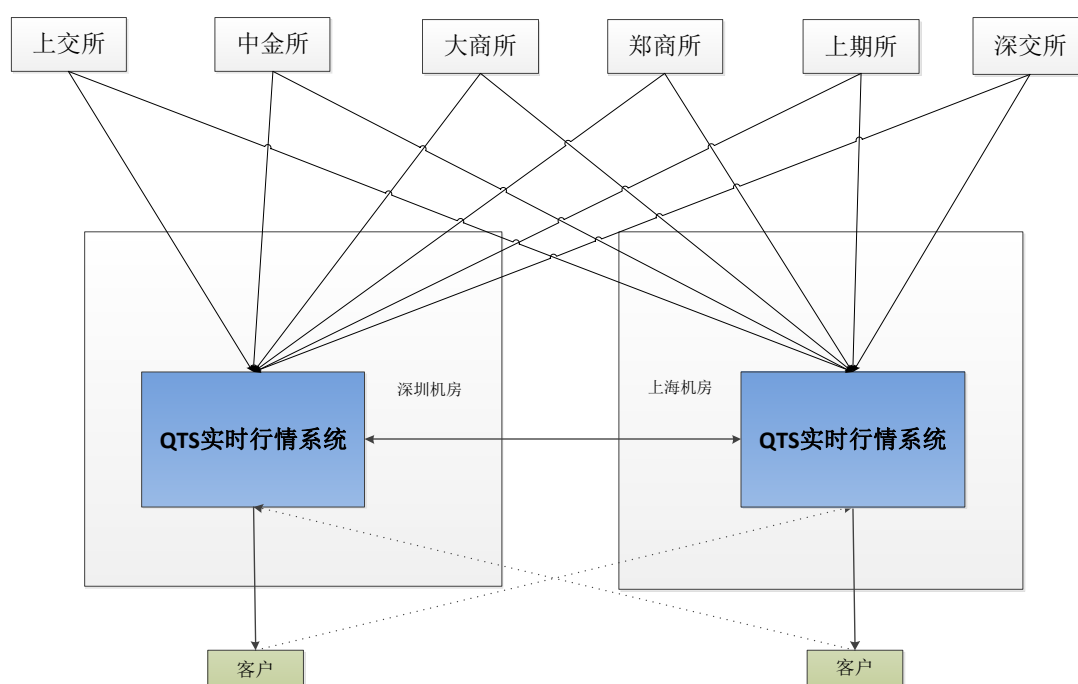


图 1-1 QTS2.6 系统示例图

QTS 系统采用双数据源设计，在任意一个数据源出现问题的情况下，系统仍然可以使用；同时为每个客户端提供主备 Fens 服务，确保客户端行情的连接，整套系统**可用性高**。QTS

同时接受两个数据源数据，并作去重处理，使得**数据更加丰富和完整**。此外 QTS 在内部加强优化处理，在外部数据传输时采用 Fast 编码技术，保障数据传输的**时效性和高效性**。

1.4 基础接口概述

QTS2.6 基础接口详细使用方法见下：

表 1.2 基础接口概述

接口 使用方 法	基础接口
订阅方式	按消息类型查询、订阅数据，查询、订阅时指定消息类型如：上交所 L1 静态数据、上交所 L1 实时行情、深交所 L1 静态数据等（具体情况见表 5.1 消息类型）。
返回数据 方式	每一个消息类型都有一个查询快照接口和实时行情回调接口，通过这些接口可返回一个涵盖该消息类型下所有字段的结构体。
订阅的市 场数量	一次只能订阅一个市场，因为任何市场的消息类型都是唯一的。
建议使用 场景	订阅单个市场的消息，速度更快
支持的语 言	C++、C#、java

2. 支持的环境及安装方法

- 本系统支持的操作系统及语言如下表所示。

User Mannual

		支持的语言		
		C++	C#	Java
支持的 环境	Window 2008 32bits	√	√	√
	Window 2008 R2 64bits	√	√	√
	Window 2012 R2 64bits	√	√	√
	CentOS 6.5(64bits)	√	×	√

注：

从 2.6 版本开始，C++、C#、Java 接口覆盖沪深 L1、沪深 L2、中金所 L2、上期所 L1、郑商所 L1、大商所 L1、大商所 L2 市场；

从 2.4 版本开始，支持的 Window 环境为 Window 2008 32bits、Window 2008 R2 64bits、Window 2012 R2 64bits。

● 安装方法：

Windows 安装：

将适合自己操作系统版本的 ReleaseLib 文件夹中的文件添加至工程 Library 目录；

将对应自己行情接口类型权限的 Include 文件夹中的头文件添加至工程 Include 目录；

将符合自己操作系统版本的 Release 文件夹中的文件拷贝至工程中可执行文件所在的目录。

Linux 安装：

添加头文件；

将 Release\Linux64 目录下的所有文件拷贝至可执行文件所在的目录。

注意：在使用 API 之前，须保证本机已正确安装相应操作系统版本的 VC2010Redist。C#接口

使用时须安装.Net Framework 4.0。

User Mannual

编译环境配置：

建议使用 Microsoft Visual Studio 2010,并安装好相应操作系统版本的 SP1 补丁包。

使用 JAVA 版本接口需要安装 JDK1.7。

客户端 API 运行环境最低配置

参数指标	说明		备注
硬件配置	CPU	4 核	
	硬盘	500G	
	内存	8G	
操作系统	<p>➤ C++：</p> <p>Window 2008 32bits、Window 2008 R2 64bits、 Window 2012 R2 64bits。</p> <p>CentOS 6.5 64 位</p> <p>➤ C#：</p> <p>Window 2008 32bits、Window 2008 R2 64bits、 Window 2012 R2 64bits。</p> <p>➤ JAVA:</p> <p>Window 2008 32bits、Window 2008 R2 64bits、 Window 2012 R2 64bits。</p>		支持 32、64 位、中英文简繁体

User Mannual

	CentOS 6.5 64 位	
平台要求	.NET Framework 4.0 (C#) ,JDK1.7	
网络要求	—	
端口要求	—	
说明	—	

3. 接口使用指引

此章节从基础接口介绍了使用本系统接口时函数的调用顺序 ,并列举了部分函数的调用方法。

基础接口使用指引

介绍了基础接口使用时函数的调用顺序及基础接口查询快照数据、订阅实时数据、取消订阅、查询代码列表及查询权限消息类型列表等函数的使用。

订阅代码数限制权限

介绍订阅代码数限制权限，与全量权限账号的区别

3.1 基础接口使用指引

QTS 基础接口的使用过程如下图所示：

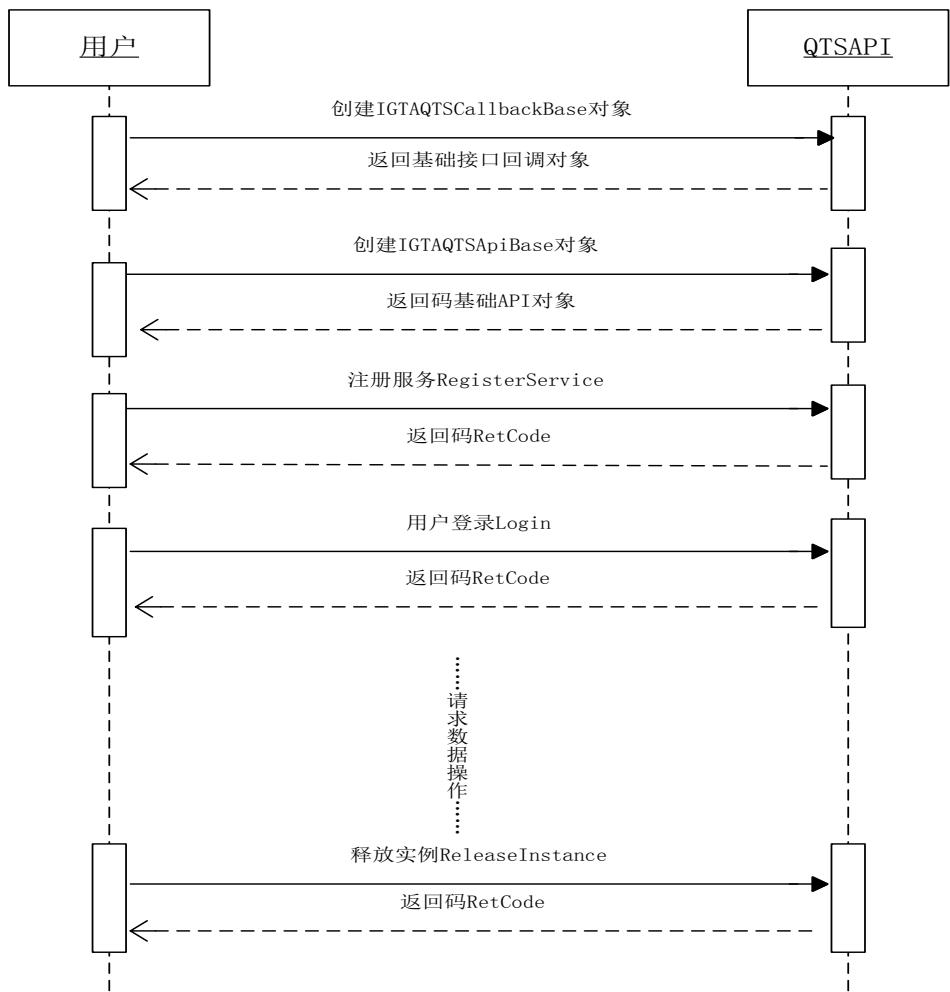


图 3-1 总体操作序列图

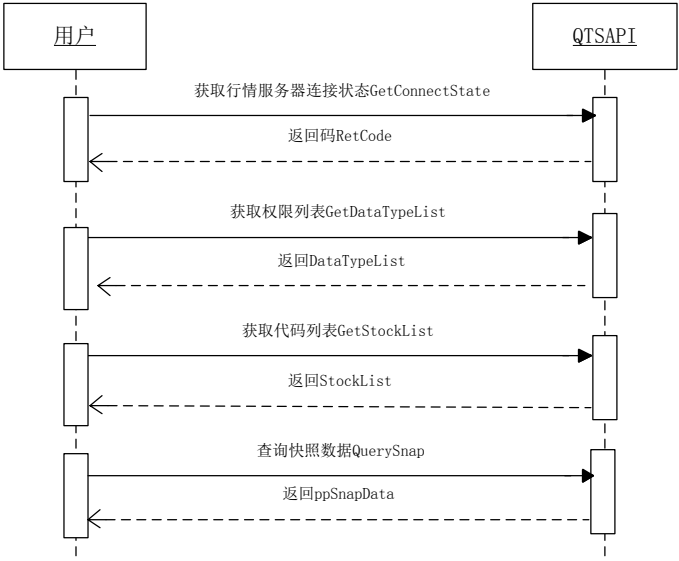


图 3-2 常用操作序列图

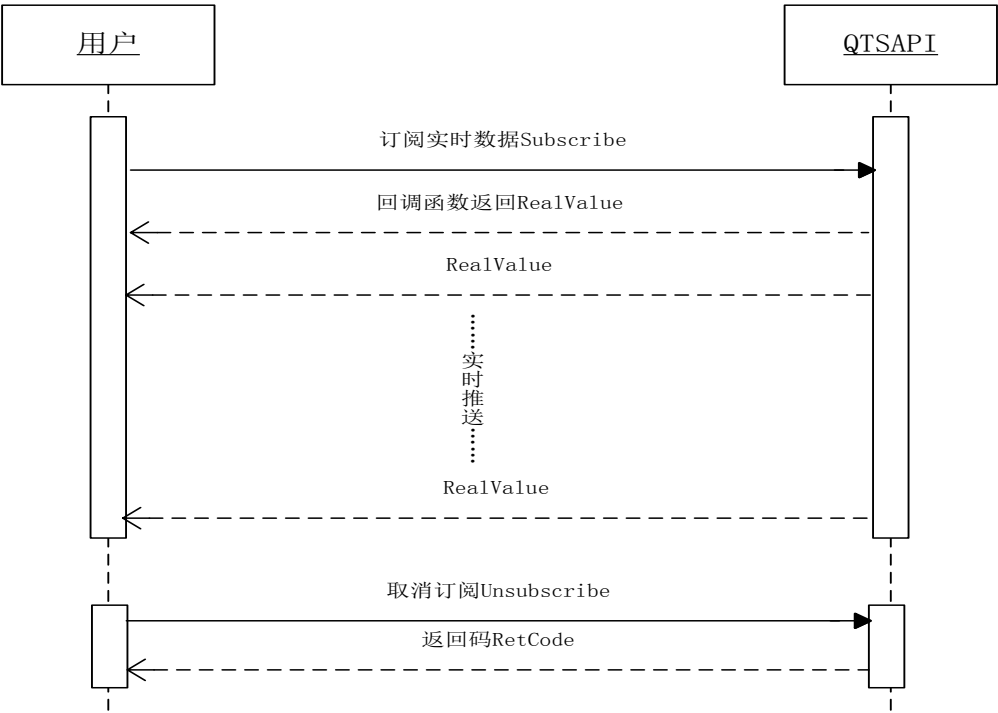


图 3-3 订阅实时数据操作序列图

User Mannual

接口使用注意事项：

(1) 请勿在回调函数接口内执行耗时操作，如：复杂运算，写文件等；否则会堵塞数据的接收。建议处理方式：把接收到的数据放至队列，再由工作线程处理接收到的数据内容；

(2) 当不再使用 API 时 需要调用释放实例接口(IGTAQTSApiBase::ReleaseInstance) 来释放资源，否则会引起内存泄漏及不可预知问题。

(3) QTS 共提供了多种语言的 API ,包括 C++、C#、Java 等。其中 C#和 Java 是在 C++ 的基础进行封装 其接口与 C++ 版本设计上几乎保持一致 在部分地方存在差异。C#和 Java 用户在使用接口时，须注意 C++ 接口中联合体(union)的转换。详情请参看附录 FAQ。

下面以一个完整的实例展示基础接口的使用。

```
////////////////////////////////////
////
/// @brief    获取代码列表和消息列表
///
////////////////////////////////////
////
#include <string>
#include "QTSDDataType.h"
#include "QTSStruct.h"
#include "GTAQTSTInterfaceBase.h"
#include <stdio.h>

int main()
{
    /**
        获取市场代码列表、获取用户权限列表示例代码
    */
    printf("Run Example_1: \n");

    //第一步：创建基础接口回调对象
    //订阅消息回调类，具体使用示例方法请参见 Example_2 工程
    IGTAQTSCallbackBase m_CallbackBase;
```

User Mannual

//第二步：创建基础 API 对象

//详见《GTA 实时行情系统 V2.X 用户手册》4.2.1.1 创建实例 CreateInstance 章节

```
IGTAQTSApiBase* pApiBase =  
IGTAQTSApiBase::CreateInstance(m_CallbackBase);
```

//第三步：注册，注册 FENS 地址

//***** 警告：实际生产环境使用时，从 GTA 公司获取到的 FENS 地址，此处需要全部通过“RegisterService”函数接口注册，

//***** 否则，在数据高可用方面，会大打折扣。

//***** 如有 4 个 FENS ip 地址，需要如下调用：

```
//      pApiBase->RegisterService("ip1", port1);  
//      pApiBase->RegisterService("ip2", port2);  
//      pApiBase->RegisterService("ip3", port3);  
//      pApiBase->RegisterService("ip4", port4);
```

//详见《GTA 实时行情系统 V2.X 用户手册》4.2.1.4 注册服务 RegisterService 章节

```
pApiBase->RegisterService("192.168.105.136", 7777);  
pApiBase->RegisterService("192.168.195.82", 8899);  
pApiBase->RegisterService("192.168.195.83", 8899);  
pApiBase->RegisterService("192.168.195.84", 8899);
```

```
do  
{
```

//第四步：登录，通过用户名与密码向服务器登陆

//详见《GTA 实时行情系统 V2.X 用户手册》4.2.1.5 用户认证 Login 章节

```
RetCode ret = pApiBase->LoginX("gta1", "123456", "NetType=0");
```

```
if ( Ret_Success != ret )
```

```
{  
    printf("Login error:%d\n", ret);  
    break;  
}
```

```
CDataBuffer<StockSymbol> StockList1;
```

```
CDataBuffer<StockSymbol> StockList2;
```

// 获取上交所和深交所代码列表，其中 SSE 表示上交所，SZSE 表示深交所，CFFEX 表示中金所

//详见《GTA 实时行情系统 V2.X 用户手册》4.2.1.11 获取代码列表

GetStockList 章节

```
ret = pApiBase->GetStockList("sse,szse", StockList1);
```

```
if ( Ret_Success != ret )
```

```
{
```

```
        printf("GetStockList(sse,szse) error:%d\n", ret);
        break;
    }

    // 获取上交所代码列表
    ret = pApiBase->GetStockList("sse", StockList2);
    if ( Ret_Success != ret )
    {
        printf("GetStockList(sse) error:%d\n", ret);
        break;
    }

    int Count = StockList2.Size();
    StockSymbol* pStock = StockList2;
    int WriCount = Count > 10 ? Count : 10;
    printf("SSE Stock Count = %d List:", Count);
    for( int i = 0; i < WriCount; ++i)
    {
        printf("%s, ", pStock[i].Symbol);
    }
    printf("\n");

    CDataBuffer<MsgType> DataTypeList;
    // 获取权限列表
    // 详见《GTA 实时行情系统 V2.X 用户手册》4.1.1.7 获取权限列表
    GetDataTypeList 章节
    ret = pApiBase->GetDataTypeList(DataTypeList);
    if ( Ret_Success != ret )
    {
        printf("GetDataTypeList(sse) error:%d\n", ret);
        break;
    }

    MsgType* pMsg = DataTypeList;
    Count = DataTypeList.Size();
    printf("MsgType Count = %d, List:", Count);
    for( int i = 0; i < Count; ++i)
    {
        printf("0x%08x, ",pMsg[i]);
    }
    printf("\n");

} while (false);
```

User Mannual

```
getchar();
```

// 释放 API 实例，当不再使用 API 时，需要调用此接口释放内部资源，否则会引起内存泄漏及不可预知问题

```
// 详见《  
实时行情系统 v2.x 用户手册》4.2.1.2 释放实例 ReleaseInstance 章节  
IGTAQTSApiBase::ReleaseInstance(pApiBase);  
return 0;  
}
```

关于回调接口类的使用，可参看[第八章例二](#)。

3.2 订阅代码数限制权限

2.4.2 版本新增功能：支持对订阅代码数进行限制，可限制每个消息类型的订阅代码数。对于被限制订阅代码数的账号，可以作为用户测试环境的测试账号，也可以根据业务需求，满足对指定代码的业务需求。

订阅代码数限制的账号与全量账号的使用方法仅在订阅上存在差异：

表 3.1 订阅代码数限制的账号与全量账号使用上的差异

使用方法		订阅代码数限制账号	全量账号
订阅	基础接口	(1)不支持按消息类型进行全市场代码订阅； (2)按代码订阅，代码数不可超过订阅代码数限制上限；	(1)支持按消息类型进行全市场代码订阅； (2)按代码订阅，无订阅数量限制；
取消订阅		相同	相同
查询快照数据		相同	相同
获取代码列表		相同	相同

4. 行情接口

此部分对 QTS 用户接口进行详细的描述，0 以 C++版本接口为主。 主要包含三个部分:

[基础接口 GTAQTSInterfaceBase.h](#)

介绍基础接口头文件 [GTAQTSInterfaceBase.h](#) 中的类及其函数。

[数据类接口 QTSDataFieldDefine.h](#)

介绍数据类头文件 [QTSDataFieldDefine.h](#) 中的类及其函数。

4.1 基础接口 GTAQTSInterfaceBase.h

基础接口文件 GTAQTSInterfaceBase.h 中包含了两类接口：一类是基础实时接口。这类接口在使用时即时返回结果给用户，如创建（释放）实例、设置超时时长、登录和订阅实时数据等等。另一类是基础回调接口。这类接口在被系统调用时返回结果或数据给用户，例如：当有新的逐笔成交数据到来时，逐笔成交的实时数据回调接口被调用，向用户推送一条逐笔成交数据；当客户 API 与系统的行情服务器状态发生变化时，订阅服务连接状态回调接口返回一条与该行情服务器相关的消息类型的连接状态给用户等。

实时接口

[创建实例 CreateInstance](#)

[释放实例 ReleaseInstance](#)

[设置超时时长 SetTimeOut](#)

[注册服务 RegisterService](#)

[用户认证 Login](#)

[获取行情服务器连接状态 GetConnectState](#)

[获取权限列表 GetDataTypeList](#)

[订阅实时数据 Subscribe](#)

[取消订阅 Unsubscribe](#)

[查询快照数据 QuerySnap](#)

[获取代码列表 GetStockList](#)

回调接口

[登录状态回调接口 OnLoginState](#)

[订阅服务连接状态回调接口 OnConnectionState](#)

[实时数据回调接口 OnSubscribe](#)

4.1.1 实时接口类 IGTAQTSApiBase

4.1.1.1 创建实例 CreateInstance

使用此接口创建一个基础接口（IGTAQTSApiBase）的全局对象。

函数原型：

```
static IGTAQTSApiBase * CreateInstance(IGTAQTSCallbackBase& Callback)
```

参数：

Callback	基础接口回调对象
----------	----------

返回：

基础接口对象

例子：

先定义一个基础接口回调对象，再创建基础 API 实例

```
CallbackBase m_CallbackBase;    //定义一个基础接口回调对象
```

```
IGTAQTSApiBase* pApiBase  = CreateInstance(m_CallbackBase)    //创建基础 API 实例
```

实现回调接口，用新的回调对象创建 API 实例

```
class Callback:public CallbackBase{    //实现基础回调接口

public:

    Callback(){}

    ~Callback(){}

    ...}

    Callback m_Callback;                //定义新的回调对象

    IGTAQTSApiBase* pApiBase  = CreateInstance(m_Callback)    //创建基础 API 实例
```

该函数在 [GTAQTSInterfaceBase.h](#) 中定义。

4.1.1.2 释放实例 ReleaseInstance

使用此接口释放一个由 CreateInstance 创建的基础接口 (IGTAQTSApiBase) 的对象。

函数原型：

```
void ReleaseInstance(IGTAQTSApiBase* pInstance)
```

参数：

pInstance 由 CreateInstance 创建的基础接口对象

例子：

释放基础 API 实例

```
CallbackBase m_CallbackBase;    //定义一个基础接口回调对象

    IGTAQTSApiBase* pApiBase  = CreateInstance(m_CallbackBase)    //创建基础 API 实例
```


User Mannual

```
ReleaseInstance(pApiBase); //释放 API 实例
```

该函数在 [GTAQTSInterfaceBase.h](#) 中定义。

4.1.1.3 设置超时时长 SetTimeout

函数原型：

```
void SetTimeout(int nSecond = TIMEOUT_DEFAULT)
```

参数：

nSecond	超时时长，单位为秒
---------	-----------

说明：

若不设置超时时长，默认为 3 秒，若设置的范围小于 1，则超时时长为 1，反之若大于 120，则为 120。

例子：

```
SetTimeOut(10) //超时时长为 10 秒
```

```
SetTimeOut(-1) //超时时长为 1 秒
```

```
SetTimeOut(1000) //超时时长为 120 秒
```

该函数在 [GTAQTSInterfaceBase.h](#) 中定义。

4.1.1.4 注册服务 RegisterService

使用此接口注册服务，支持行情订阅服务无法连接后自动切换，重连后将自动重新订

User Mannual

阅，以实现高可用。因此在配置时注册地址时需要将提供的全部 FENS 地址配齐以保证高可用实现。连接顺序为顺序连接，当连接断开，先尝试连接当前地址一次，如不成功，则按顺序循环进行连接。重连成功后将订阅消息重新请求一次。

函数原型：

```
RetCode RegisterService(const char* pIP, unsigned short uPort)
```

参数：

pIP FENS 服务 IP 地址

uPort FENS 服务端口

返回：

RetCode 错误码枚举类型，含义见表 [5.4 返回码含义列表](#)

注：

注册服务时，需保证服务 IP 格式正确，若格式错误或未设置服务 IP，均有可能导致系统报错。

例子：

注册一个服务地址

```
RegisterService("192.168.192.168","8000");
```

注册多个服务地址，程序支持自动重连，用户无需自行编写重连逻辑。

```
RegisterService("192.168.192.168","8000");
```

User Mannual

```
RegisterService("192.168.192.168","8001");
```

```
RegisterService("192.168.192.168","8002");
```

```
RegisterService("192.168.192.168","8003");
```

该函数在 [GTAQTSInterfaceBase.h](#) 中定义。

4.1.1.5 用户认证 LoginX (支持网络类型设置)

2.6 版本提供新的用户认证函数 :**LoginX** ,支持配置不同的连接网络类型(公网与特殊网络) , 通过不同的网络类型进行用户认证。配置注册服务之后使用此接口进行身份认证 , 需要输入正确的账号和密码。

老版本 Login 不作升级维护 , 因此使用 2.6 版本及之后版本 API 的用户 , 请使用 LoginX 进行用户认证。

函数原型 :

```
RetCode LoginX (const char* pUserName, const char* pPassword, const char* pOptions = NULL)
```

参数 :

pUserName	用户名 (账号) 。长度限制为 50 个字节。
pPassWord	密码。长度限制为 20 个字节。
pOptions	可选附加参数,可填充 null。当有多个参数时,用逗号 (,) 分隔。例如 : " 参数 A=取值 A , 参数 B=取值 B , " 。目前只启用 NetType 网络类型参数。

/// 参数标识	可选值	说明
/// NetType	0 , 1	连接网络类型 , 0 表示公网(默认值) , 1 表示特殊网络 (内网、专线等)
///		示例 , 如 : "NetType=1"

返回 :

RetCode 错误码枚举类型 , 含义见表 [5.4 返回码含义列表](#)

注 :

用户认证支持账户互踢机制 , 即同一个账户登录多次 , 后登录将剔除前登录 , 成为有效登录。

对于特殊网络用户 (专线接入、内网接入等) , 可以通过配置 NetType 的参数值 (0 为公网接入 , 1 为特殊网络接入) , 实现不同的网络环境接入 QTS 行情服务。

对于特殊网络用户 , 要求使用 LoginX 进行认证登录 , 不建议使用 Login。建议特殊网络用户建立应急机制 : 当特殊网络环境出现异常 , 能够通过改变 NetType 值重登录切换到公网环境 ; 当公网环境出现异常 , 能够通过改变 NetType 值重登录切换到特殊网络环境。

例子 :

通过公网网络接入 , 进行用户认证

```
LoginX("test1", "123123" , "NetType=0")
```

```
LoginX("test1", "123123")
```

通过特殊网络接入（内网、专线等），进行用户认证

```
LoginX("test1", "123123", "NetType=1")
```

该函数在 [GTAQTSInterfaceBase.h](#) 中定义。

4.1.1.7 获取行情服务器连接状态 GetConnectState

使用此接口主动获取与 API 订阅的每个消息类型对应的行情订阅服务器地址的连接状态。

函数原型：

```
RetCode GetConnectState(CDataBuffer<ServerState>& ServerStates)
```

参数：

ServerStates 存放服务器地址对象，包含服务器地址与该地址的连接状态。

返回：

RetCode 错误码枚举类型，含义见表 [5.4 返回码含义列表](#)

说明：

同时订阅多个消息类型，每个消息类型有可能在不同的服务器上。

详情请参考 [6.3 节<消息类型连接状态>](#)

User Mannual

该函数在 [GTAQTSInterfaceBase.h](#) 中定义。

4.1.1.8 获取权限列表 GetDataTypeList

使用此接口获取有权限的数据类型列表。

函数原型：

```
RetCode GetDataTypeList(CDataBuffer<MsgType>& DataTypeList)
```

参数：

DataTypeList 消息类型列表，该列表大小即为有权限的消息类型个数

返回：

RetCode 错误码枚举类型，含义见表 [5.4 返回码含义列表](#)

例子：

```
RetCode errorCode;

CDataBuffer<MsgType>& DataTypeList;

errorCode = GetDataTypeList(DataTypeList);

//然后从 DataTypeList 中取出权限消息类型列表
```

该函数在 [GTAQTSInterfaceBase.h](#) 中定义。

4.1.1.9 订阅实时数据 Subscribe

根据用户的输入订阅相应的实时数据，重复订阅（消息类型相同、代码相同）只会返回一份

User Mannual

数据。对于限制订阅代码数的账号,只支持按代码列表订阅且订阅代码数不超过限制上限时,才会订阅成功。

函数原型：

```
RetCode Subscribe(MsgType msgType, char* pCodeList = NULL)
```

参数：

msgType 需要订阅的消息类型，须指定且每条订阅语句只能指定一种消息类型，消息类型见表 [5.1 消息类型](#)

pCodeList 订阅代码列表，代码间以英文逗号“,”分割，且需正确结束字符串。该参数默认为 NULL，表示订阅当前消息类型下的所有代码。单个代码长度限制为 20 个字节，最多可输入 10000 个代码。

注：订阅时若只输入消息类型不输入代码列表或代码列表为 NULL，取消订阅无论以何种方式取消，都会取消掉该消息类型下的所有代码。

返回：

RetCode 错误码枚举类型，含义见表 [5.4 返回码含义列表](#)

说明：

使用基础行情接口订阅时，若指定的消息类型无权限，则会有消息类型无权限的错误，且订阅不会成功。

对于有代码数限制权限的账号，限制可订阅的代码数以累计订阅的代码数总数为准，当最新

一次订阅的代码数累计数量超过订阅代码数上限权限，则返回超出上限 (Ret_OutLimit); 当取消订阅时，则恢复相应的代码数可订阅数；对于限制代码数的消息类型，按全市场订阅直接返回超出上限 (Ret_OutLimit)。

例子：

订阅上交所 L1 深度所有代码的实时行情数据

```
errCode = Subscribe(Msg_SSEL1_Quotation);
```

结果 : 若消息类型有权限 , errCode 为订阅成功 (Ret_Success) ; 若消息类型无权限 , 则 errCode 为无权限 (Ret_NoPermission)。

// 通过此种方法订阅的数据，取消时无论以何种方式，该消息类型下所有代码都会被取消。

订阅中金所所有代码的实时行情（限制代码订阅数权限）

```
errCode = Subscribe(Msg_CFFEXL2_Quotation);
```

// 当消息类型有权限，且订阅代码数限制为非全量，通过此种方法按全市场代码订阅该消息类型，则返回：超出上限 (Ret_OutLimit)。需要按代码进行订阅且订阅代码数不超过上限，方可成功订阅到数据。

// 正常情况下 errCode 为订阅成功 (Ret_Success)，若消息类型无权限，则 errCode 为无权限 (Ret_NoPermission)。

订阅深交所 L2 深度下部分代码的逐笔成交数据（消息类型有权限）

```
Subscribe(Msg_SZSEL2_Transaction,"000795");
```


订阅深交所 L2 深度下部分代码的逐笔委托数据（消息类型有权限，且订阅代码限制为 2）

```
Subscribe(Msg_SZSEL2_Order,"000795",000001");
```

//当消息类型有权限，且订阅代码数限制为非全量（例如为 2），通过此种方法订阅该消息类型，且累计订阅代码数不超过上限值，则返回订阅成功（Ret_Success）；若累计订阅代码数超过上限进行订阅（例如代码数为 3），则返回返回码：超出上限（Ret_OutLimit）。实际累计订阅的代码数小于或等于该消息类型所限制的上限，方可成功订阅到数据。

先订阅上交所 L2 深度下代码为 600000 的实时行情数据（消息类型有权限），再订阅所有代码的实时行情数据

```
Subscribe(Msg_SSEL2_Quotation,"600000");
```

```
Subscribe(Msg_SSEL2_Quotation);
```

结果：600000 的数据在同一个时间点，只会返回一条。

订阅不存在的代码数据（消息类型有权限）

```
errCode = Subscribe(Msg_SSEL2_Quotaton,'999999');
```

结果：errCode 返回结果为订阅成功（Ret_Success），但是没有数据返回。

订阅

该函数在 [GTAQTSInterfaceBase.h](#) 中定义。

4.1.1.10 取消订阅 Unsubscribe

对于限制订阅代码数的账号权限，可正常按代码或者消息类型取消订阅，不受订阅代码数限制影响。并且取消订阅，将恢复对应数量的可订阅代码数。

函数原型：

指定消息类型取消：

```
RetCode Unsubscribe(msgType, pCodeList = NULL)
```

取消全部已经订阅的内容：

```
RetCode UnsubscribeAll()
```

参数：

msgType	需要取消订阅的消息类型，见表 5.1 消息类型
pCodeList	取消订阅代码列表，代码间以英文逗号“,”分割，且需正确结束字符串。 该参数默认为 NULL，表示取消订阅当前消息类型下的所有代码。单个代码长度限制为 20 个字节，最多可输入 10000 个代码。

返回：

RetCode	错误码枚举类型，含义见表 5.4 返回码含义列表
---------	--

说明：

按消息类型订阅的数据，不管取消订阅以任何方式取消，均为取消全部代码。

例子：

按消息类型取消订阅

```
errCode = Unsubscribe(Msg_SSEL1_Quotation)
```

结果：errCode 为 Ret_Success，上交所 L1 实时行情所有代码的数据都会停止推送（不检测权限）。

```
errCode = Unsubscribe(Msg_SZSEL2_Quotation)
```

结果：errCode 为 Ret_Success，深交所 L2 实时行情所有代码的数据都会停止推送（不检测权限）。

```
errCode = Unsubscribe(Msg_CFFEXL2_Quotation)
```

结果：errCode 为 Ret_Success，中金所 L2 实时行情所有代码的数据都会停止推送（不检测权限）。

按代码和消息类型取消订阅

```
//先订阅部分代码
```

```
Subscribe(Msg_SZSEL2_Quotation,"000001,000045");
```

```
Unsubscribe(Msg_SZSEL2_Quotation,"000001");
```

结果：000001 的数据停止推送，000045 的数据正常推送。

```
//先按消息类型订阅数据
```

```
Subscribe(Msg_SZSEL2_Quotation);
```

```
Unsubscribe(Msg_SZSEL2_Quotation,"000001");
```

结果：整个消息类型的数据都会停止推送。

按代码和消息类型取消订阅（限制代码订阅数权限）

```
//先订阅部分代码
```

```
Subscribe(Msg_SZSEL2_Quotation,"000001,000045,000100,300059");
```

```
Unsubscribe(Msg_SZSEL2_Quotation,"000001");
```

结果：000001 的数据停止推送，000045,000100,300059 的数据正常推送。

//再按消息类型取消订阅数据

```
Unsubscribe(Msg_SZSEL2_Quotation);
```

结果：整个消息类型的数据都会停止推送。

该函数在 [GTAQTSInterfaceBase.h](#) 中定义。

4.1.1.11 查询快照数据 QuerySnap

对于限制订阅代码数的账号权限，可正常查询对应消息类型全市场的快照数据，不受订阅代码数限制影响。

函数原型：

```
RetCode QuerySnap_XXXX(char* pCodeList = NULL, CDataBuffer<XXXX>& ppSnapData )
```

参数：

pCodeList 订阅代码列表，代码间以英文逗号“,”分割，且需正确结束字符串。该参数默认为 NULL，表示订阅当前消息类型下的所有代码。单个代码长度限制为 20 个字节，最多可输入 10000 个代码。

ppSnapData 快照数据

XXXX 消息结构体，其定义可参阅 [5.6 -消息结构体](#)

User Mannual

返回：

RetCode 错误码枚举类型，含义见表 5.4 返回码含义列表

查询接口列表：

```
RetCode QuerySnap_SSEL1_Static (char* pCodeList = NULL, CDataBuffer< SSEL1_Static >&
ppSnapData );            //查询上交所 L1 静态数据

RetCode QuerySnap_SSEL1_Quotation (char* pCodeList =NULL,CDataBuffer<SSEL1_Quotation >&
ppSnapData);            //查询上交所 L1 实时行情

RetCode QuerySnap_SSE_IndexPress (char* pCodeList =NULL,CDataBuffer< SSE_IndexPress >&
ppSnapData);            //查询上交所指数通

RetCode QuerySnap_SZSEL1_Static (char* pCodeList = NULL, CDataBuffer< SZSEL1_Static >&
ppSnapData );            //查询深交所 L1 静态数据

RetCode QuerySnap_SZSEL1_Quotation (char* pCodeList = NULL, CDataBuffer<
SZSEL1_Quotation >& ppSnapData );            //查询深交所 L1 实时行情

RetCode QuerySnap_SSEL2_Static (char* pCodeList = NULL, CDataBuffer< SSEL2_Static >&
ppSnapData );            //查询上交所 L2 静态数据

RetCode QuerySnap_SSEL2_Quotation (char* pCodeList =NULL,CDataBuffer<SSEL2_Quotation >&
ppSnapData);            //查询上交所 L2 实时行情

RetCode QuerySnap_SSEL2_Transaction (char* pCodeList = NULL, CDataBuffer<
SSEL2_Transaction >& ppSnapData );            //查询上交所 L2 逐笔成交

RetCode QuerySnap_SSEL2_Index (char* pCodeList = NULL, CDataBuffer< SSEL2_Index >&
```

User Mannual

```
ppSnapData );      //查询上交所 L2 指数行情

RetCode QuerySnap_SSEL2_Auction (char* pCodeList = NULL, CDataBuffer< SSEL2_Auction >&
ppSnapData );      //查询上交所 L2 集合竞价

RetCode QuerySnap_SSEL2_Overview (char* pCodeList = NULL, CDataBuffer<
SSEL2_Ovreview >& ppSnapData );      //查询上交所 L2 市场总览

RetCode QuerySnap_SSEIOL1_Static (char* pCodeList = NULL, CDataBuffer< SSEIOL1_Static >&
ppSnapData );      //查询上交所个股期权静态数据

RetCode QuerySnap_SSEIOL1_Quotation (char* pCodeList = NULL, CDataBuffer<
SSEIOL1_Quotation >& ppSnapData);      //查询上交所个股期权实时行情

RetCode QuerySnap_SZSEL2_Static (char* pCodeList = NULL, CDataBuffer< SZSEL2_Static >&
ppSnapData );      //查询深交所 L2 静态数据

RetCode QuerySnap_SZSEL2_Quotation (char* pCodeList = NULL, CDataBuffer<
SZSEL2_Quotation >& ppSnapData );      //查询深交所 L2 实时行情

RetCode QuerySnap_SZSEL2_Transaction (char* pCodeList = NULL, CDataBuffer<
SZSEL2_Transaction >& ppSnapData );      //查询深交所 L2 逐笔成交

RetCode QuerySnap_SZSEL2_Index (char* pCodeList = NULL, CDataBuffer< SZSEL2_Index >&
ppSnapData );      //查询深交所 L2 指数行情

RetCode QuerySnap_SZSEL2_Order (char* pCodeList = NULL, CDataBuffer< SZSEL2_Order >&
ppSnapData );      //查询深交所 L2 逐笔委托

RetCode QuerySnap_SZSEL2_Status (char* pCodeList = NULL, CDataBuffer< SZSEL2_Status >&
ppSnapData);      //查询深交所 L2 证券状态

RetCode QuerySnap_CFFEXL2_Static (char* pCodeList = NULL, CDataBuffer< CFFEXL2_Static >&
```

User Mannual

```
ppSnapData ); //查询中金所 L2 静态数据
```

```
RetCode QuerySnap_CFFEXL2_Quotation (char* pCodeList = NULL, CDataBuffer<  
CFFEXL2_Quotation >& ppSnapData); //查询中金所 L2 实时行情
```

```
RetCode QuerySnap_SHFEL1_Static (char* pCodeList = NULL, CDataBuffer< SHFEL1 _Static >&  
ppSnapData ); //查询上期所 L1 静态数据
```

```
RetCode QuerySnap_SHFEL1_Quotation (char* pCodeList = NULL, CDataBuffer< SHFEL1  
_Quotation >& ppSnapData); //查询上期所 L1 实时行情
```

```
RetCode QuerySnap_CZCEL1_Static (char* pCodeList = NULL, CDataBuffer< CZCEL1 _Static >&  
ppSnapData ); //查询郑商所 L1 静态数据
```

```
RetCode QuerySnap_CZCEL1_Quotation (char* pCodeList = NULL, CDataBuffer< CZCEL1  
_Quotation >& ppSnapData); //查询郑商所 L1 实时行情
```

```
RetCode QuerySnap_ESUNNY_Index (char* pCodeList = NULL, CDataBuffer< ESUNNY_Index >&  
ppSnapData ); //查询易盛指数行情
```

```
RetCode QuerySnap_DCEL1_Static (char* pCodeList = NULL, CDataBuffer< DCEL1 _Static >&  
ppSnapData ); //查询大商所 L1 静态数据
```

```
RetCode QuerySnap_DCEL1_Quotation (char* pCodeList = NULL, CDataBuffer< DCEL1  
_Quotation >& ppSnapData); //查询大商所 L1 最优行情
```

```
RetCode QuerySnap_DCEL1_ArbiQuotation (char* pCodeList = NULL, CDataBuffer< DCEL1 _  
ArbiQuotation >& ppSnapData ); //查询大商所 L1 套利行情
```

```
RetCode QuerySnap_DCEL2_Static (char* pCodeList = NULL, CDataBuffer< DCEL2 _Static >&  
ppSnapData); //查询大商所 L2 静态数据
```

```
RetCode QuerySnap_DCEL2_Quotation (char* pCodeList = NULL, CDataBuffer< DCEL2 _
```

```
Quotation >& ppSnapData ); //查询大商所 L2 最优深度行情
```

```
RetCode QuerySnap_ DCEL2_ ArbiQuotation (char* pCodeList = NULL, CDataBuffer< DCEL2 _  
ArbiQuotation >& ppSnapData); //查询大商所 L2 套利深度行情
```

```
RetCode QuerySnap_ DCEL2_ RealTimePrice (char* pCodeList = NULL, CDataBuffer< DCEL2 _  
RealTimePrice >& ppSnapData ); //查询大商所 L2 实时结算价
```

```
RetCode QuerySnap_ DCEL2_ OrderStatisic (char* pCodeList = NULL, CDataBuffer< DCEL2 _  
OrderStatisic >& ppSnapData); //查询大商所 L2 委托统计行情
```

```
RetCode QuerySnap_ DCEL2_ MarchPriceQty (char* pCodeList = NULL, CDataBuffer< DCEL2 _  
MarchPriceQty >& ppSnapData); //查询大商所 L2 分价成交量行情
```

该函数在 [GTAQTSInterfaceBase.h](#) 中定义。

4.1.1.12 获取代码列表 GetStockList

使用此接口获取指定市场下的代码列表。对于限制订阅代码数的账号权限，可正常获取对应消息类型全市场的代码列表，不受订阅代码数限制影响。

函数原型：

```
RetCode GetStockList(char* pMarketTag, CDataBuffer<StockSymbol>& StockList)
```

参数：

pMarketTag 市场列表，可输入单个或多个市场，不同市场列表之间用英文逗号隔开。
市场列表不允许为空。

可输入的市场：

上交所：“SSE”

深交所：“SZSE”

中金所：“CFFEX”

上期所：“SHFE”

StockList

返回的代码列表，带市场标志。

注：

（1）对于上交所，若账号权限仅有 L1（静态数据&实时行情），则获取的上交所代码列表中不包含指数代码。若账号权限有 L2 静态数据，则获取的上交所代码列表中包含指数代码。

（2）上交所个股期权的代码列表包含在上交所中。

（3）上交所指数通、易盛指数不支持获取代码列表

返回：

RetCode

错误码枚举类型，含义见表 5.4 返回码含义列表

该函数在 [GTAQTSInterfaceBase.h](#) 中定义。

4.1.2 回调接口类 IGTAQTSCallbackBase

4.1.2.1 登录状态回调接口 OnLoginState

此接口返回认证结果。

函数原型：

```
void OnLoginState( RetCode errCode)
```

参数：

errCode	错误码，含义见表 5.4 返回码含义列表
---------	--------------------------------------

4.1.2.2 订阅服务连接状态回调接口 OnConnectionState

此接口返回每个消息类型的连接状态。

函数原型：

```
void OnConnectionState(MsgType msgType, RetCode errCode)
```

参数：

msgType	消息类型
errCode	错误码，含义见表 5.4 返回码含义列表 ，在此处表示每个消息类型的连接状态。

说明：

消息类型连接状态是用户订阅的消息类型所在的行情订阅服务器与客户端 API 的连接状态，

并且以与该行情订阅服务器相关的所有消息类型为单位依次返回连接状态。这意味着,有些用户订阅了,但是并没有在当前行情订阅服务器上的消息类型有可能也会返回连接或者断开连接的状态。因此,不能完全依靠此连接状态来判断用户与系统行情订阅服务器之间的连接是否存在。

详情请参考 [6.3 节<消息类型连接状态>](#)

4.1.2.3 实时数据回调 OnSubscribe

此接口向拥有基础接口使用权的用户推送订阅的实时数据,与订阅实时行情基础接口配套使用。

函数原型：

```
void OnSubscribe_XXXX(const XXXX& RealValue)
```

参数：

RealValue 回调接口返回的消息类型为 MsgType 的实时数据。

XXXX 消息结构体,其含义可参阅 [5.6 消息类型结构体](#)

实时数据回调接口列表：

```
OnSubscribe_ SSEL1_Static (const SSEL1_Static & RealValue);        //上交所 L1 静态数据
```

```
OnSubscribe_ SSEL1_Quotation (const SSEL1_Quotation & RealValue);    //上交所 L1 实时行情
```

User Mannual

OnSubscribe_SSE_IndexPress (const SSE_IndexPress & RealValue); //上交所指数通

OnSubscribe_SZSEL1_Static (const SZSEL1_Static & RealValue); //深交所 L1 静态数据

OnSubscribe_SZSEL1_Quotation (const SZSEL1_Quotation & RealValue); //深交所 L1 实时行情

OnSubscribe_SSEL2_Static (const SSEL2_Static & RealValue); //上交所 L2 静态数据

OnSubscribe_SSEL2_Quotation (const SSEL2_Quotation & RealValue); //上交所 L2 实时行情

OnSubscribe_SSEL2_Transaction (const SSEL2_Transaction & RealValue); //上交所 L2 逐笔成交

OnSubscribe_SSEL2_Index (const SSEL2_Index & RealValue); //上交所 L2 指数行情

OnSubscribe_SSEL2_Auction (const SSEL2_Auction & RealValue); //上交所 L2 集合竞价

OnSubscribe_SSEL2_Overview (const SSEL2_Overview & RealValue); //上交所 L2 市场总览

OnSubscribe_SSEIOL1_Static (const SSEIOL1_Static & RealValue); //上交所个股期权静态数据

OnSubscribe_SSEIOL1_Quotation (const SSEIOL1_Quotation & RealValue); //上交所个股期权实时行情

OnSubscribe_SZSEL2_Static (const SZSEL2_Static & RealValue); //深交所 L2 静态数据

OnSubscribe_SZSEL2_Quotation (const SZSEL2_Quotation & RealValue); //深交所 L2 实时行

User Mannual

情

OnSubscribe_SZSEL2_Transaction (const SZSEL2_Transaction &RealValue); //深交所 L2 逐笔

成交

OnSubscribe_SZSEL2_Index (const SZSEL2_Index &RealValue); //深交所 L2 指数行情

OnSubscribe_SZSEL2_Order (const SZSEL2_Order &RealValue); //深交所 L2 逐笔委托

OnSubscribe_SZSEL2_Status (const SZSEL2_Status &RealValue); //深交所 L2 证券状态

OnSubscribe_CFFEXL2_Static (const CFFEXL2_Static & RealValue); //中金所 L2 静态数据

OnSubscribe_CFFEXL2_Quotation (const CFFEXL2_Quotation & RealValue); //中金所 L2 实时

行情

OnSubscribe_SHFEL1_Static (const SHFEL1_Static & RealValue); //上期所 L1 静态数据

OnSubscribe_SHFEL1_Quotation (const SHFEL1_Quotation & RealValue); //上期所 L1 实时行

情

OnSubscribe_CZCEL1_Static (const CZCEL1_Static & RealValue); //郑商所 L1 静态数据

OnSubscribe_CZCEL1_Quotation (const CZCEL1_Quotation & RealValue); //郑商所 L1 实时行

情

OnSubscribe_ESUNNY_Index (const ESUNNY_Index & RealValue); //易盛指数行情

OnSubscribe_DCEL1_Static (const DCEL1_Static & RealValue); //大商所 L1 静态数据

User Mannual

OnSubscribe_ DCEL1_Quotation (const DCEL1_Quotation & RealValue); //大商所 L1 最优行情

OnSubscribe_ DCEL1_ArbiQuotation (const DCEL1_ArbiQuotation & RealValue); //大商所 L1
套利行情

OnSubscribe_ DCEL2_Static (const DCEL2_Static & RealValue); //大商所 L2 静态数据

OnSubscribe_ DCEL2_Quotation (const DCEL2_Quotation & RealValue); //大商所 L2 最优深度
行情

OnSubscribe_ DCEL2_ArbiQuotation (const DCEL2_ArbiQuotation & RealValue); //大商所 L2
套利深度行情

OnSubscribe_ DCEL2_RealTimePrice (const DCEL2_RealTimePrice & RealValue); //大商所
L2 实时结算价

OnSubscribe_ DCEL2_OrderStatisic (const DCEL2_OrderStatisic & RealValue); //大商所 L2 委
托统计行情

OnSubscribe_ DCEL2_MarchPriceQty (const DCEL2_MarchPriceQty & RealValue); //大商所 L2
分价成交量行情

4.2数据类接口 QTSDataFieldDefine.h

数据类接口 [QTSDataFieldDeFine.h](#) 包含了有数据返回的实时接口和回调接口提取信息的方法。

数据缓存模板类 CDataBuffer

[重置缓存大小 ReSize](#)

[返回数据大小 Size](#)

[\[运算符\]返回数据首地址 T*](#)

4.2.1 数据缓存模板类 CDataBuffer

4.2.1.1 重置缓存大小 ReSize

该函数接受新的缓存容量值，若新的缓存容量值大于原来的容量值，则重置容量值为新值。

函数原型：

```
void ReSize(int nSize)
```

参数：

nSize 新的缓存值。

例子：

```
CDataBuffer DBuffer;
```

```
nSize = 32;
```

```
DBuffer.ReSize(nSize);
```

如果原来的缓存容量大小小于 32 字节，缓存就会被重置为 32 字节，否则缓存容量不变化。

该函数在 [QTSDataFieldDeFine.h](#) 中定义。

4.2.1.2 返回数据大小 Size

该函数返回 CDataBuffer 对象的大小。

函数原型：

int Size()

例子：

```
CDataBuffer<ServerState>& ServerStates;
```

```
GetConnectState(ServerStates);
```

```
int mSize = ServerStates. Size();
```

mSize 为 ServerStates 的大小。

该函数在 [QTSDDataFieldDeFine.h](#) 中定义。

4.2.1.3 [运算符]返回数据首地址 T*

函数原型：

operator T *()

例子：

```
CDataBuffer< SSEIOL1_Static >& ppSnapData );
```

```
QuerySnap_SSEIOL1_Static(NULL, ppSnapData );
```

```
Point = T* ppSnapData;
```

Point 指向 ppSnapData 申请的内存的首地址。

该函数在 [QTSDDataFieldDeFine.h](#) 中定义。

5. 数据定义

此部分针对用户在使用 QTS2.6 时需要用到的概念及相关数据进行解释 ,主要包含以下几点 :

消息类型 MsgType

消息类型适用于基础 API 的订阅、取消订阅及查询 ,由“市场+深度+(交易所)消息名”组成 ,此部分说明本系统所支持的消息类型。

市场列表 MarketTag

网络状态含义 ConnectState

此部分包含对网络状态返回码的解释。

返回码含义列表 RetCode

此部分包含对返回码的解释。

消息结构体的字段类型 FIELD_TYPE

此部分包含对基础 API 返回的消息结构体中额字段类型进行说明。

消息结构体 (基础 API 适用)

此部分包含对基础 API 返回的消息结构体 ,并列举了每个结构体中应包含的数据及相应的解释。

5.1消息类型 MsgType:QTSDDataType.h

消息类型是指各个市场 (及深度) 和及该市场下所提供的数据类型的组合 ,交易所加深度(若有)加数据类型组成一个消息类型。当前系统所支持的消息类型及其具体含义如下表。

表 5.1 消息类型

消息类型	枚举变量	枚举值	对应消息结构体
错误消息类型	Msg_Unknown	0x00000000	N/A
上交所L1静态数据	Msg_SSEL1_Static	0x00101000	SSEL1_Static

User Mannual

上交所L1实时行情	Msg_SSEL1_Quotation	0x00101001	SSEL1_Quotation
上交所指数通	Msg_SSE_IndexPress	0x0010100D	SSE_IndexPress
深交所L1静态数据	Msg_SZSEL1_Static	0x00201000	SZSEL1_Static
深交所L1实时行情	Msg_SZSEL1_Quotation	0x00201001	SZSEL1_Quotation
上交所L2静态数据	Msg_SSEL2_Static	0x00102000	SSEL2_Static
上交所L2实时行情	Msg_SSEL2_Quotation	0x00102001	SSEL2_Quotation
上交所L2逐笔成交	Msg_SSEL2_Transaction	0x00102002	SSEL2_Transaction
上交所L2指数行情	Msg_SSEL2_Index	0x00102003	SSEL2_Index
上交所L2虚拟集合竞价	Msg_SSEL2_Auction	0x00102004	SSEL2_Auction
上交所L2市场总览	Msg_SSEL2_Ovreview	0x00102005	SSEL2_Ovreview
上交所个股期权静态数	Msg_SSEIOL1_Static	0x00103000	SSEIOL1_Static
上交所个股期权实时行	Msg_SSEIOL1_Quotation	0x00103001	SSEIOL1_Quotation
深交所L2静态数据	Msg_SZSEL2_Static	0x00202000	SZSEL2_Static
深交所L2实时行情	Msg_SZSEL2_Quotation	0x00202001	SZSEL2_Quotation
深交所L2逐笔成交	Msg_SZSEL2_Transaction	0x00202002	SZSEL2_Transaction
深交所L2指数行情	Msg_SZSEL2_Index	0x00202003	SZSEL2_Index
深交所L2逐笔委托	Msg_SZSEL2_Order	0x00202006	SZSEL2_Order
深交所L2证券状态	Msg_SZSEL2_Status	0x00202007	SZSEL2_Status
中金所L2静态数据	Msg_CFFEXL2_Static	0x00302000	CFFEXL2_Static
中金所L2实时行情	Msg_CFFEXL2_Quotation	0x00302001	CFFEXL2_Quotation
上期所L1静态数据	Msg_SHFEL1_Static	0x00401000	SHFEL1_Static
上期所L1实时行情	Msg_SHFEL1_Quotation	0x00401001	SHFEL1_Quotation
郑商所L1静态数据	Msg_CZCEL1_Static	0x00501000	CZCEL1_Static
郑商所L1实时行情	Msg_CZCEL1_Quotation	0x00501001	CZCEL1_Quotation
易盛指数行情	Msg_ESUNNY_Index	0x00501003	ESUNNY_Index
大商所L1静态数据	Msg_DCEL1_Static	0x00601000	DCEL1_Static
大商所L1最优行情	Msg_DCEL1_Quotation	0x00601001	DCEL1_Quotation
大商所L1套利行情	Msg_DCEL1_ArbiQuotation	0x00601008	DCEL1_ArbiQuotation
大商所L2静态数据	Msg_DCEL2_Static	0x00602000	DCEL2_Static
大商所L2最优深度行情	Msg_DCEL2_Quotation	0x00602001	DCEL2_Quotation

User Manual

大商所L2套利深度行情	Msg_DCEL2_ArbiQuotation	0x00602008	DCEL2_ArbiQuotation
大商所L2实时结算价	Msg_DCEL2_RealTimePrice	0x00602009	DCEL2_RealTimePrice
大商所L2委托统计行情	Msg_DCEL2_OrderStatistic	0x0060200A	DCEL2_OrderStatistic
大商所L2分价成交量行	Msg_DCEL2_MarchPriceQty	0x0060200B	DCEL2_MarchPriceQty

该枚举类型在 [QTSDDataType.h](#) 中定义

5.2 市场列表 MarketTag:QTSDDataType.h

下表为目前 QTS 系统所能提供行情的交易所，上交所个股期权已纳入上交所中。

表 5.2 市场列表

市场	枚举变量	枚举值
错误市场	Market_Unknown	0
上交所	Market_SSE	1
深交所	Market_SZSE	2
中金所	Market_CFFEX	3
上期所	Market_SHFE	4
郑商所	Market_CZCE	5
大商所	Market_DCE	6

该枚举类型在 [QTSDDataType.h](#) 中定义

5.3 网络状态含义 ConnectState:QTSDDataType.h

表 5.3 网络状态含义列表

网络状态	枚举变量	枚举值
正在连接	Connecting	0
已连接	Connected	1
正在关闭	ConnectClosing	2
已关闭	ConnectClosed	3

该枚举类型在 [QTSDDataType.h](#) 中定义

5.4 返回码含义列表 RetCode:QTSDDataType.h

调用 QTS 接口时，若接口返回类型为 RetCode，用户可根据下表查出 RetCode 的含义。

表 5.4 返回码含义列表

含义	枚举变量	枚举值
失败	Ret_Error	-1
成功	Ret_Success	0
请先设置行情服务器地址	Ret_NoAddress	1
没有权限，请联系客服	Ret_NoPermission	2
参数无效	Ret_ParamInvalid	3
账号或密码错误	Ret_AccountError	4
账号不在有效期内	Ret_AccountOutDate	5
连接失败	Ret_ConnectFail	6
账号重复登录	Ret_LoginRepeat	7
超时	Ret_OutTime	8

User Mannual

连接断开	Ret_CloseConnect	9
超出订阅代码数上限	Ret_OutLimit	10
版本过低	Ret_LowVersion	11

该枚举类型在 [QTSDDataType.h](#) 中定义

5.5 消息结构体的字段类型 FIELD_TYPE:QTSDDataType.h

下表包含了本系统规定的字段类型。

表 5.5 消息结构体的字段类型

字段类型	字段类型枚举变量	枚举值
未知类型	Type_ERROR	0
char	Type_Char	1
short	Type_Short	2
int	Type_Int	3
Unsigned int	Type_UInt	4
long long	Type_Longlong	5
unsigned long long	Type_ULonglong	6
string	Type_String	7
double	Type_Double	8
queue队列（如50档队列）	Type_Sequence	9

该枚举类型在 [QTSDDataType.h](#) 中定义

5.6 消息结构体 (基础 API 适用) :QTSStruct.h

[上交所 L1 静态数据 SSEL1_Static](#)

[上交所 L1 实时行情 SSEL1_Quotation](#)

[上交所指数通 SSE_IndexPress](#)

[深交所 L1 静态数据 SZSEL1_Static](#)

[深交所 L1 实时行情 SZSEL1_Quotation](#)

[上交所个股期权静态数据 SSEIOL1_Static](#)

[上交所个股期权实时行情 SSEIOL1_Quotation](#)

[上交所 L2 静态数据 SSEL2_Static](#)

[上交所 L2 实时行情 SSEL2_Quotation](#)

[上交所 L2 虚拟集合竞价 SSEL2_Auction](#)

[上交所 L2 指数行情 SSEL2_Index](#)

[上交所 L2 市场总览 SSEL2_Overview](#)

[上交所 L2 逐笔成交 SSEL2_Transaction](#)

[深交所 L2 静态数据 SZSEL2_Static](#)

[深交所 L2 证券状态 SZSE L2_Status](#)

[深交所 L2 实时行情 SZSEL2_Quotation](#)

[深交所 L2 指数行情 SZSEL2_Index](#)

[深交所 L2 逐笔委托 SZSEL2_Order](#)

[深交所 L2 逐笔成交 SZSEL2_Transaction](#)

[中金所 L2 静态数据 Msg_CFFEXL2_Static](#)

[中金所 L2 实时行情 Msg_CFFEXL2_Quotation](#)

[上期所 L1 静态数据 Msg_SHFEL1_Static](#)

[上期所 L1 实时行情 Msg_SHFEL1_Quotation](#)

[郑商所 L1 静态数据 Msg_CZCEL1_Static](#)

[郑商所 L1 实时行情 Msg_CZCEL1_Quotation](#)

[易盛指数行情 Msg_ESUNNY_Index](#)

[大商所 L1 静态数据 Msg_DCEL1_Static](#)

[大商所 L1 最优行情 Msg_DCEL1_Quotation](#)

[大商所 L1 套利行情 Msg_DCEL1_ArbiQuotation](#)

[大商所 L2 静态数据 Msg_DCEL2_Static](#)

[大商所 L2 最优深度行情 Msg_DCEL2_Quotation](#)

[大商所 L2 套利深度行情 Msg_DCEL2_ArbiQuotation](#)

[大商所 L2 实时结算价 Msg_DCEL2_RealTimePrice](#)

[大商所 L2 委托统计行情 Msg_DCEL2_OrderStatic](#)

[大商所 L2 分价成交量行情 Msg_DCEL2_MarchPriceQty](#)

5.6.1 上交所 L1 静态数据 SSEL1_Static

表 5.6 上交所 L1 静态数据表

字段名	字段含义	用户类型	Tag	描述
LocalTime	采集时间	int32	1120	数据到达时本系统记录的时间，精确到毫秒

User Mannual

Quotation Flag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填空格：第 1 位对应：‘1’ 表示上海机房行情源；‘2’ 表示深圳机房行情源
Time	数据生成时间	int32	1191	标识接口中本记录更新时间 HH:MM:SS
Symbol	证券代码	string(40)	1187	
ISINCode	ISIN 代码	String(12)	1080	交易所预留
Security Name	证券名称	string(20)	1145	UTF-8 编码
SecurityEN	英文证券名称	string(24)	1143	交易所预留
Symbol Underlying	基础证券代码	string(20)	1189	
MarketType	市场种类	string(6)	1100	‘ASHR’ 表示 A 股市场；‘BSHR’ 表示 B 股市场；‘CSHR’ 表示国际版市场。
CFICode	证券类别	string(6)	1043	‘ES’ 表示股票；‘EU’ 表示基金；‘D’ 表示债券；‘RWS’ 表示权证；‘FF’ 表示期货。
SecuritySubType	证券子类别	string(6)	1150	自定义详细证券类别，参考：(1) GBF 国债；(2) GBZ 无息国债；(3) DST 国债分销（仅用于分销阶段）；(4) DVP 公司债（地方债）分销；(5) CBF 企业债券；(6) CCF 可转换企业债券；(7) CPF 公司债券（或地方债券）；(8) FBF 金融机构发行债券；(9) CRP 质押式国债回购；(10) BRP 质押式企债回购；(11) ORP 买断式债券回购；(12) CBD 分离式可转债；(13) OBD 其它债券；(14) CEF 封闭式基金；(15) OEF 开放式基金；(16) EBS 交易所交易基金（买卖）；(17) FBL 跨市场/跨境资金；(18) OFN 其它基金；(19) ASH 以人民币交易的股票；(20) BSH 以美元交易的股票；(21) CSH 国际版股票；(22) OEQ 其它股票；(23) CIW 企业发行权证；(24) COV 备兑权证；(25) FEQ 个股期货；(26) FBD 债券期货；(27) OFT 其它期货；(28) AMP 集合资产管理计划；(29) WIT 国债预发行；(30) LOF LOF 基金；(31) OPS 公开发行优先股；(32) PPS 非公开发行优先股；(33) QRP 报价回购；
Currency	币种	string(5)	1061	美元：USD；人民币：CNY

User Mannual

ParValue	面值	Double	1128	3 位有效小数位数, 债券当前面值, 单位元, 其他产品取
Tradable MarketNo	可 流 通 未 上市数量	int64	1063	交易所预留字段
EndDate	最后 交 易 日期	int 32	1063	对于国债预发行产品, 为最后交易日期, 格式为 YYYYMMDD
ListingDate	上市日期	int 32	1085	在上交所首日交易日期, YYYYMMDD
SetNo	产 品 集 编 号	uint32	1180	取值范围从 1 到 999,。用来表明产品的一种分组方式, 用于在多主机间记性负载均衡分配。该值在一个交易日 内不会变化。
BuyVolumeUni t	买 数 量 单 位	uint32	1041	买订单的申报数量必须是该字段的整数倍。
SellVolumeUni t	卖 数 量 单 位	uint32	1178	卖订单的申报数量必须是该字段的整数倍。
DeclareVolum eFloor	申 报 量 下 限	uint32	1106	申报数量下限
DeclareVolum eCeiling	申 报 量 上 限	uint32	1103	申报数量上限
PreClose Price	昨收价	double	1131	3 位有效小数位数; 前收盘价格 (如有除权除息, 为除权 除息后的收盘价); 对于货币市场基金实时申赎, 取值为
TickSize	最 小 报 价	double	1136	3 位有效小数位数。申报价格的最小变动单位
UpDown LimitType	涨 跌 幅 限 制类型	string(1)	1138	‘N’ 表示交易规则 3.4.13 规定的有涨跌幅限制类型或 者权证管理办法第 22 条规定; ‘R’表示交易规则 3.4.15 和 3.4.16 规定的无涨跌幅限制类型; ‘S’表示回购涨跌 幅控制类型。
PriceUp Limit	涨幅价格	double	1139	3 位有效小数位数。对于 N 类型涨跌幅限制的产品, 该 字段当日不会更改, 基于前收盘价 (已首日上市交易产 品为发行价) 计算; 对于 R 类型五涨跌幅限制的产品, 该字段取开盘时基于参考价格计算的上限价格。
PriceDown	跌幅价格	double	1135	3 位有效小数位数。计算方式参考涨幅上限价格
XRRatio	除权比例	double	1233	6 位有效小数位数。每股送股比例 ; 对于国债预发行产品,
XDAmount	除息金额	double	1232	6 位有效小数位数。每股分红金额
CrdBuy Underlying	融 资 标 的 标志	string(1)	1057	‘ T ’表示是融资标的证券; ‘ F ’表示不是融资标的证券。

User Mannual

CrdSell Underlying	融 券 标 的 标志	string(1)	1060	‘ T ’表示是融券标的证券；‘ F ’表示不是融券标的证券。
Security Status	产 品 状 态 标识	string(20)	1148	该字段为 20 位字符串，每位表示允许对应的业务，无定义则填空格。 第 0 位对应：‘ N ’表示首日上市。 第 1 位对应：‘ D ’表示除权。 第 2 位对应：‘ R ’表示除息。 第 3 位对应：‘ D ’表示国内主板正常交易产品；‘ S ’表示股票风险警示产品；‘ P ’表示退市整理产品；‘ T ’表示退市转让产品；‘ U ’表示优先股产品。 第 4 位对应：‘ S ’表示债券风险警示类。第 4 位未来改为不启用，此改动暂未实施，具体以交易所通知为准。 第 5 位对应：‘ L ’表示债券投资者适当性要求类。

该消息类型结构体在 QTSStruct.h 中定义

5.6.2 上交所 L1 实时行情 SSEL1_Quotation

表 5.8 上交所 L1 实时行情表

字段名	字段含	用户类型	Tag	描述
LocalTime Stamp	采 集 时 间	int32	1120	数据到达时本系统记录的时间，精确到毫秒
QuotationFlag	行 情 源 标志	String(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填空格：第 1 位对应：‘ 1 ’表示上海机房行情源；‘ 2 ’表示深圳机房行情源

User Mannual

PacketTimeStamp	包头时间	int64	1125	交易所发包时间，格式为：日期时间YYYYMMDDHHMMSSMMM
Time	数据生成时间	int32	1191	表示成交时间，HHMMssmmm。发生成交，时间错才会刷新。
Symbol	证券代码	string(40)	1187	
Security Name	证券简称	string(20)	1145	UTF-8 编码
SecurityType	证券类别	INT32	1362	1：表示指数行情数据； 2：表示股票（A、B 股）行情数据； 3：表示债券行情数据； 4：表示基金行情数据；
OpenPrice	开盘价	double	1118	指数：4 位有效小数位；股票基金债券：3 位有效小数位
HighPrice	最高价	double	1074	指数：4 位有效小数位；股票基金债券：3 位有效小数位
LowPrice	最低价	double	1088	指数：4 位有效小数位；股票基金债券：3 位有效小数位
LastPrice	现价	double	1083	指数：4 位有效小数位；股票基金债券：3 位有效小数位
TotalVolume	成交总量	UINT64	1202	成交量，单位：张或股 股票为股，基金为份，债券与回购为手，权证为 100 份； 对于指数行情数据，表示参与计算相应指数的交易数量， 股票的指数交易数量单位是 100 股，基金指数的交易数量单位是 100 份，债券指数的交易数量单位是手。
TotalNo	成交笔数	uint64	1197	

User Mannual

Total Amount	成交总额	double	1192	<p>成交金额，单位：元</p> <p>当产品代码为债券（国债、企债、可转债）时，由于债券交易的数量以手为单位，成交金额为该债券每笔成交的价格*数量*10 的总和；当产品代码为席位质押式国债回购代码 201***、席位质押式企业债回购代码 202***以及账户质押式国债回购代码 204***时，成交金额为 100*成交数量*10；</p> <p>当产品代码为买断式国债回购代码 203***时，成交金额为其基础产品昨日收盘价 *成交数量*10；对于指数行情数据，表示参与计算相应指数的成交金额，单位均为人民币元。</p>
PreClose Price	昨收价	double	1131	指数：4 位有效小数位；股票基金债券：3 位有效小数位
ClosePrice	今收盘	double	1046	指数：4 位有效小数位；股票基金债券：3 位有效小数位
SellPrice01	申卖价	double	1157	3 位有效小数位
SellVolume 01	申卖量	UINT64	1168	
SellPrice02	申卖价	double	1158	3 位有效小数位
SellVolume 02	申卖量	UINT64	1169	
SellPrice03	申卖价	double	1159	3 位有效小数位
SellVolume 03	申卖量	UINT64	1170	
SellPrice04	申卖价	double	1160	3 位有效小数位
SellVolume 04	申卖量	UINT64	1171	
SellPrice05	申卖价	double	1161	3 位有效小数位
SellVolume 05	申卖量	UINT64	1172	
BuyPrice01	申买价	double	1019	3 位有效小数位
BuyVolume01	申买量	UINT64	1031	
BuyPrice02	申买价	double	1020	3 位有效小数位
BuyVolume02	申买量	UINT64	1032	
BuyPrice03	申买价	double	1021	3 位有效小数位
BuyVolume03	申买量	UINT64	1033	
BuyPrice04	申买价	double	1022	3 位有效小数位
BuyVolume04	申买量	UINT64	1034	
BuyPrice05	申买价	double	1023	3 位有效小数位

User Mannual

BuyVolume05	申 买 量	UINT64	1035	
NAV	基 金 T-1 日	double	1114	3 位有效小数位。对基金有意义，取前一日 nav 净值。
IOPV	ETF 净 值估值	double	1079	3 位有效小数位。对基 金有意义
SecurityPhaseTag	当 前 品 种 交 易 状态	string(8)	1147	<p>该字段为 8 位字符串，左起每位表示特定的含义，无定义则填空格。</p> <p>第 1 位：‘S’ 表示启动（开市前）时段，‘C’ 表示集合竞价时段，‘T’ 表示连续交易时段，‘B’ 表示休市时段，‘E’ 表示闭市时段，‘P’ 表示产品停牌，‘M’ 表示可恢复交易的熔断时段（盘中集合竞价），‘N’ 表示不可恢复交易的熔断时段（暂停交易至闭市），‘D’ 表示集合竞价阶段结束到连续竞价阶段开始之前的时段（如有）。</p> <p>第 2 位：‘0’ 表示此产品不可正常交易，‘1’ 表示此产品可正常交易，无意义填空格。</p> <p>第 3 位：‘0’ 表示未上市，‘1’ 表示已上市。</p> <p>第 4 位：‘0’ 表示此产品在当前时段不接受进行新订单申报，‘1’ 表示此产品在当前时段可接受进行新订单申报。无意义填空格。</p>

该消息类型结构体在 QTSStruct.h 中定义

注：2.5 版本上交所 L1 实时行情新增字段：**SecurityType** 证券类别、**TotalNo** 成交笔数

5.6.3 上交所指数通行行情 SSE_IndexPress

表 5.9 上交所指数通行行情表

字段名	字段含	用户类型	Tag	描述
LocalTime Stamp	采 集 时 间	int32	1120	数据到达时本系统记录的时间，精确到毫秒

User Mannual

QuotationFlag	行情源标志	String(4)	1280	该字段为 4 位字符串,起每位表示特定的含义,无定义则填空格:第 1 位对应: '1' 表示上海机房行情源; '2' 表示深圳机房行情源
PacketTimeStamp	包头时间	int64	1125	交易所发包时间,格式为:日期时间 YYYYMMDDHHMMSSMMM
TradeDate	交易日	INT32	1208	行情文件所代表交易日期,内容为被用于计算的那天的交易日期。日期的格式为
ActionDay	业务发生日	INT32	1285	行情文件所代表的自然日期(北京时间),内容为被用于计算的那天的自然日期(北京时间)。日期的格式为 "YYYYMMDD"。如交易日为 2012 年 1 月 19 日
Time	数据生成时间	int32	1191	行情文件的更新时间戳(北京时间),格式为 "HHMMSS"。
Symbol	指数代码	string(40)	1187	
SecurityName	指数简	string(40)	1145	
MarketType	市场代码	string(6)	1100	1: 上证所; 2: 深交所; 3: 沪深; 4: 香港; 5: 亚太; 0: 全球
LastPrice	最新价	double	1083	当日开盘值,当前交易日开盘指数值。初始值为 0.0000。
OpenPrice	开盘价	double	1118	
HighPrice	最高价	double	1074	
LowPrice	最低价	double	1088	
ClosePrice	今收盘价	double	1046	当日收盘值,当前交易日收盘值。初始值为 0.0000。当值不为 0.0000 时,说明指数已收盘。
PreClosePrice	昨收价	double	1131	
PriceUpdown1	涨跌	double	1282	LastPrice-PreClosePrice
UpDownRate	涨跌幅	double	1286	

User Mannual

TotalVolume	成交总	UINT64	1202	单位：股
TotalAmount	成交金	double	1192	单位：万元
ExchangeRate	汇率	double	1329	汇 率 ， 该 汇 率 在 盘 中 时 为 0.00000000 ， 收
Currency	币种标	string(5)	1061	使用货币。 0：人民币； 1：港币； 2：
DisplayNum	指数展	INT32	1344	展示指数的顺序按指数展示序号进行排列。
ClosePrice2	当日收	double	1346	当日收盘值 2，若该指数为全球指数，该收盘值为当日亚
ClosePrice3	当日收 盘值 3	double	1347	当日收盘值 3，若该指数为全球指数，该收盘值为当日欧 洲区收盘值。初始值为 0.0000。当值不为 0.0000 时， 说明指数欧洲区已收盘。

5.6.4 深交所 L1 静态数据 SZSEL1_Static

表 5.10 深交所 L1 静态数据表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32	1120	格式为：HHMMSSsss
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充格。 第 1 位对应：' 1' 表示上海机房行情源，' 2' 表示深圳机房行情源
Symbol	证券代码	string(40)	1187	
Security Name	证券名称	string(40)	1145	可能包含中文字符，表示最多 40 个 UTF-8 字符
SymbolSource	证券代码源	string(5)	1188	102 = 深圳证券交易所
Security EN	证券英文简称	string(40)	1143	

User Mannual

ISINCode	ISIN 代码	string(12)	1080	
Symbol Underlying	基础证券代码	string(20)	1189	
UnderlyingSecurityIDSource	基础证券代码源	string(4)	1361	102 = 深圳证券交易所
Security Type	证券类别代码	INT32	1362	1 主板 A 股； 2 中小板股票； 3 创业板股票； 4 主板 B 股； 5 国债（含地方债）； 6 企业债； 7 公司债； 8 可转债； 9 私募债； 10 可交换私募债； 11 证券公司次级债； 12 质押式回购； 13 资产支持证券； 14 本市场股票 ETF； 15 跨市场 ETF； 16 跨境 ETF； 17 本市场实物债券 ETF； 18 现金债券 ETF； 19 黄金 ETF； 20 货币 ETF； 21（预留）杠杆 ETF； 22（预留）商品期货 ETF； 23 标准 LOF； 24 分级子基金； 25 封闭式基金； 26 仅申赎基金； 28 权证； 29 个股期权； 30 ETF 期权； 33 优先股； 34 证券公司短期债； 35 可交换公司债
Security StatusTag	证券状态标识	string(20)	1364	该字段为 20 位字符串（后六位备用），每位表示特定的含义，“1”表示位数有业务意义，“0”表示该位数无业务意义。第 1 位对应：“1”表示停牌；第 2 位对应：“1”表示除权；第 3 位对应：“1”表示除息；第 4 位对应：“1”表示 ST；第 5 位对应：“1”表示*ST；第 6 位对应：“1”表示上市首日；第 7 位对应：“1”表示公司再融资；第 8 位对应：“1”表示恢复上市首日；第 9 位对应：“1”表示网络投票；第 10 位对应：“1”表示退市整理期；第 12 位对应：“1”表示增发股份上市；第 13 位对应：“1”表示合约调整；第 14 位对应：“1”表示暂停上市后协议转让。
PreClose Price	昨收价	double	1131	现货 4 位有效小数位；指数 5 位有效小数位
ListingDate	上市日期	INT32	1085	

User Mannual

Currency	币种	string(5)	1061	币种：CNY = 人民币；HKD = 港币
ParValue	每股面值	double	1128	
IssuedVolume	总发行量	double	1082	2 位有效小数位
Outstanding Share	流通股数	double	1044	2 位有效小数位
Industry Type	行业种类	string(5)	1075	
PreYearEPS	上年每股利润	double	1134	
YearEPS	本年每股利润	double	1235	
Offering Flag	收购(转股、行权) 标志	string(1)	1117	股票：要约收购；债券、优先股：转股回售；Y=是 N=否
NAV	基金 T-1 日 累计净值	double	1114	
Coupon Rate	票面利率	double	1055	
IssuePrice	贴现发行价	double	1369	
Interest	每百元应计利息	double	1370	对于优先股：8 位小数， 0.0000 表示浮动股息率
Interest Accrual Date	起息日	INT32	1076	
Maturity Date	到期交割日	INT32	1101	

User Mannual

Conversi onPrice	行权价格	double	1053	
Conversi onRatio	行权比例	double	1054	
Conversi on BeginDa te	行权开始日	INT32	1051	
Conversi onEndD ate	行权结束日	INT32	1052	
CallOrPu t	认购认沽	string(1)	1042	C = Call ; P = Put
Warrant Clearing Type	权证结算方 式	string(1)	1219	S = 证券结算 C = 现金结算
Clearing Price	结算价格	double	1396	适用于权证
OptionT ype	行权类型	string(1)	1119	A = 美式 E = 欧式 B = 百慕大式
EndDate	最后交易日	int32	1063	
Expiratio nDays	购回期限	int32	1371	
DayTradi ng	回转交易标 志	string (1)	1363	是否支持当日回转交易：Y=支持 N=不支持
GageFla g	保证金证券 标志	string(1)	1365	是否可作为融资融券可充抵保证金证券：Y=是 N=否
GageRat e	担保品折算 率	double	1073	

User Mannual

CrdBuyUnderlying	融资标的标志	string(1)	1057	Y=是 N=否
CrdSellUnderlying	融券标的标志	string(1)	1060	Y=是 N=否
CrdPriceCheckType	提价检查方式	string(2)	1058	0=不检查；1=不低于最近成交价；2=不低于昨收价；3=不低于最高叫买；4=不低于最低叫卖
PledgeFlag	质押入库标志	string(1)	1366	是否可质押入库:Y=是 N=否
ContractMultiplier	债券折回购标准券比例	double	1048	
RegularShare	对应回购标准券	string(8)	1367	
QualificationFlag	投资者适当性管理标志	string(1)	1368	是否需要对该证券作投资者适当性管理 Y=是 N=否
MarketMakerFlag	做市商标志	string(1)	1095	标识是否有做市商 Y=是 N=否
RoundLot	整手数	double	1142	2 位有效小数位。对于某一证券申报的委托，其委托数量字段必须为该证券数量单位的整数倍
TickSize	最小报价单位	double	1136	
BuyQtyUpperLimit	买数量上限	double	1388	2 位有效小数位。买委托数量的上限
SellQtyUpper	卖数量上限	double	1389	2 位有效小数位。卖委托数量的上限

User Mannual

Limit				
BuyVolumeUnit	买数量单位	double	1041	2 位有效小数位。每笔买委托的委托数量必须是买数量单位的整数倍
SellVolumeUnit	卖数量单位	double	1178	2 位有效小数位。每笔卖委托的委托数量必须是卖数量单位的整数倍
LimitUpRateO	开盘集合竞价上涨幅度	double	1372	
LimitDownRateO	开盘集合竞价下跌幅度	double	1373	
LimitUpAbsoluteO	开盘集合竞价上涨限价	double	1374	预留字段，深交所未有实际应用
LimitDownAbsoluteO	开盘集合竞价下跌限价	double	1375	预留字段，深交所未有实际应用
AuctionUpDownRateO	开盘集合竞价有效范围涨跌幅度	double	1376	
AuctionUpDownAbsoluteO	开盘集合竞价有效范围涨跌价格	double	1377	
LimitUpRateT	连续竞价上涨幅度	double	1378	
LimitDownRateT	连续竞价下跌幅度	double	1379	
LimitUpAbsoluteT	连续竞价上涨限价	double	1394	预留字段，深交所未有实际应用

User Mannual

LimitDown AbsoluteT	连续竞价下跌限价	double	1395	预留字段，深交所未有实际应用
AuctionUp DownRateT	连续竞价有效范围涨跌幅度	double	1380	
AuctionUp DownAbsoluteT	连续竞价有效范围涨跌价格	double	1381	
LimitUp RateC	收盘集合竞价上涨幅度	double	1382	
LimitDown RateC	收盘集合竞价下跌幅度	double	1383	
LimitUp AbsoluteC	收盘集合竞价上涨限价	double	1384	预留字段，深交所未有实际应用
LimitDown AbsoluteC	收盘集合竞价下跌限价	double	1385	预留字段，深交所未有实际应用
AuctionUp DownRateC	收盘集合竞价有效范围涨跌幅度	double	1386	
AuctionUp DownAbsoluteC	收盘集合竞价有效范围涨跌价格	double	1387	

该消息类型结构体在 QTSStruct.h 中定义

5.6.5 深交所 L1 实时行情 SZSEL1_Quotation

表 5.11 深交所 L1 实时行情表

字段名	字段含义	用户类型	Tag	描述
LocalTimeStamp	采集时间	INT32	1120	格式为：HHMMSSsss
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充格。 第 1 位对应：' 1' 表示上海机房行情源，' 2' 表示深圳机房行情源
Time	数据生成时间	INT64	1191	数据生成时间 YYYYMMDDHHMMSSsss
Symbol	证券代码	string(40)	1187	
SymbolSource	证券代码源	string(5)	1188	102=深圳证券交易所 103=香港交易所
PreClosePrice	昨收价	double	1131	
OpenPrice	开盘价	double	1118	
LastPrice	现价	double	1083	
HighPrice	最高价	double	1074	
LowPrice	最低价	double	1088	
PriceUpLimit	涨停价	double	1139	
PriceDownLimit	跌停价	double	1135	
PriceUpdown1	升降一	double	1282	LastPrice-PreClosePrice ;

User Mannual

PriceUpdown 2	涨跌二	double	1200	LastPrice-上一笔 LastPrice ; 对于当天的第一笔成交, 该字段 =LastPrice-PreClosePrice
TotalNo	成交笔数	uint64	1197	
TotalVolume	成交总量	double	1202	两位有效小数(深交所数量均带两位小数);单位:股票 为股,基金为份,债券为张,指数为股。
TotalAmount	成交总额	double	1192	
ClosePrice	今收盘价	double	1046	QTS2.4 将静态数据文件中的现货证券收盘行情 cashsecurityclosemd.xml 中的收盘价填充到实时行情 以提供收盘价信息(在最后一笔行情数据基础上填充收 盘价,且原有数据生成时间基础上加 1 秒作为新的 time),但 cashsecurityclosemd.xml 文件交易所 3 点 半左右才推送导致最后一笔收盘数据较晚推送。
SecurityPhaseTag	当前品种交易状态	string(8)	1147	产品所处的交易阶段代码: 第 0 位: S=启动(开市前) O=开盘集合竞价 T=连续 B=休市 C=收盘集合竞价 E=已闭市 H=临时停牌 A=盘后交易 V=波动性中断 第 1 位: 0=正常状态 1=全天停牌
PERatio1	市盈率 1	double	1129	
PERatio2	市盈率 2	double	1130	
NAV	基金 T-1 日净值	double	1114	基金 T-1 日净值
IOPV	基金实时参考净值	double	1079	基金实时参考净值(包括 ETF 的 IOPV)

User Mannual

PremiumRate	权证溢价率	double	1391	
SampleNo	样本个数	UINT32	1273	统计量指标样本个数,成交量统计指标代码详见附录 7.3
SellPrice01	申卖价一	double	1157	
SellVolume01	申卖量一	double	1168	两位有效小数(深交所数量均带两位小数)
SellPrice02	申卖价二	double	1158	
SellVolume02	申卖量二	double	1169	两位有效小数(深交所数量均带两位小数)
SellPrice03	申卖价三	double	1159	
SellVolume03	申卖量三	double	1170	两位有效小数(深交所数量均带两位小数)
SellPrice04	申卖价四	double	1160	
SellVolume04	申卖量四	double	1171	两位有效小数(深交所数量均带两位小数)
SellPrice05	申卖价五	double	1161	
SellVolume05	申卖量五	double	1172	两位有效小数(深交所数量均带两位小数)
BuyPrice01	申买价一	double	1019	
BuyVolume01	申买量一	double	1031	两位有效小数(深交所数量均带两位小数)
BuyPrice02	申买价二	double	1020	
BuyVolume02	申买量二	double	1032	两位有效小数(深交所数量均带两位小数)
BuyPrice03	申买价三	double	1021	
BuyVolume03	申买量三	double	1033	两位有效小数(深交所数量均带两位小数)
BuyPrice04	申买价四	double	1022	

User Mannual

BuyVolume04	申买量四	double	1034	两位有效小数（深交所数量均带两位小数）
BuyPrice05	申买价五	double	1023	
BuyVolume05	申买量五	double	1035	两位有效小数（深交所数量均带两位小数）

该消息类型结构体在 QTSStruct.h 中定义

5.6.6 上交所个股期权静态数据 SSEIOL1_Static

表 5.12 上交所个股期权静态数据表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32	1120	数据到达时本系统记录的时间，精确到毫秒
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填空格。 第 1 位对应：' 1 ' 表示上海机房行情源，' 2 ' 表示深圳机房行情源
PacketTime Stamp	包头时间	INT64	1125	交易所发包时间，格式为：日期时间 YYYYMMDDHHMMSSMMM
Symbol	证券代码	string(40)	1187	合约期权产品代码，8 位字符;唯一标示
ContractID	合约交易代码	string(19)	1047	合约交易代码 17 位，按以下顺序编写： 1、第 1 至第 6 位为数字，取标的证券代码，如工商银行 601398，取“601398”； 2、第 7 位为 C(Call)或者 P(Put)，分别表示认购期权或者认沽期权； 3、第 8、9 位表示到期年份； 4、第 10、11 位表示到期月份； 5、第 12 位期初设为“M”，表示月份合约。当合约首次调整后，“M”修改为“A”，以表示该合约被

User Mannual

				调整过一次，如发生第二次调整，则“A”修改为“B”、“M”修改为“A”，以此类推； 6、第 13 至 17 位表示期权行权价格.留两位备用
SecurityName	期权合约简称	string(20)	1145	UTF-8 编码
Symbol Underlying	基础证券代码	string(20)	1189	期权标的证券代码
NameUnderlying	基础证券名称	string(20)	1113	UTF-8 编码
Underlying Type	标的证券类型	string(5)	1217	EBS – ETF , ASH – A 股
OptionType	行权类型	string(1)	1119	若为欧式期权，则本字段为“E”；若为美式期权，则本字段为“A”
CallOrPut	认购认沽	string(1)	1042	认购，则本字段为“C”；若为认沽，则本字段为“P”
Contract MultiplierUnit	合约单位	int32	1049	经过除权除息调整后的合约单位，一定是整数
ExercisePrice	期权行权价	double	1071	4 位有效小数位。经过除权除息调整后的期权行权价，精确到 0.0001 元
StartDate	首个交易日	int32	1184	期权首个交易日,YYYYMMDD (年*10000+月*100+日)
EndDate	最后交易日	int32	1063	期权最后交易日/行权日，YYYYMMDD
ExerciseDate	期权行权日	int32	1070	期权行权日，YYYYMMDD
DeliveryDate	行权交割日	int32	1062	行权交割日，默认为行权日的下一个交易日，YYYYMMDD
ExpireDate	期权到期日	int32	1072	期权到期日，YYYYMMDD
Version	合约版本号	string(1)	1218	期权合约的版本号。新挂合约是‘1’

User Mannual

TotalLong Position	合约未平仓 数	UINT32	1196	单位是（张），此值为前一交易日日终持仓轧差对冲之后持仓数据
PreClosePri ce	昨收价	double	1131	昨日收盘价，4 位有效小数位。如遇除权除息则为调整后的收盘价格（上市首日的文件中，填写参考价格），右对齐，精确到 0.0001 元。
PreSettlePri ce	昨结算	double	1133	昨日结算价，4 位有效小数位。如遇除权除息则为调整后的结算价（合约上市首日填写参考价），右对齐，单位：元（精确到 0.0001 元。）
PreClosePri ce Underlying	标的昨收价	double	1132	4 位有效小数位。期权标的证券除权除息调整后的前收盘价格，右对齐，单位：元（精确到 0.0001 元）
UpDownLi mit Type	涨跌幅限制 类型	string(1)	1138	‘N’ 有涨跌幅限制类型
PriceUpLim it	涨幅价格	double	1139	4 位有效小数位。当日期权涨停价格，单位：元（精确到 0.1 厘）
PriceDown Limit	跌幅价格	double	1135	4 位有效小数位。当日期权跌停价格，单位：元（精确到 0.1 厘）
MarginUnit	单位保证金	double	1091	2 位有效小数位。当日持有一张合约所需要的保证金数量，精确到分
MarginRati oParam1	保证金计算 参数一	double	1089	2 位有效小数位。保证金计算参数，单位：%
MarginRati oParam2	保证金计算 参数二	double	1090	2 位有效小数位。保证金计算参数，单位：%
RoundLot	整手数	UINT32	1142	一手等于几张合约
LmtOrdFlo or	限价申报下 限	int64	1087	单笔限价申报的申报张数下限。
LmtOrdCeil ing	限价申报上 限	int64	1086	单笔限价申报的申报张数上限。

User Mannual

MktOrdFloor	市价申报下限	int64	1112	单笔市价申报的申报张数下限。
MktOrdCeiling	市价申报上限	int64	1111	单笔市价申报的申报张数上限。
TickSize	最小报价单位	double	1136	4 位有效小数位。单位：元（精确到 0.1 厘）
SecurityStatus Flag	期权合约状态	string(8)	1149	该字段为 8 位字符串，左起每位表示特定的含义，无定义则填充空格。第 1 位：‘0’ 表示可开仓，‘1’ 表示限制卖出开仓（包括备兑开仓）和买入开仓。 第 2 位：‘0’ 表示未连续停牌，‘1’ 表示连续停牌。 第 3 位：‘0’ 表示未临近到期日，‘1’ 表示距离到期日不足 5 个交易日。 第 4 位：‘0’ 表示近期末做调整，‘1’ 表示最近 5 个交易日内合约发生过调整。 第 5 位：‘A’ 表示当日新挂牌的合约，‘E’ 表示存续的合约，‘D’ 表示当日摘牌的合约。 第 6 位：‘1’ 表示合约不能进行垂直价差组合策略，‘0’ 表示可以进行所有的组合策略
AutoSplitDate	垂直价差组合策略到期解除日期	int32	1283	该日期表示垂直价差组合策略到期解除的日期，YYYYMMDD； 对于 E-2 后波动加挂的合约，此字段仍为 E-2 日。到期日调整时，此字段信息自动同步调整。

该消息类型结构体在 QTSStruct.h 中定义

5.6.7 上交所个股期权实时行情 SEIOL1_Quotation

表 5.13 上交所个股期权实时行情表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32	1120	数据到达时本系统记录的时间，精确到毫秒
Quotation	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充空格。

User Mannual

Flag				第 1 位对应：' 1' 表示上海机房行情源，' 2' 表示深圳机房行情源
PacketTime Stamp	包头时间	INT64	1125	交易所发包时间，格式为：日期时间 YYYYMMDDHHMMSSMMM
Time	数据生成时间	int32	1191	时间戳 143025 表示 14:30:25：（1）如果期权合约的产品代码为“00000000”，则表示行情时间，格式为 HHMMSS；（2）如果为其他合约代码，则表示该期权的行情时间（最后更新时间），如行情时间为 000000 则表示该期权尚未更新更新过
Symbol	合约代码	string(40)	1187	期权合约的产品代码;唯一标示
PreSettlePrice	昨结算	double	1133	4 位有效小数位
SettlePrice	结算价	double	1181	4 位有效小数位。QTS2.0 将个股期权收盘价格文件 clpr03MMDD.txt 中的结算价填充到实时行情以提供结算价，但收盘价文件交易所 4 点半左右才推送导致最后一笔结算数据较晚推送。
OpenPrice	开盘价	double	1118	4 位有效小数位
HighPrice	最高价	double	1074	4 位有效小数位
LowPrice	最低价	double	1088	4 位有效小数位
LastPrice	现价	double	1083	4 位有效小数位 如果期权合约的产品代码为“00000000”，则表示记录数
ClosePrice	收盘价	double	1046	单位为元，精确到 0.0001 元。 QTS2.0 将个股期权收盘价格文件 clpr03MMDD.txt 中的收盘价填充到实时行情以提供收盘价，但收盘价文件交易所 4 点半左右才推送导致最后一笔收盘数据较晚推送。个股期权收盘价也可以通过闭市后最后一笔数据的最新价获取。
AuctionPrice	波动性中断参考价	double	1077	4 位有效小数位

User Mannual

Auction Volume	波动性中断 虚拟匹配量	int64	1078	
TotalPosition	持仓量	uint64	1198	单位是（张），交易时间代表此处总持仓量值为持仓余额轧差之前的数值;收盘后为闭市轧差后的未平仓合约数量。
BuyLevelNo	买盘价位数量	UINT32	1015	五档:总是 5，空档位用 0 填充
BuyLevel	申买五档	BuySellLevelInfo3	--	以下编号从 01 至 05 的申买价、申买量包含在此结构中。
BuyPrice01	申买价 1	Double	1019	4 位有效小数位。在集合竞价时间段，申买价一和申卖价一中同时为虚拟动态参考价格，即根据集合竞价算法计算得出的虚拟撮合价格；
BuyVolume01	申买量 1	UINT64	1031	在集合竞价时间段，申买量一和申卖量一分别为行情发布时刻的买方和卖方虚拟匹配量。
BuyPrice02	申买价 2	Double	1020	4 位有效小数位
BuyVolume02	申买量 2	UINT64	1032	在集合竞价时间段，申买量二和申卖量二分别为行情发布时刻的买方和卖方虚拟未匹配量。
BuyPrice03	申买价 3	Double	1021	4 位有效小数位
BuyVolume03	申买量 3	UINT64	1033	
BuyPrice04	申买价 4	Double	1022	4 位有效小数位
BuyVolume04	申买量 4	UINT64	1034	
BuyPrice05	申买价 5	Double	1023	4 位有效小数位
BuyVolume	申买量 5	UINT64	1035	

User Mannual

e05				
SellLevelNo	卖盘价位数量	uint32	1153	五档:总是 5，空档位用 0 填充
SellLevel	申卖五档	BuySellLevelInfo3	--	以下编号从 01 至 05 的申卖价、申卖量和申卖总委托笔数均包含在此结构体中。
SellPrice01	申卖价 1	Double	1157	4 位有效小数位。在集合竞价时间段，申买价一和申卖价一中同时为虚拟动态参考价格，即根据集合竞价算法计算得出的虚拟撮合价格；
SellVolume01	申卖量 1	UINT64	1168	在集合竞价时间段，申买量一和申卖量一分别为行情发布时刻的买方和卖方虚拟匹配量。
SellPrice02	申卖价 2	Double	1158	4 位有效小数位
SellVolume02	申卖量 2	UINT64	1169	在集合竞价时间段，申买量二和申卖量二分别为行情发布时刻的买方和卖方虚拟未匹配量。
SellPrice03	申卖价 3	Double	1159	4 位有效小数位
SellVolume03	申卖量 3	UINT64	1170	
SellPrice04	申卖价 4	Double	1160	4 位有效小数位
SellVolume04	申卖量 4	UINT64	1171	
SellPrice05	申卖价 5	Double	1161	4 位有效小数位
SellVolume05	申卖量 5	UINT64	1172	
TotalVolume	成交总量	UINT64	1202	当日累计成交总量，如果期权合约的产品代码为“00000000”，则表示行情日期，格式为 YYYYMMDD

User Mannual

TotalAmount	成交总额	Double	1192	2 位有效小数位
SecurityPhaseTag	交易状态	string(8)	1147	<p>该字段为 4 位字符串，左起每位表示特定的含义，无定义则填充空格。</p> <p>第 1 位：‘S’ 表示启动（开市前）时段，‘C’ 表示集合竞价时段，‘T’ 表示连续交易时段，‘B’ 表示休市时段，‘E’ 表示闭市时段，‘V’ 表示波动性中断，‘P’ 表示临时停牌、‘U’ 表示收盘集合竞价。</p> <p>‘M’ 表示可恢复交易的熔断（盘中集合竞价），‘N’ 表示不可恢复交易的熔断（暂停交易至闭市）</p> <p>第 2 位：‘0’ 表示未连续停牌，‘1’ 表示连续停牌。（预留，暂填充格）。</p> <p>第 3 位：‘0’ 表示不限制开仓，‘1’ 表示限制备兑开仓，‘2’ 表示卖出开仓，‘3’ 表示限制卖出开仓、备兑开仓，‘4’ 表示限制买入开仓，‘5’ 表示限制买入开仓、备兑开仓，‘6’ 表示限制买入开仓、卖出开仓，‘7’ 表示限制买入开仓、卖出开仓、备兑开仓。</p> <p>第 4 位：‘0’ 表示此产品在当前时段不接受进行新订单申报，‘1’ 表示此产品在当前时段可接受进行新订单申报。</p>

该消息类型结构体在 QTSStruct.h 中定义

5.6.8 上交所 L2 静态数据 SSEL2_Static

表 5.14 上交所 L2 静态数据表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32	1120	数据到达时本系统记录的时间，精确到毫秒
	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定

User Mannual

QuotationFlag				义则填充格。 第 1 位对应：' 1' 表示上海机房行情源，' 2' 表示深圳机房行情源
PacketTime Stamp	包头时间	INT64	1125	交易所发包时间，格式为：日期时间 YYYYMMDDHHMMSSMMM
Time	数据生成时间	int32	1191	标识接口中本记录更新时间 HH:MM:SS
Symbol	证券代码	string(40)	1187	
ISINCode	ISIN 代码	String(12)	1080	
SecurityName	证券名称	string(20)	1145	UTF-8 编码
SecurityEN	英文证券名称	string(20)	1143	
SymbolUnderlying	基础证券代码	string(20)	1189	
MarketType	市场种类	string(6)	1100	'ASHR'表示 A 股市场；'BSHR'表示 B 股市场；'CSHR'表示国际版市场。
CFIcode	证券类别	string(6)	1043	'ES' 表示股票；'EU' 表示基金；'D' 表示债券；'RWS' 表示权证；'FF' 表示期货。
SecuritySub Type	证券子类别	string(6)	1150	自定义详细证券类别，参考：(1) GBF 国债；(2) GBZ 无息国债；(3) DST 国债分销（仅用于分销阶段）；(4) DVP 公司债（地方债）分销；(5) CBF 企业债券；(6) CCF 可转换企业债券；(7) CPF 公司债券（或地方债券）；(8) FBF 金融机构发行债券；(9) CRP 质押式国债回购；(10) BRP 质押式企债回购；(11) ORP 买断式债券回购；(12) CBD 分离式可转债；(13) OBD 其它债券；(14) CEF 封闭式基金；(15) OEF 开放式基金；(16) EBS 交易所交易基金（买卖）；(17) FBL 跨市场/跨境资金；(18) OFN 其它基金；(19) ASH 以人民币交易的股票；(20) BSH 以美元交易的股票；(21) CSH 国际版股票；(22) OEQ 其它股票；(23) CIW 企业发行权证；(24) COV 备兑权证；(25) FEQ 个股

User Mannual

				期货 ; (26) FBD 债券期货 ; (27) OFT 其 它 期 货 ; (28) AMP 集合资产管理计划 ; (29) WIT 国 债预发行 ; (30) LOF LOF 基金 ; (31) OPS 公 开发行优先股 ; (32) PPS 非公开发行优先股 ; (33) QRP 报价回购 ;
Currency	币种	string(5)	1061	美元 : USD ; 人民币 : CNY
ParValue	面值	Double	1128	3 位有效小数位数, 债券当前面值, 单位元, 其他产 品取 0.000
TradableNo	可流通未上 市数量	INT64	1205	交易所预留字段
EndDate	最后交易日 期	int 32	1063	对于国债预发行产品, 为最后交易日期, 格式为 YYYYMMDD
ListingDate	上市日期	INT32	1085	在上交所首日交易日期, YYYYMMDD
SetNo	产品集编号	UINT32	1180	取值范围从 1 到 999,。用来表明产品的一种分组方 式, 用于在多主机间进行负载均衡分配。该值在一 个交易日内不会变化。
BuyVolume Unit	买数量单位	UINT32	1041	买订单的申报数量必须是该字段的整数倍。
SellVolume Unit	卖数量单位	UINT32	1178	卖订单的申报数量必须是该字段的整数倍。
DeclareVolu meFloor	申报量下限	UINT32	1106	申报数量下限
DeclareVolu meCeiling	申报量上限	UINT32	1103	申报数量上限
PreClosePric e	昨收价	Double	1131	3 位有效小数位数, ; 前收盘价格 (如有除权除息, 为除权除息后的收盘价) ; 对于货币市场基金实时申赎, 取值为 0.010
TickSize	最小报价单 位	Double	1136	3 位有效小数位数, 申报价格的最小变动单位

User Mannual

UpDownLimit Type	涨跌幅限制类型	string(1)	1138	<p>‘N’表示交易规则规定的有涨跌幅限制类型或者权证管理办法第 22 条规定；</p> <p>‘R’表示交易规则规定的无涨跌幅限制类型；‘S’表示回购涨跌幅控制类型。</p>
PriceUpLimit	涨幅价格	double	1139	<p>3 位有效小数位数，对于 N 类型涨跌幅限制的产品，该字段当日不会更改，基于前收盘价（已首日上市交易产品为发行价）计算；</p> <p>对于 R 类型无涨跌幅限制的产品，该字段取开盘时基于参考价格计算的上限价格。</p>
PriceDownLimit	跌幅价格	Double	1135	3 位有效小数位数，计算方式参考涨幅上限价格
XRRatio	除权比例	Double	1233	6 位有效小数位数，每股送股比例；对于国债预发行产品，为保证金比例。
XDAmount	除息金额	Double	1232	6 位有效小数位数，每股分红金额
CrdBuyUnderlying	融资标的标志	string(1)	1057	‘T’表示是融资标的证券；‘F’表示不是融资标的证券。
CrdSellUnderlying	融券标的标志	string(1)	1060	‘T’表示是融券标的证券；‘F’表示不是融券标的证券。
SecurityStatus	产品状态标识	string(20)	1148	<p>该字段为 20 位字符串，每位表示允许对应的业务，无定义则填充格。</p> <p>第 0 位对应：‘N’表示首日上市。</p> <p>第 1 位对应：‘D’表示除权。</p> <p>第 2 位对应：‘R’表示除息。</p> <p>第 3 位对应：‘D’表示国内主板正常交易产品，‘S’表示股票风险警示产品，‘P’表示退市整理产品，‘T’表示退市转让产品，‘U’表示优先股产品。</p> <p>第 4 位对应：‘S’表示债券风险警示类。第 4 位未来改为不启用，此改动暂未实施，具体以交易所通知为准。</p> <p>第 5 位对应：‘L’表示债券投资者适当性要求类。</p>
SampleNo	样本数量	UINT32	1273	重点指数当前的样本数量

User Mannual

SampleAvgPrice	样本均价	double	1274	重点指数当前样本的均价 市价总值/发行股本 若该指标不统计则输出 N/A
TradeAmount	成交金额	double	1190	重点指数当前样本的当日成交金额（单位：亿元）
AvgCapital	平均股本	double	1275	重点指数当前样本的平均股本（算术平均，单位：亿股） 若该指标不统计则输出 N/A
TotalMarketValue	总市值	double	1276	重点指数当前样本的总市值汇总（算术和，单位：万亿元） 若该指标不统计则输出 N/A
MarketValueRatio	占比%	double	1277	重点指数当前样本的总市值占上证综指全样本的总市值，百分比 若该指标不统计则输出 N/A
StaticPERatio	静态市盈率	double	1278	重点指数当前样本的静态市盈率。公式：合计（人民币收盘价*发行量）/合计（每股收益*发行量） 若该指标未统计则输出 N/A
IndexLevelStatus	指数级别标识	string(20)	1279	前 5 位为指数排序数值，最后一位即指数级别信息： 1 为重点指数；2 为全貌指数；其他可根据需要扩展。

该消息类型结构体在 QTSStruct.h 中定义

5.6.9 上交所 L2 实时行情 SSEL2_Quotation

表 5.15 上交所 L2 实时行情表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32	1120	数据到达时本系统记录的时间，精确到毫秒

User Mannual

Quotation Flag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充空格。 第 1 位对应：' 1 ' 表示上海机房行情源，' 2 ' 表示深圳机房行情源
PacketTime Stamp	包头时间	INT64	1125	交易所发包时间，格式为：日期时间 YYYYMMDDHHMMSSMMM
Time	数据生成时间	int32	1191	最新订单时间（秒）;143025 表示 14:30:25
Symbol	证券代码	string(20)	1187	
PreClosePrice	昨收价	double	1131	
OpenPrice	开盘价	double	1118	
HighPrice	最高价	double	1074	
LowPrice	最低价	double	1088	
LastPrice	现价	double	1083	
ClosePrice	今收盘价	double	1046	
TradeStatus	当前品种交易状态	string(8)	1212	具体含义可参考附录- 当前品种交易状态
SecurityPhaseTag	当前产品状态	string(8)	1147	该字段为 8 位字符串，左起每位表示特定的含义，无定义则填充空格。 第 1 位：'S' 表示启动（开市前）时段，'C' 表示集合竞价时段，'T' 表示连续交易时段，'B' 表示休市时段，'E' 表示闭市时段，'P' 表示产品停牌，'M' 表示可恢复交易的熔断时段（盘中集合竞价），'N' 表示不可恢复交易的熔断时段（暂停交易至闭市），'D' 表示集合竞价阶段结束到连续竞价阶段开始之前的时段（如有）。 第 2 位：'0' 表示此产品不可正常交易，'1' 表示此产品可正常交易，无意义填充空格。 第 3 位：'0' 表示未上市，'1' 表示已上市。

User Mannual

				第 4 位：‘0’ 表示此产品在当前时段不接受进行新订单申报，‘1’ 表示此产品在当前时段可接受进行新订单申报。无意义填充格。
TotalNo	成交笔数	UINT64	1197	
TotalVolume	成交总量	UINT64	1202	股票：股； 权证：份； 债券：手
TotalAmount	成交总额	double	1192	(元)
TotalBuyOrderVolume	委托买入总量	UINT64	1195	股票：股； 权证：份； 债券：手
WtAvgBuyPrice	加权平均委托买价	double	1230	(元) 非债券代码适用
BondWtAvgBuyPrice	债券加权平均委托买价	double	1013	(元) 债券代码适用
TotalSellOrderVolume	委托卖出总量	UINT64	1201	
WtAvgSellPrice	加权平均委托卖价	double	1231	(元) 非债券代码适用
BondWtAvgSellPrice	债券加权平均委托卖价	double	1014	(元) 债券代码适用
IOPV	ETF 净值估值	double	1079	
ETFBuyNo	ETF 申购笔数	INT32	1065	
ETFBuyVolume	ETF 申购量	int64	1066	

User Mannual

ETFBuyAmount	ETF 申购额	double	1064	
ETFSellNo	ETF 赎回笔数	INT32	1068	
ETFSellVolume	ETF 赎回量	int64	1069	
ETFSellAmount	ETF 赎回额	double	1067	
YTM	债券到期收益率	double	1236	
TotalWarrantExecVol	权证执行的总数量	int64	1203	
WarrantDownLimit	权证跌停价格	double	1221	(元)
WarrantUpLimit	权证涨停价格	double	1223	(元)
WithdrawBuyNo	买入撤单笔数	INT32	1225	
WithdrawBuyVolume	买入撤单量	int64	1226	
WithdrawBuyAmount	买入撤单额	double	1224	
WithdrawSellNo	卖出撤单笔数	INT32	1228	
WithdrawSellVolume	卖出撤单量	int64	1229	

User Mannual

Withdraw SellAmount	卖出撤单额	double	1227	
TotalBuyNo	买入总笔数	INT32	1193	
TotalSellNo	卖出总笔数	INT32	1199	
MaxBuy Duration	买入成交最大等待时间	int32	1102	未成交时则显示为-1
MaxSell Duration	卖出成交最大等待时间	int32	1104	未成交时则显示为-1
BuyOrder No	买方委托价位数	int32	1018	
SellOrder No	卖方委托价位数	int32	1156	
SellLevelNo	卖盘价位数量	UINT32	1153	10 档行情，不足时补空
SellLevel	申卖十档	BuySellLevelInfo3	--	以下编号从 01 至 10 的申卖价、申卖量和申卖总委托笔数均包含在此结构体中。
SellPrice01	申卖价	double	1157	
SellVolume01	申卖量	UINT64	1168	
TotalSellOrder No01	卖出总委托笔数	UINT32	1261	
SellPrice02	申卖价	double	1158	
SellVolume02	申卖量	UINT64	1169	

User Mannual

TotalSellOrder No02	卖出总委托笔数	UINT32	1262	
SellPrice03	申卖价	double	1159	
SellVolume03	申卖量	UINT64	1170	
TotalSellOrder No03	卖出总委托笔数	UINT32	1263	
SellPrice04	申卖价	double	1160	
SellVolume04	申卖量	UINT64	1171	
TotalSellOrder No04	卖出总委托笔数	UINT32	1264	
SellPrice05	申卖价	double	1161	
SellVolume05	申卖量	UINT64	1172	
TotalSellOrder No05	卖出总委托笔数	UINT32	1265	
SellPrice06	申卖价	double	1162	
SellVolume06	申卖量	UINT64	1173	
TotalSellOrder No06	卖出总委托笔数	UINT32	1266	
SellPrice07	申卖价	double	1163	

User Mannual

SellVolume07	申卖量	UINT64	1174	
TotalSellOrder No07	卖出总委托笔数	UINT32	1267	
SellPrice08	申卖价	double	1164	
SellVolume08	申卖量	UINT64	1175	
TotalSellOrder No08	卖出总委托笔数	UINT32	1268	
SellPrice09	申卖价	double	1165	
SellVolume09	申卖量	UINT64	1176	
TotalSellOrder No09	卖出总委托笔数	UINT32	1269	
SellPrice10	申卖价	double	1166	
SellVolume10	申卖量	UINT64	1177	
TotalSellOrder No10	卖出总委托笔数	UINT32	1270	
SellLevelQueueNo01	卖一档揭示委托笔数	UINT32	1271	卖一档队列中揭示的笔数。为 0 表示不揭示
SellLevelQueue	卖 1 档队列	UINT32	1154	
BuyLevelNo	买盘价位数量	UINT32	1015	10 档行情，不足时补空

User Mannual

BuyLevel	申买十档	BuySellLevelInfo3	--	以下编号从 01 至 10 的申买价、申买量和申买总委托笔数均包含在此结构体中。
BuyPrice01	申买价	double	1019	
BuyVolume01	申买量	UINT64	1031	股票：股； 权证：份； 债券：手
TotalBuyOrder No01	买入总委托笔数	UINT32	1251	
BuyPrice02	申买价	double	1020	
BuyVolume02	申买量	UINT64	1032	
TotalBuyOrder No02	买入总委托笔数	UINT32	1252	
BuyPrice03	申买价	double	1021	
BuyVolume03	申买量	UINT64	1033	
TotalBuyOrder No03	买入总委托笔数	UINT32	1253	
BuyPrice04	申买价	double	1022	
BuyVolume04	申买量	UINT64	1034	
TotalBuyOrder No04	买入总委托笔数	UINT32	1254	
BuyPrice05	申买价	double	1023	

User Mannual

BuyVolume05	申买量	UINT64	1035	
TotalBuyOrder No05	买入总委托笔数	UINT32	1255	
BuyPrice06	申买价	double	1024	
BuyVolume06	申买量	UINT64	1036	
TotalBuyOrder No06	买入总委托笔数	UINT32	1256	
BuyPrice07	申买价	double	1025	
BuyVolume07	申买量	UINT64	1037	
TotalBuyOrder No07	买入总委托笔数	UINT32	1257	
BuyPrice08	申买价	double	1026	
BuyVolume08	申买量	UINT64	1038	
TotalBuyOrder No08	买入总委托笔数	UINT32	1258	
BuyPrice09	申买价	double	1027	
BuyVolume09	申买量	UINT64	1039	
TotalBuyOrder No09	买入总委托笔数	UINT32	1259	

User Mannual

BuyPrice10	申买价	double	1028	
BuyVolume10	申买量	UINT64	1040	
TotalBuyOrder No10	买入总委托笔数	UINT32	1260	
BuyLevelQueueNo01	买一档揭示委托笔数	UINT32	1272	买一档队列中揭示的笔数。为 0 表示不揭示
BuyLevelQueue	买 1 档队列	UINT32	1016	50 档，不足时补空

该消息类型结构体在 QTSStruct.h 中定义

注：2.4.2 版本上交所 L2 实时行情新增字段：**SecurityPhaseTag** 当前产品状态

5.6.10 上交所 L2 虚拟集合竞价 SSEL2_Auction

表 5.16 上交所 L2 虚拟集合竞价表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32	1120	数据到达时本系统记录的时间，精确到毫秒
Quotation Flag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充格。 第 1 位对应：' 1' 表示上海机房行情源，' 2' 表示深圳机房行情源
PacketTime Stamp	包头时间	INT64	1125	交易所发包时间，格式为：日期时间 YYYYMMDDHHMMSSMMM
Time	数据生成时间	int32	1191	143025 表示 14:30:25

User Mannual

Symbol	证券代码	string(40)	1187	
Open Price	虚拟开盘参考价	double	1118	
AuctionVolume	虚拟匹配量	int64	1009	
LeaveVolume	虚拟未匹配量	int64	1084	
Side	买卖方向	string(1)	1183	0=无未匹配量，买卖两边的未匹配量都为 0; 1=买方有未匹配量，卖方未匹配量=0; 2=卖方有未匹配量，买方未匹配量=0

该消息类型结构体在 QTSStruct.h 中定义

5.6.11 上交所 L2 指数行情 SSE_L2_Index

表 5.17 上海 L2 指数行情表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32	1120	数据到达时本系统记录的时间，精确到毫秒
Quotation Flag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充空格。 第 1 位对应：' 1' 表示上海机房行情源，' 2' 表示深圳机房行情源
PacketTimeStamp	包头时间	INT64	1125	交易所发包时间，格式为：日期时间 YYYYMMDDHHMMSSMMM
Time	数据生成时间	int32	1191	143025 表示 14:30:25
TradeTime	成交时间	int32	1213	

User Mannual

Symbol	证券代码	string(40)	1187	
PreClosePrice	昨收价	double	1131	
OpenPrice	开盘价	double	1118	
TotalAmount	成交总额	double	1192	
HighPrice	最高价	double	1074	
LowPrice	最低价	double	1088	
LastPrice	现价	double	1083	
TotalVolume	成交总量	UINT64	1202	单位 :股。其中债券类指数需要除以 10 后单位为张。
ClosePrice	今收盘价	double	1046	

该消息类型结构体在 QTSStruct.h 中定义

5.6.12 上交所 L2 市场总览 SSEL2_Overview

表 5.18 上交所 L2 市场总览表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32	1120	数据到达时本系统记录的时间，精确到毫秒
Quotation Flag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充格。 第 1 位对应：' 1' 表示上海机房行情源，' 2' 表示深圳机房行情源
PacketTime Stamp	包头时间	INT64	1125	交易所发包时间，格式为：日期时间 YYYYMMDDHHMMSSMMM

User Mannual

Time	数据生成时间	int32	1191	143025 表示 14:30:25
Symbol	证券代码	string(40)	1187	
MarketTime	市场时间	int32	1099	百分之一秒
TradeDate	市场日期	int32	1208	

该消息类型结构体在 QTSStruct.h 中定义

5.6.13 上交所 L2 逐笔成交 SSEL2_Transaction

表 5.19 上交所 L2 逐笔成交表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32	1120	数据到达时本系统记录的时间，精确到毫秒
Quotation Flag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充格。 第 1 位对应：' 1' 表示上海机房行情源，' 2' 表示深圳机房行情源
PacketTimeStamp	包头时间	INT64	1125	交易所发包时间，格式为：日期时间 YYYYMMDDHHMMSSMMM
TradeTime	成交时间	int32	1213	14302506 表示 14:30:25.06
RecID	业务索引	UINT32	1140	从 1 开始，按 TradeChannel 连续
TradeChannel	成交通道	int32	1206	
Symbol	证券代码	string(40)	1187	
TradePrice	成交价格	double	1210	

User Mannual

TradeVolume	成交数量	UINT32	1215	股票：股;权证、基金：份;债券：手
TradeAmount	成交金额	double	1190	非债券代码（Symbol 非 0,1,2 开头）： TradeAmount=TradePrice*TradeVolume 非回购债券（Symbol 非 0,1 开头）： TradeAmount=TradePrice*TradeVolume*10 回购债券代码（Symbol 以 2 开头）： TradeAmount=TradePrice*TradeVolume*1000
BuyRecID	买方订单号	int64	1029	
SellRecID	卖方订单号	int64	1167	
BuySellFlag	内外盘标志	string(1)	1030	B – 外盘,主动买;S – 内盘,主动卖;N – 未知

该消息类型结构体在 QTSStruct.h 中定义

5.6.14 深交所 L2 静态数据 SZSEL2_Static

表 5.20 深交所 L2 静态数据表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32	1120	格式为：HHMMSSsss
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充格。 第 1 位对应：' 1' 表示上海机房行情源，' 2' 表示深圳机房行情源
Symbol	证券代码	string(40)	1187	

User Mannual

Security Name	证券名称	string(40)	1145	可能包含中文字符，表示最多 40 个 UTF-8 字符
SymbolSource	证券代码源	string(5)	1188	102 = 深圳证券交易所
Security EN	证券英文简称	string(40)	1143	
ISINCode	ISIN 代码	string(12)	1080	
Symbol Underlying	基础证券代码	string(20)	1189	
UnderlyingSecurityIDSource	基础证券代码源	string(4)	1361	102 = 深圳证券交易所
Security Type	证券类别代码	INT32	1362	1 主板 A 股；2 中小板股票；3 创业板股票；4 主板 B 股；5 国债（含地方债）；6 企业债；7 公司债；8 可转债；9 私募债；10 可交换私募债；11 证券公司次级债；12 质押式回购；13 资产支持证券；14 本市场股票 ETF；15 跨市场 ETF；16 跨境 ETF；17 本市场实物债券 ETF；18 现金债券 ETF；19 黄金 ETF；20 货币 ETF；21（预留）杠杆 ETF；22（预留）商品期货 ETF；23 标准 LOF；24 分级子基金；25 封闭式基金；26 仅申赎基金；28 权证；29 个股期权；30ETF 期权；33 优先股；34 证券公司短期债；35 可交换公司债
Security StatusTag	证券状态标识	string(20)	1364	该字段为 20 位字符串（后六位备用），每位表示特定的含义，“1”表示位数有业务意义，“0”表示该位数无业务意义。第 1 位对应：“1”表示停牌；第 2 位对应：“1”表示除权；第 3 位对应：“1”表示除息；第 4 位对应：“1”表示 ST；第 5 位对应：“1”表示*ST；第 6 位对应：“1”表示上市首日；第 7 位对应：“1”表示公司再融资；第 8 位对应：“1”表示恢复上市首日；第 9 位对应：“1”表示网络投票；第 10 位对应：“1”表示退市整理期；第 12 位对应：“1”表示增发股份上市；第 13 位

User Mannual

				对应：“1”表示合约调整；第14位对应：“1”表示暂停上市后协议转让。
PreClose Price	昨收价	double	1131	现货4位有效小数位；指数5位有效小数位
ListingDate	上市日期	INT32	1085	
Currency	币种	string(5)	1061	币种：CNY = 人民币；HKD = 港币
ParValue	每股面值	double	1128	
IssuedVolume	总发行量	double	1082	2位有效小数位
Outstanding Share	流通股数	double	1044	2位有效小数位
Industry Type	行业种类	string(5)	1075	
PreYearEPS	上年每股利润	double	1134	
YearEPS	本年每股利润	double	1235	
Offering Flag	收购(转股、行权)标志	string(1)	1117	股票：要约收购；债券、优先股：转股回售；Y=是 N=否
NAV	基金T-1日累计净值	double	1114	
Coupon Rate	票面利率	double	1055	
IssuePrice	贴现发行价	double	1369	
Interest	每百元应计利息	double	1370	对于优先股：8位小数，0.0000表示浮动股息率

User Mannual

Interest AccrualDate	起息日	INT32	1076	
Maturity Date	到期交割日	INT32	1101	
ConversionPrice	行权价格	double	1053	
ConversionRatio	行权比例	double	1054	
ConversionBeginDate	行权开始日	INT32	1051	
ConversionEndDate	行权结束日	INT32	1052	
CallOrPut	认购认沽	string(1)	1042	C = Call ; P = Put
Warrant Clearing Type	权证结算方式	string(1)	1219	S = 证券结算 C = 现金结算
Clearing Price	结算价格	double	1396	适用于权证
OptionType	行权类型	string(1)	1119	A = 美式 E = 欧式 B = 百慕大式
EndDate	最后交易日	int32	1063	
ExpirationDays	购回期限	int32	1371	
DayTrade	回转交易标	string (1)	1363	是否支持当日回转交易：Y=支持 N=不支持

User Mannual

ng	志			
GageFlag	保证金证券标志	string(1)	1365	是否可作为融资融券可充抵保证金证券：Y=是 N=否
GageRate	担保品折算率	double	1073	
CrdBuyUnderlying	融资标的标志	string(1)	1057	Y=是 N=否
CrdSellUnderlying	融券标的标志	string(1)	1060	Y=是 N=否
CrdPriceCheckType	提价检查方式	string(2)	1058	0=不检查；1=不低于最近成交价；2=不低于昨收价；3=不低于最高叫买；4=不低于最低叫卖
PledgeFlag	质押入库标志	string(1)	1366	是否可质押入库:Y=是 N=否
ContractMultiplier	债券折合回购标准券比例	double	1048	
RegularShare	对应回购标准券	string(8)	1367	
Qualification Flag	投资者适当性管理标志	string(1)	1368	是否需要对该证券作投资者适当性管理 Y=是 N=否
Market Maker Flag	做市商标志	string(1)	1095	标识是否有做市商 Y=是 N=否
RoundLot	整手数	double	1142	2 位有效小数位。对于某一证券申报的委托，其委托数量字段必须为该证券数量单位的整数倍
TickSize	最小报价单位	double	1136	

User Mannual

BuyQtyUpper Limit	买数量上限	double	1388	2 位有效小数位。买委托数量的上限
SellQtyUpper Limit	卖数量上限	double	1389	2 位有效小数位。卖委托数量的上限
BuyVolumeUnit	买数量单位	double	1041	2 位有效小数位。每笔买委托的委托数量必须是买数量单位的整数倍
SellVolumeUnit	卖数量单位	double	1178	2 位有效小数位。每笔卖委托的委托数量必须是卖数量单位的整数倍
LimitUpRateO	开盘集合竞价上涨幅度	double	1372	
LimitDownRateO	开盘集合竞价下跌幅度	double	1373	
LimitUpAbsoluteO	开盘集合竞价上涨限价	double	1374	预留字段，深交所未有实际应用
LimitDownAbsoluteO	开盘集合竞价下跌限价	double	1375	预留字段，深交所未有实际应用
AuctionUpDownRateO	开盘集合竞价有效范围涨跌幅度	double	1376	
AuctionUpDownAbsoluteO	开盘集合竞价有效范围涨跌价格	double	1377	
LimitUpRateT	连续竞价上涨幅度	double	1378	

User Mannual

LimitDownRateT	连续竞价下跌幅度	double	1379	
LimitUpAbsoluteT	连续竞价上涨限价	double	1394	预留字段，深交所未有实际应用
LimitDownAbsoluteT	连续竞价下跌限价	double	1395	预留字段，深交所未有实际应用
AuctionUpDownRateT	连续竞价有效范围涨跌幅度	double	1380	
AuctionUpDownAbsoluteT	连续竞价有效范围涨跌价格	double	1381	
LimitUpRateC	收盘集合竞价上涨幅度	double	1382	
LimitDownRateC	收盘集合竞价下跌幅度	double	1383	
LimitUpAbsoluteC	收盘集合竞价上涨限价	double	1384	预留字段，深交所未有实际应用
LimitDownAbsoluteC	收盘集合竞价下跌限价	double	1385	预留字段，深交所未有实际应用
AuctionUpDownRateC	收盘集合竞价有效范围涨跌幅度	double	1386	

User Mannual

Auction Up DownAb soluteC	收盘集合竞 价有效范围 涨跌价格	double	1387	
------------------------------------	------------------------	--------	------	--

该消息类型结构体在 QTSStruct.h 中定义

5.6.15 深交所 L2 证券状态 SZSEL2_Status

表 5.21 深交所 L2 证券状态表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32	1120	格式为：HHMMSSsss
Quotation Flag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充空格。 第 1 位对应：' 1 ' 表示上海机房行情源，' 2 ' 表示深圳机房行情源
Symbol	证券代码	string(40)	1187	
SymbolSo urce	证券代码源	string(5)	1188	102=深圳证券交易所 103=香港交易所
Time	数据生成时间	INT64	1191	数据生成时间 YYYYMMDDHHMMSSsss
FinancialS tatus	证券状态信息	string(8)	1345	A=上市公司早间披露提示；B=上市公司午间披露提示 每个字节揭示一种状态，最多可同时揭示八种状态。 每个字节对应的状态不固定，状态按出现的先后顺序在该字段从左到右排列,目前深交所只揭示两种状态。
CrdbuySta tus	当前融资开 放状态	string(1)	1056	适用于融资标的证券。 1= 是,0=否，空格无意义

User Mannual

CrdSellStatus	当前融券开放状态	string(1)	1059	适用于融券标的证券。 1= 是,0=否, 空格无意义
SubScribeStatus	申购标志	string(1)	1186	适用于 ETF, LOF 等开放式基金,对于黄金 ETF 是指现金申购。 1= 是,0=否, 空格无意义
RedemptionStatus	赎回标志	string(1)	1141	适用于 ETF, LOF 等开放式基金,对于黄金 ETF 是指现金赎回开关。 1= 是,0=否, 空格无意义
PurchasingStatus	认购标志	string(1)	1348	适用于网上发行认购代。 1= 是,0=否, 空格无意义
StockDiviStatus	转股标志	string(1)	1349	适用于处于转股回售期的可转债。 1= 是,0=否, 空格无意义
PutableStatus	回售标志	string(1)	1350	适用于处于转股回售期的可转债。 1= 是,0=否, 空格无意义
ExerciseStatus	行权标志	string(1)	1343	适用于处于行权期的权证。 1= 是,0=否, 空格无意义
GoldPurchase	黄金 ETF 实物申购标志	string(1)	1392	适用于黄金 ETF 实物申购。 1= 是,0=否, 空格无意义
GoldRedemption	黄金 ETF 实物赎回标志	string(1)	1393	适用于黄金 ETF 实物赎回。 1= 是,0=否, 空格无意义
AcceptedStatus	预受要约标志	string(1)	1351	适用于处于要约收购期的股票。 1= 是,0=否, 空格无意义
ReleaseStatus	解除要约标志	string(1)	1352	适用于处于要约收购期的股票。 1= 是,0=否, 空格无意义
CancStockDiviStatus	转股撤单标志	string(1)	1353	适用于处于转股回售期的可转债。 1= 是,0=否, 空格无意义
CancPutableStatus	回售撤单标志	string(1)	1354	适用于处于转股回售期的可转债。 1= 是,0=否, 空格无意义
PledgeStatus	质押标志	string(1)	1355	适用于质押式回购可质押入库证券。 1= 是,0=否, 空格无意义

User Manual

RemovePledge	解押标志	string(1)	1356	适用于质押式回购可质押入库证券。1= 是,0=否, 空格无意义
VoteStatus	表决权标志	string(1)	1357	适用于优先股。1= 是,0=否, 空格无意义
StockPledge Repo	股票质押式回购标志	string(1)	1358	适用于可开展股票质押式回购业务的证券。1= 是,0=否, 空格无意义
DivideStatus	实时分拆标志	string(1)	1359	适用于分级基金。1= 是,0=否, 空格无意义
MergerStatus	实时合并标志	string(1)	1360	适用于分级基金。1= 是,0=否, 空格无意义

该消息类型结构体在 QTSStruct.h 中定义

5.6.16 深交所 L2 实时行情 SZSEL2_Quotation

表 5.22 深交所 L2 实时行情表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32	1120	格式为：HHMMSSsss
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充空格。 第 1 位对应：' 1' 表示国上海机房行情源，' 2' 表示深圳机房行情源
Time	数据生成时间	INT64	1191	数据生成时间 YYYYMMDDHHMMSSsss
Symbol	证券代码	string(40)	1187	
SymbolSource	证券代码源	string(5)	1188	102=深圳证券交易所 103=香港交易所

User Mannual

PreClose Price	昨收价	double	1131	
OpenPrice	开盘价	double	1118	
LastPrice	现价	double	1083	
HighPrice	最高价	double	1074	
LowPrice	最低价	double	1088	
PriceUpLimit	涨停价	double	1139	
PriceDownLimit	跌停价	double	1135	
PriceUpdown1	涨跌一	double	1282	LastPrice-PreClosePrice ;
PriceUpdown2	涨跌二	double	1200	LastPrice-上一笔 LastPrice ; 对于当天的第一笔成交, 该字段=LastPrice-PreClosePrice
TotalNo	成交笔数	uint64	1197	
TotalVolume	成交总量	double	1202	2 位有效小数位。单位: 股票为股, 基金为份, 债券与回购为张。
TotalAmount	成交总额	double	1192	
ClosePrice	今收盘价	double	1046	QTS2.3 将静态数据文件中的现货证券收盘行情 cashsecurityclosemd.xml 中的收盘价填充到实时行情以提供收盘价信息 (在最后一笔行情数据基础上填充收盘价, 且原有数据生成时间基础上加 1 秒作为新的 time), 但 cashsecurityclosemd.xml 文件交易所 3 点半左右才推送导致最后一笔收盘数据较晚推送。
SecurityPhase	当前品种交易状态	string(8)	1147	产品所处的交易阶段代码: 第 0 位: S=启动 (开市前); O=开盘集合竞价;

User Mannual

Tag				T=连续；B=休市； C=收盘集合竞价；E=已闭市；H=临时停牌；A=盘后交易；V=波动性中断 第 1 位：0=正常状态；1=全天停牌
PERatio1	市盈率 1	double	1129	
NAV	基金 T-1 日净值	double	1114	基金 T-1 日净值
PERatio2	市盈率 2	double	1130	
IOPV	基金实时参考净值	double	1079	基金实时参考净值（包括 ETF 的 IOPV）
Premiu mRate	权证溢价率	double	1391	
TotalSell Order Volume	委托卖出总量	double	1201	2 位有效小数位。
WtAvgS ellPrice	加权平均委托卖价	double	1231	
SellLevel No	卖盘价位数	UINT32	1153	
SellPrice 01	申卖价一	double	1157	
SellVolu me01	申卖量一	double	1168	2 位有效小数位。
TotalSell Order No01	卖出总委托笔数	uint64	1261	
SellPrice 02	申卖价二	double	1158	
SellVolu me02	申卖量二	double	1169	2 位有效小数位。

User Mannual

TotalSell Order No02	卖出总委托 笔数	uint64	1262	
SellPrice 03	申卖价三	double	1159	
SellVolume03	申卖量三	double	1170	2 位有效小数位。
TotalSell Order No03	卖出总委托 笔数	uint64	1263	
SellPrice 04	申卖价四	double	1160	
SellVolume04	申卖量四	double	1171	2 位有效小数位。
TotalSell Order No04	卖出总委托 笔数	uint64	1264	
SellPrice 05	申卖价五	double	1161	
SellVolume05	申卖量五	double	1172	2 位有效小数位。
TotalSell Order No05	卖出总委托 笔数	uint64	1265	
SellPrice 06	申卖价六	double	1162	
SellVolume06	申卖量六	double	1173	2 位有效小数位。
TotalSell Order	卖出总委托 笔数	uint64	1266	

User Mannual

No06				
SellPrice 07	申卖价七	double	1163	
SellVolume 07	申卖量七	double	1174	2 位有效小数位。
TotalSell Order No07	卖出总委托 笔数	uint64	1267	
SellPrice 08	申卖价八	double	1164	
SellVolume 08	申卖量八	double	1175	2 位有效小数位。
TotalSell Order No08	卖出总委托 笔数	uint64	1268	
SellPrice 09	申卖价九	double	1165	
SellVolume 09	申卖量九	double	1176	2 位有效小数位。
TotalSell Order No09	卖出总委托 笔数	uint64	1269	
SellPrice 10	申卖价十	double	1166	
SellVolume 10	申卖量十	double	1177	2 位有效小数位。
TotalSell Order No10	卖出总委托 笔数	uint64	1270	

User Mannual

SellLevel QueueNo01	卖一档揭示 委托笔数	UINT32	1271	
SellLevel Queue	卖 1 档队列	double	1154	2 位有效小数位。揭示卖一价前 50 笔委托。50 档，不足补为 0。
TotalBuy Order Volume	委托买入总 量	double	1195	2 位有效小数位。
WtAvgB uyPrice	加权平均买 入价	double	1230	
BuyLevel No	买 盘 价 位 数：	UINT32	1015	
BuyPrice 01	申买价一	double	1019	
BuyVolu me01	申买量一	double	1031	2 位有效小数位。
TotalBuy Order No01	买入总委托 笔数	uint64	1251	
BuyPrice 02	申买价二	double	1020	
BuyVolu me02	申买量二	double	1032	2 位有效小数位。
TotalBuy Order No02	买入总委托 笔数	uint64	1252	
BuyPrice 03	申买价三	double	1021	
BuyVolu me03	申买量三	double	1033	2 位有效小数位。

User Mannual

TotalBuy Order No03	买入总委托 笔数	uint64	1253	
BuyPrice 04	申买价四	double	1022	
BuyVolume04	申买量四	double	1034	2 位有效小数位。
TotalBuy Order No04	买入总委托 笔数	uint64	1254	
BuyPrice 05	申买价五	double	1023	
BuyVolume05	申买量五	double	1035	2 位有效小数位。
TotalBuy Order No05	买入总委托 笔数	uint64	1255	
BuyPrice 06	申买价六	double	1024	
BuyVolume06	申买量六	double	1036	2 位有效小数位。
TotalBuy Order No06	买入总委托 笔数	uint64	1256	
BuyPrice 07	申买价七	double	1025	
BuyVolume07	申买量七	double	1037	2 位有效小数位。
TotalBuy Order	买入总委托 笔数	uint64	1257	

User Mannual

No07				
BuyPrice08	申买价八	double	1026	
BuyVolume08	申买量八	double	1038	2 位有效小数位。
TotalBuyOrderNo08	买入总委托笔数	uint64	1258	
BuyPrice09	申买价九	double	1027	
BuyVolume09	申买量九	double	1039	2 位有效小数位。
TotalBuyOrderNo09	买入总委托笔数	uint64	1259	
BuyPrice10	申买价十	double	1028	
BuyVolume10	申买量十	double	1040	2 位有效小数位。
TotalBuyOrderNo10	买入总委托笔数	uint64	1260	
BuyLevelQueueNo01	买一档揭示委托笔数	UINT32	1272	
BuyLevelQueue	买一档委托队列	double	1016	2 位有效小数位。揭示买一价前 50 笔委托。50 档，不足补为 0。

该消息类型结构体在 QTSStruct.h 中定义

5.6.17 深交所 L2 指数快照 SZSEL2_Index

表 5.23 深交所 L2 指数快照表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32	1120	格式为：HHMMSSsss
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充空格。 第 1 位对应：' 1' 表示国上海机房行情源，' 2' 表示深圳机房行情源
Time	数据生成时间	INT64	1191	数据生成时间 YYYYMMDDHHMMSSsss
Symbol	证券代码	string(40)	1187	
SymbolSource	证券代码源	string(5)	1188	102=深圳证券交易所；103=香港交易所
PreClosePrice	昨收价	double	1131	
OpenPrice	开盘价	double	1118	
HighPrice	最高价	double	1074	
LowPrice	最低价	double	1088	
LastPrice	现价	double	1083	
TotalAmount	成交金额	double	1192	
TotalNo	成交笔数	uint64	1197	

User Mannual

TotalVolume	成交总量	double	1202	2 位有效小数位。单位：股
Security Phase Tag	当前品种交易状态	string(8)	1147	产品所处的交易阶段代码 第 0 位：S=启动（开市前）；O=开盘集合竞价；T=连续；B=休市； C=收盘集合竞价；E=已闭市；H=临时停牌；A=盘后交易；V=波动性中断 第 1 位：0=正常状态；1=全天停牌
Sample No	样本个数	UINT32	1273	统计量指标样本个数，成交量统计指标代码详见 深圳市场统计指标定义表

该消息类型结构体在 QTSStruct.h 中定义

5.6.18 深交所 L2 逐笔委托 SZSEL2_Order

表 5.24 深圳 L2 逐笔成交表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32	1120	格式为：HHMMSSsss
Quotation Flag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充格。 第 1 位对应：'1'表示国上海机房行情源，'2'表示深圳机房行情源
SetID	频道代码	UINT32	1179	频道代码
RecID	消息记录号	UINT64	1140	消息记录号：从 1 开始计数，同一频道连续
Symbol	证券代码	string(40)	1187	
SymbolSource	证券代码源	string(5)	1188	102=深圳证券交易所；103=香港交易所
Time	委托时间	int64	1191	格式为：YYYYMMDDHHMMSSsss

User Mannual

OrderPrice	委托价格	double	1122	
OrderVolume	委托数量	double	1124	2 位有效小数位。股票为股，基金为份，债券为张
OrderCode	委托买卖方向	string(1)	1121	买卖方向：1=买 2=卖 G=借入 F=出借
OrderType	订单类别	string(1)	1123	订单类别：1=市价 2=限价 U=本方最优

该消息类型结构体在 QTSStruct.h 中定义

5.6.19 深交所 L2 逐笔成交 SZSEL2_Transaction

表 5.25 深交所 L2 逐笔成交表

字段名	字段含义	用户类型	Tag	描述
LocalTimeStamp	采集时间	INT32	1120	格式为：HHMMSSsss
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充格。 第 1 位对应：'1'表示上海机房行情源，'2'表示深圳机房行情源
SetID	频道代码	UINT32	1179	频道代码
RecID	消息记录号	UINT64	1140	消息记录号 从 1 开始计数，同一频道连续
BuyOrderID	买方委托索引	UINT64	1017	买方委托索引：从 1 开始计数,0 表示无对应委托
SellOrderID	卖方委托索引	UINT64	1155	卖方委托索引：从 1 开始计数,0 表示无对应委托
Symbol	证券代码	string(40)	1187	

User Mannual

SymbolSource	证券代码源	string(5)	1188	102=深圳证券交易所；103=香港交易所
TradeTime	成交时间	int64	1213	成交时间：YYYYMMDDHHMMSSsss
TradePrice	成交价格	double	1210	
TradeVolume	成交数量	double	1215	2 位有效小数位。股票为股，基金为份，债券为张
TradeType	成交类型	string(1)	1214	成交类别：4=撤销，主动或自动撤单执行报告；F=成交，成交执行报告

该消息类型结构体在 QTSStruct.h 中定义

5.6.20 中金所 L2 静态数据 CFFEXL2_Static

表 5.26 中金所 L2 静态数据表

字段名	字段含义	用户类型	Tag	描述
LocalTime Stamp	采集时间	INT32		数据到达时本系统记录的时间，精确到毫秒
TradeDate	成交日期	INT32	1208	YYYYMMDD，统一为交易归属日期，后续交易所均遵循此原则
Time	数据生成时间	INT32	1191	更新时间和最后修改毫秒两个字段合成一个：HHMMSSMMM
Symbol	证券代码	string(40)	1145	
PreClosePric	昨收价	double	1131	指数有效小数位为 2 位，国债有效小数位为 3 位

User Mannual

e				
PreSettlePrice	昨结算	double	1133	指数有效小数位为 2 位，国债有效小数位为 3 位
OpenPrice	开盘价	double	1118	指数有效小数位为 2 位，国债有效小数位为 3 位
HighPrice	最高价	double	1074	指数有效小数位为 2 位，国债有效小数位为 3 位
LowPrice	最低价	double	1088	指数有效小数位为 2 位，国债有效小数位为 3 位
TradePrice	最新价	double	1210	指数有效小数位为 2 位，国债有效小数位为 3 位
ClosePrice	收盘价	double	1046	指数有效小数位为 2 位，国债有效小数位为 3 位
SettlePrice	结算价	double	1181	指数有效小数位为 2 位，国债有效小数位为 3 位
PriceUpLimit	涨停价	double	1139	指数有效小数位为 2 位，国债有效小数位为 3 位
PriceDownLimit	跌停价	double	1135	指数有效小数位为 2 位，国债有效小数位为 3 位
PreTotalPosition	昨持仓量	UINT64	1092	
TotalPosition	持仓量	UINT64	1198	
PreDelta	昨虚实度	double	1093	前一日对冲值
Delta	今虚实度	double	1094	对冲值：期权价格变化/期货价格变化
SettleGroupID	结算组代码	string(10)	1096	
SettleID	结算编号	UINT32	1097	
TotalVolume	总成交量	UINT64	1202	

User Mannual

TotalAmount	总成交额	double	1192	3 位有效小数
SellPrice01	卖价一	double	1157	指数有效小数位为 2 位，国债有效小数位为 3 位
SellVolume01	卖量一	UINT64	1168	
SellPrice02	卖价二	double	1158	指数有效小数位为 2 位，国债有效小数位为 3 位
SellVolume02	卖量二	UINT64	1169	
SellPrice03	卖价三	double	1159	指数有效小数位为 2 位，国债有效小数位为 3 位
SellVolume03	卖量三	UINT64	1170	
SellPrice04	卖价四	double	1160	指数有效小数位为 2 位，国债有效小数位为 3 位
SellVolume04	卖量四	UINT64	1171	
SellPrice05	卖价五	double	1161	指数有效小数位为 2 位，国债有效小数位为 3 位
SellVolume05	卖量五	UINT64	1172	
BuyPrice01	买价一	double	1019	指数有效小数位为 2 位，国债有效小数位为 3 位
BuyVolume01	买量一	UINT64	1031	
BuyPrice02	买价二	double	1020	指数有效小数位为 2 位，国债有效小数位为 3 位
BuyVolume02	买量二	UINT64	1032	
BuyPrice03	买价三	double	1021	指数有效小数位为 2 位，国债有效小数位为 3 位
BuyVolume03	买量三	UINT64	1033	

User Mannual

BuyPrice04	买价四	double	1022	指数有效小数位为 2 位，国债有效小数位为 3 位
BuyVolume04	买量四	UINT64	1034	
BuyPrice05	买价五	double	1023	指数有效小数位为 2 位，国债有效小数位为 3 位
BuyVolume05	买量五	UINT64	1035	

该消息类型结构体在 QTSStruct.h 中定义

注：中金所静态数据消息类型中定义的字段在 QTS2.1 版本相和 QTS1.9 版本中存在变动，主要是名称的变动和个别字段的删除，具体情况请参见附录 [QTS1.x 与 QTS2.x 字段对比](#)。

5.6.21 中金所 L2 实时行情 CFFEXL2_Quotation

中金所 L2 实时行情 Msg_CFFEXL2_Quotation 消息类型的结构体定义与中金所 L2 静态数据消息类型的结构体定义相同，请参见[表 5.6.20 中金所 L2 静态数据 CFFEXL2_Static](#)。

5.6.22 上期所 L1 静态数据 SHFEL1_Static

表 5.27 上期所 L1 静态数据表

字段名	字段含义	用户类型	Tag	描述
LocalTimeStamp	采集时间	INT32	1120	
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填空格。 第 1 位对应：'1'表示上海机房行情源，'2'表示深圳

User Mannual

				机房行情源
ActionDay	业务发生日期	INT32	1285	交易实际发生的日期，如周五的夜盘业务发生日期为周五的日期，但市场日期属于下周一对应的日期。
Time	数据生成时间	INT32	1191	将最后修改时间和最后修改毫秒合并成新字段——数据生成时间
TradeDate	交易日期	INT32	1208	
SettleGroupID	结算组代码	string(10)	1096	
SettleID	结算编号	UINT32	1097	
Symbol	合约代码	string(20)	1187	
PreSettlePrice	昨结算	double	1133	
PreClosePrice	昨收价	double	1131	
PreTotalPosition	昨持仓量	double	1092	
OpenPrice	开盘价	double	1118	
PriceUpLimit	涨停价格	double	1139	
PriceDownLimit	跌停价格	double	1135	
LastPrice	最新价	double	1083	
HighPrice	最高价	double	1074	
LowPrice	最低价	double	1088	

User Manual

TotalVolume	成交总量	UINT64	1202	该字段在交易所文档显示为数量字段，该字段值随着成交逐渐累加，作为成交总量下发
TotalAmount	成交总额	double	1192	
TotalPosition	持仓量	double	1198	
ClosePrice	今收盘价	double	1046	
SettlePrice	今结算价	double	1181	
PreDelta	昨虚实度	double	1093	
Delta	今虚实度	double	1094	
BuyPrice01	申买价一	double	1019	
BuyVolume01	申买量一	UINT64	1031	
SellPrice01	申卖价一	double	1157	
SellVolume01	申卖量一	UINT64	1168	

5.6.23 上期所 L1 实时行情 SHFEL1_Quotation

上期所 L1 实时行情 Msg_SHFEL1_Quotation 消息类型的结构体定义与上期所 L1 静态数据消息类型的结构体定义相同，请参见[表 5.6.22 上期所 L1 静态数据 SHFEL1_Static](#)。

5.6.24 郑商所 L1 静态数据 CZCEL1_Static

表 5.28 郑商所 L1 静态数据表

User Mannual

字段名	字段含义	用户类型	Tag	描述
LocalTimeStamp	采集时间	INT32	1120	hh:mm:ss:rrr,精确到毫秒
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串,起每位表示特定的含义,无定义则填空格。 第 1 位对应: '1'表示上海南汇机房行情源, '2'表示深圳福永机房行情源
TradeDate	交易日期	INT32	1208	计算字段: 日盘成交日期为本地接收数据的日期,夜盘输出的日期应为交易归属日期,计算方法为:若夜盘开盘次日是交易日,则将该日(当晚 21:00 持续到 23:30)的夜盘的交易归属日期算成次日,若夜盘开盘当日为周五,则将该日(当晚 21:00 持续到 23:30)的夜盘交易归属日期算成下个交易日。节假日前一天没有夜盘。
Time	数据生成时间	int32	1191	成交时间
Symbol	合约代码	string(40)	1187	合约编码 例如 期货为 WS509 期权为 WS509C1600
Version	合约版本号	string(1)	1218	合约版本号, 目前为 0
PreClosePrice	昨收价	double	1131	
PreSettlePrice	昨结算	double	1133	
PreTotalPosition	昨持仓量	UINT64	1092	昨日空盘量, 双向计算
OpenPrice	开盘价	double	1118	
PriceUpLimit	涨停价格	double	1139	

User Mannual

PriceDownLimit	跌停价格	double	1135	
LastPrice	最新价	double	1083	
AveragePrice	均价	double	1291	
HighPrice	最高价	double	1074	
LowPrice	最低价	double	1088	
LifeHigh	历史最高成交价格	double	1288	
LifeLow	历史最低成交价格	double	1289	
BuyPrice01	申买价	double	1019	
BuyVolume01	申买量	UINT64	1031	
SellPrice01	申卖价	double	1157	
SellVolume01	申卖量	UINT64	1168	
TotalPosition	持仓量	uint64	1198	
TotalVolume	成交总量	UINT64	1202	该字段在交易所文档中显示为数量，为实时更新数据，在 api 作为成交总量数据下发。
TotalAmount	成交总额	double	1192	该字段非实时更新，只在与交易所进行重登录时才更新。
ClosePrice	今收盘价	double	1046	
ClearPrice	当日交割结算价	double	1290	交割日结算价，与结算价的数值相同

User Mannual

SettlePrice	结算价	double	1181	
DeriveBidPrice	组合买入价	double	1297	
DeriveAskPrice	组合卖出价	double	1298	
DeriveBidLot	组合买入数量	UINT64	1299	
DeriveAskLot	组合卖出数量	UINT64	1300	
LastLot	最后一笔成交手数	UINT64	1398	该字段非实时更新，只在与交易所进行重登录时才更新。

5.6.25 郑商所 L1 实时行情 CZCEL1_Quotation

郑商所 L1 实时行情 Msg_CZCEL1_Quotation 消息类型的结构体定义与郑商所 L1 静态数据消息类型的结构体定义相同，请参见[表 5.6.24 郑商所 L1 静态数据 CZCEL1 Static](#)。

5.6.26 易盛指数行情 ESUNNY_Index

表 5.29 易盛指数行情表

字段名	字段含义	用户类型	Tag	描述
LocalTimeStamp	采集时间	INT32	1120	hh:mm:ss:rrr,精确到毫秒
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充格。 第 1 位对应：'1'表示上海机房行情源，'2'表示深圳机房行情源

User Mannual

symbol	合约代码	string(40)	1187	
Time	数据生成时间	UInt64	1191	格式：YYYYMMDDhhmmss000 不同的位置代表年月日时分秒毫秒，其中 YYYYMMDD 为业务发生日期。 eg: 20150819093015123
PreClosePrice	昨收价	double	1131	
PreSettlePrice	昨结算	double	1133	
PreTotalPosition	昨持仓量	UInt64	1092	
OpenPrice	开盘价	double	1118	
LastPrice	最新价	double	1083	
HighPrice	最高价	double	1074	
LowPrice	最低价	double	1088	
LifeHigh	历史最高成交价格	double	1288	
lifeLow	历史最低成交价格	double	1289	
TotalVolume	成交总量	UInt64	1202	
TotalPosition	持仓量	UInt64	1198	
AveragePrice	均价	double	1291	预留字段，暂未启用
ClosePrice	今收盘价	double	1046	
TotalAmount	成交总额	double	1192	

User Mannual

t				
SettlePrice	结算价	double	1181	
LastMatchQty	瞬时成交量	Uint64	1341	预留字段，暂未启用
PriceUpLimit	涨停价格	double	1139	预留字段，暂未启用
PriceDownLimit	跌停价格	double	1135	预留字段，暂未启用

5.6.27大商所 L1 静态数据 DCEL1_Static

表 5.30 大商所 L1 静态数据表

字段名	字段含义	用户类型	Tag	描述
LocalTimeStamp	采集时间	INT32	1120	采集服务接收的本地时间,格式为 :HH:MM:SS.MMM
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串,起每位表示特定的含义,无定义则填充格。 第 1 位对应: '1'表示上海南汇机房行情源, '2'表示深圳福永机房行情源
TradeDate	交易日期	INT32	1208	
Time	数据生成时间	INT32	1191	
Symbol	合约代码	string(40)	1187	
RoutineNo	事务编号	Uint32	1328	
SecurityName	合约名称	string(40)	1145	

User Mannual

me				
PreClosePrice	昨收价	double	1131	
PreSettlePrice	昨结算	double	1133	
PreTotalPosition	昨持仓量	UINT64	1092	
OpenPrice	开盘价	double	1118	指某一期货合约开市前五分钟内经集合竞价产生的成交价格。集合竞价未产生成交价格的，以集合竞价后第一笔成交价为开盘价。
PriceUpLimit	涨停价格	double	1139	
PriceDownLimit	跌停价格	double	1135	
LastPrice	最新价	double	1083	
AveragePrice	均价	double	1291	某合约当天的成交价格按照成交量的加权平均。计算公式：当日均价=成交额/成交量。有可能出现均价在最高价和最低价之外，因为，成交量和成交额包括套利定单的成交，但是套利定单直接成交不更新最新价，最高价和最低价。
HighPrice	最高价	double	1074	
LowPrice	最低价	double	1088	
LifeHigh	历史最高成交价格	double	1288	
LifeLow	历史最低成交价格	double	1289	
LastMatchQty	最新成交量	UINT64	1341	指某交易日某一期货合约交易期间的即时成交双边数量，包含作为套利合约的单腿成交的成交量。
TotalVolum	成交总量	UINT64	1202	

User Mannual

e				
TotalAmount	成交总额	double	1192	指某一合约在当日交易期间所有成交合约的双边金额，包含作为套利合约的单腿成交的成交额。计算公式：成交额=Σ成交价*成交量*交易单位
TotalPosition	持仓量	UINT64	1198	
InterestChg	持仓量变化	INT64	1287	计算公式：持仓量变化=持仓量-初始持仓量
BuyPrice01	申买价	double	1019	指某一期货合约当日买方申请买入的即时最高价格。
BuyVolume01	申买量	UINT64	1031	指某一期货合约当日交易所交易系统中未成交的最高价位申请买入的下单数量，包含了申买推导量。
BidImpliedQty01	申买推导量	UINT64	1330	由套利定单推导出来的该合约的申买数量
SellPrice01	申卖价	double	1157	
SellVolume01	申卖量	UINT64	1168	
AskImpliedQty01	申卖推导量	UINT64	1335	
ClosePrice	今收盘价	double	1046	
SettlePrice	结算价	double	1181	指某一期货合约当日成交价格按成交量的加权平均价。当日无成交的，以上一交易日的结算价作为当日结算价。结算价是进行当日未平仓合约盈亏结算和制定下一交易日涨跌停板额的依据。
Delta	Delta	double	1094	期权或资产价格变动对其标的资产价格变动的比率
Gamma	Gamma	double	1293	证券组合价值 Delta 变化与标的资产价格变化的比率
Rho	Rho	double	1294	交易组合价值变化与利率变化的比率
Theta	Theta	double	1295	证券组合价值变化与时间变化的比率

Vega	Vega	double	1296	交易组合价值变化与标的资产波动率变化的比率
------	------	--------	------	-----------------------

5.6.28大商所 L1 最优行情 DCEL1_Quotation

大商所 L1 实时行情 Msg_DCEL1_Quotation 消息类型的结构体定义与大商所 L1 静态数据消息

类型的结构体定义相同，请参见[表 5.6.27 大商所 L1 静态数据 DCEL1_Static](#)。

5.6.29大商所 L1 套利行情 DCEL1_ArbiQuotation

表 5.32 大商所 L1 套利行情表

字段名	字段含义	用户类型	Tag	描述
LocalTimeStamp	采集时间	INT32	1120	采集服务接收的本地时间,格式为 :HH:MM:SS.MMM
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串,起每位表示特定的含义,无定义则填充格。 第 1 位对应: '1'表示上海南汇机房行情源, '2'表示深圳福永机房行情源
TradeDate	交易日期	INT32	1208	
Time	数据生成时间	INT32	1191	
RoutineNo	事务编号	Uint32	1328	该值始终是 0, 没有用
Symbol	套利合约号	string(40)	1187	格式 : 套利策略 腿 1 合约代码&腿 2 合约代码 套利策略有 SP 跨期套利, SPC 跨品种套利
LastPrice	最新价	double	1083	套利定单直接成交, 不刷新最新价。

User Mannual

HighPrice	最高价	double	1074	套利定单直接成交，不刷新该价。
LowPrice	最低价	double	1088	套利定单直接成交，不刷新该价。
LifeHigh	历史最高成交价格	double	1288	套利定单直接成交，不刷新该价。
LifeLow	历史最低成交价格	double	1289	套利定单直接成交，不刷新该价。
PriceUpLimit	涨停价格	double	1139	第一个合约涨停板价 - 第二个合约跌停板价
PriceDownLimit	跌停价格	double	1135	第一个合约跌停板价 - 第二个合约涨停板价
BuyPrice01	申买价	double	1019	指当日买方申请套利定单买入的即时最高价格。
BuyVolume01	申买量	UINT64	1031	指当日交易所交易系统中未成交的最高价位申请买入的套利定单的下单数量。
SellPrice01	申卖价	double	1157	指当日卖方申请套利定单卖出的即时最低价格。
SellVolume01	申卖量	UINT64	1168	指当日交易所交易系统中未成交的最低价位申请卖出的套利定单的下单数量。

5.6.30大商所 L2 静态数据 DCEL2_Static

表 5.33 大商所 L2 静态数据表

字段名	字段含义	用户类型	Tag	描述
LocalTimeStamp	采集时间	INT32	1120	采集服务接收的本地时间,格式为 :HH:MM:SS.MMM
QuotationFl	行情源标志	string(4)	1280	该字段为 4 位字符串,起每位表示特定的含义,无定义则填充格。 第 1 位对应: '1'表示上海南汇机房行情源, '2'表示

User Mannual

ag				深圳福永机房行情源
TradeDate	交易日期	INT32	1208	格式：YYYYMMDD
Time	数据生成时间	INT32	1191	HH:MM:SS.MMM，取自 bestquot 域的 Gentime
Symbol	合约代码	string(40)	1187	含期货合约品种、期权合约品种
RoutineNo	事务编号	Uint32	1328	该值始终是 0，没有用
SecurityName	合约名称	string(40)	1145	
PreClosePrice	昨收价	double	1131	
PreSettlePrice	昨结算	double	1133	
PreTotalPosition	昨持仓量	UINT64	1092	是上一交易日结算后，期货交易者所持有的未平仓合约的双边数量
OpenPrice	开盘价	double	1118	指某一期货合约开市前五分钟内经集合竞价产生的成交价格。集合竞价未产生成交价格的，以集合竞价后第一笔成交价为开盘价。
PriceUpLimit	涨停价格	double	1139	当前交易日价格涨幅的上限
PriceDownLimit	跌停价格	double	1135	当前交易日价格跌幅的下限
LastPrice	最新价	double	1083	指某交易日某一期货合约交易期间的即时成交价格
AveragePrice	均价	double	1291	某合约当天的成交价格按照成交量的加权平均。计算公式：当日均价=成交额/成交量。有可能出现均价在最高价和最低价之外，因为，成交量和成交额包括套利定单的成交，但是套利定单直接成交不更新最新价，最高价和最低价。

User Mannual

HighPrice	最高价	double	1074	指当前交易期间某一期货合约成交价中的最高成交价格
LowPrice	最低价	double	1088	指当前交易期间某一期货合约成交价中的最低成交价格
LifeHigh	历史最高成交价格	double	1288	某一期货合约从上市以来的最低成交价格
LifeLow	历史最低成交价格	double	1289	某一期货合约从上市以来的最高成交价格
LastMatchQty	最新成交量	UINT64	1341	指某交易日某一期货合约交易期间的即时成交双边数量，包含作为套利合约的单腿成交的成交量
TotalVolume	成交总量	UINT64	1202	指某一合约在当日交易期间所有成交合约的双边数量，包含作为套利合约的单腿成交的成交量。
TotalAmount	成交总额	double	1192	指某一合约在当日交易期间所有成交合约的双边金额，包含作为套利合约的单腿成交的成交额。计算公式：成交额=∑成交价*成交量*交易单位
TotalPosition	持仓量	UINT64	1198	指某交易日某一期货合约交易期间期货交易者所持有的未平仓合约的双边数量。
InterestChg	持仓量变化	INT64	1287	计算公式：持仓量变化=持仓量-初始持仓量
ClosePrice	今收盘价	double	1046	指某一期货合约当日交易的最后一笔成交价格。
SettlePrice	结算价	double	1181	指某一期货合约当日成交价格按成交量的加权平均价。当日无成交的，以上一交易日的结算价作为当日结算价。结算价是进行当日未平仓合约盈亏结算和制定下一交易日涨跌停板额的依据。
MBLQuotBuyNum	深度买行情数量	UINT32	1301	接收到的“买 深度行情域”数量，买 5 个，不足时补 0
BuyPrice01	申买价一	double	1019	委托价格（最优买委托价格和最高买与此字段重复）
BuyVolume01	申买量一	UINT64	1031	在该委托价格上的委托量，包含了由套利定单推导出来的推导量（最优行情域中的申买量与此字段重复）

User Mannual

BidImPLYQty 01	申买推导量 一	UINT64	1330	由套利定单推导出来的推导量（最优行情域中申买推导量与此字段重复）
BuyPrice02	申买价二	double	1020	
BuyVolume 02	申买量二	UINT64	1032	
BidImPLYQty 02	申买推导量 二	UINT64	1331	
BuyPrice03	申买价三	double	1021	
BuyVolume 03	申买量三	UINT64	1033	
BidImPLYQty 03	申买推导量 三	UINT64	1332	
BuyPrice04	申买价四	double	1022	
BuyVolume 04	申买量四	UINT64	1034	
BidImPLYQty 04	申买推导量 四	UINT64	1333	
BuyPrice05	申买价五	double	1023	
BuyVolume 05	申买量五	UINT64	1035	
BidImPLYQty 05	申买推导量 五	UINT64	1334	
MBLQuotSel INum	深度卖行情 数量	UINT32	1302	接收到的“卖 深度行情域”数量，卖 5 个，不足时补 0
SellPrice01	申卖价一	double	1157	委托价格（最优卖委托价格和最低卖与此字段重复）
SellVolume0 1	申卖量一	UINT64	1168	在该委托价格上的委托量，包含了由套利定单推导出来的推导量（最优行情域中申卖量与此字段重复）

User Mannual

AskImPLYQt y01	申卖推导量 一	UINT64	1335	由套利定单推导出来的推导量（最优行情域中申卖 推导量与此字段重复）
SellPrice02	申卖价二	double	1158	
SellVolume0 2	申卖量二	UINT64	1169	
AskImPLYQt y02	申卖推导量 二	UINT64	1336	
SellPrice03	申卖价三	double	1159	
SellVolume0 3	申卖量三	UINT64	1170	
AskImPLYQt y03	申卖推导量 三	UINT64	1337	
SellPrice04	申卖价四	double	1160	
SellVolume0 4	申卖量四	UINT64	1171	
AskImPLYQt y04	申卖推导量 四	UINT64	1338	
SellPrice05	申卖价五	double	1161	
SellVolume0 5	申卖量五	UINT64	1172	
AskImPLYQt y05	申卖推导量 五	UINT64	1339	
BuyLevelQu eueNo01	买一档揭示 委托笔数	UINT32	1272	实际接收到的“买最优价委托量”数量，最多 10 个
BuyLevelQu eue	买最优价委 托量队列	UINT32	1016	买一价前 10 笔委托
SellLevelQu eueNo01	卖一档揭示 委托笔数	UINT32	1271	实际接收到的“卖最优价委托量”数量，最多 10 个

User Mannual

SellLevelQueue	卖最优价委托量队列	UINT32	1154	卖一价前 10 笔委托
Delta	Delta	double	1094	期权或资产价格变动对其标的资产价格变动的比率
Gamma	Gamma	double	1293	证券组合价值 Delta 变化与标的资产价格变化的比率
Rho	Rho	double	1294	交易组合价值变化与利率变化的比率
Theta	Theta	double	1295	证券组合价值变化与时间变化的比率
Vega	Vega	double	1296	交易组合价值变化与标的资产波动率变化的比率

5.6.31 大商所 L2 最优深度行情 DCEL2_Quotation

大商所 L2 实时行情 Msg_DCEL2_Quotation 消息类型的结构体定义与大商所 L2 静态数据消息

类型的结构体定义相同，请参见[表 5.6.30 大商所 L2 静态数据 DCEL2_Static](#)。

5.6.32 大商所 L2 套利深度行情 DCEL2_ArbiQuotation

表 5.34 大商所 L2 套利深度行情表

字段名	字段含义	用户类型	Tag	描述
LocalTimeStamp	采集时间	INT32	1120	
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充空格。 第 1 位对应：'1'表示上海南汇机房行情源，'2'表示深圳福永机房行情源
TradeDate	交易日期	INT32	1208	格式：YYYYMMDD

User Mannual

Time	数据生成时间	INT32	1191	
Symbol	套利合约号	string(40)	1187	格式 : 套利策略 腿 1 合约代码&腿 2 合约代码 套利策略有 SP 跨期套利, SPC 跨品种套利
RoutineNo	事务编号	Uint32	1328	该值始终是 0
LastPrice	最新价	double	1083	套利定单直接成交, 不刷新最新价。
HighPrice	最高价	double	1074	套利定单直接成交, 不刷新该价。
LowPrice	最低价	double	1088	套利定单直接成交, 不刷新该价。
LifeHigh	历史最高成交价格	double	1288	套利定单直接成交, 不刷新该价。
LifeLow	历史最低成交价格	double	1289	套利定单直接成交, 不刷新该价。
PriceUpLimit	涨停价格	double	1139	第一个合约涨停板价 - 第二个合约跌停板价
PriceDownLimit	跌停价格	double	1135	第一个合约跌停板价 - 第二个合约涨停板价
MBLQuotBuyNum	深度买行情数量	UINT32	1301	接收到的" 买 深度行情域" 数量, 买 5 个, 不足时补 0
BuyPrice01	申买价一	double	1019	委托价格 (最优买委托价格和最高买与此字段重复)
BuyVolume01	申买量一	UINT64	1031	在该委托价格上的套利定单的委托量 (套利最优行情域申买量与此字段重复)
BidImpleQty01	申买推导量一	UINT64	1330	目前各档推导量都显示为 0
BuyPrice02	申买价二	double	1020	
BuyVolume02	申买量二	UINT64	1032	

User Mannual

BidImPLYQty02	申买推导量二	UINT64	1331	
BuyPrice03	申买价三	double	1021	
BuyVolume03	申买量三	UINT64	1033	
BidImPLYQty03	申买推导量三	UINT64	1332	
BuyPrice04	申买价四	double	1022	
BuyVolume04	申买量四	UINT64	1034	
BidImPLYQty04	申买推导量四	UINT64	1333	
BuyPrice05	申买价五	double	1023	
BuyVolume05	申买量五	UINT64	1035	
BidImPLYQty05	申买推导量五	UINT64	1334	
MBLQuotSelINum	深度卖行情数量	UINT32	1302	接收到的“卖 深度行情域”数量，卖 5 个，不足时补 0
SellPrice01	申卖价一	double	1157	委托价格（最优卖委托价格和最低买与此字段重复）
SellVolume01	申卖量一	UINT64	1168	在该委托价格上的套利定单的委托量（套利最优行情域申卖量与此字段重复）
AskImPLYQty01	申卖推导量一	UINT64	1335	目前各档推导量都显示为 0
SellPrice02	申卖价二	double	1158	
SellVolume02	申卖量二	UINT64	1169	

User Mannual

AskImPLYQty02	申卖推导量二	UINT64	1336	
SellPrice03	申卖价三	double	1159	
SellVolume03	申卖量三	UINT64	1170	
AskImPLYQty03	申卖推导量三	UINT64	1337	
SellPrice04	申卖价四	double	1160	
SellVolume04	申卖量四	UINT64	1171	
AskImPLYQty04	申卖推导量四	UINT64	1338	
SellPrice05	申卖价五	double	1161	
SellVolume05	申卖量五	UINT64	1172	
AskImPLYQty05	申卖推导量五	UINT64	1339	
BuyLevelQueueNo01	买一档揭示委托笔数	UINT32	1272	实际接收到的“买最优价委托量”数量，最多 10 个
BuyLevelQueue	买最优价委托量队列	UINT32	1016	买一价前 10 笔委托
SellLevelQueueNo01	卖一档揭示委托笔数	UINT32	1271	实际接收到的“卖最优价委托量”数量，最多 10 个
SellLevelQueue	卖最优价委托量队列	UINT32	1154	卖一价前 10 笔委托

5.6.33大商所 L2 实时结算价 DCEL2_RealTimePrice

表 5.35 大商所 L2 实时结算价表

字段名	字段含义	用户类型	Tag	描述
LocalTimeStamp	采集时间	INT32	1120	采集服务接收的本地时间 ,格式为 :HH:MM:SS.MMM
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串, 起每位表示特定的含义, 无定义则填空格。 第 1 位对应: '1'表示上海南汇机房行情源, '2'表示深圳福永机房行情源
TradeDate	交易日期	INT32	1208	格式: YYYYMMDD, 用 L2 最优深度行情的最新值填充
Time	数据生成时间	INT32	1191	HH:MM:SS.MMM, 用 L2 最优深度行情的最新值填充
Symbol	合约代码	string(40)	1187	含期货合约品种、期权合约品种
SettlePrice	实时结算价	double	1181	某合约当天的成交价格按照成交量的加权平均。计算公式: 当日均价=成交额/成交量。

5.6.34大商所 L2 委托统计行情 DCEL2_OrderStatistic

表 5.36 大商所 L2 委托统计行情表

字段名	字段含义	用户类型	Tag	描述
LocalTimeStamp	采集时间	INT32	1120	采集服务接收的本地时间 ,格式为 :HH:MM:SS.MMM

User Mannual

QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充空格。 第 1 位对应：'1'表示上海南汇机房行情源，'2'表示深圳福永机房行情源
TradeDate	交易日期	INT32	1208	格式：YYYYMMDD，用 L2 最优深度行情的最新值填充
Time	数据生成时间	INT32	1191	HH:MM:SS.MMM，用 L2 最优深度行情的最新值填充
Symbol	合约代码	string(40)	1187	含期货合约品种、期权合约品种
TotalBuyOrderVolume	委托买入总量	uint64	1195	买方向的所有委托的总委托量，不包括推导量
TotalSellOrderVolume	委托卖出总量	uint64	1201	卖方向的所有委托的总委托量，不包括推导量
WtAvgBuyPrice	加权平均委托买价	double	1230	买方向所有委托的加权平均价，计算公式= $\sum(\text{委托价} \times \text{委托量}) / \text{买委托总量}$
WtAvgSellPrice	加权平均委托卖价	double	1231	卖方向所有委托的加权平均价，计算公式= $\sum(\text{委托价} \times \text{委托量}) / \text{卖委托总量}$

5.6.35大商所 L2 分价成交量行情 DCEL2_MarchPriceQty

表 5.37 大商所 L2 分价成交量表

字段名	字段含义	用户类型	Tag	描述
LocalTimeStamp	采集时间	INT32	1120	采集服务接收的本地时间，格式为 :HH:MM:SS.MMM
QuotationFlag	行情源标志	string(4)	1280	该字段为 4 位字符串，起每位表示特定的含义，无定义则填充空格。 第 1 位对应：'1'表示上海南汇机房行情源，'2'表示

User Mannual

				深圳福永机房行情源
TradeDate	交易日期	INT32	1208	格式：YYYYMMDD，用 L2 最优深度行情的最新值填充
Time	数据生成时间	INT32	1191	HH:MM:SS.MMM，用 L2 最优深度行情的最新值填充
Symbol	合约代码	string(40)	1187	含期货合约品种、期权合约品种
Price01	价格 1	double	1303	成交量最大的 5 个价位区间
PriceBOQty01	买开数量	UINT64	1304	此消息类型中数量字段的值均为累计值。
PriceBEQty01	买平数量	UINT64	1305	
PriceSOQty01	卖开数量	UINT64	1306	
PriceSEQty01	卖平数量	UINT64	1307	
Price02	价格 2	double	1308	
PriceBOQty02	买开数量	UINT64	1309	
PriceBEQty02	买平数量	UINT64	1310	
PriceSOQty02	卖开数量	UINT64	1311	
PriceSEQty02	卖平数量	UINT64	1312	
Price03	价格 3	double	1313	
PriceBOQty	买开数量	UINT64	1314	

User Mannual

03				
PriceBEQty0 3	买平数量	UINT64	1315	
PriceSOQty0 3	卖开数量	UINT64	1316	
PriceSEQty0 3	卖平数量	UINT64	1317	
Price04	价格 4	double	1318	
PriceBOQty 04	买开数量	UINT64	1319	
PriceBEQty0 4	买平数量	UINT64	1320	
PriceSOQty0 4	卖开数量	UINT64	1321	
PriceSEQty0 4	卖平数量	UINT64	1322	
Price05	价格 5	double	1323	
PriceBOQty 05	买开数量	UINT64	1324	
PriceBEQty0 5	买平数量	UINT64	1325	
PriceSOQty0 5	卖开数量	UINT64	1326	
PriceSEQty0 5	卖平数量	UINT64	1327	

1、大连商品交易所 Level-2 行情提供分价位的成交量统计。Level-2 提供 5 个成交量最大的价位区间，分别显示每个价位区间上的买开、卖开、买平、卖平成交量；

2、请注意，这里的成交量只统计基本定单的成交；

3、将涨跌停板范围按照 3*tick 划分成多个子区间，根据区间内基本定单产生的成交量降序

排列，价格为区间的下限。假设跌停板价格是 1，涨停板价格是 9，tick 是 1，那么可以划分成[1,2,3]、[4,5,6]、[7,8,9]三个区间，第一笔成交的成交价为 5，买开与卖平成交 2 手（买卖各 1 手），则将划分到第 2 个区间里，这个区间的价格为 4，则此时计算分价成交行情的话，价格 1 为 4，买开数量为 1，买平数量为 0，卖开数量为 0，卖平数量为 1，其他价格及量均为 0。

6. 接口使用注意事项

介绍本系统接口在使用过程中需知的注意事项，主要包括：

[用户权限](#)

[订阅/取消订阅](#)

[消息类型连接状态](#)

[浮点数精度](#)

[断线重连](#)

[系统延时](#)

[查询数据超时](#)

6.1 用户权限

本系统中的用户权限按消息类型（见表 5.1 消息类型）划分。

6.2 订阅/取消订阅

订阅时，会对基础接口中的消息类型进行检测，请务必保证输入参数及格式正确，若订阅不存在的代码，虽返回订阅成功，但不会有实际的行情数据推送。

取消订阅时，不做权限检测，不管有没有权限的消息类型会提示取消订阅成功。亦不做代码检测，不管取消订阅的代码是否存在、是否已经订阅，都会提示取消订阅成功。

6.3 消息类型连接状态

消息类型连接状态是用户订阅的消息类型所在的行情订阅服务器与客户端 API 的连接状态，并且以与该行情订阅服务器相关的所有消息类型为单位依次返回连接状态。这意味着，有些用户订阅了，但是并没有在当前行情订阅服务器上的消息类型有可能也会返回连接或者断开连接的状态。如下图：

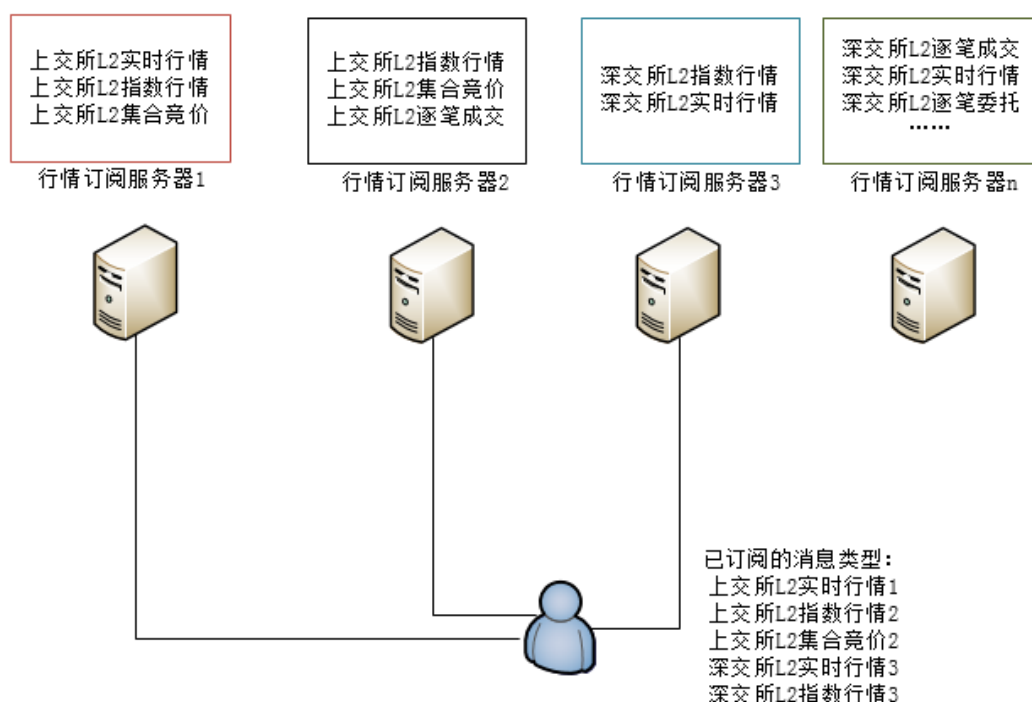


图 6.1 消息类型连接状态图

我们将用户订阅的第 n 个行情订阅服务器上的消息类型标识为“消息类型 n ”。如图，用户已订阅的消息类型为行情订阅服务器 1 上的上交所 L2 实时行情、行情订阅服务器 2 上的上交所 L2 指数行情和上交所 L2 集合竞价还有行情订阅服务器 3 上的深交所 L2 实时行情和深交所 L2 指数行情。当用户 API 与行情订阅服务器 1 断开连接，会收到的连接状态变化的通知：上交所 L2 实时行情、上交所 L2 指数行情和上交所 L2 集合竞价均断开连接。但实际上用户已经订阅的上交所 L2 指数行情和上交所 L2 集合竞价来自于行情订阅服务器 2，并没有断连。所以，此连接状态只能作为参考，并不完全准确。

6.4 浮点数精度

(1) 在使用价格、金额等 double 类型的字段，会出现数值精度与交易所文档不一致情况。

这与 FAST 编解码整型转浮点型有关。因此客户在使用 double 类型字段时，根据自身业务需

求，结合用户手册的字段说明（含精度说明），进行四舍五入处理后便可正常使用。

(2)java 接口: java 语言的浮点数据类型(double 类型)采用 IEEE 754 标准规范，尾数使用 52bit 表示，支持的精度为 15.65 位，即最大有效位数为 15 位。交易所文档对接口对成交总量、成交金额的精度一般大于 15 位（例如上交所 L1 实时行情为 16 位精度，深交所 L2 价格方面字段为 18 位精度）。如交易所传输数据超过 15 位精度，将会出现数据被截取丢失问题。

6.5 断线重连

如接口[基础接口-注册服务](#)所述，正常情况下本系统支持自动重连，用户无需再自写逻辑，但是需注意的是，系统在两种情况下不会自动重连：

第一次调用订阅接口返回不成功（`errCode != Ret_Success`）

同一个账号重复登录时，先登录的用户被断开，且不会自动重连

6.6 系统延时

QTS 内部测延时方案 测试 QTS 采集接收到交易所发来的数据到 QTS 行情订阅服务器下发此数据到客户端 API 的延时，与交易所无关。具体延时部署方法见下图：

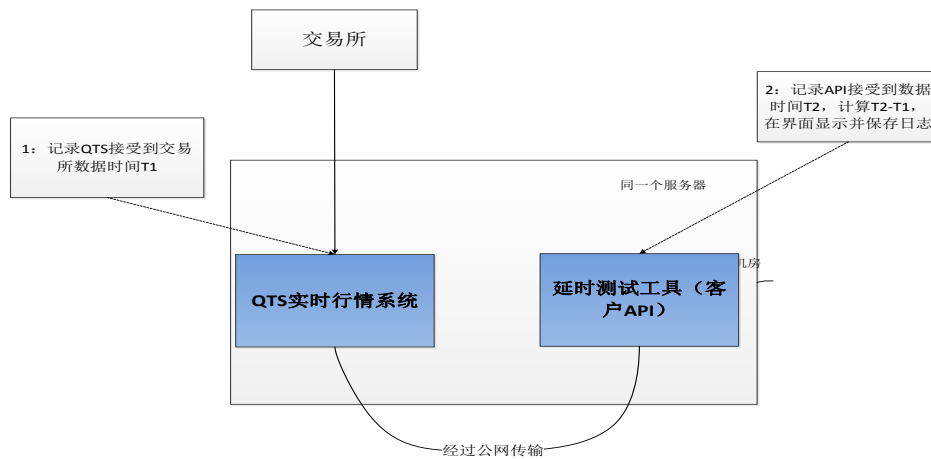


图 6.2 QTS 系统内部测延时图

6.7 查询数据超时

由于查询股票列表、静态数据快照、行情数据快照等快照类数据，一次性发的数据包较大，在超时设置时长不够大或者网络环境不够稳定时，容易返回超时错误。此时，需对返回结果进行判断，如若返回超时错误，则继续请求数据，直至成功。获取股票列表，建议每个市场单独进行获取，否则在超时设置时长不够大或者网络环境不够稳定时，容易返回超时错误。示例代码如下：

获取股票列表：

```
ret = pApiBase->GetStockList("sse", StockList1);
if ( Ret_Success != ret )
{
    if ( Ret_OutTime == ret )
    {
        ret = pApiBase->GetStockList("sse", StockList1);
    }
}
```

```
    }

    if ( Ret_Success != ret )
    {
        printf("GetStockList(sse) error:%d\n", ret);
        break;
    }
}
```

请求静态数据快照：

```
CDataBuffer<SZSEL2_Static> Snap_Static;
ret = pApiBase->QuerySnap_SZSEL2_Static(NULL, Snap_Static);
if ( Ret_Success != ret )
{
    if ( Ret_OutTime == ret )
    {
        ret = pApiBase->QuerySnap_SZSEL2_Static(NULL, Snap_Static);
    }

    if ( Ret_Success != ret )
    {
        printf("QuerySnap_SZSEL2_Static error:%d\n", ret);
        break;
    }
}
```

7. 附录

此处对数据定义中由于篇幅问题未能展开详细描述的细节做进一步解释，主要包括：

[消息类型结构体与基础接口的对应](#)

[当前品种交易状态（只适用上交所 L2 实时行情）](#)

[深圳市场统计指标定义表](#)

[QTS1.X 与 QTS2.X 中金所字段对比](#)

[QTS1.X 与 QTS2.X 上交所 L1 字段对比](#)

[QTS1.X 与 QTS2.X 上期所 L1 字段对比](#)

[QTS1.X 与 QTS2.X 郑商所 L1 字段对比](#)

[QTS1.X 与 QTS2.X 大商所 L1 字段对比](#)

[QTS1.X 与 QTS2.X 大商所 L2 字段对比](#)

[业务 FAQ](#)

7.1消息类型结构体与基础接口对应

每个结构体对应一个查询快照数据函数，且对应一个实时数据回调函数，对应关系如下表：

表 7.1 各个消息类型对应的接口名称

结构体名称	对应的查询函数	对应的实时回调函数
SSEL1_Static	CDataBuffer< SSEL1_Static >& ppSnapData; QuerySnap_SSEL1_Static (Codelist,ppSnapData);	const SSEL1_Static & RealValue; OnSubscribe_SSEL1_Static (RealValue);
SSEL1_Quotation	CDataBuffer< SSEL1_Quotation >& ppSnapData; QuerySnap_SSEL1_Quotation (Codelist,ppSnapData);	const SSEL1_Quotation & RealValue; OnSubscribe_SSEL1_Quotation (RealValue);

User Mannual

SSE_IndexPress	CDataBuffer< SSE_IndexPress >& ppSnapData; QuerySnap_SSE_IndexPress (Codelist,ppSnapData);	const SSE_IndexPress & RealValue; OnSubscribe_SSE_IndexPress (RealValue);
SZSEL1_Static	CDataBuffer< SZSEL1_Static >& ppSnapData QuerySnap_SZSEL1_Static (Codelist,ppSnapData);	const SZSEL1_Static & RealValue; OnSubscribe_SZSEL1_Static (RealValue);
SZSEL1_Quotation	CDataBuffer< SZSEL1_Quotation >& ppSnapData QuerySnap_SZSEL1_Quotation (Codelist,ppSnapData);	const SZSEL1_Quotation & RealValue; OnSubscribe_SZSEL1_Quotation (RealValue);
SSEL2_Static	CDataBuffer< SSEL2_Static >& ppSnapData; QuerySnap_SSEL2_Static (Codelist,ppSnapData);	const SSEL2_Static & RealValue; OnSubscribe_SSEL2_Static (RealValue);
SSEL2_Quotation	CDataBuffer< SSEL2_Quotation >& ppSnapData; QuerySnap_SSEL2_Quotation (Codelist,ppSnapData);	const SSEL2_Quotation & RealValue; OnSubscribe_SSEL2_Quotation (RealValue);
SSEL2_Transaction	CDataBuffer< SSEL2_Transaction >& ppSnapData QuerySnap_SSEL2_Transaction (Codelist,ppSnapData);	const SSEL2_Transaction & RealValue; OnSubscribe_SSEL2_Transaction (RealValue);
SSEL2_Index	CDataBuffer< SSEL2_Index >& ppSnapData; QuerySnap_SSEL2_Index (Codelist,ppSnapData);	const SSEL2_Index & RealValue; OnSubscribe_SSEL2_Index (RealValue);
SSEL2_Auction	CDataBuffer< SSEL2_Auction >& ppSnapData; QuerySnap_SSEL2_Auction (Codelist,ppSnapData);	const SSEL2_Auction & RealValue; OnSubscribe_SSEL2_Auction (RealValue);
SSEL2_Ovreview	CDataBuffer< SSEL2_Ovreview >& ppSnapData; QuerySnap_SSEL2_Ovreview (Codelist,ppSnapData);	const SSEL2_Ovreview & RealValue; OnSubscribe_SSEL2_Ovreview (RealValue);
SSEIOL1_Static	CDataBuffer< SSEIOL1_Static >& ppSnapData	const SSEIOL1_Static & RealValue; OnSubscribe_SSEIOL1_Static

User Mannual

	QuerySnap_SSEIOL1_Static (Codelist,ppSnapData);	(RealValue);
SSEIOL1_Quotation	CDataBuffer< SSEIOL1_Quotation >& ppSnapData QuerySnap_SSEIOL1_Quotation (Codelist,ppSnapData);	const SSEIOL1_Quotation & RealValue; OnSubscribe_SSEIOL1_Quotation (RealValue);
SZSEL2_Static	CDataBuffer< SZSEL2_Static >& ppSnapData QuerySnap_SZSEL2_Static (Codelist,ppSnapData);	const SZSEL2_Static & RealValue; OnSubscribe_SZSEL2_Static (RealValue);
SZSEL2_Quotation	CDataBuffer< SZSEL2_Quotation >& ppSnapData QuerySnap_SZSEL2_Quotation (Codelist,ppSnapData);	const SZSEL2_Quotation & RealValue; OnSubscribe_SZSEL2_Quotation (RealValue);
SZSEL2_Transaction	CDataBuffer< SZSEL2_Transaction >& ppSnapData; QuerySnap_SZSEL2_Transaction (Codelist,ppSnapData);	const SZSEL2_Transaction & RealValue; OnSubscribe_SZSEL2_Transaction (RealValue);
SZSEL2_Index	CDataBuffer< SZSEL2_Index >& ppSnapData QuerySnap_SZSEL2_Index (Codelist,ppSnapData);	const SZSEL2_Index & RealValue; OnSubscribe_SZSEL2_Index (RealValue);
SZSEL2_Order	CDataBuffer< SZSEL2_Order >& ppSnapData QuerySnap_SZSEL2_Order (Codelist,ppSnapData);	const SZSEL2_Order & RealValue; OnSubscribe_SZSEL2_Order (RealValue);
SZSEL2_Status	CDataBuffer< SZSEL2_Status >& ppSnapData QuerySnap_SZSEL2_Status (Codelist,ppSnapData);	const SZSEL2_Status & RealValue; OnSubscribe_SZSEL2_Status (RealValue);
CFFEXL2_Static	CDataBuffer< CFFEXL2_Static >& ppSnapData QuerySnap_CFFEXL2_Static (Codelist,ppSnapData);	const CFFEXL2_Static & RealValue; OnSubscribe_CFFEXL2_Static (RealValue);
CFFEXL2_Quotation	CDataBuffer< CFFEXL2_Quotation >& ppSnapData QuerySnap_CFFEXL2_Quotation (Codelist,ppSnapData);	const CFFEXL2_Quotation & RealValue; OnSubscribe_CFFEXL2_Quotation (RealValue);

User Mannual

SHFEL1_Static	CDataBuffer< SHFEL1_Static >& ppSnapData QuerySnap_ SHFEL1_Static (Codelist,ppSnapData);	const SHFEL1_Static & RealValue; OnSubscribe_ SHFEL1_Static (RealValue);
SHFEL1_Quotation	CDataBuffer< SHFEL1_Quotation >& ppSnapData QuerySnap_ SHFEL1_Quotation (Codelist,ppSnapData);	const SHFEL1_Quotation & RealValue; OnSubscribe_ SHFEL1_Quotation (RealValue);

7.2当前品种交易状态（只适用上交所 L2 实时行情）

表 7.1 当前品种交易状态

TradeStatus	Description
ADD	新产品
BETW	交易间，禁止任何交易活动
BREAK	休市，例如：午餐休市。无撮合和市场内部信息披露。
CLOSE	闭市，自动计算闭市价格
DEL	产品待删除
ENDTR	交易结束
FCALL	固定价格集合竞价
HALT	暂停，除了自有订单和交易的查询之外，任何交易活动都被禁止
SUSP	停牌（SUSP 和 HALT 的区别在于 SUSP 时可以撤单）
ICALL	盘中集合竞价
IOBB	盘中集合竞价订单簿平衡（OBB）
IPOBB	盘中集合竞价 PreOBB
OCALL	开市集合竞价
OBB	开市集合竞价 OBB
OPOBB	开市集合竞价订单簿平衡（OBB）前期阶段

User Mannual

NOTRD	非交易支持非交易服务
POSTR	盘后处理
PRETR	盘前处理
START	启动
TRADE	连续自动撮合
VOLA	连续交易和集合竞价交易的波动性中断 (VOLA)

7.3 深圳市场统计指标定义表

表 7.2 深圳市场统计指标定义表

序号	证券代码	证券简称	计算口径
1	395001	主板 A 股	不含中小板、创业板的所有 A 股的成交量
2	395002	主板 B 股	所有 B 股的成交量
3	395003	中小板	所有中小企业板股票的成交量
4	395004	创业板	所有创业板股票的成交量
5	395011	封闭基金	所有封闭基金的成交量
6	395012	LOFs	所有 LOF 的成交量，但不含分级和非上市的部分
7	395013	ETFs	所有 ETF 的成交量
8	395014	分级基金	所有分级基金的成交量
9	395021	可转债	所有可转债的成交量
10	395022	企业债	所有企业债（非上市公司债券）现券的成交量
11	395024	公司债	所有公司债（上市公司债券）现券的成交量
12	395031	国债	所有国债现券的成交量
13	395032	债券回购	所有债券回购的成交量
14	395041	股票权证	所有以股票为标的的权证的成交量

User Mannual

15	395099	总成交	市场总成交量
----	--------	-----	--------

7.4 QTS1.x 与 QTS2.x 中金所字段对比

表 7.3 中金所 1.X 与 2.X 字段对比表

序号	QTS1.x 字段名	QTS2.x 字段名 (—表示无该字段)	QTS1.x 数据类型	QTS2.x 数据类型	描述
1	szStockNo	Symbol	C6	string(20)	期货代码
2	dChg	——	double	——	涨跌，计算方法： 成交价 — 昨结算价
3	dChgPct	——	double	——	涨跌幅，计算方法： 涨跌 / 昨结算价
4	dVolume	TotalVolume	double	UINT64	当日总量
5	dAmount	TotalAmount	double	double	成交总金额
6	dLastTrade	TradePrice	double	double	成交价
7	dLastVolume	——	double	——	瞬量，计算方法：本笔 当日总量 — 上笔当日 总量
8	dUpB	PriceUpLimit	double	double	涨停价
9	dLowB	PriceDownLimit	double	double	跌停价
10	dHigh	HighPrice	double	double	当日最高
11	dLow	LowPrice	double	double	当日最低
12	dYclose	PreClosePrice	double	double	昨收
13	dOpen	OpenPrice	double	double	开盘

User Mannual

14	dAvgPrice	——	double	——	均价，计算方法为：成交总金额/当日总量/300
15	dPriceGap	——	double	——	档差，无意义，QTS1.9中固定为 0.002。
16	dBuyprice1	BuyPrice01	double	double	买价一
17	dBuyprice2	BuyPrice02	double	double	买价二
18	dBuyprice3	BuyPrice03	double	double	买价三
19	dBuyprice4	BuyPrice04	double	double	买价四
20	dBuyprice5	BuyPrice05	double	double	买价五
21	dBuyvol1	BuyVolume01	double	UINT64	买量一
22	dBuyvol2	BuyVolume02	double	UINT64	买量二
23	dBuyvol3	BuyVolume03	double	UINT64	买量三
24	dBuyvol4	BuyVolume04	double	UINT64	买量四
25	dBuyvol5	BuyVolume05	double	UINT64	买量五
26	dSellprice1	SellPrice01	double	double	卖价一
27	dSellprice2	SellPrice02	double	double	卖价二
28	dSellprice3	SellPrice03	double	double	卖价三
29	dSellprice4	SellPrice04	double	double	卖价四
30	dSellprice5	SellPrice05	double	double	卖价五
31	dSellvol1	SellVolume01	double	UINT64	卖量一
32	dSellvol2	SellVolume02	double	UINT64	卖量二
33	dSellvol3	SellVolume03	double	UINT64	卖量三
34	dSellvol4	SellVolume04	double	UINT64	卖量四
35	dSellvol5	SellVolume05	double	UINT64	卖量五

User Mannual

36	dPreOpenInterest	PreTotalPosition	double	UINT64	昨持仓量
37	dOpenInterest	TotalPosition	double	UINT64	持仓量
38	dSettlementPrice	SettlePrice	double	double	今结算价格
39	szTradingDay	TradeDate	C9	INT32	交易日期
40	szUpdateTime	Time	C9	INT32	更新时间，在 2.1 中与最后修改毫秒合并，称为数据生成时间
41	dPreSettlementPrice	PreSettlePrice	double	double	昨结算价
42	SettlementGroupID	SettleGroupID	C9	string(10)	结算组代码
43	SettlementID	SettleID	int	UINT32	结算编号
44	ClosePrice	ClosePrice	double	double	今收盘
45	PreDelta	PreDelta	double	double	昨虚实度
46	CurrDelta	Delta	double	double	今虚实度
47	UpdateMillisec	——	int	——	最后修改毫秒，在 2.1 中合并于数据生成时间中

7.5 QTS1.x 与 QTS2.x 上交所 L1 字段对比

表 7.4 上交所 L1 静态数据 1.x 与 2.x 字段对比表

User Mannual

序号	QTS1.x 字段名 (一表示无该字段)	QTS2.x 字段名 (——表示无该字段)	QTS1.x 数据类型	QTS2.x 数据类型	描述
1	Symbol	SecurityName	C10	string(20)	
2	SecurityID	Symbol	C7	string(20)	
3	SecurityIDSource	——	C5	——	交易所旧字段,无值
4	szSecurityAltID[7]	——	char	——	交易所旧字段,无值
5	szSecurityAltIDSource[4]	——	char	——	交易所旧字段,无值
6	szCFIcode	CFIcode	C5	string(6)	‘ES’表示股票；‘EU’表示基金；‘D’表示债券；‘RWS’表示权证；‘FF’表示期货。
7	SecurityExchange[7]	——	char	——	交易所旧字段,无值
8	SecuritySubType	SecuritySubType	C5	string(6)	GBV 浮动利率国债 GBF 固定利率国债 GBZ 无息国债 DST 国债分销（仅用于分销阶段） DVP 公司债分销 CBV 浮动利率企业债券（不包括可转换企业债券） CBF 固定利率企业债券（不包括可转换企业债券） CCV 浮动利率可转换企业债券 CCF 固定利率可转换企

User Mannual

					业债券 CPV 浮动利率债券 CPF 固定利率公司债券 FBV 浮动利率金融机构 (即银行、保险公司、 债 券公司)发行债券 FBF 固定利率金融机构 (即银行、保险公司、 债 券公司)发行债券 CRP 质押式国债回购 BRP 质押式企债回购 ORP 买断式债券回购 CBD 分离式可转债 BBL 债券借贷 OBD 其它债券 CEF 封闭式基金 OEF 开放式基金 EBS 交易所交易基金 (买卖) FBL 基金借贷 OFN 其它基金 ASH 以人民币交易的 股票 BSH 以美元交易的股 票 CSH 国际版股票 EBL 股票借贷 OEQ 其它股票 CIW 企业发行权证 COV 备兑权证 CER 凭证式权证 OWR 其它权证 FIX 指数期货 FEQ 个股期货 FBD 债券期货 OFT 其它期货
9	ContractM ultiplier	——	double	——	交易所旧字段,无值

User Mannual

10	SecurityDesc[20]	——	char	——	交易所旧字段,无值
11	PreClosePx	PreClosePrice	double	Double	
12	RoundLot[5]	——	char	——	交易所旧字段,无值
13	InterestAccrualDate	——	DWORD	——	交易所旧字段,无值
14	price	——	double	——	交易所旧字段,无值
15	IndustryClassification[5]	——	char	——	
16	Currency	Currency	C5	string(5)	美元： USD；人民币： CNY
17	ShareFaceValue	——	double	——	
18	OutstandingShares	——	UINT64	——	
19	PublicFloatShareQuantity	——	UINT64	——	
20	SecurityTradingStatus	——	int	——	交易所旧字段
21	CorporateAction	——	C5	——	第1位设置，为D 第2位设置，为R 两个都设置，为 DR
22	MaturityDate	——	DWORD	——	交易所旧字段,无值
23	CouponRate[2]	——	char	——	交易所旧字段,无值

User Mannual

24	CouponPaymentDate	——	DWORD	——	交易所旧字段,无值
25	LastInterestPaymentDate	——	DWORD	——	交易所旧字段,无值
26	NextInterestPaymentDate	——	DWORD	——	交易所旧字段,无值
27	ConversionPrice	——	double	——	交易所旧字段,无值
28	ResidualMaturity[13]	——	char	——	交易所旧字段,无值
29	AccruedInterestAmt	——	double	——	交易所旧字段,无值
30	BondType[2]	——	char	——	交易所旧字段,无值
31	American_European[2]	——	char	——	交易所旧字段,无值
32	CallOrPut[2]	——	char	——	交易所旧字段,无值
33	Underlying	SymbolUnderlying	C7	string(20)	基础证券
34	Issuer[20]	——	char	——	交易所旧字段,无值
35	CVRatio	——	double	——	交易所旧字段,无值
36	_CouponRate	——	double	——	交易所旧字段,无值
37	ConversionPeriod	——	DWORD	——	交易所旧字段,无值
38	InterestRate	——	double	——	交易所旧字段,无值

User Mannual

	e				
39	szStateID[8]]	——	char	——	交易所旧字段,无值
40	fUpperLmt Prc	PriceUpLimit	float	double	
41	fLowerLmt Prc	PriceDownLimit	float	Double	
42	——	LocalTimeStamp	——	int32	数据到达时本系统记录的时间,精确到毫秒
43	——	QuotationFlag	——	string(4)	该字段为 4 位字符串,起每位表示特定的含义,无定义则填空格。第 1 位对应 '1' 表示上海南汇机房行情源; '2' 表示深圳福永机房行情源
44	——	Time	——	int32	标识接口中本记录更新时间 HH:MM:SS
45	——	ISINCode	——	String(12)	交易所预留
46	——	SecurityEN	——	string(24)	交易所预留
47	——	MarketType	——	string(6)	'ASHR'表示 A 股市场; 'BSHR'表示 B 股市场; 'CSHR'表示国际版市场。
48	——	ParValue	——	int64	债券当前面值,单位元,其他产品取 0.000
49	——	TradableMarketNo	——	int64	交易所预留字段
50	——	EndDate	——	int 64	对于国债预发行产品,为最后交易日期,格式为 YYYYMMDD

User Mannual

51	——	ListingDate	——	int 32	上市日期
52	——	SetNo	——	uint32	产品集编号
53	——	BuyVolumeUnit	——	uint32	买数量单位
54	——	SellVolumeUnit	——	uint32	买数量单位
55	——	DeclareVolumeFloor	——	uint32	申报数量下限
56	——	DeclareVolumeCeiling	——	uint32	申报数量上限
57	——	TickSize	——	double	申报价格的最小变动单位
58	——	UpDownLimitType	——	string(1)	‘N’ 表示交易规则 3.4.13 规定的有涨跌幅限制类型或者权证管理办法第 22 条规定；‘R’ 表示交易规则 3.4.15 和 3.4.16 规定的无涨跌幅限制类型；‘S’ 表示回购涨跌幅控制类型。
59	——	XRRatio	——	double	每股送股比例；对于国债预发行产品，为保证金比例。
60	——	XDAmount	——	double	每股分红金额
61	——	CrdBuyUnderlying	——	string(1)	‘T’ 表示是融资标的证券；‘F’ 表示不是融资标的证券。
62	——	CrdSellUnderlying	——	string(1)	‘T’ 表示是融券标的证券；‘F’ 表示不是融券标的证券。
63	——	SecurityStatus	——	string(20)	该字段为 20 位字符串，每位表示允许对应的业务，无定义则填空格。第 0 位对应：‘N’ 表示首日上市。

User Mannual

					<p>第 1 位对应：‘D’ 表示除权。</p> <p>第 2 位对应：‘R’ 表示除息。</p> <p>第 3 位对应：‘D’ 表示国内主板正常交易产品；‘S’ 表示股票风险警示产品；‘P’ 表示退市整理产品；‘T’ 表示退市转让产品；‘U’ 表示优先股产品。</p> <p>第 4 位对应：‘S’ 表示债券风险警示类。第 4 位未来改为不启用，此改动暂未实施，具体以交易所通知为准。</p> <p>第 5 位对应：‘L’ 表示债券投资者适当性要求类。</p>
--	--	--	--	--	---

表 7.5 上交所 L1 实时行情 1.X 与 2.X 字段对比表

序号	QTS1.x 字段名（一表示无该字段）	QTS2.x 字段名（一表示无该字段）	QTS1.x 数据类型	QTS2.x 数据类型	描述
1	szStockNo	Symbol	C6	string(20)	证券代码
2	szName	SecurityName	C8	string(20)	证券简称
3	szDate	——	C9	——	
4	szTime	——	C9	——	
5	fLasttrade	LastPrice	float	double	现价
6	dVolume	TotalVolume	double	UINT64	成交总量

User Mannual

7	dAmount	TotalAmount	double	double	成交总额
8	fpTrans	——	float	——	
9	fOpen	OpenPrice	float	double	开盘价
10	fYClose	PreClosePrice	float	double	昨收价
11	fBuyprice1	BuyPrice01	float	Double	
12	fBuyprice2	BuyPrice02	float	Double	
13	fBuyprice3	BuyPrice03	float	Double	
14	fBuyprice4	BuyPrice04	float	Double	
15	fBuyprice5	BuyPrice05	float	double	
16	fBuyvol1	BuyVolume01	float	UINT64	
17	fBuyvol2	BuyVolume02	float	UINT64	
18	fBuyvol3	BuyVolume03	float	UINT64	
19	fBuyvol4	BuyVolume04	float	UINT64	
20	fBuyvol5	BuyVolume05	float	UINT64	
21	fSellprice1	SellPrice01	float	Double	
22	fSellprice2	SellPrice02	float	Double	
23	fSellprice3	SellPrice03	float	Double	
24	fSellprice4	SellPrice04	float	Double	
25	fSellprice5	SellPrice05	float	double	
26	fSellvol1	SellVolume01	float	UINT64	
27	fSellvol2	SellVolume02	float	UINT64	
28	fSellvol3	SellVolume03	float	UINT64	
29	fSellvol4	SellVolume04	float	UINT64	
30	fSellvol5	SellVolume05	float	UINT64	

User Mannual

31	fLow	LowPrice	float	double	
32	fHigh	HighPrice	float	double	
33	fPE	——	float	——	市盈率
34	——	LocalTimeSta mp	——	int32	数据到达时本系统记录的时间， 精确到毫秒
35	——	PacketTimeSt amp	——	int64	交易所发包时间，格式为：日期 时间 YYYYMMDDHHMMSSMMM
36	——	QuotationFlag	——	String(4)	该字段为 4 位字符串，起每位表 示特定的含义，无定义则填充格。 第 1 位对应：‘1’ 表示上海南汇 机房行情源；‘2’ 表示深圳福永 机房行情源
37	——	Time	——	int32	时间戳 143025 表示 14:30:25
38	——	ClosePrice	——	double	
39	——	NAV	——	double	四位小数
40	——	IOPV	——	double	
41	——	TradeStatus	——	string(8)	该字段为 8 位字符串，左起每位 表示特定的含义，无定义则填充 格。 第 1 位：‘S’ 表示启动（开市前） 时段，‘C’ 表示集合竞价时段， ‘T’ 表示连续交易时段，‘B’ 表 示休市时段，‘E’ 表示闭市时段， ‘P’ 表示产品停牌。 第 2 位：‘0’ 表示未连续停牌， ‘1’ 表示连续停牌。无意义填充 格。 第 3 位：‘0’ 表示未上市，‘1’ 表示已上市。

7.6 QTS1.X 与 QTS2.X 上期所 L1 字段对比

表 7.5 上期所 1.X 与 2.X 字段对比表

序号	QTS1.x 字段名 (一表示无该字段)	QTS2.x 字段名 (一表示无该字段)	QTS1.x 数据类型	QTS2.x 数据类型	描述
1	TradingDay	TradeDate	C9	INT32	交易日期
2	SettlementGroupID	SecurityName	SettleGroupID	string(10)	结算组代码
3	SettlementID	SettleID	int	UINT32	结算编号
4	LastPrice	LastPrice	double	double	最新价
5	PreSettlementPrice	PreSettlePrice	double	double	昨结算
6	PreClosePrice	PreClosePrice	double	double	昨收盘
7	PreOpenInterest	PreTotalPosition	double	double	昨持仓量
8	OpenPrice	OpenPrice	double	double	今开盘
9	HighestPrice	HighPrice	double	double	最高价
10	LowestPrice	LowPrice	double	double	最低价
11	Volume	TotalVolume	int	UINT64	数量
12	Turnover	TotalAmount	double	double	成交金额
13	OpenInterest	TotalPosition	double	double	持仓量
14	ClosePrice	ClosePrice	double	double	今收盘

User Mannual

15	SettlementPrice	SettlePrice	double	double	今结算
16	PriceUpLimit	PriceUpLimit	double	double	涨停板价
17	LowerLimitPrice	PriceDownLimit	double	double	跌停板价
18	PreDelta	PreDelta	double	double	昨虚实度
19	CurrDelta	Delta	double	double	今虚实度
20	UpdateTime	Time	C9	INT32	最后修改时间
21	UpdateMilliSec		int		最后修改毫秒
22	InstrumentID	Symbol	C31	string(20)	合约代码
23	BidPrice1	BuyPrice01	double	double	申买价一
24	BidVolume1	BuyVolume01	int	UINT64	申买量一
25	AskPrice1	SellPrice01	double	double	申卖价一
26	AskVolume1	SellVolume01	int	UINT64	申卖量一
27	BidPrice2	——	double	——	申买价二
28	BidVolume2	——	int	——	申买量二
29	AskPrice2	——	double	——	申卖价二
30	AskVolume2	——	int	——	申卖量二
31	BidPrice3	——	double	——	申买价三
32	BidVolume3	——	int	——	申买量三
33	AskPrice3	——	double	——	申卖价三
34	AskVolume3	——	int	——	申卖量三
35	BidPrice4	——	double	——	申买价四

User Mannual

36	BidVolume4	——	int	——	申买量四
37	AskPrice4	——	double	——	申卖价四
38	AskVolume4	——	int	——	申卖量四
39	BidPrice5	——	double	——	申买价五
40	BidVolume5	——	int	——	申买量五
41	AskPrice5	——	double	——	申卖价五
42	AskVolume5	——	int	——	申卖量五
43	dwTransacti on	——	dword	——	成交笔数
44	ActionDay	ActionDay	C9	INT32	业务发生日期
45	——	LocalTimeSta mp	——	INT32	采集时间
46	——	QuotationFlag	——	string(4)	行情源标志

7.7 QTS1.X 与 QTS2.X 郑商所 L1 字段对比

表 7.6 郑交所 L1 1.X 与 2.X 字段对比表

序 号	QTS1.x 字 段名（一表 示无该字段）	QTS2.x 字段 名（一表示无该 字段）	QTS1.x 数据 类型	QTS2.x 数据类型	描述
1	szInstrumentID	Symbol	C20	string(40)	合约编码
2	dPreClosePr	PreClosePrice	double	double	前收盘价格

User Mannual

	ice				
3	dPreSettlePrice	PreSettlePrice	double	double	前结算价格
4	dOpenPrice	OpenPrice	double	double	开盘价
5	dBidPrice	BuyPrice01	double	double	买入价格
6	dwBidSize	BuyVolume01	DWORD	UINT64	买入数量
7	dAskPrice	SellPrice01	double	double	卖出价
8	dwAskSize	SellVolume01	DWORD	UINT64	卖出数量
9	dClosePrice	LastPrice	double	double	最新价
10	dwOpenInterest	TotalPosition	DWORD	uint64	持仓量
11	dHighPrice	HighPrice	double	double	最高价
12	dLowPrice	LowPrice	double	double	最低价
13	dClosePrice2	ClosePrice	double	double	收盘价
14	dSettlePrice	ClearPrice	double	double	当日交割结算价
15	dClearPrice	SettlePrice	double	double	结算价
16	dAveragePrice	AveragePrice	double	double	均价
17	dListHighPrice	LifeHigh	double	double	历史最高成交价格
18	dListLowPrice	LifeLow	double	double	历史最低成交价格
19	dUpperLimit	PriceUpLimit	double	double	涨停板
20	dLowerLimit	PriceDownLimit	double	double	跌停板

User Mannual

21	dwTotalVolume	TotalVolume	double	UINT64	总成交量
22	dwTradingDate	TradeDate	DWORD	TradeDate	交易日期
23	dwTradingTime	Time	DWORD	int32	交易时间
24	dwTransaction	——	DWORD	——	成交笔数
25	——	LocalTimeStamp	——	INT32	采集时间
26	——	QuotationFlag	——	string(4)	行情源标志
27	——	Version	——	string(1)	合约版本号
28	——	PreTotalPosition	——	UINT64	昨持仓量
29	——	TotalAmount	——	double	成交总额
30	——	DeriveBidPrice	——	double	组合买入价
31	——	DeriveAskPrice	——	double	组合卖出价
32	——	DeriveBidLot	——	UINT64	组合买入数量
33	——	DeriveAskLot	——	UINT64	组合卖出数量
34	——	LastLot	——	UINT64	最后一笔成交手数

7.8 QTS1.X 与 QTS2.X 大商所 L1 字段对比

表 7.6 上期所 1.X 与 2.X 字段对比表

User Mannual

序号	QTS1.x 字段名 (一表示无该字段)	QTS2.x 字段名 (一表示无该字段)		QTS1.x 数据类型	QTS2.x 数据类型	描述
		当 btQuotFlag=1, 映射 2.6 大商所 L1 最优行情	当 btQuotFlag=2, 映射 2.6 大商所 L1 套利行情			
1	wSize	——	——	WORD	——	数据包大小
2	btQuotFlag	——	——	BYTE	——	消息类型标志
3	szTradingDate	TradeDate	TradeDate	C8	C8	交易日期
4	szContractID	Symbol	Symbol	C20	C20	合约号
5	nID	RoutineNo	RoutineNo	UINT	Uint32	事务编号
6	szContractName	SecurityName	——	C40	string(40)	合约名称
7	dClosePrice	LastPrice	LastPrice	double	double	最新价
8	dHighPrice	HighPrice	HighPrice	double	double	最高价
9	dLowPrice	LowPrice	LowPrice	double	double	最低价
10	nTotalVolume	LastMatchQty	——	UINT64	UINT	最新成交量
11	nPrevTotalVolume	TotalVolume	——	UINT	UINT64	成交量
12	dTurnover	TotalAmount	——	double	double	成交额
13	nPrevOpenInterest	PreTotalPosition	——	UINT	UINT64	初始持仓量

User Mannual

	nterest					
14	nOpenInterest	TotalPosition	——	UINT	UINT64	持仓量
15	nInterestChg	InterestChg	——	INT	INT	持仓量变化
16	dClearPrice	SettlePrice	——	double	double	今结算价
17	dListLowPrice	LifeHigh	LifeHigh	double	double	历史最低价
18	dListHighPrice	LifeLow	LifeLow	double	double	历史最高价
19	dUpperLimit	PriceUpLimit	PriceUpLimit	double	double	涨停板
20	dLowerLimit	PriceDownLimit	PriceDownLimit	double	double	跌停板
21	dLastClearPrice	PreSettlePrice	——	double		上日结算价
22	dClosePrice3	PreClosePrice	——	double	double	上日收盘价
23	dBidPrice	BuyPrice01	BuyPrice01	double	double	最高买
24	nBidSize	BuyVolume01	BuyVolume01	UINT	UINT64	申买量
25	nBidImpliedQty	BidImpliedQty01	——	UINT	UINT64	申买推导量
26	dAskPrice	SellPrice01	SellPrice01	double	double	最低卖
27	nAskSize	SellVolume01	SellVolume01	UINT	UINT64	申卖量

User Mannual

28	nAskImPLYQty	AskImPLYQty01	——	UINT	UINT64	申卖推导量
29	dAveragePrice	AveragePrice	——	double	double	当日均价
30	szTradingTime	Time	Time	C12	INT32	生成时间
31	dOpenPrice	OpenPrice	——	double	double	开盘价
32	dClosePrice2	ClosePrice	——	double	double	收盘价
33	——	LocalTimeStamp	——	——	INT32	采集时间
34	——	QuotationFlag	——	——	string(4)	行情源标志
35	——	Delta	——	——	double	Delta
36	——	Gamma	——	——	double	Gamma
37	——	Rho	——	——	double	Rho
38	——	Theta	——	——	double	Theta
39	——	Vega	——	——	double	Vega

7.9 QTS1.X 与 QTS2.X 大商所 L2 字段对比

表 7.7 大商所 L2 套利深度行情 1.X 与 2.X 字段对比表

序号	QTS1.x 字段名（—表示无该字段）	QTS2.x 字段名（—表示无该字段）	QTS1.x 数据类型	QTS2.x 数据类型	描述

User Mannual

1	szTradeDate	TradeDate	C8	INT32	交易日期
2	sz	Symbol	C80	string(40)	套利合约号
3	nTID	RoutineNo	UINT	Uint32	事务编号
4	dLastPrice	LastPrice	double	double	最新价
5	dLowPrice	LowPrice	double	double	最低价
6	dHighPrice	HighPrice	double	double	最高价
7	dListLowPrice	LifeLow	double	double	历史最低价
8	dListHighPrice	LifeHigh	double	double	历史最高价
9	dUpperLimit	PriceUpLimit	double	double	涨停板
10	dLowerLimit	PriceDownLimit	double	double	跌停板
11	dBidPrice	BuyPrice01	double	double	最高买
12	nBidVolume	BuyVolume01	UINT	UINT64	申买量
13	dAskPrice	SellPrice01	double	double	最低卖
14	nAskVolume	SellVolume01	UINT	UINT64	申卖量

User Mannual

15	szGenTime	Time	C8	INT32	生成时间
16	dBestBuyOrderPrice	BuyPrice01	double	double	买委托价格
17	nBestBuyOrderQtyOne	BuyLevelQueue	UINT	UINT32	买委托量 1
18	nBestBuyOrderQtyTwo		UINT	UINT32	买委托量 2
19	nBestBuyOrderQtyThree		UINT	UINT32	买委托量 3
20	nBestBuyOrderQtyFour		UINT	UINT32	买委托量 4
21	nBestBuyOrderQtyFive		UINT	UINT32	买委托量 5
22	nBestBuyOrderQtySix		UINT	UINT32	买委托量 6
23	nBestBuyOrderQtySeven		UINT	UINT32	买委托量 7
24	nBestBuyOrderQtyEight		UINT	UINT32	买委托量 8
25	nBestBuyOrderQtyNine		UINT	UINT32	买委托量 9
26	nBestBuyOrderQtyTen		UINT	UINT32	买委托量 10
27	dBestSellOrderPrice	SellPrice01	double	double	卖委托价格
28	nBestSellOrder	SellLevelQueue	UINT	UINT32	卖委托量 1

User Mannual

	derQtyOne	e			
29	nBestSellOrderQtyTwo		UINT	UINT32	卖委托量 2
30	nBestSellOrderQtyThree		UINT	UINT32	卖委托量 3
31	nBestSellOrderQtyFour		UINT	UINT32	卖委托量 4
32	nBestSellOrderQtyFive		UINT	UINT32	卖委托量 5
33	nBestSellOrderQtySix		UINT	UINT32	卖委托量 6
34	nBestSellOrderQtySeven		UINT	UINT32	卖委托量 7
35	nBestSellOrderQtyEight		UINT	UINT32	卖委托量 8
36	nBestSellOrderQtyNine		UINT	UINT32	卖委托量 9
37	nBestSellOrderQtyTen		UINT	UINT32	卖委托量 10
38	szGenTime	——	C12	——	生成时间
39	MBLQuoteBuyNum	MBLQuoteBuyNum	int	UINT32	深度行情买数量
40	OrderPrice	BuyPrice01~BuyPrice05	Double	double	价格
	OrderQty	BuyVolume01~BuyVolume01	Uint	UINT64	委托量

User Mannual

	ImPLYQty	BidImPLYQty0 1~ BidImPLYQty0 15	Uint	UINT64	推导量
	BsFlag	——	Uint	——	买卖标志
	GenTime	——	Char(12)	——	生成时间
41	MBLQuotSel INum	MBLQuottSell Num	int	UINT32	深度行情卖数量
42	OrderPrice	SellPrice01~ SellPrice05	Double	double	价格
	OrderQty	SellVolume01 ~ SellVolume01	Uint	UINT64	委托量
	ImPLYQty	AskImPLYQty0 1~ AskImPLYQty0 15	Uint	UINT64	推导量
	BsFlag	——	Uint	——	买卖标志
	GenTime	——	Char(12)	——	生成时间
43	LocalTimeSt amp	——	INT32	——	采集时间
44	QuotationFl ag	——	string(4)	——	行情源标志

表 7.8 大商所 L2 最优深度行情 1.x 与 2.x 字段对比表

User Mannual

序号	QTS1.x 字段名 (一表示无该字段)	QTS2.x 字段名 (一表示无该字段)	QTS1.x 数据类型	QTS2.x 数据类型	描述
1	szTradingDate	TradeDate	C8	INT32	交易日期
2	szContractID	Symbol	C20	string(40)	合约号
3	nTID	RoutineNo	UINT	Uint32	事务编号
4	szContractName	SecurityName	C40	string(40)	合约名称
5	dLastPrice	LastPrice	double	double	最新价
6	dHighPrice	HighPrice	double	double	最高价
7	dLowPrice	LowPrice	double	double	最低价
8	nTotalVolum	LastMatchQty	UINT	UINT64	最新成交量
9	nPrevTotalVolume	TotalVolume	UINT	UINT64	成交量
10	dTurnover	TotalAmount	double	double	成交额
11	nPrevOpenInterest	PreTotalPosition	UINT	UINT64	初始持仓量
12	nOpenInterest	TotalPosition	UINT	UINT64	持仓量
13	nInterestChg	InterestChg	int	UINT64	持仓量变化
14	dClearPrice	SettlePrice	double	double	今结算价
15	dListLowPrice	LifeHigh	double	double	历史最低价
16	dListHighPrice	LifeLow	double	double	历史最高价

User Mannual

	ce				
17	dUpperLimit	PriceUpLimit	double	double	涨停板
18	dLowerLimit	PriceDownLimit	double	double	跌停板
19	dLastClearPrice	PreSettlePrice	double	double	上日结算价
20	dLastClose	PreClosePrice	double	double	上日收盘价
21	dBidPrice	BuyPrice01	double	double	最高买
22	nBidSize	UINT64	UINT	double	申买量
23	nBidImPLYQty	UINT64	UINT	double	申买推导量
24	dAskPrice	SellPrice01	double	double	最低卖
25	nAskSize	UINT64	UINT	double	申卖量
26	nAskImPLYQty	UINT64	UINT	double	申卖推导量
27	dAveragePrice	AveragePrice	double	double	当日均价
28	szTradingTime	Time	C12	Int32	交易时间
29	dOpenPrice	OpenPrice	double	double	开盘价
30	dClosePrice	ClosePrice	double	double	收盘价
31	dBestBuyOrderPrice	BuyPrice01	double	double	买委托价格
32	nBestBuyOrderQtyOne	BuyLevelQueue	UINT	UINT32	买委托量 1
	nBestBuyOrderQtyTwo		UINT		买委托量 2

User Mannual

	nBestBuyOrderQtyThree		UINT		买委托量 3
	nBestBuyOrderQtyFour		UINT		买委托量 4
	nBestBuyOrderQtyFive		UINT		买委托量 5
	nBestBuyOrderQtySix		UINT		买委托量 6
	nBestBuyOrderQtySeven		UINT		买委托量 7
	nBestBuyOrderQtyEight		UINT		买委托量 8
	nBestBuyOrderQtyNine		UINT		买委托量 9
	nBestBuyOrderQtyTen		UINT		买委托量 10
33	dBestSellOrderPrice	SellPrice01	double	double	卖委托价格
34	nBestSellOrderQtyOne	SellLevelQueue	UINT	UINT32	卖委托量 1
	nBestSellOrderQtyTwo		UINT		卖委托量 2
	nBestSellOrderQtyThree		UINT		卖委托量 3
	nBestSellOrderQtyFour		UINT		卖委托量 4
	nBestSellOrderQtyFive		UINT		卖委托量 5

User Mannual

	derQtyFive				
	nBestSellOrderQtySix		UINT		卖委托量 6
	nBestSellOrderQtySeven		UINT		卖委托量 7
	nBestSellOrderQtyEight		UINT		卖委托量 8
	nBestSellOrderQtyNine		UINT		卖委托量 9
	nBestSellOrderQtyTen		UINT		卖委托量 10
35	szGenTime		C12		生成时间
35	MBLQuotBuyNum	MBLQuotBuyNum	----	UINT32	深度行情买数量
36	OrderPrice	BuyPrice01~BuyPrice05	Double	double	价格
37	OrderQty	BuyVolume01~BuyVolume05	Uint	UINT64	委托量
38	ImpleyQty	BidImpleyQty01~BidImpleyQty05	Uint	UINT64	推导量
39*	BsFlag	——	Uint	——	买卖标志
40	GenTime	——	Char(12)	——	生成时间
41	MBLQuotSellNum	MBLQuotSellNum	----	UINT32	深度行情卖数量

User Mannual

42	OrderPrice	BuyPrice01~ BuyPrice05	Double	double	价格
43	OrderQty	BuyVolume01 ~ BuyVolume05	Uint	double	委托量
44	ImPLYQty	BidImPLYQty0 1~ BidImPLYQty0 5	Uint	double	推导量
45	BsFlag	——	Uint	——	买卖标志
46	GenTime	——	Char(12)	——	生成时间
47	——	LocalTimeSta mp	——	INT32	采集时间
48	——	QuotationFlag	——	string(4)	行情源标志
49	——	BuyLevelQueu eNo01	——	UINT32	买一档揭示委托笔数
50	——	SellLevelQueu eNo01	——	UINT32	卖一档揭示委托笔数
51	——	Delta	——	double	Delta
52	——	Gamma	——	double	Gamma
53	——	Rho	——	double	Rho
54	——	Theta	——	double	Theta
55	——	Vega	——	double	Vega

表 7.8 大商所 L2 实时结算价 1.X 与 2.X 字段对比表

User Mannual

序号	QTS1.x 字段名 (一表示无该字段)	QTS2.x 字段名 (一表示无该字段)	QTS1.x 数据类型	QTS2.x 数据类型	描述
1	szContractID	合约号	C80	string(40)	合约号
2	dRealTimePrice	实时结算价	double	double	实时结算价
3	——	LocalTimeStamp	——	INT32	采集时间
4	——	QuotationFlag	——	string (4)	行情源标志
5	——	TradeDate	——	INT32	交易日期
6	——	Time	——	INT32	数据生成时间

表 7.8 大商所 L2 委托统计行情 1.x 与 2.x 字段对比表

序号	QTS1.x 字段名 (一表示无该字段)	QTS2.x 字段名 (一表示无该字段)	QTS1.x 数据类型	QTS2.x 数据类型	描述
1	szContractID	Symbol	C80	string(40)	合约号
2	nTotalBuyOrderNum	TotalBuyOrderVolume	UINT	uint64	买委托总量
3	nTotalSellOrderNum	TotalSellOrderVolume	UINT	uint64	卖委托总量
4	dWeightedAverageBuyO	WtAvgBuyPrice	double	int64	加权平均委买价格

User Mannual

	rderPrice				
5	dWeightedAverageSelloOrderPrice	WtAvgSellPrice	double	int64	加权平均委卖价格
6	——	LocalTimeStamp	——	INT32	采集时间
7	——	QuotationFlag	——	string (4)	行情源标志
8	——	TradeDate	——	INT32	交易日期
9	——	Time	——	INT32	数据生成时间

表 7.8 大商所 L2 分价成交量行情 1.X 与 2.X 字段对比表

序号	QTS1.x 字段名 (一表示无该字段)	QTS2.x 字段名 (一表示无该字段)	QTS1.x 数据类型	QTS2.x 数据类型	描述
1	szContractID	Symbol	C80	string(40)	合约号
2	dPriceOne	Price01	double	double	价格 1
3	nPriceOneBOQty	PriceBOQty01	UINT	UINT64	买开数量
4	nPriceOneBEQty	PriceBEQty01	UINT	UINT64	买平数量
5	nPriceOneSOQty	PriceSOQty01	UINT	UINT64	卖开数量
6	nPriceOneSEQty	PriceSEQty01	UINT	UINT64	卖平数量

User Mannual

7	dPriceTwo	Price02	double	double	价格 2
8	nPriceTwoB OQty	PriceBOQty02	UINT	UINT64	买开数量
9	nPriceTwoB EQty	PriceBEQty02	UINT	UINT64	买平数量
10	nPriceTwoS OQty	PriceSOQty02	UINT	UINT64	卖开数量
11	nPriceTwoS EQty	PriceSEQty02	UINT	UINT64	卖平数量
12	dPriceThree	Price03	double	double	价格 3
13	nPriceThree BOQty	PriceBOQty03	UINT	UINT64	买开数量
14	nPriceThree BEQty	PriceBEQty03	UINT	UINT64	买平数量
15	nPriceThree SOQty	PriceSOQty03	UINT	UINT64	卖开数量
16	nPriceThree SEQty	PriceSEQty03	UINT	UINT64	卖平数量
17	dPriceFour	Price04	double	double	价格 4
18	nPriceFourB OQty	PriceBOQty04	UINT	UINT64	买开数量
19	nPriceFourB EQty	PriceBEQty04	UINT	UINT64	买平数量
20	nPriceFourS OQty	PriceSOQty04	UINT	UINT64	卖开数量
21	nPriceFourS EQty	PriceSEQty04	UINT	UINT64	卖平数量
22	dPriceFive	Price05	double	double	价格 5

User Mannual

23	nPriceFiveB OQty	PriceBOQty05	UINT	UINT64	买开数量
24	nPriceFiveB EQty	PriceBEQty05	UINT	UINT64	买平数量
25	nPriceFiveS OQty	PriceSOQty05	UINT	UINT64	卖开数量
26	nPriceFiveS EQty	PriceSEQty05	UINT	UINT64	卖平数量
27	——	LocalTimeSta mp	——	INT32	采集时间
29	——	QuotationFlag	——	string (4)	行情源标志
29	——	TradeDate	——	INT32	交易日期
30	——	Time	——	INT32	数据生成时间

7.10 业务 FAQ

序号	正文
1	Q: QTS 提供了哪些交易所的数据
	A: 交易所发送的数据的内容分为两类：静态数据（包含盘前、盘后信息，正常情况下每天交易所只推送一次）、实时行情（包含盘中交易的量价信息，交易时间段即时更新）；数据内容按深度分为 Level1 和 Level2：Level1 为基本行情数据（五档行情、高开低收、成交量、成交金额等），Level2 在 Level-1 行情数据的基础上增加增值信息（10 档行情、逐笔成交（每一笔成交明细）、逐笔委托（含第 1 档前 50 笔委托）、统计数据（买卖方累计撤单笔数）等深度更大的数据）。
2	Q: QTS 接口中实时订阅、快照查询的区别？
	A: 实时订阅：实时接收交易所推送的数据，如果交易所无推送相关数据，则 API 接收不到数据。

User Mannual

	快照查询：通过在 QTS 保存的缓存数据，获取最新一笔数据记录。			
3	Q:上交所 L2 如何获取逐笔委托？			
	A:深交所 L2 有提供逐笔委托消息类型，上交所 L2 无单独的逐笔委托消息类型，但是可以从上交所 L2 实时行情获取（包含买卖一档的前 50 委托队列）			
4	Q:各个交易所数据接收频率及时间段			
	A:行情数据推送时间			
	消息类型	交易所推送频率	API 接收时间	行情品种类型
	上 交 所 L1 静态数据	每天 1 次	0830 左右可以进行订阅请求（ 其余时间调用快照查询可获取 ）	股票、基金、债券
	上 交 所 L1 实时行情	5 秒/1 次	0800~1500，指数推送至 1510 左右（0915 之前为初始化的盘前数据）	股票、基金、债券、指数
	深 交 所 L1 静态数据	每天 1 次	0800 左右可以进行订阅请求（ 其余时间调用快照查询可获取 ）	股票、基金、债券、指数
	深 交 所 L1 实时行情	3 秒/1 次	0800~1600	股票、基金、债券、指数
	上交所个股期权静态数据	每天 1 次	0830 左右进行订阅请求（ 其余时间调用快照查询可获取 ）	个股期权
	上交所个股期权实时行情	1 秒/2 次	0830~1500,1500-1600 为时间戳变动，其余字段不变的行情，到 1620 左右可以获取到最后一笔收盘数据	个股期权
	上 交 所 L2 静态数据	每天 1 次	0830 左右进行订阅请求（其余时间调用快照查询可获取）	股票、基金、债券、指数
	上 交 所 L2 实时行情	3 秒/1 次	0925~1500	股票、基金、债券
	上 交 所 L2 虚拟集合竞价	3 秒/1 次	0915~0925	股票、债券

User Mannual

上交所 L2 指数行情	3 秒/1 次	0820~0925~1500(0925 之前为初始 化数据 , 闭市后除两岸三地指数 00099 还会推送数据)	指数
上交所 L2 逐笔成交	3 秒/1 次	0925~1500	股票、基金、债券
上交所 L2 市场总览	5 秒/1 次	0830~1500	代码 000000(全市 场)
深交所 L2 静态数据	每天 1 次	0800 左右进行订阅请求 (其余时间调 用快照查询可获取)	股票、基金、债券、 指数
深交所 L2 证券状态	15 秒/1 次	0800~1600	股票、债券、基金
深交所 L2 实时行情	3 秒/1 次	0800~1600	股票、基金、债券
深交所 L2 指数快照	3 秒/1 次	0800~1600	指数
深交所 L2 逐笔委托	实时推送, 消息驱 动	0915~1500	股票、基金、债券
深交所 L2 逐笔成交	实时推送, 消息驱 动	0925~1500	股票、基金、债券
中金所 L2 静态数据	每天 1 次	0730 左右 (其余时间调用快照查询可 获取)	股指期货、国债期货
中金所 L2 实时行情	1 秒/2 次	0915~1600 (1530 已经可以获取到结 算价)	股指期货、国债期货
上交所指数 通	3 秒/1 次	0800-1620	指数
上期所 L1 静态数据	每个交易日 2 次	夜盘 : 1830 左右 ; 早盘 : 0730 左右	商品期货
上期所 L1 实时行情	1 秒/2 次	夜盘 : 2100-0230 ; 早盘 : 0900-1500 (具体品种以实际交易时间为准)	商品期货

User Mannual

	郑商所L1静态数据	每个交易日 2 次	夜盘：1930 左右；早盘：0700 左右	商品期货//期权
	郑商所L1实时行情	1 秒/2 次	夜盘：2100-2330；早盘：0900-1500 (具体品种以实际交易时间为准)	商品期货//期权
	易盛指数行情	1 秒/4 次	夜盘：2100-2330；早盘：0900-1500 (具体品种以实际交易时间为准)	商品期货指数
	大商所L1静态数据	每个交易日 2 次	夜盘：1930 左右；早盘：0700 左右	商品期货/期权
	大商所L1最优行情	1 秒/2 次	夜盘：2100-2330；早盘：0900-1500 (具体品种以实际交易时间为准)	商品期货//期权
	大商所L1套利行情	1 秒/2 次	夜盘：2100-2330；早盘：0900-1500 (具体品种以实际交易时间为准)	商品期货//期权
	大商所L2静态数据	每个交易日 2 次	夜盘：1930 左右；早盘：0700 左右	商品期货//期权
	大商所L2最优深度行情	1 秒/4 次	夜盘：2100-2330；早盘：0900-1500 (具体品种以实际交易时间为准)	商品期货//期权
	大商所L2套利深度行情	1 秒/4 次	夜盘：2100-2330；早盘：0900-1500 (具体品种以实际交易时间为准)	商品期货//期权
	大商所L2实时结算价	6 秒/次	夜盘：2100-2330；早盘：0900-1500 (具体品种以实际交易时间为准)	商品期货//期权
	大商所L2委托统计行情	1 秒/4 次	夜盘：2100-2330；早盘：0900-1500 (具体品种以实际交易时间为准)	商品期货//期权
	大商所L2分价成交量行情	1 秒/4 次	夜盘：2100-2330；早盘：0900-1500 (具体品种以实际交易时间为准)	商品期货//期权
5	A: 如何判断获取当天停牌个股?			

User Mannual

	<p>Q: (1) 上交所 L1: 通过实时行情获取, 在 0900 左右查询行情快照, 交易状态为 P 1 的个股即为当天停牌个股 (当天复牌的, 交易状态 0800 左右为 P010, 0840 刷新为 S011; 当天临时停牌的, 交易状态 0800 左右为 S 11, 0840 刷新为 P010; 连续停牌的, 交易状态 0800 为 P010); (2) 深交所 L1: 通过实时行情中的 SecurityPhaseTag (第 1 位数值为 '1') 判断停牌; (3) 上交所 L2: 通过当前品种交易状态 TradeStatus, 在 0840 查询快照仅可以获取到当天开始停牌的个股 (状态为 SUSP); 如果获取完整的当天停牌个股, 方法如下: 上海 L2 实时行情, 在 0925 之前发送的数据均为当天停牌或近几天 (非连续停牌), 0925-0930: 交易状态会先转变为 BETW 再变成 TRADE (对于当天参与交易的股票, 竞价产生开盘价, 时间戳在 0930 之前, 可以在时间戳为 0930 之前获取快照, 快照的代码集合为 A), 静态数据的代码集合为 B (含交易以及停牌的证券)。B-A 的代码集合即为停牌的个股 (含当天停牌和连续停牌的), 此方法仅对股票、基金有效。(债券由于部分品种不参与竞价撮合) (4) 深交所 L2: 盘前获取实时行情数据, 交易状态第二位为 1 即为全天停牌的个股 (停牌个股一分钟一次全量快照) + 交易状态为 H0 (临时停牌), 两者并集为当天停牌个股;</p>
6	<p>Q: QTS 时间字段的含义</p> <p>A: LocalTimeStamp 【GTA 机器时间】采集时间为 GTA 机房服务器时间 (上海南汇机房、深圳福永机房)、PacketTimeStamp 【交易所机器时间】包头时间为交易所发包时间、time 【交易所机器时间】为交易所业务时间 (具体含义参见字段说明)。</p>
7	<p>Q: 什么市面上行情终端显示的 399001 成交量与 QTS 接收的数据不一致</p> <p>Q: 深交所推送的指数除了常规指数外, 还有以 395 开头的成交量统计指标 (此类代码成交量、成交笔数、成交金额均按照全样本计算, 例如 395003 为中小板, 计算标的为深交所中小板约 800 只个股, 而中小板指数 399005 计算对象为 100 只成分股)。市面上一些行情软件会用 395003 去填充 399005, 作为参考。</p>
8	<p>Q: 沪深 L1 实时行情, 指数闭市之后还继续有数据推送</p> <p>A: 闭市之后, 由于指数需要在个股收盘后进行计算, 因此指数一般在 1510 左右才无行情推送, 除特殊指数 000999; 此指数为两岸三地指数, 标的为港澳台个股 (港交所交易结束时间为 1600), 此指数将一直推送至 1530。</p>
9	<p>Q: 与其他行情供应商, QTS 的数据量为什么比较多?</p> <p>A: 与其他行情商对比, GTA 采用双源双线备份的方式, API 输出的数据为两个源的数据经过过去重后所得到的, 数据会更完整。</p>
10	<p>Q: LocalTimeStamp、PacketTimeStamp、time 是否实现了时钟同步</p> <p>A: (1) 交易所不允许外部机器接入进行时钟同步, 因此三个时间字段是没有进行时钟同步的。(2) LocalTimeStamp 包含两个机房 (深圳福永、上海南汇) 服务器的时间, 两个源经过过滤后的 API</p>

User Mannual

	<p>出口数据，time 不一定是时间顺序的；PacketTimeStamp 为交易所发包时间，不能接入端口出口的数据 PacketTimeStamp 可能不一致，因此 PacketTimeStamp 也不一定时间顺序；time 为交易所业务时间，经过 QTS 过滤去重后，能够保证 time 是绝对顺序的。但是 SHL2 实时行情出现过 PacketTimeStamp<time,因此交易所也无对两个时间字段进行时钟同步。所以三个时间字段没有横向对比的意义。</p>
11	<p>Q:客户测试行情网络环境的要求</p> <p>A: 公网连接无限制、网络流量无限制、电脑/服务器有专用带宽：沪深 L2 需保证 7M-10M，沪深 L1 需保证 4M 以上，其余需保证 2M 以上。</p>
12	<p>Q:新一代深交所静态数据如何获取涨跌停价格</p> <p>A:(1)对于存在涨跌幅限制的现货集中竞价品种，以昨收价作为基准价格，乘以对应的涨跌幅度，计算的结果均四舍五入精确到价格档位。其中开盘集合竞价、连续竞价、收盘集合竞价关于涨跌幅设置是一致的，深交所未来可能根据业务需要，出现不一样的配置。(2)对于不存在涨跌幅限制的债券（现货交易、回购交易），有效范围涨跌幅度、有效范围涨跌价格才有业务意义。</p>
13	<p>Q:在 java 或者 C#接口中如何获取买卖档位上的数据。</p> <p>A:由于在 C++中采用了 union 结构来存储买卖档位的信息，而将 c++接口封装成 java 和 C#接口后，由于语言的差异，所以在获取 union 里面的信息也存在差异，以上海 L1 实时行情为例，在 c++中定义的结构体如下，</p> <pre> struct SSEL1_Quotation { int LocalTimeStamp; char QuotationFlag[QUOTATION_FLAG_LEN]; long long PacketTimeStamp; int Time; char Symbol[SYMBOL_LEN]; char SecurityName[SECURITY_NAME_LEN]; double OpenPrice; double HighPrice; double LowPrice; double LastPrice; unsigned long long TotalVolume; double TotalAmount; double PreClosePrice; double ClosePrice; union { BuySellLevelInfo SellLevel[LEVEL_FIVE]; ///< 五档卖行情 struct { double SellPrice01; ///< 申卖价 1 } } } </pre>

	<pre> unsigned long long SellVolume01; ///< 申卖量 1 double SellPrice02; ///< 申卖价 2 unsigned long long SellVolume02; ///< 申卖量 2 double SellPrice03; ///< 申卖价 3 unsigned long long SellVolume03; ///< 申卖量 3 double SellPrice04; ///< 申卖价 4 unsigned long long SellVolume04; ///< 申卖量 4 double SellPrice05; ///< 申卖价 5 unsigned long long SellVolume05; ///< 申卖量 5 };}; union { BuySellLevelInfo BuyLevel[LEVEL_FIVE]; ///< 五档买行情 struct { double BuyPrice01; ///< 申买价 1 unsigned long long BuyVolume01; ///< 申买量 1 double BuyPrice02; ///< 申买价 2 unsigned long long BuyVolume02; ///< 申买量 2 double BuyPrice03; ///< 申买价 3 unsigned long long BuyVolume03; ///< 申买量 3 double BuyPrice04; ///< 申买价 4 unsigned long long BuyVolume04; ///< 申买量 4 double BuyPrice05; ///< 申买价 5 unsigned long long BuyVolume05; ///< 申买量 5 };}; double NAV; double IOPV; char SecurityPhaseTag[PHRASE_TAG_LEN];}; </pre> <p>为获取买五档的数据，在使用 java 接口时需要通过 union 中的结构体数组获取，即类似于 SSEI1_Quotation. SellLevel[0].price 来获取卖一档行情。</p> <p>而使用 C#接口时需要通过 union 中的结构体来获取，即类似于采用 SSEI1_Quotation. SellPrice01 来获取卖一档行情。</p> <p>目前采用了 union 结构的数据包括：SSEL1_Quotation、SSEL2_Quotation、SSEIOL1_Quotation、SZSEL1_Quotation、SZSEL2_Static、SZSEL2_Quotation、CFFEXL2_Quotation。</p>
14	<p>Q:沪深 L2 怎么判断撤单记录</p> <p>A: (1) 深交所 L2 逐笔成交：当成交类型 TradeType=4 时，表示撤单记录，此时成交价格为 0，且买卖方索引至少一个为 0；(2) 上海 L2 逐笔成交：上交所不下发撤单成交记录</p>

User Mannual

15	Q:怎么通过证券代码命名规则区分不同的证券品种			
	A :			
	(1) 沪市代码区间			
	证 券 类 型	第 1 位	第 2-3 位	描 述
	国 债 / 指 数	0	00	上证指数、沪深 300、中证指数
			09	国债 (2 0 0 0 年前发行)
			10	国债 (2 0 0 0 年 - 2 0 0 9 年发行)
			18	国家开发银行在上交所增发金融债券, 例如 018003 为国开 1401
			19	固定收益电子平台交易国债
			20	记账式贴现国债
			90	新国债质押式回购质押券出入库 (对应 0 1 0 * * * 国债)
			99	新国债质押式回购质押券出入库 (对应 0 0 9 * * * 国债)
	债 券	1	00	可转债 (对应 6 0 0 * * *), 其中 1 0 0 9 * * 用于转债回售
			02	为公司债券质押券出入库分配 102000-102999 代码段
			03	103000 - 103999 代码段, 用于质押式回购交易中的公司债券质押券出入库。
			04	公司债及国家发改委等核准发行的、登记在证券账户的债券 (对应 1 2 2 * * *) 出入库
			05	1 0 5 0 0 0 - 1 0 5 8 9 9 用于分离债 (对应 1 2 6 * * *) 出入库, ? 1 0 5 9 0 0 - 1 0 5 9 9 9 用于企业债 (1 2 0 * * * 、 1 2 9 * * *) 出入库
			06	地方政府债出入库 (对应 1 3 0 * * *)
			07	记账式贴现国债出入库 (对应 0 2 0 * * *)
			08	对应 018***号码段, 例如 108003 为国开 1401 的质押券申报和转回代码

User Mannual

			10	可转债 (对应 6 0 0 * * *)
			12	可转债 (对应 6 0 0 * * *)
			13	可转债 (对应 6 0 1 * * *)
			20	企业债
			21	资产证券化
			22	1 2 2 0 0 0 - 1 2 2 4 9 9 用于公司债 , 1 2 2 5 0 0 - 1 2 2 9 9 用于国家发改委等核准发行的、登记在证券账户的债券
			23	123000-123499 用于不能同时满足以下条件的债券 : (一) 发行人的债项评级不低于 AA; (二) 债券上市前 , 发行人最近一期末的净资产不低于 15 亿元人民币 ; (三) 债券上市前 , 发行人最近三个会计年度实现的年均可分配利润不少于债券一年利息的 1.5 倍 ; (四) 本所规定的其他条件。
			24	124000 - 124999 代码段 , 公司债券
			25	中小企业私募债券
			26	分离交易的可转换公司债
			27	为公司债券交易分配 127000-127999 代码段
			28	可交换公司债 , 为信贷资产支持证券交易分配 128000-128999 代码段
			29	企业债
			30	地方政府债
			32	可交换公司债券
			33	可交换公司债券质押券出入库
			35	13500-135499 证券公司短期债券挂牌 135500-135999 并购重组私募债券挂牌转让
			81	可转债转股 (对应 6 0 0 * * *) , 已不再增用
			90	可转债转股 (对应 6 0 0 * * *)
			91	可转债转股 (对应 6 0 1 * * *)

User Mannual

			92	可交换公司债转股 (对应 1 2 8 * * *)
	回购	2	01	国债回购 (席位托管方式)
			02	企业债回购
			03	国债买断式回购
			04	债券质押式回购 (账户托管方式)
			05	债券质押式报价回购
			06	债券质押式协议回购
	优先股	3	30	优先股交易
			60	优先股转让
	期货		10	国债期货 (暂停交易)
	基金 / 权证	5	00	契约型封闭式基金(高频有记录)
			01	上市开放式基金
			10	交易型开放式指数证券投资基金 (不一定有高频记录) /ETF
			11	交易型货币市场基金、债券 ETF 等上市交易的固定收益类基金产品
			12	ETF
			13	ETF
			18	黄金 ETF(交易代码 : 518**0 , 申赎代码 : 518**1 , 申赎资金 : 518**5 , 认购代码 : 518**3 , 认购资金 : 518**4)
			19	开放式基金申赎 (无高频记录)
			21	开放式基金认购 (无高频记录)
			22	开放式基金跨市场转托管 (无高频记录)
			23	开放式基金分红 (无高频记录)
			24	开放式基金基金转换 (行情软件中无该类型代码)
			80	权证 (含股改权证、公司权证)(有高频记录)

User Mannual

			82	权证行权（无高频记录）
	A 股	6	00	A 股证券
			01	A 股证券
			03	A 股交易
	非交易业务(发行、权益分配)	7	00	配股（对应6 0 0 * * *）
			02	职工股配股（对应6 0 0 * * *）
			04	持股配债
			05	基金扩募
			06	要约收购
			30	申购、增发（对应6 0 0 * * *）
			31	持股增发（对应6 0 0 * * *）
			32	A 股申购
			33	可转债申购（对应6 0 0 * * *）
			34	A 股申购款
			35	基金申购
			36	A 股申购配号
			38	网上投票（对应6 0 0 * * *）
			40	申购款或增发款（对应6 0 0 * * *）
			41	申购或增发配号（对应6 0 0 * * *）
			42	配股
			43	可转债发债款（对应6 0 0 * * *）
			44	可转债配号（对应6 0 0 * * *）
			45	基金申购款

User Mannual

			46	基金申购配号
			51	7 5 1 0 0 0 - 7 5 1 1 9 9用于国债分销,751800-751899 公司债券网上分销业务,7 5 1 9 0 0 - 7 5 1 9 6 9用于地方政府债(对应1 3 0 * * *)分销,7 5 1 9 7 0 - 7 5 1 9 9 9用于公司债券及国家发改委等核准发行的、登记在证券账户的债券(对应1 2 2 * * *)分销
			52	网络投票
			53	持股配债
			54	可转债申购
			55	可转债发债款
			56	可转债配号
			60	配股(对应6 0 1 * * *)
			62	职工股配股(对应6 0 1 * * *)
			70	优先股申购
			71	优先股配股/配售
			72	优先股申购款分配
			73	优先股申购配号分配
			80	申购、增发(对应6 0 1 * * *)
			81	持股增发(对应6 0 1 * * *)
			83	可转债申购(对应6 0 1 * * *)
			88	网络投票(对应6 0 1 * * *)
			90	申购款或增发款(对应6 0 1 * * *)
			91	申购或增发配号(对应6 0 1 * * *)
			93	可转债申购款(对应6 0 1 * * *)
			94	可转债配号(对应6 0 1 * * *)

User Mannual

		99	指定交易（含指定交易、撤销指定、回购指定撤销、A股密码服务等）
		99	799981-799984 证券代码段为融资融券业务服务：其中 799981 为余券划转、799982 为还券划转、799983 为担保物划转、799984 位券源划转。
B 股	9	00	B 股证券
		38	网上投票（B 股）
		39	B 股网络投票密码服务（现仅用 9 3 9 9 8 8）
（2）深市代码区间：			
第 1 位	第 2 位	第 3 位	定义
0	0	0-1	主板 A 股
		2-4	中小企业板股票
	3	0-2	主板 A 股及中小企业板股票认购权证
		6	创业板股权激励计划涉及的员工认股权
		7	【037000，037499】主板 A 股权激励计划涉及的员工认股权 【037500，037999】中小企业板股权激励计划涉及的员工认股权
		8-9	主板 A 股及中小企业板票认沽权证
	7	0-1	主板 A 股增发
			主板可转换公司债券申购
		2-4	中小企业板股票增发
			中小企业板可转换公司债券申购
	8	0-1	主板 A 股配股优先权
			主板可转换公司债券的优先权认购
		2-4	中小企业板配股优先权
			中小企业板可转换公司债券的优先权认购

User Mannual

	1	0	0-7	付息式国债 其中【101600，101799】是债券分销代码区间。 【101650,101699】是债券分销代码区间，分销代码位于【101660,101674】区间时实行投资者适当性管理
			8	贴现式国债
			9	地方政府债
		1	1	企业债
			2	公司债
			3	可分离交易的可转换公司债券
			7	"【117000,117499】中小企业可交换私募债 【117500,117999】为证券公司短期债"
			8	【118900，118999】为证券公司次级债 【118000，118899】为其他中小企业私募债
			9	【119000，119499】为资产支持证券
		2	0-9	可转换公司债券（已发行的可转换公司债券继续保留原代码） 【123000，123999】是创业板可转换公司债券代码区间 【127000，127999】是主板可转换公司债券代码区间 【128000，128999】是中小企业板可转换公司债券代码
		3	1	【131800，131899】为债券回购交易代码； 131990 为债券回购的标准券代码
		4	0	优先股
		5	0-1	分级基金子基金
			9	ETF(其中,XXZQJB 字段值为 'E' 的是交易型开放式指数基金(ETF); XXZQJB 字段值为 'M' 的是 实时申购赎回的货币市场基金(以下简称货币基金。货币基金不交易,按照 ETF 申购赎回模式进行申购赎回), ETF 和货币基金的共同代码区间为【159001,159999】。

User Mannual

16				XXZQJB 字段值 为 ‘ L ’ 时，ETF 或 货币基金处于发行阶段。
		6	0-9	开放式基金(其中 ,XXZQJB 字段值为 ‘L’ 的 ,是上市开放式基金(LOF) ; XXZQJB 字段值为 ‘F’ 的是非交易型开放式基金 (暂不交易，仅揭示基金净值及开放申购赎回业务); XXZQJB 字段值为 ‘O’ 的是净值揭示服务开放式基金。
		8	4	封闭式证券投资基金
	2	0	0-9	B 股
		3	8	B 股现金选择权
		8	0-9	B 股配股优先权
	3	0	0-9	创业板股票
			0-1	主板股东大会网络投票
			2-4	中小企业板股东大会网络投票
			6	5-8
		9		369991 用于中国结算网络服务身份认证业务的密码激活/密码重置 369999 用于深交所身份认证业务的密码激活/密码挂失
		7	0-9	创业板股票增发 创业板可转换公司债券申购
		8	0-9	创业板配股优先权 创业板可转换公司债券的优先权认购
		9	5	成交量统计指标
			9	指数
		Q: 上交所指数通与上交所 L2 指数行情、上交所 L1 实时行情里面的指数代码有什么区别		
A: 上交所指数通为上交所转发的中证指数行情，里面的代码除了包括上交所部分指数、深交所部分指数，还包括以 H/CE/9 开头的特色指数，指数品种类型比上交所 L2 指数行情、上交所 L1 实				

	时行情里面的指数要丰富。
17	Q:64 位工程编译 C#示例代码中的动态库文件失败
	A:QTS C#接口的动态库默认为 32 位，因此如果使用 64 位工程编译，需要在工程删除 32 位动态库文件，重新加载 64 位动态库文件，才可以编译成功。

8. 实例

8.1 例一（C++示例代码）

路径：QTSApiPub\Example\Example_1

```
////////////////////////////////////
////
/// @brief    获取代码列表和消息列表
///
////////////////////////////////////
////
#include <string>
#include "QTSDDataType.h"
#include "QTSStruct.h"
#include "GTAQTSInterfaceBase.h"
#include <stdio.h>

int main()
{
    /**
        获取市场代码列表、获取用户权限列表示例代码
    */
    printf("Run Example_1: \n");

    //订阅消息回调类，具体使用示例方法请参见 Example_2 工程
    IGTAQTSCallbackBase m_CallbackBase;

    //创建基础 API 对象
    //详见《GTA 实时行情系统 V2.X 用户手册》4.2.1.1 创建实例 CreateInstance 章节
}
```

User Mannual

```
IGTAQTSApiBase* pApiBase =
IGTAQTSApiBase::CreateInstance(m_CallbackBase);

//注册 FENS 地址
//***** 警告：实际生产环境使用时，从 GTA 公司获取到的 FENS 地址，此处需要全部通
过“RegisterService”函数接口注册，
//***** 否则，在数据高可用方面，会大打折扣。
//***** 如有 4 个 FENS ip 地址，需要如下调用：
//      pApiBase->RegisterService("ip1", port1);
//      pApiBase->RegisterService("ip2", port2);
//      pApiBase->RegisterService("ip3", port3);
//      pApiBase->RegisterService("ip4", port4);
//详见《GTA 实时行情系统 V2.X 用户手册》4.2.1.4 注册服务 RegisterService 章
节
pApiBase->RegisterService("192.168.105.136", 7777);
pApiBase->RegisterService("192.168.195.82", 8899);
pApiBase->RegisterService("192.168.195.83", 8899);
pApiBase->RegisterService("192.168.195.84", 8899);

do
{
    //通过用户名与密码向服务器登陆
    //详见《GTA 实时行情系统 V2.X 用户手册》4.2.1.5 用户认证 Login 章节
    RetCode ret = pApiBase->LoginX("gta1", "123456", "NetType=0");
    if ( Ret_Success != ret )
    {
        printf("Login error:%d\n", ret);
        break;
    }

    CDataBuffer<StockSymbol> StockList1;
    CDataBuffer<StockSymbol> StockList2;

    // 获取上交所和深交所代码列表，其中 SSE 表示上交所，SZSE 表示深交所，CFFEX
    表示中金所
    //详见《GTA 实时行情系统 V2.X 用户手册》4.2.1.11 获取代码列表
    GetStockList 章节
    ret = pApiBase->GetStockList("sse,szse", StockList1);
    if ( Ret_Success != ret )
    {
        printf("GetStockList(sse,szse) error:%d\n", ret);
        break;
    }
}
```

User Mannual

```
// 获取上交所代码列表
ret = pApiBase->GetStockList("sse", StockList2);
if ( Ret_Success != ret )
{
    printf("GetStockList(sse) error:%d\n", ret);
    break;
}

int Count = StockList2.Size();
StockSymbol* pStock = StockList2;
int WriCount = Count > 10 ? Count : 10;
printf("SSE Stock Count = %d List:", Count);
for( int i = 0; i < WriCount; ++i)
{
    printf("%s, ", pStock[i].Symbol);
}
printf("\n");

CDataBuffer<MsgType> DataTypeList;
// 获取权限列表
//详见《GTA 实时行情系统 V2.X 用户手册》4.1.1.7 获取权限列表
GetDataTypeList 章节
ret = pApiBase->GetDataTypeList(DataTypeList);
if ( Ret_Success != ret )
{
    printf("GetDataTypeList(sse) error:%d\n", ret);
    break;
}

MsgType* pMsg = DataTypeList;
Count = DataTypeList.Size();
printf("MsgType Count = %d, List:", Count);
for( int i = 0; i < Count; ++i)
{
    printf("0x%08x, ",pMsg[i]);
}
printf("\n");

} while (false);

getchar();

// 当不再使用 API 时, 需要调用此接口释放内部资源, 否则会引起内存泄漏及不可预知问
```

User Mannual

题

//详见《GTA 实时行情系统 v2.x 用户手册》4.2.1.2 释放实例 ReleaseInstance 章节

```
IGTAQTSApiBase::ReleaseInstance(pApiBase);  
return 0;  
}
```

8.2 例二 (C++示例代码)

路径：QTSApiPub\Example\Example_2

Example_2.cpp

```
/////////////////////////////////////  
/////  
/// @brief    基础 API  
///          实现不同方式的订阅和取消订阅及快照查询  
/////////////////////////////////////  
/////  
#include "GTAQTSInterfaceBase.h"  
#include "CallBackBase.h"  
#include <stdio.h>  
#include "../Common/IOService.h"  
#include "../Common/TinyThread.h"  
  
#ifdef _WIN32  
#include <windows.h>  
#define SLEEP(t) Sleep((t)*1000)  
#else  
#include <unistd.h>  
#define SLEEP(t) sleep(t)  
#endif  
  
int main()  
{  
    IOService ios;  
  
    //订阅消息回调类
```

User Mannual

```
CallBackBase CallbackBase;
CallBackBase.SetIoService(&ios);

//启动处理数据服务
ios.Start();

//创建基础 API 对象
//详见《GTA 实时行情系统 V2.X 用户手册》4.2.1.1 创建实例 CreateInstance 章节
IGTAQTSApiBase* pApiBase =
IGTAQTSApiBase::CreateInstance(CallbackBase);
// 注册 FENS 地址
//***** 警告：实际生产环境使用时，从 GTA 公司获取到的 FENS 地址，此处需要全部通过“RegisterService”函数接口注册，
//***** 否则，在数据高可用方面，会大打折扣。
//***** 如有 4 个 FENS ip 地址，需要如下调用：
//      pApiBase->RegisterService("ip1", port1);
//      pApiBase->RegisterService("ip2", port2);
//      pApiBase->RegisterService("ip3", port3);
//      pApiBase->RegisterService("ip4", port4);
//详见《GTA 实时行情系统 V2.X 用户手册》4.2.1.4 注册服务 RegisterService 章节
pApiBase->RegisterService("10.1.134.43", 8888);
pApiBase->RegisterService("192.168.195.82", 8899);
pApiBase->RegisterService("192.168.195.83", 8899);
pApiBase->RegisterService("192.168.195.84", 8899);

do
{
    //通过用户名与密码向服务器登陆
    //详见《GTA 实时行情系统 V2.X 用户手册》4.2.1.5 用户认证 Login 章节
    RetCode ret = pApiBase->LoginX("gta1", "123456", "NetType=0");
    if ( Ret_Success != ret )
    {
        printf("Login error:%d\n", ret);
        break;
    }

    //***** 获取证券代码列表及权限列表
    *****

    CDataBuffer<StockSymbol> StockList1;
    CDataBuffer<StockSymbol> StockList2;
```

User Mannual

// 获取上交所和深交所代码列表，其中 SSE 表示上交所，SZSE 表示深交所，CFFEX 表示中金所

//详见《GTA 实时行情系统 V2.X 用户手册》4.2.1.11 获取代码列表

GetStockList 章节

```
ret = pApiBase->GetStockList("sse,szse", StockList1);
if ( Ret_Success != ret )
{
    printf("GetStockList(sse,szse) error:%d\n", ret);
    break;
}
```

// 获取深交所代码列表

```
ret = pApiBase->GetStockList("szse", StockList2);
if ( Ret_Success != ret )
{
    printf("GetStockList(szse) error:%d\n", ret);
    break;
}
```

```
int Count = StockList2.Size();
StockSymbol* pStock = StockList2;
int WriCount = Count > 10 ? Count : 10;
printf("SSE Stock Count = %d List:", Count);
for( int i = 0; i < WriCount; ++i)
{
    printf("%s, ", pStock[i].Symbol);
}
printf("\n");
```

```
CDataBuffer<MsgType> DataTypeList;
```

// 获取权限列表

//详见《GTA 实时行情系统 V2.X 用户手册》4.1.1.7 获取权限列表

GetDataTypeList 章节

```
ret = pApiBase->GetDataTypeList(DataTypeList);
if ( Ret_Success != ret )
{
    printf("GetDataTypeList(sse) error:%d\n", ret);
    break;
}
```

```
MsgType* pMsg = DataTypeList;
Count = DataTypeList.Size();
printf("MsgType Count = %d, List:", Count);
```

User Mannual

```
for( int i = 0; i < Count; ++i)
{
    printf("0x%08x, ",pMsg[i]);
}
printf("\n");

//*****订阅行情数据
*****

// 按代码订阅深交所实时行情数据
//详见《GTA 实时行情系统 v2.x 用户手册》4.1.1.8 订阅实时数据 Subscribe 章节
节

ret = pApiBase->Subscribe(Msg_SZSEL2_Quotation,
"000001,000651,000039");
if ( Ret_Success != ret )
{
    printf("Subscribe by code error:%d\n", ret);
    break;
}

printf("press any key to continue...(Unsubscribe all data)");
getchar();

// 按代码取消深交所实时行情数据
//详见《GTA 实时行情系统 v2.x 用户手册》4.1.1.9 取消订阅 Unsubscribe 章节
节

ret = pApiBase->Unsubscribe(Msg_SZSEL2_Quotation,
"000001,000651,000039");
if ( Ret_Success != ret )
{
    printf("Unsubscribe by code error:%d\n", ret);
}
// 取消上交所实时所有代码
ret = pApiBase->Unsubscribe(Msg_SZSEL2_Quotation, NULL);
if ( Ret_Success != ret )
{
    printf("Unsubscribe by market error:%d\n", ret);
}

// 深交所实时行情快照查询，快照数据通过 Snap_Quotation 结构返回
//详见《GTA 实时行情系统 v2.x 用户手册》4.1.1.10 查询快照数据 QuerySnap 章节
节
```


User Mannual

```
CDataBuffer<SZSEL2_Quotation> Snap_Quotation;
ret =
pApiBase->QuerySnap_SZSEL2_Quotation(NULL, Snap_Quotation);
if ( Ret_Success != ret )
{
    printf("Login error:%d\n", ret);
    break;
}

SZSEL2_Quotation* pSse = Snap_Quotation;
if( pSse != NULL)
{
    printf("QuerySnap_Base:Count = %d, LocalTimeStamp = %d, Symbol
= %s, OpenPrice = %f, TotalAmount = %f\n",
        Snap_Quotation.Size(), pSse->LocalTimeStamp,
pSse->Symbol, pSse->OpenPrice, pSse->TotalAmount);
}

// 获取全部快照
for (int i = 0; i < Snap_Quotation.Size(); ++i)
{
    SZSEL2_Quotation& SSEQuot = Snap_Quotation[i];
    printf("LocalTimeStamp = %d, Symbol = %s, OpenPrice = %f,
TotalAmount = %f\n",
        SSEQuot.LocalTimeStamp, SSEQuot.Symbol,
SSEQuot.OpenPrice, SSEQuot.TotalAmount);
}

} while (false);

//退出工作线程
ios.Stop();

getchar();

// 当不再使用 API 时，需要调用此接口释放内部资源，否则会引起内存泄漏及不可预知问题
//详见《GTA 实时行情系统 V2.X 用户手册》4.2.1.2 释放实例 ReleaseInstance 章节
IGTAQTSApiBase::ReleaseInstance(pApiBase);
return 0;
}
```

CallBackBase.cpp

```
#include "CallBackBase.h"
#include <stdio.h>
#include "../Common/public.h"

CallBackBase::CallBackBase(void)
{
    m_ios = 0;
}

CallBackBase::~CallBackBase(void)
{
}

void CallBackBase::SetIoService(IoService *ios)
{
    m_ios = ios;
}

/// 连接状态返回，连接成功/失败
/// @param msgType    -- 消息类型
/// @param errCode    -- 失败原因，成功时返回 0
/// 详见《GTA 实时行情系统 V2.X 用户手册》5.5 返回码含义列表
RetCode 章节
void CallBackBase::OnConnectionState(MsgType msgType, RetCode errCode)
{
    printf("MsgType:0x%x ConnectionState:%d\n", msgType, errCode);
}

void CallBackBase::OnSubscribe_SSEL2_Quotation(const SSEL2_Quotation&
RealSSEL2Quotation)
{
    /**
    警告：请勿在回调函数接口内执行耗时操作，如：复杂运算，写文件等；否则会堵塞数据的
    接收。
    建议处理方式：把接收到的数据放至队列，再由工作线程处理接收到的数据内容；
    可参照本回调函数的处理方法
    */

    if ( 0 == m_ios )
```

User Mannual

```
{
    printf("IoService object is null");
    return ;
}

UTIL_Time stTime;
PUTIL_GetLocalTime(&stTime);

//接收到数据后，先放入本地队列，等待数据处理接口处理
SubData *subdata = new SubData;
subdata->msgType = Msg_SSEL2_Quotation;
subdata->cur_time = PUTIL_SystemTimeToDateTime(&stTime);
subdata->data.assign((const char*)&RealSSEL2Quotation,
sizeof(RealSSEL2Quotation));

    TaskPtr task(new Task(std::bind(&CallBackBase::Deal_Message, this,
subdata)));
    m_ios->Post(task);
}

/// 深交所 L1 实时行情订阅数据实时回调接口
/// @param RealSZSEL1Quotation -- 实时数据
void CallBackBase::OnSubscribe_SZSEL1_Quotation(const
SZSEL1_Quotation& RealSZSEL1Quotation)
{
    /**
    警告：请勿在回调函数接口内执行耗时操作，如：复杂运算，写文件等；否则会堵塞数据的
    接收。
    建议处理方式：把接收到的数据放至队列，再由工作线程处理接收到的数据内容；
    可参照本回调函数的处理方法
    */

    if ( 0 == m_ios )
    {
        printf("IoService object is null");
        return ;
    }

    UTIL_Time stTime;
    PUTIL_GetLocalTime(&stTime);

    //接收到数据后，先放入本地队列，等待数据处理接口处理
    SubData *subdata = new SubData;
    subdata->msgType = Msg_SZSEL1_Quotation;
```

User Mannual

```
subdata->cur_time = PUTIL_SystemTimeToDateTime(&stTime);
subdata->data.assign((const char*)&RealSZSEl1Quotation,
sizeof(RealSZSEl1Quotation));

TaskPtr task(new Task(std::bind(&CallBackBase::Deal_Message, this,
subdata)));
m_ios->Post(task);
}

/// 上交所 L2 逐笔成交订阅数据实时回调接口
/// @param RealSSEL2Transaction    -- 实时数据
void CallBackBase::OnSubscribe_SSEL2_Transaction(const
SSEL2_Transaction& RealSSEL2Transaction)
{
    /**
    警告：请勿在回调函数接口内执行耗时操作，如：复杂运算，写文件等；否则会堵塞数据的
    接收。
    建议处理方式：把接收到的数据放至队列，再由工作线程处理接收到的数据内容；
    可参照本回调函数的处理方法
    */

    if ( 0 == m_ios )
    {
        printf("IoService object is null");
        return ;
    }

    UTIL_Time stTime;
    PUTIL_GetLocalTime(&stTime);

    //接收到数据后，先放入本地队列，等待数据处理接口处理
    SubData *subdata = new SubData;
    subdata->msgType = Msg_SSEL2_Transaction;
    subdata->cur_time = PUTIL_SystemTimeToDateTime(&stTime);
    subdata->data.assign((const char*)&RealSSEL2Transaction,
sizeof(RealSSEL2Transaction));

    TaskPtr task(new Task(std::bind(&CallBackBase::Deal_Message, this,
subdata)));
    m_ios->Post(task);
}

/// 深交所 L2 实时行情订阅数据实时回调接口
/// @param RealSZSEl2Quotation -- 实时数据
```

User Mannual

```
void CallbackBase::OnSubscribe_SZSEL2_Quotation(const
SZSEL2_Quotation& RealsSZSEL2Quotation)
{
    /**
    警告：请勿在回调函数接口内执行耗时操作，如：复杂运算，写文件等；否则会堵塞数据的
    接收。
    建议处理方式：把接收到的数据放至队列，再由工作线程处理接收到的数据内容；
    可参照本回调函数的处理方法
    */

    if ( 0 == m_ios )
    {
        printf("IoService object is null");
        return ;
    }

    UTIL_Time stTime;
    PUTIL_GetLocalTime(&stTime);

    //接收到数据后，先放入本地队列，等待数据处理接口处理
    SubData *subdata = new SubData;
    subdata->msgType = Msg_SZSEL2_Quotation;
    subdata->cur_time = PUTIL_SystemTimeToDateTime(&stTime);
    subdata->data.assign((const char*)&RealsSZSEL2Quotation,
sizeof(RealsSZSEL2Quotation));

    TaskPtr task(new Task(std::bind(&CallbackBase::Deal_Message, this,
subdata)));
    m_ios->Post(task);
}

/// 处理上交所 L2 实时行情快照订阅数据
void CallbackBase::Deal_Message(SubData *subdata)
{
    static int count = 0;// 防止输出太多内容

    //智能指针，会自动释放指针指向的对象，释放内存空间
    std::unique_ptr<SubData> subdataptr(subdata);

    switch ( subdataptr->msgType )
    {
    case Msg_SSEL2_Quotation:
        {
            SSEL2_Quotation *RealSSEL2Quotation =
```

User Mannual

```
(SSEL2_Quotation*)&subdata->data[0];

    if( count < 10 )// 防止输出太多内容
    {
        count++;
        printf("Subscribe Data: PacketTimeStamp = %lld, Symbol
= %s, OpenPrice = %f, HighPrice = %f, LowPrice = %f, LastPrice = %f,
ClosePrice = %f\n",
            RealSSEL2Quotation->PacketTimeStamp,
RealSSEL2Quotation->Symbol, RealSSEL2Quotation->OpenPrice,
RealSSEL2Quotation->HighPrice,
            RealSSEL2Quotation->LowPrice,
RealSSEL2Quotation->LastPrice, RealSSEL2Quotation->ClosePrice);
    }
}
break;
case Msg_SSEL2_Transaction:
{
    SSEL2_Transaction *RealSSEL2Transaction =
(SSEL2_Transaction*)&subdata->data[0];

    if( count < 10 )// 防止输出太多内容
    {
        count++;
        printf("Subscribe Data: PacketTimeStamp = %lld, Symbol
= %s, TradeTime = %d, TradePrice = %f, TradeVolume = %u, TradeAmount
= %f\n",
            RealSSEL2Transaction->PacketTimeStamp,
RealSSEL2Transaction->Symbol, RealSSEL2Transaction->TradeTime,
RealSSEL2Transaction->TradePrice,
            RealSSEL2Transaction->TradeVolume,
RealSSEL2Transaction->TradeAmount);
    }
}
break;
case Msg_SZSEL1_Quotation:
{
    SZSEL1_Quotation *RealSZSEL1Quotation =
(SZSEL1_Quotation*)&subdata->data[0];

    if( count < 10 )// 防止输出太多内容
    {
        count++;
        printf("Subscribe Data: Time = %lld, Symbol = %s, OpenPrice
```

User Mannual

```
= %f, HighPrice = %f, LowPrice = %f, LastPrice = %f\n",
        RealsSZSEL1Quotation->Time,
RealsSZSEL1Quotation->Symbol, RealsSZSEL1Quotation->OpenPrice,
RealsSZSEL1Quotation->HighPrice,
        RealsSZSEL1Quotation->LowPrice,
RealsSZSEL1Quotation->LastPrice);
    }
}
break;
case Msg_SZSEL2_Quotation:
{
    SZSEL2_Quotation *RealSSEL2Quotation =
(SZSEL2_Quotation*)&subdata->data[0];

    if( count < 10 )// 防止输出太多内容
    {
        count++;
        printf("Subscribe Data: LocalTimeStamp = %d, Symbol = %s,
OpenPrice = %f, HighPrice = %f, LowPrice = %f, LastPrice = %f\n",
            RealsSSEL2Quotation->LocalTimeStamp,
RealsSSEL2Quotation->Symbol, RealsSSEL2Quotation->OpenPrice,
RealsSSEL2Quotation->HighPrice,
            RealsSSEL2Quotation->LowPrice,
RealsSSEL2Quotation->LastPrice);
    }
}
break;
default:
    printf("unknown message type:0x%x\n", subdataptr->msgType);
    break;
}
}
```

8.3Java 示例代码

本节通过一个完整的例子展示了 QTS 实时行情系统 Java 版本 API 的使用过程。

Example.java 中主要展示了从创建 API 实例，到注册、登陆、订阅数据、释放实例等一系列完整的操作。

GTACallbackBase.java 中定义了订阅数据实时回调接口

User Mannual

AFileOutBase.java 中包含了一些以文件形式存储数据所需要的函数。

SZSEL2_Quotation_FileOut.java 实现了以文件形式存储深圳 L2 实时行情数据。

SZSEL2_Transaction_FileOut.java 实现了以文件形式存储深圳 L2 逐笔成交数据。

Example.java

```
package test;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

import com.gta.qts.c2j.adaptee.impl.GTAQTSApiBaseImpl;
import com.gta.qts.c2j.adaptee.structure.QTSDDataType.MsgType;
import com.gta.qts.c2j.adaptee.structure.QTSDDataType.RetCode;
import com.gta.qts.c2j.adaptee.structure.QTSDDataType;
import com.gta.qts.c2j.adaptee.structure.SSEL2_Quotation;
import com.gta.qts.c2j.adaptee.structure.SZSEL2_Quotation;
import com.gta.qts.c2j.adaptee.structure.StockSymbol;
import com.gta.qts.c2j.adaptee.IGTAQTSApi;
import com.gta.qts.c2j.adaptee.IGTAQTSCallbackBase;

public class Example {

    public static void main(String[] args) throws IOException {
        // 创建消息回调对象，用于接收实时数据
        IGTAQTSCallbackBase callback = new GTACallbackBase();

        //创建 API 对象，与服务端交互使用
        IGTAQTSApi baseService =
            GTAQTSApiBaseImpl.getInstance().CreateInstance(callback);

        //基础 API 环境初始化，在开始使用 API 操作函数前，只调用一次
        baseService.BaseInit();
    }
}
```

User Mannual

```
//注册 FENS 地址
//***** 警告：实际生产环境使用时，从 GTA 公司获取到的 FENS 地址，此处需要全部通过“RegisterService”函数接口注册，
//***** 否则，在数据高可用方面，会大打折扣。
//***** 如有 4 个 FENS ip 地址，需要如下调用：
//      baseService.BaseRegisterService("192.168.105.136",
(short)7777);
//      baseService.BaseRegisterService("192.168.105.137",
(short)7777);
//      baseService.BaseRegisterService("192.168.105.138",
(short)7777);
//      baseService.BaseRegisterService("192.168.105.139",
(short)7777);
baseService.BaseRegisterService("192.168.203.35", (short)8888);
baseService.BaseRegisterService("192.168.105.137",
(short)7777);
baseService.BaseRegisterService("192.168.105.138",
(short)7777);
baseService.BaseRegisterService("192.168.105.139",
(short)7777);

do{
    //通过用户名与密码向服务器登陆
    int ret = baseService.BaseLoginX("test1",
"123456", "NetType=0");
    if ( QTSDDataType.RetCode.Ret_Success !=
QTSDDataType.RetCode.fetchByCode(ret) ){
        System.out.println("Login error:" + ret);
        break;
    }
    System.out.println("Login success");

    List<StockSymbol> outList = new ArrayList<StockSymbol>();

    // 获取上交所和深交所代码列表，其中 SSE 表示上交所，SZSE 表示深交所 sse,
    ret = baseService.BaseGetStockList("szse,sse", outList);
    if ( QTSDDataType.RetCode.Ret_Success !=
QTSDDataType.RetCode.fetchByCode(ret) ){
        System.out.println("GetStockList(sse,szse) error:" +
ret);
        break;
    }
    System.out.println("GetStockList success");
```

```
//输出获取到的证券代码
System.out.print("StockList:" + outList.size() + " ");
for (int idx = 0; idx < outList.size(); idx++) {
    if (idx > 0) {
        System.out.print(",");
    }
    byte[] bytesymbol = outList.get(idx).Symbol;
    String Symbol = new
String(bytesymbol,0,bytesymbol.length,"UTF-8").trim();
    System.out.print(Symbol);
}
System.out.println("");

List<Integer> msgtypeList = new ArrayList<Integer>();

//获取用户权限列表
ret = baseService.BaseGetMsgTypeList(msgtypeList);
if ( RetCode.Ret_Success != RetCode.fetchByCode(ret) ){
    System.out.println("GetMsgTypeList error:" + ret);
    break;
}
System.out.println("GetMsgTypeList success");

//输出获取到的用户权限列表
System.out.print("MsgType:" + msgtypeList.size() + " ");
for (int idx = 0; idx < msgtypeList.size(); idx++) {
    MsgType msgType =
MsgType.fetchByCode(msgtypeList.get(idx));
    if (msgType != null) {
        System.out.print(((idx > 0) ? ",Msg_" : "Msg_") +
msgType.name());
    }
}
System.out.println("");

// 按代码订阅深交所实时行情数据
ret =
baseService.BaseSubscribe(MsgType.SZSEL2_Quotation.code,
"000001,000002,000003");
if ( RetCode.Ret_Success != RetCode.fetchByCode(ret) ){
    System.out.println("Subscribe error:" + ret);
    break;
}
```

User Mannual

```
System.out.println("Subscribe success");

//订阅成功, 启动处理数据线程
SZSEL2_Quotation_FileOut.startThread();

//等待客户输入后再继续执行
System.out.println("press Enter to continue...(Unsubscribe
all data)");
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
br.readLine();

// 按代码取消深交所实时行情数据
ret =
baseService.BaseUnsubscribe(MsgType.SZSEL2_Quotation.code, "000001");
if ( RetCode.Ret_Success != RetCode.fetchByCode(ret) ){
    System.out.println("Unsubscribe error:" + ret);
    break;
}
System.out.println("Unsubscribe 000001 success");

// 取消深交所实时所有代码
ret =
baseService.BaseUnsubscribe(MsgType.SZSEL2_Quotation.code, null);
if ( RetCode.Ret_Success != RetCode.fetchByCode(ret) ){
    System.out.println("Unsubscribe error:" + ret);
    break;
}
System.out.println("Unsubscribe all success");

List<SZSEL2_Quotation> snapList = new
ArrayList<SZSEL2_Quotation>();

//深交所实时行情快照查询
ret =
baseService.QuerySnap_SZSEL2_Quotation("000001,000002,000003",
snapList);
if ( RetCode.Ret_Success != RetCode.fetchByCode(ret) ){
    System.out.println("QuerySnap_SZSEL2_Quotation error:" +
ret);
    break;
}
System.out.println("QuerySnap_SZSEL2_Quotation success");
```

User Mannual

```
//输出查询快照结果
System.out.println("QuerySnap_SZSEL2_Quotation : count=" +
snapList.size());
    for (int idx = 0; idx < snapList.size(); idx++) {
        byte[] symbol = snapList.get(idx).Symbol;
        String strSymbol = new
String(symbol,0,symbol.length,"UTF-8").trim();
        System.out.println("LocalTimeStamp=" +
snapList.get(idx).LocalTimeStamp
            + " Symbol=" + strSymbol
            + " OpenPrice=" + snapList.get(idx).OpenPrice
            + " TotalAmount=" + snapList.get(idx).TotalAmount
            );
    }

}while(false);

System.out.print("press Enter to end...");
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
br.readLine().trim();

//基础 API 环境反初始化
baseService.BaseUninit();

System.out.println("GTA API exit");
}

}
```

GTACallbackBase.java

```
package test;

import com.gta.qts.c2j.adaptee.IGTAQTSCallbackBase;
import com.gta.qts.c2j.adaptee.structure.SSEIOL1_Quotation;
import com.gta.qts.c2j.adaptee.structure.SSEIOL1_Static;
import com.gta.qts.c2j.adaptee.structure.SSEL2_Auction;
import com.gta.qts.c2j.adaptee.structure.SSEL2_Index;
import com.gta.qts.c2j.adaptee.structure.SSEL2_Overview;
```

User Mannual

```
import com.gta.qts.c2j.adaptee.structure.SSEL2_Quotation;
import com.gta.qts.c2j.adaptee.structure.SSEL2_Static;
import com.gta.qts.c2j.adaptee.structure.SSEL2_Transaction;
import com.gta.qts.c2j.adaptee.structure.SZSEL2_Index;
import com.gta.qts.c2j.adaptee.structure.SZSEL2_Order;
import com.gta.qts.c2j.adaptee.structure.SZSEL2_Quotation;
import com.gta.qts.c2j.adaptee.structure.SZSEL2_Static;
import com.gta.qts.c2j.adaptee.structure.SZSEL2_Status;
import com.gta.qts.c2j.adaptee.structure.SZSEL2_Transaction;

/**
 * 订阅消息数据回调接口
 * */
public class GTACallbackBase implements IGTAQTSCallbackBase {

    @Override
    public void OnConnectionState(int msgType, int errCode) {
        System.out.println("OnConnectionState msgType:" + msgType + "
errCode:" + errCode);
    }

    @Override
    public void OnLoginState(int errCode) {

    }

    @Override
    public void OnSubscribe_SSEIOL1_Quotation(SSEIOL1_Quotation data) {

    }

    @Override
    public void OnSubscribe_SSEIOL1_Static(SSEIOL1_Static data) {

    }

    @Override
    public void OnSubscribe_SSEL2_Auction(SSEL2_Auction data) {

    }

    @Override
    public void OnSubscribe_SSEL2_Index(SSEL2_Index data) {
```

User Mannual

```
}

@Override
public void OnSubscribe_SSEL2_Overview(SSEL2_Overview data) {

}

@Override
public void OnSubscribe_SSEL2_Quotation(SSEL2_Quotation data) {
}

@Override
public void OnSubscribe_SSEL2_Static(SSEL2_Static data) {

}

@Override
public void OnSubscribe_SSEL2_Transaction(SSEL2_Transaction data) {

}

@Override
public void OnSubscribe_SZSEL2_Index(SZSEL2_Index data) {

}

@Override
public void OnSubscribe_SZSEL2_Order(SZSEL2_Order data) {

}

@Override
public void OnSubscribe_SZSEL2_Quotation(SZSEL2_Quotation data) {
    //System.out.println("OnSubscribe_SZSEL2_Quotation");
    SZSEL2_Quotation_FileOut.printData(data);
}

@Override
public void OnSubscribe_SZSEL2_Static(SZSEL2_Static data) {

}

@Override
public void OnSubscribe_SZSEL2_Status(SZSEL2_Status data) {
```

User Mannual

```
    }

    @Override
    public void OnSubscribe_SZSEL2_Transaction (SZSEL2_Transaction data)
    {
        //System.out.println("OnSubscribe_SZSEL2_Transaction");
        SZSEL2_Transaction_FileOut.printData(data);
    }
}
```

AFileOutBase.java

```
package test;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;
import java.math.RoundingMode;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

import com.gta.qts.c2j.adaptee.structure.BuySellLevelInfo;
import com.gta.qts.c2j.adaptee.structure.BuySellLevelInfo3;
import com.gta.qts.c2j.adaptee.structure.SZSE_BuySellLevelInfo3;

public abstract class AFileOutBase {
    public static final String seperator =
System.getProperty("file.separator");
    public static final String path = System.getProperty("user.dir");
    public static boolean isCsv = false; // .csv or .txt
    public static final String TAB = isCsv ? "," : "\t";
    public static final String fileDir = "JavaFileData" + seperator + new
SimpleDateFormat("yyyyMMdd_HH:mm:ss").format(new Date());
    public static boolean stopping;
```

User Mannual

```
public static String getFileHHmm(Long yyyyMMddHHmmssMMM, int freq)
{
    if (freq == 0) {
        return new SimpleDateFormat("HHmm").format(new Date());
    }

    String timeStr = null;
    if (yyyyMMddHHmmssMMM == null || yyyyMMddHHmmssMMM == 0L) {
        timeStr = new
SimpleDateFormat("yyyyMMddHHmmssMMM").format(new Date());
    }
    else {
        timeStr = String.valueOf(yyyyMMddHHmmssMMM);
    }

    int currHour = Integer.parseInt(timeStr.substring(8, 10));
    int currMinute = Integer.parseInt(timeStr.substring(10, 12));
    int currentMinutes = currHour * 60 + currMinute;
    int splitMinutes = (currentMinutes / freq) * freq;
    int splitHour = splitMinutes / 60;
    int splitMinute = splitMinutes % 60;

    return (splitHour < 10 ? "0" : "") + splitHour + (splitMinute <
10 ? "0" : "") + splitMinute;
}

public static <T> String getFileName(Class<?> clazz, String HHmm) {
    String yyyyMMdd = new SimpleDateFormat("yyyyMMdd").format(new
Date());
    return
clazz.getSimpleName()+"_"+yyyyMMdd+"_"+HHmm+"_Base"+(isCsv ? ".csv" :
".txt");
}

public static <T> String getSnapFileName(Class<?> clazz) {
    String date = new
SimpleDateFormat("yyyyMMdd_HHmmssSSS").format(new Date());
    return clazz.getSimpleName()+"_"+date+"_Base_snap"+(isCsv ?
".csv" : ".txt");
}

public static <T> String getSimpleListFileName(String prefix) {
    String date = new
```


User Mannual

```
SimpleDateFormat("yyyyMMdd_HH:mm:ssSSS").format(new Date());
    return prefix+"_"+date+"_Base"+(isCsv ? ".csv" : ".txt");
}

public static void createFolder() {
    File dirDirectory= new File(path+separator+fileDir+separator) ;
    if(!dirDirectory.exists()) {
        dirDirectory.mkdirs() ;
    }
}

public static File CreateFile(String fileName){
    File dirDirectory= new File(path+separator+fileDir+separator) ;
    if(!dirDirectory.exists()) {
        dirDirectory.mkdirs() ;
    }
    String filePath = path+separator+fileDir+separator+fileName;
    // System.out.println("CreateFile "+filePath);
    File file= new File(filePath) ;
    if(!file.exists()){
        try {
            file.createNewFile();
            return file ;
        } catch (IOException e) {
            e.printStackTrace();
        }
    }else{
        return file ;
    }
    return null ;
}

public static String byteArr2StringAndTrim(byte[] bytes){
    try{
        return new String(bytes,0,bytes.length,"UTF-8").trim();
    }catch(UnsupportedEncodingException e){
        e.printStackTrace();
        return null ;
    }
}

public static DecimalFormat getDecimalFormat() {
    DecimalFormat df = new DecimalFormat("#.#####");
    df.setMinimumFractionDigits(6);
}
```

User Mannual

```
        df.setMaximumFractionDigits(6);
        df.setRoundingMode(RoundingMode.DOWN);
        return df;
    }

    public static DecimalFormat getDecimalFormat8() {
        DecimalFormat df = new DecimalFormat("#.#####");
        df.setMinimumFractionDigits(8);
        df.setMaximumFractionDigits(8);
        df.setRoundingMode(RoundingMode.DOWN);
        return df;
    }

    public static String unsigned2String(int v) {
        return String.valueOf((long) (0xFFFFFFFF & v));
    }

    public static String double2String(DecimalFormat df, double fieldVal)
    {
        if (df == null) {
            df = getDecimalFormat();
        }
        return df.format(fieldVal);
    }

    public static String double2String8(DecimalFormat df, double
fieldVal) {
        if (df == null) {
            df = getDecimalFormat8();
        }
        return df.format(fieldVal);
    }

    public static String levelInfo2StringWithTab(DecimalFormat df,
BuySellLevelInfo[] fieldVal) {
        StringBuilder sb = new StringBuilder();
        BuySellLevelInfo info = null;
        for(int i=0;i<fieldVal.length;i++){
            info = fieldVal[i];
            sb.append(double2String(df,info.Price)).append(TAB);
            sb.append(info.Volume).append(TAB);
        }
        return sb.toString();
    }
}
```

User Mannual

```
public static String levelInfo3ToStringWithTab(DecimalFormat df,
BuySellLevelInfo3[] fieldVal) {
    StringBuilder sb = new StringBuilder();
    BuySellLevelInfo3 info = null;
    for(int i=0;i<fieldVal.length;i++){
        info = fieldVal[i];
        sb.append(double2String(df,info.Price)).append(TAB);
        sb.append(info.Volume).append(TAB);
        sb.append(info.TotalOrderNo).append(TAB);
    }
    return sb.toString();
}

public static String szseLevelInfo3ToStringWithTab(DecimalFormat df,
SZSE_BuySellLevelInfo3[] fieldVal) {
    StringBuilder sb = new StringBuilder();
    SZSE_BuySellLevelInfo3 info = null;
    for(int i=0;i<fieldVal.length;i++){
        info = fieldVal[i];
        sb.append(double2String(df,info.Price)).append(TAB);
        sb.append(double2String(df,info.Volume)).append(TAB);
        sb.append(info.TotalOrderNo).append(TAB);
    }
    return sb.toString();
}

public static BufferedWriter createWriter(String fileName, String
title) {
    BufferedWriter bw = null;
    try {
        File f = CreateFile(fileName);
        FileOutputStream fos = new FileOutputStream(f,true);
        if (f.length() < 1) {
            byte[] bom = new byte[] {(byte) 0xEF, (byte) 0xBB, (byte) 0xBF};
            fos.write(bom);
        }
        bw = new BufferedWriter(new OutputStreamWriter(fos,"UTF-8"));
        if (title != null) {
            bw.write(title);
            bw.newLine();
            bw.flush();
        }
    } catch (IOException e) {
```

User Mannual

```
        e.printStackTrace();
    }
    return bw;
}

public static SimpleDateFormat getLocalTimeFormat() {
    return new SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS");
}

public static void closeWriter(BufferedWriter bw) {
    if (bw != null) {
        try {
            bw.flush();
            bw.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        bw = null;
    }
}

public static String getLocalTimeTitle() {
    return getLocalTimeFormat().format(new Date());
}

public static boolean isStopping() {
    return stopping;
}

public static void setStopping(boolean stopping) {
    AFileOutBase.stopping = stopping;
}
}
```

Formatters.java

```
package test;

import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
```

User Mannual

```
public class Formatters {
    private SimpleDateFormat sdf = AFileOutBase.getLocalTimeFormat();
    private DecimalFormat df6 = AFileOutBase.getDecimalFormat();
    private DecimalFormat df8 = AFileOutBase.getDecimalFormat8();

    public SimpleDateFormat getSdf() {
        return sdf;
    }
    public DecimalFormat getDf6() {
        return df6;
    }
    public DecimalFormat getDf8() {
        return df8;
    }
}
```

SZSEL2_Quotation_FileOut.java

```
package test;

import java.io.BufferedWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.Queue;
import java.util.concurrent.LinkedBlockingQueue;

import com.gta.qts.c2j.adaptee.structure.QTSDDataType;
import com.gta.qts.c2j.adaptee.structure.SZSEL2_Order;
import com.gta.qts.c2j.adaptee.structure.SZSEL2_Quotation;

public class SZSEL2_Quotation_FileOut extends AFileOutBase {
    public static Queue<SZSEL2_Quotation> QUEUE11 = new
LinkedBlockingQueue<SZSEL2_Quotation>();
    public static FileOutThread11 FILE_THREAD;

    private static class FileOutThread11 implements Runnable {
        private boolean stopped;
```

User Mannual

```
@Override
public void run() {
    int freq = 30;
    boolean saveFile = true;
    Formatters fmt = new Formatters();
    String prevHHmm = null;
    String currHHmm = null;
    BufferedWriter bw = null;
    SZSEL2_Quotation data = null;
    while (!stopped) {
        data = QUEUE11.poll();
        if (data == null) {
            if (stopping) {
                if (saveFile) {
                    closeWriter(bw);
                }
                stopped = true;
                break;
            }
            else {
                try {
                    Thread.sleep(10);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                continue;
            }
        }

        if (!saveFile) {
            continue;
        }

        currHHmm = getFileHHmm(data.Time, freq);
        if ((bw == null) || !currHHmm.equals(prevHHmm)) {
            closeWriter(bw);
            bw = createWriter(getFileName(SZSEL2_Quotation.class,
currHHmm), title());
            prevHHmm = currHHmm;
        }

        flushData(data, bw, fmt);
    }
}
```

User Mannual

```
    }

    public boolean isStopped() {
        return stopped;
    }
}

public static void startThread() {
    if (FILE_THREAD == null) {
        FILE_THREAD = new FileOutThread11();
        new Thread(FILE_THREAD).start();
    }
}

public static boolean isStopped() {
    return (FILE_THREAD == null) || FILE_THREAD.isStopped();
}

public static void printData(SZSEL2_Quotation data){
    if (!stopping) {
        QUEUE11.offer(data);
    }
}

public static BufferedWriter createSnapWriter() {
    String fileName = getSnapFileName(SZSEL2_Quotation.class);
    return createWriter(fileName, title());
}

public static void flushData(SZSEL2_Quotation data, BufferedWriter
bw, Formatters fmt) {
    SimpleDateFormat sdf = fmt.getSdf();
    DecimalFormat df = fmt.getDf6();
    try {
        bw.write(sdf.format(new java.util.Date()) + TAB);
        bw.write(data.LocalTimeStamp + TAB);
        bw.write(byteArr2StringAndTrim(data.QuotationFlag) + TAB);
        bw.write(data.Time + TAB);
        bw.write(byteArr2StringAndTrim(data.Symbol) + TAB);
        bw.write(byteArr2StringAndTrim(data.SymbolSource) + TAB);
        bw.write(double2String(df,data.PreClosePrice) + TAB);
        bw.write(double2String(df,data.OpenPrice) + TAB);
        bw.write(double2String(df,data.LastPrice) + TAB);
    }
```

User Mannual

```
        bw.write(double2String(df,data.HighPrice) + TAB);
        bw.write(double2String(df,data.LowPrice) + TAB);
        bw.write(double2String(df,data.PriceUpLimit) + TAB);
        bw.write(double2String(df,data.PriceDownLimit) + TAB);
        bw.write(double2String(df,data.PriceUpdown1) + TAB);
        bw.write(double2String(df,data.PriceUpdown2) + TAB);
        bw.write(data.TotalNo + TAB);
        bw.write(double2String(df,data.TotalVolume) + TAB);
        bw.write(double2String(df,data.TotalAmount) + TAB);
        bw.write(double2String(df,data.ClosePrice) + TAB);
        bw.write(byteArr2StringAndTrim(data.SecurityPhaseTag) +
TAB);

        bw.write(double2String(df,data.PERatio1) + TAB);
        bw.write(double2String(df,data.NAV) + TAB);
        bw.write(double2String(df,data.PERatio2) + TAB);
        bw.write(double2String(df,data.IOPV) + TAB);
        bw.write(double2String(df,data.PremiumRate) + TAB);
        bw.write(double2String(df,data.TotalSellOrderVolume) + TAB);
        bw.write(double2String(df,data.WtAvgSellPrice) + TAB);
        bw.write(unsigned2String(data.SellLevelNo) + TAB);
        bw.write(szseLevelInfo3ToStringWithTab(df,data.SellLevel));
        bw.write(unsigned2String(data.SellLevelQueueNo01) + TAB);
        for(int i=0;i<50;i++) {
            bw.write(double2String(df,data.SellLevelQueue[i]) + TAB);
        }
        bw.write(double2String(df,data.TotalBuyOrderVolume) + TAB);
        bw.write(double2String(df,data.WtAvgBuyPrice) + TAB);
        bw.write(unsigned2String(data.BuyLevelNo) + TAB);
        bw.write(szseLevelInfo3ToStringWithTab(df,data.BuyLevel));
        bw.write(unsigned2String(data.BuyLevelQueueNo01) + TAB);
        for(int i=0;i<50;i++) {
            bw.write(double2String(df,data.BuyLevelQueue[i]) + TAB);
        }
        bw.newLine();
        bw.flush();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static String title() {
    StringBuilder sb = new StringBuilder();
    sb.append(AFileOutBase.getLocalTimeTitle()).append(TAB);
    sb.append("1LocalTimeStamp 采集时间").append(TAB)
```

```
.append("2QuotationFlag 行情源标志").append(TAB)
.append("3Time 数据生成时间").append(TAB)
.append("4Symbol 证券代码").append(TAB)
.append("5SymbolSource 证券代码源").append(TAB)
.append("6PreClosePrice 昨收价").append(TAB)
.append("7OpenPrice 开盘价").append(TAB)
.append("8LastPrice 现价").append(TAB)
.append("9HighPrice 最高价").append(TAB)
.append("10LowPrice 最低价").append(TAB)
.append("11PriceUpLimit 涨停价").append(TAB)
.append("12PriceDownLimit 跌停价").append(TAB)
.append("13PriceUpdown1 涨跌一").append(TAB)
.append("14PriceUpdown2 涨跌二").append(TAB)
.append("15TotalNo 成交笔数").append(TAB)
.append("16TotalVolume 成交总量").append(TAB)
.append("17TotalAmount 成交总额").append(TAB)
.append("18ClosePrice 今收盘价").append(TAB)
.append("19SecurityPhaseTag 当前品种交易状态").append(TAB)
.append("20PERatio1 市盈率 1").append(TAB)
.append("21NAV 基金 T-1 日净值").append(TAB)
.append("22PERatio2 市盈率 2").append(TAB)
.append("23IOPV 基金实时参考净值").append(TAB)
.append("24PremiumRate 权证溢价率").append(TAB)
.append("25TotalSellOrderVolume 委托卖出总量").append(TAB)
.append("26WtAvgSellPrice 加权平均委卖价").append(TAB)
.append("27SellLevelNo 卖盘价位数").append(TAB)
.append("28SellPrice01 申卖价").append(TAB)
.append("29SellVolume01 申卖量").append(TAB)
.append("30TotalSellOrderNo01 卖出总委托笔数").append(TAB)
.append("31SellPrice02 申卖价").append(TAB)
.append("32SellVolume02 申卖量").append(TAB)
.append("33TotalSellOrderNo02 卖出总委托笔数").append(TAB)
.append("34SellPrice03 申卖价").append(TAB)
.append("35SellVolume03 申卖量").append(TAB)
.append("36TotalSellOrderNo03 卖出总委托笔数").append(TAB)
.append("37SellPrice04 申卖价").append(TAB)
.append("38SellVolume04 申卖量").append(TAB)
.append("39TotalSellOrderNo04 卖出总委托笔数").append(TAB)
.append("40SellPrice05 申卖价").append(TAB)
.append("41SellVolume05 申卖量").append(TAB)
.append("42TotalSellOrderNo05 卖出总委托笔数").append(TAB)
.append("43SellPrice06 申卖价").append(TAB)
.append("44SellVolume06 申卖量").append(TAB)
.append("45TotalSellOrderNo06 卖出总委托笔数").append(TAB)
```

```
.append("46SellPrice07 申卖价").append(TAB)
.append("47SellVolume07 申卖量").append(TAB)
.append("48TotalSellOrderNo07 卖出总委托笔数").append(TAB)
.append("49SellPrice08 申卖价").append(TAB)
.append("50SellVolume08 申卖量").append(TAB)
.append("51TotalSellOrderNo08 卖出总委托笔数").append(TAB)
.append("52SellPrice09 申卖价").append(TAB)
.append("53SellVolume09 申卖量").append(TAB)
.append("54TotalSellOrderNo09 卖出总委托笔数").append(TAB)
.append("55SellPrice10 申卖价").append(TAB)
.append("56SellVolume10 申卖量").append(TAB)
.append("57TotalSellOrderNo10 卖出总委托笔数").append(TAB)
.append("58SellLevelQueueNo01 卖一档揭示委托笔数").append(TAB);
for (int idx = 1; idx <= 50; idx++) {
    sb.append(58 + idx).append("SellLevelQueue" + idx + "卖1档队
列").append(TAB);
}
sb.append("109TotalBuyOrderVolume 委托买入总量").append(TAB)
.append("110WtAvgBuyPrice 加权平均买入价").append(TAB)
.append("111BuyLevelNo 买盘价位数").append(TAB)
.append("112BuyPrice01 申买价").append(TAB)
.append("113BuyVolume01 申买量").append(TAB)
.append("114TotalBuyOrderNo01 买入总委托笔数").append(TAB)
.append("115BuyPrice02 申买价").append(TAB)
.append("116BuyVolume02 申买量").append(TAB)
.append("117TotalBuyOrderNo02 买入总委托笔数").append(TAB)
.append("118BuyPrice03 申买价").append(TAB)
.append("119BuyVolume03 申买量").append(TAB)
.append("120TotalBuyOrderNo03 买入总委托笔数").append(TAB)
.append("121BuyPrice04 申买价").append(TAB)
.append("122BuyVolume04 申买量").append(TAB)
.append("123TotalBuyOrderNo04 买入总委托笔数").append(TAB)
.append("124BuyPrice05 申买价").append(TAB)
.append("125BuyVolume05 申买量").append(TAB)
.append("126TotalBuyOrderNo05 买入总委托笔数").append(TAB)
.append("127BuyPrice06 申买价").append(TAB)
.append("128BuyVolume06 申买量").append(TAB)
.append("129TotalBuyOrderNo06 买入总委托笔数").append(TAB)
.append("130BuyPrice07 申买价").append(TAB)
.append("131BuyVolume07 申买量").append(TAB)
.append("132TotalBuyOrderNo07 买入总委托笔数").append(TAB)
.append("133BuyPrice08 申买价").append(TAB)
.append("134BuyVolume08 申买量").append(TAB)
.append("135TotalBuyOrderNo08 买入总委托笔数").append(TAB)
```

User Mannual

```
.append("136BuyPrice09 申买价").append(TAB)
.append("137BuyVolume09 申买量").append(TAB)
.append("138TotalBuyOrderNo09 买入总委托笔数").append(TAB)
.append("139BuyPrice10 申买价").append(TAB)
.append("140BuyVolume10 申买量").append(TAB)
.append("141TotalBuyOrderNo10 买入总委托笔数").append(TAB)
.append("142BuyLevelQueueNo01 买一档揭示委托笔数").append(TAB);
for (int idx = 1; idx <= 50; idx++) {
    sb.append(142 + idx).append("BuyLevelQueue" + idx + "买1档队
列");
    if (idx <= 49) {
        sb.append(TAB);
    }
}
return sb.toString();
}
}
```

SZSEL2_Transaction_FileOut.java

```
package test;

import java.io.BufferedWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import java.util.Queue;
import java.util.concurrent.LinkedBlockingQueue;

import com.gta.qts.c2j.adaptee.structure.SZSEL2_Transaction;

public class SZSEL2_Transaction_FileOut extends AFileOutBase {
    public static Queue<SZSEL2_Transaction> QUEUE14 = new
LinkedBlockingQueue<SZSEL2_Transaction>();
    public static FileOutThread14 FILE_THREAD;

    private static class FileOutThread14 implements Runnable {
        private boolean stopped;
    }
}
```

User Mannual

```
@Override
public void run() {
    int freq = 30;
    boolean saveFile = true;
    Formatters fmt = new Formatters();
    String prevHHmm = null;
    String currHHmm = null;
    BufferedWriter bw = null;
    SZSEL2_Transaction data = null;
    while (!stopped) {
        data = QUEUE14.poll();
        if (data == null) {
            if (stopping) {
                if (saveFile) {
                    closeWriter(bw);
                }
                stopped = true;
                break;
            }
            else {
                try {
                    Thread.sleep(10);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                continue;
            }
        }

        if (!saveFile) {
            continue;
        }

        currHHmm = getFileHHmm(data.TradeTime, freq);
        if ((bw == null) || !currHHmm.equals(prevHHmm)) {
            closeWriter(bw);
            bw =
createWriter(getFileName(SZSEL2_Transaction.class, currHHmm),
title());

            prevHHmm = currHHmm;
        }

        flushData(data, bw, fmt);
    }
}
```

```
    }

    public boolean isStopped() {
        return stopped;
    }
}

public static void startThread() {
    if (FILE_THREAD == null) {
        FILE_THREAD = new FileOutThread14();
        new Thread(FILE_THREAD).start();
    }
}

public static boolean isStopped() {
    return (FILE_THREAD == null) || FILE_THREAD.isStopped();
}

public static void printData(SZSEL2_Transaction data){
    if (!stopping) {
        QUEUE14.offer(data);
    }
}

public static BufferedWriter createSnapWriter() {
    String fileName = getSnapFileName(SZSEL2_Transaction.class);
    return createWriter(fileName, title());
}

public static void flushData(SZSEL2_Transaction data, BufferedWriter
bw, Formatters fmt) {
    SimpleDateFormat sdf = fmt.getSdf();
    DecimalFormat df = fmt.getDf6();
    try {
        bw.write(sdf.format(new java.util.Date()) + TAB);
        bw.write(data.LocalTimeStamp + TAB);
        bw.write(byteArr2StringAndTrim(data.QuotationFlag) + TAB);
        bw.write(unsigned2String(data.SetID) + TAB);
        bw.write(data.RecID + TAB);
        bw.write(data.BuyOrderID + TAB);
        bw.write(data.SellOrderID + TAB);
        bw.write(byteArr2StringAndTrim(data.Symbol) + TAB);
        bw.write(byteArr2StringAndTrim(data.SymbolSource) + TAB);
```

User Mannual

```
        bw.write(data.TradeTime + TAB);
        bw.write(double2String(df,data.TradePrice) + TAB);
        bw.write(double2String(df,data.TradeVolume) + TAB);
        bw.write(byteArr2StringAndTrim(new
byte[] {data.TradeType}));
        bw.newLine();
        bw.flush();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static String title() {
    StringBuilder sb = new StringBuilder();
    sb.append(AFileOutBase.getLocalTimeTitle()).append(TAB);
    sb.append("1LocalTimeStamp 采集时间").append(TAB)
    .append("2QuotationFlag 行情源标志").append(TAB)
    .append("3SetID 频道代码").append(TAB)
    .append("4RecID 消息记录号").append(TAB)
    .append("5BuyOrderID 买方委托索引").append(TAB)
    .append("6SellOrderID 卖方委托索引").append(TAB)
    .append("7Symbol 证券代码").append(TAB)
    .append("8SymbolSource 证券代码源").append(TAB)
    .append("9TradeTime 成交时间").append(TAB)
    .append("10TradePrice 成交价格").append(TAB)
    .append("11TradeVolume 成交数量").append(TAB)
    .append("12TradeType 成交类型");
    return sb.toString();
}
}
```

8.4 C#示例代码

注意：QTS C#接口的动态库默认为 32 位，因此如果使用 64 位工程编译，需要在工程删除 32 位动态库文件，重新加载 64 位动态库文件，才可以编译成功。

User Mannual

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using GTAQTS;
using GTAQTS.Model;

namespace ExampleDotNet
{
    class Program
    {
        static void Main(string[] args)
        {
            //回调类
            CallbackBase Callback = new CallbackBase();
            IGTAQTSApiBase pApiBase =
            IGTAQTSApiBase.CreateInstance(Callback);
            // 注册于登陆
            pApiBase.RegisterService("192.168.105.136", 7777);

            pApiBase.LoginX("gtal", "123456", "NetType=0");

            // 按代码订阅上交所实时行情数据
            pApiBase.Subscribe(MsgType.Msg_SSEL2_Quotation,
            "000001,000002,000003");
            // 订阅上交所实时行情所有数据
            pApiBase.Subscribe(MsgType.Msg_SSEL2_Quotation, "");

            Console.ReadLine();
            // 按代码取消上交所实时行情数据
            pApiBase.Unsubscribe(MsgType.Msg_SSEL2_Quotation,
            "000001,000002,000003");
            // 取消上交所实时所有代码
            pApiBase.Unsubscribe(MsgType.Msg_SZSEL2_Quotation, "");

            // 上交所实时行情快照查询
            IEnumerable<SSEL2_Quotation> Snap_Quotation;
            pApiBase.QuerySnap_SSEL2_Quotation("",out Snap_Quotation);

            String strQuot = String.Format("QuerySnap_Base:Count = {0}",
```

User Mannual

```
Snap_Quotation.Count ( ) ;

        Console.WriteLine (strQuot) ;

        Console.ReadLine ( ) ;
        IGTAQTSApiBase.ReleaseInstance ( ) ;
    }
}
}
```

CallbackBase.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using GTAQTS;
using GTAQTS.Model;

namespace ExampleDotNet
{
    class CallbackBase : IGTAQTSCallbackBase
    {
        // 防止输出太多内容
        static int count = 0;
        /// 上交所 L2 实时行情快照订阅数据实时回调接口
        /// @param RealSSEL2Quotation -- 实时数据
        public override void OnSubscribe_SSEL2_Quotation (SSEL2_Quotation
RealSSEL2Quotation)
        {
            if ( count < 10 )
            {
                count++;
                String strQuot = String.Format ("Subscribe Data:
PacketTimeStamp = {0}, Symbol = {1}, OpenPrice = {2}, HighPrice = {3},
LowPrice = {4}, LastPrice = {5}, ClosePrice = {6}",
                    RealSSEL2Quotation.PacketTimeStamp,
                    RealSSEL2Quotation.Symbol, RealSSEL2Quotation.OpenPrice,
                    RealSSEL2Quotation.HighPrice,
                    RealSSEL2Quotation.LowPrice,
                    RealSSEL2Quotation.LastPrice, RealSSEL2Quotation.ClosePrice);
```

User Mannual

```
        System.Console.WriteLine(strQuot);  
    }  
}  
}  
}
```
