中国银河证券格物金融服务平台
金融数据功能开发手册
（Python 版）

目录

1. 版本说明

1.1 文档管理信息表

主题	中国银河证券格物金融服务平台金融数据功能开发手册（Python 版）
文档版本	V1.0.0
Python SDK 版本	V1.0.1
创建时间	2023 年 1 月 28 日
创建人	中国银河证券信息技术部量化交易服务研发团队
最新发布日期	2023 年 3 月 10 日

1.2 文档变更记录表

修改人	修改时间	修改内容
张德高	2023 年 3 月 10 日	文档整体结构调整和细节表述优化

2. 功能介绍

本文档是 tgw 的 SDK 开发指南，包含了对 API 接口的说明以及示例，用于指引开发人员通过 tgw 金融数据功能接口进行数据接收和查询的开发。

2.1 术语和缩略语

术语、缩写	解释
tgw	tgw，格物机构金融服务平台
SDK	Software Development Kit，金融数据功能软件开发工具包

2.2 金融数据服务

金融数据功能，是指用户使用 C++、Python 以及其他本功能可支持的程序设计语言或用户端页面，获取公司通过对证券交易所等渠道的公开信息加工而成的行情数据、金融资讯数据等金融数据的功能。

2.3 两种使用场景

tgw 包含了托管机房模式和互联网模式两种使用场景。

2.3.1 托管机房模式使用场景

在银河证券机房内部网络环境内，用户除 Level-1、K 线、资讯数据、因子数据等中低频数据外，通过 SDK 提供获取需要增强 Level-2 等高频数据。

2.3.2 互联网模式使用场景

在互联网网络环境内下，用户通过 SDK 提供获取 Level-1、K 线、资讯数据、因子数据等中低频数据。

2.4 数据内容

托管机房模式和互联网模式都能提供的数据包括：

- 实时推送 K 线数据（1/3/5/10/15/30/60/120 分钟线）
- K 线历史数据（日/周/月/季年线、1/3/5/10/15/30/60/120 分钟线）
- 实时推送快照衍生指标数据
- 实时推送加工因子数据

- 证券个股基本信息查询
- 证券代码表信息查询
- 复权因子表信息查询
- 加工因子历史数据
- 金融资讯数据查询

托管机房模式下的数据包括：

- Level2 现货快照历史数据
- 指数快照历史数据
- Level2 历史委托队列数据
- Level2 历史逐笔委托数据
- Level2 历史逐笔成交数据

互联网模式下的数据包括：

- 实时推送 Level1 现货快照数据
- 实时推送指数快照数据
- 实时推送期权快照数据
- 实时推送期货快照数据
- 实时推送港股通快照数据
- 实时推送盘后定价交易快照数据
- 实时推送中证指数快照数据
- 实时推送深交所国证指数快照数据
- 实时推送港股通实时额度数据
- 实时推送港股通产品状态快照数据
- 港股市场波动调节机制(VCM)推送数据
- Level1 快照历史数据

3. python 开发指南

3.1 SDK

3.1.1 wheel 文件版本

wheel 文件名	操作系统	Python 版本
tgw-1.*.*-cp38-none-win_amd64.whl	Windows	Python3.8
tgw-1.*.*-cp36-none-win_amd64.whl	Windows	Python3.8
tgw-1.*.*-cp38-none-linux_x86_64old.whl	Linux	Python3.6
tgw-1.*.*-cp36-none-linux_x86_64old.whl	Linux	Python3.6

3.1.2 SDK 目录结构

各操作系统版本 SDK 下载解压首次运行 tgw_install 脚本构建 tgw 运行环境后，再运行 pip 安装脚本 pip-install 即可使用，python3.6 下面目录结构如下：

Linux 系统下 **python 版本 SDK 目录**

```
.\python3.6
|   tgw.py ( API 接口文件)
|   tgw_demo.py（开发样例程序）
|   tgw_test.py（演示程序，使用时可直接运行./run_test36.sh
|                运行前需更改配置文件）
|   run_demo.sh （样例程序运行脚本）
|   run_test36.sh （演示程序运行脚本）
|   pip-install.sh （tgw 的 pip 安装脚本）
├──etc
|       test.json   (tgw_test 所需配置文件)
└──dist
|       tgw-1.*.*-cp36-none-linux_x86_64.whl(pip 安装文件)
└──lib
        libtgw_python36.so(依赖库文件)
```

Windows 系统下 **python 版本 SDK 目录**

```
.\python3.6
|   tgw.py ( API 接口文件)
|   tgw_demo.py（开发样例程序）
|   tgw_test.py（演示程序，使用时可直接运行./run_test36.bat
|                运行前需更改配置文件）
|   run_demo.bat （样例程序运行脚本）
```

```
|   run_test36.bat （演示程序运行脚本）
|   pip-install.bat （tgw 的 pip 安装脚本）
├──etc
|       test.json(tgw_test 所需配置文件)
└──dist
|       tgw-1.*.*-cp36-none-win_amd64.whl(pip 安装文件)
└──lib
        _tgw.pyd(依赖库文件)
```

使用前必须有中国银河证券股份有限公司授权的登录账户密码以及服务 IP 地址和端口。建议使用 Telnet 命令验证网络连通性。

3.1.3 SDK 安装

pip install twg*（某个 wheel 文件名）

3.2 运行环境

3.2.1 操作系统及编译器版本

支持系统版本	编译器	备注
RedHat7.2 RedHat7.4 RedHat7.6	gcc4.8.5	支持多种 Level 1 或 Level 2 及自定义因子数据的订阅推送，支持快照，k 线，因子等数据查询，代码表查询，金融资讯数据查询
Windows10（64 位）	VC++ 2017	支持多种 Level 1 或 Level 2 及自定义因子数据的订阅推送，支持快照，k 线，因子等数据查询，代码表查询，金融资讯数据查询

3.2.2 Linux 推荐运行环境配置

类型	最低配置	推荐配置
处理器	2.10GHz,4 核	2.10GHz,8 核
内存	DDR4 4GB	DDR4 4GB
硬盘	200G 机械硬盘/SSD	480G 机械硬盘/SSD
网卡	普通网卡	普通万兆网卡
操作系统	REDHAT 7.2/7.4/7.6	REDHAT 7.2/7.4/7.6

3.2.3 Windows 推荐运行环境配置

类型	最低配置	推荐配置
处理器	2.60GHz，4 核	2.60GHz，8 核
内存	DDR4 4GB	DDR4 4GB
硬盘	200G 机械硬盘/SSD	480G 机械硬盘/SSD
网卡	普通网卡	普通万兆网卡
操作系统	Windows 10(64 位)	Windows 10(64 位)

3.3 运行模式说明

tgw 包含托管机房模式和互联网模式。
通过在初始化启动时设置 api_mode 区分

3.3.1 托管机房模式

三种通道 TCP、QTCP、RTCP，可单独使用，可两两结合，也可三个同时使用。
参数设置：

（1） 通道模式 api_mode 设置
tgw.ApiMode.kColocationMode 为托管机房模式

（2） 通道线程数 coloca_cfg.qtcp_channel_thread
最小值为 1，最大值为 8。

（3） 通道线程内数据条数 coloca_cfg.qtcp_max_req_cnt
最小值为 1000，最大值为 3000。

3.3.1.1 TCP 通道

（1） 通道设置
coloca_cfg.channel_mode 为 ColocatChannelMode.kTCP

（2） 数据获取范围
只支持订阅接口的数据，不包含订阅范围外的数据。

（3） 异常情况说明

1) 网络问题导致下游连接断开，接口会自动发起重连，无需调用特殊接口处理。
2) 网络堵塞或者下游处理速度太慢会导致消息堆积，如果一直没有改善，上游会将处理慢的下游连接断开，保证不同的连接之间不会互相影响。

3.3.1.2 QTCP 通道

（1） 通道设置

coloca_cfg.channel_mode 为 ColocatChannelMode.kQTCP

（2） 数据获取范围

只支持查询接口的数据，不包含查询范围外的数据。

（3） 异常情况说明

网络问题导致下游连接断开，接口会自动发起重连，无需调用特殊接口处理。

3.3.1.3 RTCP 通道

（1） 通道设置

coloca_cfg.channel_mode 为 ColocatChannelMode.kRTCP

（2） 数据获取范围

只支持回放接口的数据，不包含回放范围外的数据。

（3） 异常情况说明

网络问题导致下游连接断开，接口会自动发起重连，无需调用特殊接口处理。

3.3.1.4 TCP、QTCP、RTCP 通道同时使用的通道设置

（1） 三个通道 TCP、QTCP、RTCP，可两两结合，也可三个同时使用。

（2） 通过设置 coloca_cfg.channel_mode，用“|”分开即可。

（3） 例如“tgw.ColocatChannelMode.kQTCP | tgw.ColocatChannelMode.kTCP”。

3.3.2 互联网模式

（1） 数据接入方式

通过 websocket 方式和上游保持连接，并进行行情数据的推送和查询。

（2） 通道设置

设置 api_mode 为 tgw.ApiMode. kInternetMode

3.4 Python 开发步骤

流程描述：

1. tgw 初始化配置

   包含服务器地址、用户账号密码和运行模式配置

2. 基础接口包含初始化、订阅、退订、释放等方法，无需创建实例；

3. 数据接口包含订阅、查询和回放等三类方法；

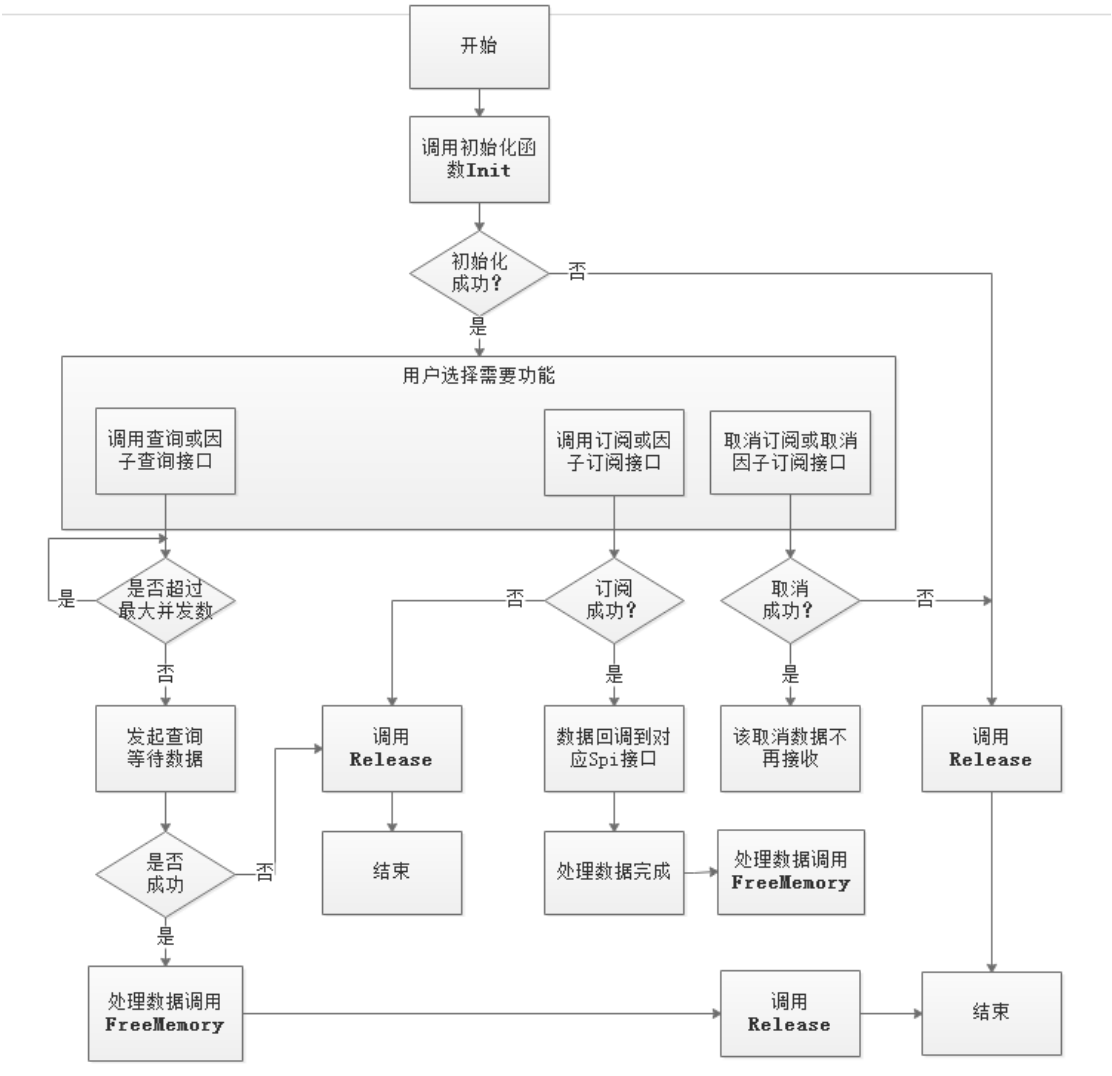
   （1） 继承所需的 tgw 中的 IGMDSpi 类，在派生类中重载回调函数；

   （2） 派生类实例化；

   （3） 派生类的实例传递给 tgw.IGMDApi 函数，供 tgw 回调使用；

   （4） 派生类的实例生命周期长于 tgw，即派生类实例化必须在 init 初始化之前；

查询和订阅接口使用流程图



3.4.1 初始化 tgw

1. tgw.Cfg()实例化为 cfg
2. 服务器地址配置
3. 用户登录账号配置
4. 运行模式配置初始化，
5. tgw.IGMDApi_Init 完成登录验证、运行模式设置、传实例到订阅方法三个功能
6. 如初始化失败，IGMDApi_Release 函数释放资源后退出

```
cfg = tgw.Cfg()

# 服务器地址配置
# cfg.server_vip = "223.70.124.220"
cfg.server_vip = "10.4.47.21"
cfg.server_port = 8600
# 用户登录账号配置
cfg.username = "tgw_zdg_test"   # 账号
cfg.password = "123456"   # 密码
# 运行模式配置
# api_mode = tgw.ApiMode.kColocationMode #  设置 api 模式 托管机房模式
api_mode = tgw.ApiMode.kInternetMode   # 设置 api 模式 互联网模式
if (api_mode == tgw.ApiMode.kColocationMode):
```

```
    cfg.coloca_cfg.channel_mode = tgw.ColocatChannelMode.kQTCP   # tcp 查询模式
    cfg.coloca_cfg.qtcp_channel_thread = 2
    cfg.coloca_cfg.qtcp_max_req_cnt = 1000

# 初始化返回错误码，完成登录验证、运行模式设置、传实例到订阅方法三个功能
error_code = tgw.IGMDApi_Init(spi, cfg, api_mode)
# 如初始化失败，退出流程
if error_code != tgw.ErrorCode.kSuccess:
    print("Init tgw failed")
    tgw.IGMDApi_Release()
    exit(-1)
```

3.4.2 继承 IGMDSpi 类

继承所需的 tgw 中的 IGMDSpi 类，在派生类中重载回调函数，示例如下

```
# 订阅 spi
class IAMDSpiApp(tgw.IGMDSpi):
    def OnLog(self, level, log, len):
        print("tgw log: ", "level: ", level, end='')
        print("        log:    ", log)

    def OnLogon(self, data):
        print("tgw Logon information:   : ")
        print("api_mode : ", data.api_mode)
        print("logon json : ", data.logon_json)
        tgw.IGMDApi_FreeMemory(data)
```

3.4.3 派生类实例化

示例：spi = IAMDSpiApp()

```
# 订阅 spi
class IAMDSpiApp(tgw.IGMDSpi):
    def OnLog(self, level, log, len):
        print("tgw log: ", "level: ", level, end='')
        print("        log:    ", log)

    def OnLogon(self, data):
        print("tgw Logon information:   : ")
        print("api_mode : ", data.api_mode)
        print("logon json : ", data.logon_json)
        tgw.IGMDApi_FreeMemory(data)
```

3.4.4 IGMDApi 函数获取数据

```
sub_item = tgw.SubscribeItem()
sub_item.market = GetMarkert("All")
sub_item.flag = GetSubType("All")
sub_item.category_type = GetCategory("All")
sub_item.security_code = "000001"
tgw.IGMDApi_Subscribe(sub_item, 1)   # 订阅全市场，全类型，全品种，全代码

sub_factor_item = tgw.SubFactorItem()
sub_factor_item.factor_type = "all"
sub_factor_item.factor_sub_type = "all"
sub_factor_item.factor_name = "all"

tgw.IGMDApi_SubFactor(sub_factor_item, 1)
```

3.4.5 取消订阅数据

取消订阅调用接口 IGMDApi_UnSubscribe，参数设置与订阅接口相同，取消订阅后将不再接收该类型的数据。

3.5 API 接口详细

3.5.1 参数说明

3.5.1.1 参数类型

- 入参（in）代表必须输入的参数。
- 出参（out）表示程序输出的参数。

3.5.1.2 精度说明

tgw 对各行情字段的数值相对实际值的倍数关系，具体如下：

类型	实际数值扩大倍数
数量（如申买量、申卖量等）	100 倍
价格（如开盘价、最新价等）	1000000 倍
金额（如总成交额等）	100000 倍
汇率	100000000 倍
比例（如涨跌幅、市盈率等）	1000000 倍

3.5.1.3 单位说明

市场 / 证券品种	上海	深圳
股票	1 股	1 股
债券	10 张	1 张
基金	1 份	1 份
期权	1 张	1 张

3.5.1.4 订阅数据类型定义

名称	值	说明
kNone	0	订阅全部数据

k1MinKline	1	订阅 1 分钟 k 线数据
k3MinKline	2	订阅 3 分钟 k 线数据
k5MinKline	3	订阅 5 分钟 k 线数据
k10MinKline	4	订阅 10 分钟 k 线数据
k15MinKline	5	订阅 15 分钟 k 线数据
k30MinKline	6	订阅 30 分钟 k 线数据
k60MinKline	7	订阅 60 分钟 k 线数据
k120MinKline	8	订阅 120 分钟 k 线数据
kSnapshotDerive	9	订阅快照衍生数据
kSnapshot	10	订阅现货快照数据
kOptionSnapshot	11	订阅期权快照数据
kHKTSnapshot	12	订阅港股快照数据
kIndexSnapshot	13	订阅指数快照数据
kAfterHourFixedPriceSnapshot	14	订阅盘后快照数据
kCSIIndexSnapshot	15	订阅中证指数数据
kCnIndexSnapshot	16	订阅国证指数快照数据
kHKTRealtimeLimit	17	订阅港股通实时额度数据
kHKTProductStatus	18	订阅港股通产品状态数据
kHKTVCM	19	订阅港股 VCM 数据
kFutureSnapshot	20	订阅期货数据

3.5.1.5 查询数据类型定义

名称	类型	说明
k1MinKline	10000	1 分钟 k 线数据
k3MinKline	10001	3 分钟 k 线数据
k5MinKline	10002	5 分钟 k 线数据
k10MinKline	10003	10 分钟 k 线数据
k15MinKline	10004	15 分钟 k 线数据
k30MinKline	10005	30 分钟 k 线数据
k60MinKline	10006	60 分钟 k 线数据
k120MinKline	10007	120 分钟 k 线数据
kDayKline	10008	日 K 线
kWeekKline	10009	周 K 线
kMonthKline	10010	月 K 线
kSeasonKline	10011	季 K 线
kYearKline	10012	年 K 线
kTickExecution	10013	逐笔成交
kSnapshot	10014	现货快照

3.5.2 基础接口

tgw 的接口，直接调用方法即可，无需创建实例。如调用 tgw.IGMDApi_ GetVersion ()。

1) 查询 tgw 版本号方法（GetVersion）

函数功能：获取当前 tgw 版本号
适用运行模式：互联网和托管机房模式
函数原型：

```
def IGMDApi_GetVersion()
```

返回值：返回当前 tgw 版本号。

2) 初始化 tgw 方法（Init）

函数功能：初始化 tgw
适用运行模式：互联网和托管机房模式
函数原型：

```
def IGMDApi_Init(pSpi, cfg, api_mode)
```

参数：

参数	解释
Spi	tgw.IGMDSpi 的派生类实例
cfg	tgw. Cfg()的实例，需对其属性赋值
api_mode	模式，1：托管机房 2：互联网

返回值：返回错误码，参照公共数据字典中的 ErrorCode。

cfg 的属性定义如下：

名称	数据类型	说明
server_vip	str	服务 ip 地址
server_port	int	服务端口
username	str	用户名
password	str	用户密码，明文填入，密文使用
coloca_cfg	class	托管机房配置（此配置在选择托管机房模式才能生效）

coloca_cfg 属性定义如下：

名称	类型	说明
channel_mode	int	通道模式的集合，请参考公共字典中 ChannelMode 定义，该配置为各通道模式的集合
qtcp_channel_thread	int	tcp 查询通道数据消费线程数，默认开启 2 个线程
qtcp_req_time_out	int	tcp 查询通道数据请求检查超时时间(单位为分钟)，默认为 10 分钟
qtcp_max_req_cnt	int	tcp 查询通道实时请求最大数，默认 1000

3) 退出释放 tgw 方法（Release）

函数功能：释放 tgw 所用线程资源，完成整个程序的退出流程
适用运行模式：互联网和托管机房模式
函数原型：

def IGMDApi_Release()

返回值：返回错误码，参照公共数据字典中的 **ErrorCode**。

4) 内存释放方法（FreeMemory）

函数功能：释放数据占用的内存，每次接收到数据处理完之后必须调用此方法，否则会造成内存泄漏
适用运行模式：互联网和托管机房模式
函数原型：

def IGMDApi_FreeMemory(data)

参数：

参数	解释
data(in)	class，数据对象

5) 获取 task_id 编号（GetTaskID）方法

函数功能：获取 task_id 编号,用于部分需要 task_id 入参的查询，回放
适用运行模式：互联网和托管机房模式
函数原型：

def IGMDApi_ GetTaskID ()

参数：无
返回值：task_id

6) 更新密码方法（UpdatePassWord）

函数功能：更新密码接口。
适用运行模式：互联网和托管机房模式，托管机房模式需开启 TCP 通道
函数原型：

def IGMDApi_UpdatePassWord(req)

参数	解释
req(in)	请求指针（返回错误码参考 ErrorCode）

req 属性定义如下：

名称	类型	说明
username	str	用户名
old_password	str	旧密码
new_password	str	新密码

返回值：返回错误码，参照公共数据字典中的 **ErrorCode**。
错误码为超时 kTimeoutExit 则大概率是失败，仍有小概率网络问题会修改成功，需要注意该错误码。

3.5.3 订阅数据方法

Subscribe 方法

函数功能：订阅行情数据
适用运行模式：互联网和托管机房模式
函数原型：

def IGMDApi_Subscribe(item, cnt)

参数	解释
item(in)	订阅行情数据的实例
cnt(in)	订阅行情数据结构体个数

item 属性定义如下：

名称	类型	说明
market	int	市场，取值参考 **MarketType**
flag	int	各数据类型的集合，为 0 表示订阅所有支持的数据类型，参考 **SubscribeDataType**
security_code	str	证券代码，为空表示所有代码
category_type	int	参考 **VarietyCategory**，为 0 表示订阅所有支持的品种，仅该参数仅互联网有效

返回值：返回错误码，参照公共数据字典中的 **ErrorCode**。

UnSubscribe 方法

函数功能：取消订阅行情数据
适用运行模式：互联网和托管机房模式
函数原型：

def IGMDApi_UnSubscribe(item, cnt))

参数	解释
item(in)	取消订阅行情数据的实例
cnt(in)	取消订阅行情数据结构体个数

item 属性定义：参考 Subscribe 方法
返回值：返回错误码，参照公共数据字典中的 **ErrorCode**

SubFactor 方法

函数功能：订阅因子行情数据
适用运行模式：互联网和托管机房模式
函数原型：

def IGMDApi_SubFactor(item, cnt)

参数	解释
item(in)	订阅因子数据的实例
cnt(in)	订阅因子数据结构体个数

item 属性定义如下：

名称	类型	说明
factor_type	str	因子父类型
factor_sub_type	str	因子子类型
factor_name	str	因子名称

返回值：返回错误码，参照公共数据字典中的 **ErrorCode**

UnSubFactor 方法

函数功能：取消订阅因子行情数据
适用运行模式：互联网和托管机房模式
函数原型：

def IGMDApi_UnSubFactor(item, cnt)

参数	解释
item(in)	取消订阅因子数据的实例
cnt(in)	取消订阅因子数据结构体个数

item 属性定义：参考 SubFactor 方法

3.5.4 订阅回调方法

（1） 继承所需的 tgw.IGMDSpi 类，在派生类中重载回调函数；
（2） 派生类实例化；
（3） 派生类的实例传递给 tgw. IGMDApi_Init 函数，供 tgw 回调使用；
（4） 派生类的实例生命周期长于 tgw，即派生类实例化必须在 init 初始化之前；

1) 接收日志数据方法

OnLog 方法

函数功能：接收日志数据的回调函数
适用运行模式：互联网和托管机房模式
函数原型：

def OnLog(self, level, log, len)

参数：

参数	类型	解释
level(out)	int	日志数据级别（参考 **LogLevel**）
log(out)	str	日志内容
len(out)	int	日志内容长度

2) 接收登陆成功时数据方法

OnLogon 方法

函数功能：接收登录成功时的数据，可根据需求对该回调信息做相应的处理

适用运行模式：互联网和托管机房模式

函数原型：

def OnLogon(self, data)

参数：

参数	解释
data(out)	登陆成功后的校验信息

data 属性定义如下：

名称	类型	说明
api_mode	int	模式，1：托管机房　2：互联网
logon_msg_len	int	登录返回信息长度
logon_json	str	登录返回信息，格式为 json

3) 接收快照数据方法

OnMDSnapshot 方法

函数功能：接收现货快照数据回调

适用运行模式：互联网模式

函数原型：

def OnMDSnapshot(self, data,cnt)

参数：

参数	解释
data(out)	现货快照数据 object，参考 **MDSnapshotL1**。
cnt(out)	数据个数

OnMDIndexSnapshot 方法

函数功能：接收指数快照数据回调

适用运行模式：互联网模式

函数原型：

def OnMDIndexSnapshot(self, data, cnt)

参数：

参数	解释
data(out)	指数快照数据 object，参考 **MDIndexSnapshot**。
cnt(out)	数据个数

OnMDOptionSnapshot 方法

函数功能：接收期权数据回调

适用运行模式：互联网模式

函数原型：

def OnMDOptionSnapshot(self, data,cnt)

参数：

参数	解释

data(out)	期权快照数据 object，参考 **MDOptionSnapshot**。
cnt(out)	数据个数

OnMDHKTSnapshot 方法

函数功能：接收港股快照数据回调
适用运行模式：互联网模式
函数原型：

def OnMDHKTSnapshot(self, data, cnt)

参数：

参数	解释
data(out)	港股快照数据 object，参考 **MDHKTSnapshot**。
cnt(out)	数据个数

OnMDFutureSnapshot 方法

函数功能：接收期货快照推送数据回调
适用运行模式：互联网模式
函数原型：

def OnMDFutureSnapshot(self, data, cnt)

参数	解释
data(out)	推送期货快照数据 object，参考 **MDFutureSnapshot**。
cnt(out)	期货快照数据个数

OnMDAfterHourFixedPriceSnapshot 方法

函数功能：接收盘后定价交易快照数据回调
适用运行模式：互联网模式
函数原型：

def OnMDAfterHourFixedPriceSnapshot(self, data, cnt)

参数	解释
data(out)	接收盘后定价交易快照数据 object， 参考 **MDAfterHourFixedPriceSnapshot**。
cnt(out)	数据个数

OnMDCSIIndexSnapshot 方法

函数功能：接收中证指数快照数据回调
适用运行模式：互联网模式
函数原型：

def OnMDCSIIndexSnapshot(self, data, cnt)

参数	解释
data(out)	接收中证指数快照数据 object，

	MDCSIIndexSnapshot。
cnt(out)	数据个数

OnMDCnIndexSnapshot 方法

函数功能：接收深交所国证指数快照数据回调

适用运行模式：互联网模式

函数原型：

def OnMDCnIndexSnapshot(self, data, cnt)

参数	解释
data(out)	接收深交所国证指数快照数据 object， 参考 **MDCnIndexSnapshot**。
cnt(out)	数据个数

4) 接收港股通数据方法

OnMDHKTRealtimeLimit 方法

函数功能：接收港股通实时额度推送数据回调

适用运行模式：互联网模式

函数原型：

def OnMDHKTRealtimeLimit(self, data, cnt)

参数	解释
data(out)	港股通实时额度推送数据 object， 参考 **MDHKTRealtimeLimit**。
cnt(out)	数据个数

OnMDHKTProductStatus 方法

函数功能：接收港股通产品状态快照数据回调

适用运行模式：互联网模式

函数原型：

def OnMDHKTProductStatus(self, data, cnt)

参数	解释
data(out)	接收港股通产品状态快照数据 object， 参考 **MDHKTProductStatus**。
cnt(out)	数据个数

OnMDHKTVCM 方法

函数功能：接收港股 VCM 推送数据回调

适用运行模式：互联网模式

函数原型：

def OnMDHKTVCM(self, data, cnt)

参数	解释

data(out)	接收港股 VCM 推送数据 object， 参考 **MDHKTVCM**。
cnt(out)	数据个数

5) 接收 K 线数据方法

OnKline 方法

函数功能：接收 k 线推送数据回调
适用运行模式：互联网模式
函数原型：

def OnKLine(self, data, cnt, kline_type)

参数	解释
data(out)	接收 k 线推送数据 object，参考 **MDKline**。
Kline_type(out)	标识 k 线数据类型
cnt(out)	数据个数

6) 接收快照衍生数据方法

OnSnapshotDerive 方法

函数功能：接收快照衍生数据推送数据回调
适用运行模式：互联网模式和托管机房模式
函数原型：

def OnSnapshotDerive(self,data,cnt)

参数	解释
data(out)	接收快照衍生推送数据 object，参考 **MDSnapshotDerive**。
cnt(out)	数据个数

7) 接收因子数据方法

OnFactor 方法

函数功能：接收因子数据推送数据回调
适用运行模式：互联网模式和托管机房模式
函数原型：

def OnFactor(self,data)

参数	解释
data(out)	接收因子推送数据 object，参考 **Factor**。

3.5.5 查询数据方法

QueryKline 方法

函数功能：查询 k 线数据接口

适用运行模式：互联网模式和托管机房模式

函数原型：

def IGMDApi_QueryKline(kline_spi, req_kline)

参数	解释
kline_spi(in)	tgw.IGMDKlineSpi 的派生类实例
req_kline(in)	设置 k 线查询信息

req_kline 属性定义如下：

名称	类型	说明
security_code	str	证券代码
market_type	int	市场类型
cq_flag	int	除权标志（0:不复权;1:向前复权;2:向后复权; 默认 0 不复权）
qj_flag	int	全价标志（债券）（0：净价，1：全价）
auto_complete	int	自动补全（0:不补齐，1：补齐 默 1）
cyc_type	int	数据周期,参考 **MDDatatype**
begin_date	int	开始日期（必须为 yyyyMMdd）
end_date	int	结束日期（必须为 yyyyMMdd）
begin_time	int	开始时间（格式 HHmm）
security_code	str	证券代码

返回数据说明：

参考 **MDKline**。

接口返回值：

返回错误码，参照 **ErrorCode**。

QuerySnapshot 方法

函数功能：查询快照数据接口

适用运行模式：互联网模式和托管机房模式

函数原型：

def IGMDApi_QuerySnapshot(snapshot_spi，req_snapshot)

参数	解释
snapshot_spi(in)	tgw.IGMDSnapshotSpi 的派生类实例
req_snapshot(in)	设置快照查询信息

req_snapshot 属性定义如下：

名称	类型	说明
security_code	str	证券代码
market_type	int	市场类型

date	int	日期（必须为 yyyyMMdd）
begin_time	int	开始时间（格式 HHmmssSSS）
end_time	int	结束时间（格式 HHmmssSSS）

返回数据说明：

参考 **MDSnapshotL1**、**MDSnapshotL2**、**MDIndexSnapshot**、**MDOptionSnapshot**、**MDHKTSnapshot**、**MDFutureSnapshot**。

QueryOrderQueue 方法

函数功能：查询委托队列数据接口

适用运行模式：托管机房模式

函数原型：

def IGMDApi_QueryOrderQueue(order_queue_spi，req_order_queue)

参数	解释
order_queue_spi(in)	tgw.IGMDOrderQueueSpi 的派生类实例
req_order_queue(in)	设置委托队列查询信息

req_order_queue 属性定义如下：

名称	类型	说明
security_code	str	证券代码
market_type	int	市场类型
date	int	日期（必须为 yyyyMMdd）
begin_time	int	开始时间（格式 HHmmssSSS）
end_time	int	结束时间（格式 HHmmssSSS）

返回数据说明：

参考 **MDOrderQueue**。

接口返回值：

返回错误码，参照 **ErrorCode**。

QueryTickExecution 方法

函数功能：查询逐笔成交数据接口

适用运行模式：托管机房模式

函数原型：

def IGMDApi_QueryTickExecution(tick_exec_spi，req_tick_execution)

参数	解释
tick_exec_spi(in)	tgw.IGMDTickExecutionSpi 的派生类实例
req_tick_execution(in)	设置逐笔成交查询信息

req_tick_execution 属性定义如下：

名称	类型	说明
security_code	str	证券代码
market_type	int	市场类型
date	int	日期（必须为 yyyyMMdd）
begin_time	int	开始时间（格式 HHmmssSSS）

end_time	int	结束时间（格式 HHmmssSSS）

返回数据说明：

参考 **MDTickExecution**。

接口返回值：

返回错误码，参照 **ErrorCode**。

QueryTickOrder 方法

函数功能：查询逐笔委托数据接口

适用运行模式：托管机房模式

函数原型：

def IGMDApi_QueryTickOrder(tick_order_spi，req_tick_order)

参数	解释
tick_order_spi(in)	tgw.IGMDTickOrderSpi 的派生类实例
req_tick_order(in)	设置逐笔委托查询信息

req_tick_order 属性定义如下：

名称	类型	说明
security_code	str	证券代码
market_type	int	市场类型
date	int	日期（必须为 yyyyMMdd）
begin_time	int	开始时间（格式 HHmmssSSS）
end_time	int	结束时间（格式 HHmmssSSS）

返回数据说明：

参考 **MDTickOrder**。

接口返回值：

返回错误码，参照 **ErrorCode**

QueryCodeTable 方法

函数功能：查询代码表数据接口

适用运行模式：托管机房模式

函数原型：

def IGMDApi_QueryCodeTable(code_table_spi)

参数	解释
code_table_spi(in)	tgw.IGMDCodeTableSpi 的派生类实例

返回数据说明：

参考 **MDCodeTable**。

接口返回值：

返回错误码，参照 **ErrorCode**

QuerySecuritiesInfo 方法

函数功能：查询证券代码信息数据接口

适用运行模式：互联网和托管机房模式

函数原型：

def IGMDApi_QuerySecuritiesInfo(code_table_spi，item，cnt)

参数	解释
code_table_spi(in)	tgw.IGMDSecuritiesInfoSpi 的派生类实例
item(in)	设置代码表查询信息
cnt(in)	item 入参个数

item 属性定义如下：

名称	类型	说明
market	int	市场
cnt	str	证券代码,为空表示查询所有代码

返回数据说明：

参考 **MDCodeTableRecord**。

接口返回值：

返回错误码，参照 **ErrorCode**

QueryExFactorTable 方法

函数功能：查询复权因子数据接口

适用运行模式：互联网和托管机房模式

函数原型：

def IGMDApi_QueryExFactorTable(ex_factor_spi，code)

参数	解释
ex_factor_spi (in)	tgw.IGMDExFactorSpi 的派生类实例
code(in)	复权因子查询信息，例如 “000001”

返回数据说明：

参考 **MDExFactorTable**。

接口返回值：

返回错误码，参照 **ErrorCode**

QueryFactor 方法

函数功能：查询加工因子数据接口

适用运行模式：互联网和托管机房模式

函数原型：

def IGMDApi_QueryFactor(factor_spi，req_factor)

参数	解释
factor_spi(in)	tgw.IGMDFactorSpi 的派生类实例
req_factor(in)	设置加工因子查询信息

Req_factor 属性定义如下：

名称	类型	说明
factor_type	str	因子父类型(英文)
factor_sub_type	str	因子子类型(英文)

factor_name	str	因子名称(英文)
begin_date	int	开始日期（yyyyMMdd 当前版本暂不支持跨天）
end_date	int	结束日期（yyyyMMdd 当前版本暂不支持跨天）
begin_time	int	开始时间（格式 HHmmssSSS） 取值说明:毫秒(HHmmssSSS)查询，开始时间和结束时间需同时为 HHmmssSSS; 开始（结束）时间 HHmmssSSS 输入示例：930（表示 0 点 0 分 0 秒 930 毫秒） 1031（表示 0 点 0 分 1 秒 31 毫秒） 93000000（表示 9 点 30 分 0 秒 0 毫秒）、 103001100（表示 10 点 30 分 1 秒 100 毫秒）
end_time	int	结束时间（格式 HHmmssSSS）
Key1	str	预留字段
Key2	str	预留字段

返回数据说明：

参考 **Factor**。

接口返回值：

返回错误码，参照 **ErrorCode**

SetThirdInfoParam 方法

函数功能：设置查询金融资讯数据接口

适用运行模式：互联网和托管机房模式

函数原型：

def IGMDApi_SetThirdInfoParam(task_id，key，value)

参数	解释
task_id(in)	金融资讯数据请求编号，调用 GetTaskId 获取
key(in)	金融资讯数据请求 json 的 key
value(in)	金融资讯数据请求 json 的 value

QueryThirdInfo 方法

函数功能：查询金融资讯数据接口

适用运行模式：互联网和托管机房模式

函数原型：

def IGMDApi_QueryThirdInfo(third_info_spi, task_id)

参数	解释
third_info_spi(in)	tgw.IGMDThirdInfoSpi 的派生类实例
task_id(in)	金融资讯数据请求编号 task_id

返回数据说明：

参考 **ThirdInfoData**。

接口返回值：

返回错误码，参照 **ErrorCode**

3.5.6 查询回调方法

1) IGMDKlineSpi 接口

接收 K 线数据。

OnMDKLine 方法

函数功能：接收 K 线数据回调接口

适用运行模式：互联网和托管机房模式

函数原型：

def OnMDKLine(self, klines, cnt, date_type)

参数	解释
klines(out)	K 线数据 object，数据解析方式见 demo，需要显式地调用 FreeMemory，
cnt(out)	K 线数据个数
date_type(out)	数据类型，取值参考数据字典 **MDDatatype** 中定义

返回数据说明：

参考 **MDKLine**。

OnStatus 方法

函数功能：接收 k 线数据状态回调接口

适用运行模式：互联网和托管机房模式

函数原型：

def OnStatus(self, status)

参数	解释
status(out)	k 线数据状态，需要显式地调用 FreeMemory

status 属性定义如下：

名称	类型	说明
error_code	int	状态为失败的错误码，参考 **ErrorCode**
error_msg_len	int	错误消息长度
error_msg	str	错误消息
rsp_union_status	object	快照、逐笔成交、逐笔委托、委托队列、k 线查询状态（具体类型通过 req_type 区分）
rsp_factor_status	object	加工因子查询状态
rsp_stockinfo_status	object	个股信息查询状态

rsp_union_status 属性定义如下

名称	类型	说明
req_type	int	请求类型，参考 MDDataType
market_type	int	市场，参考 MarketType
security_code	str	证券代码(000001)

rsp_factor_status 属性定义如下

名称	类型	说明
factor_type	str	因子父类型(英文)
factor_sub_type	str	因子子类型(英文)
factor_name	str	因子名称(英文)

rsp_stockinfo_status 属性定义如下

名称	类型	说明
code_table_item_cnt	int	SubCodeTableItem 总个数
codes	结构体数组类型	SubCodeTableItem 数组首地址

codes 属性定义如下

名称	类型	说明
market	int	市场类型,参考 MarketType，为 kNone 表示查询所有支持的市场(代码表目前只支持上交所、深交所与北交所)
security_code	str	证券代码,为空表示查询所有代码

rsp_thirdinfo_status 属性定义如下

名称	类型	说明
task_id	int	任务 id，格式为 MMDDHHmmSS+序列号（1~1000000）

2) IGMDSnapshotSpi 接口

接收现货、指数、港股通、期货快照数据。

OnMDSnapshotL2 方法

函数功能：现货快照 L2 回调接口

适用运行模式：托管机房模式

函数原型：

def OnMDSnapshotL2(self, snapshots, cnt)

参数	解释
snapshots(out)	快照数据 object，数据解析方式见 demo，需要显式地调用 FreeMemory
cnt(out)	快照数据个数

返回数据说明：

参考 MDSnapshotL2

OnMDSnapshotL1 方法

函数功能：现货快照 L1 回调接口

适用运行模式：互联网模式

函数原型：

def OnMDSnapshotL1(self, snapshots, cnt)

参数	解释
snapshots(out)	快照数据 object，数据解析方式见 demo， 需要显式地调用 FreeMemory
cnt(out)	快照数据个数

返回数据说明：

参考 **MDSnapshotL1**

OnMDIndexSnapshot 方法

函数功能：指数快照回调接口

适用运行模式：互联网模式

函数原型：

def OnMDIndexSnapshot(self, index_snapshots, cnt)

参数	解释
index_snapshots(out)	指数快照数据 object，数据解析方式见 demo， 需要显式地调用 FreeMemory
cnt(out)	指数快照数据个数

返回数据说明：

参考 **MDIndexSnapshot**

OnMDHKTSnapshot 方法

函数功能：港股通快照回调接口

适用运行模式：互联网模式和托管机房模式

函数原型：

def OnMDHKTSnapshot(self, hk_snapshots, cnt)

参数	解释
hkt_snapshots(out)	港股通快照数据 object，数据解析方式见 demo， 需要显式地调用 FreeMemory
cnt(out)	港股通快照数据个数

返回数据说明：

参考 **MDHKTSnapshot**

OnMDOptionSnapshot 方法

函数功能：期权快照回调接口

适用运行模式：互联网模式和托管机房模式

函数原型：

def OnMDOptionSnapshot(self, opt_snapshots, cnt)

参数	解释
opt_snapshots(out)	期权快照数据 object，数据解析方式见 demo，需要显式地调用 FreeMemory
cnt(out)	期权快照数据个数

返回数据说明：

参考 **MDOptionSnapshot**

OnMDFutureSnapshot 方法

函数功能：期货快照回调接口

适用运行模式：互联网模式和托管机房模式

函数原型：

def OnMDFutureSnapshot(self, future_snapshot, cnt)

参数	解释
future_ snapshot (out)	期货快照数据 object，数据解析方式见 demo，需要显式地调用 FreeMemory
cnt(out)	期货快照数据个数

返回数据说明：

参考 **MDFutureSnapshot**

OnStatus 方法

函数功能：接收快照数据状态回调接口。

适用运行模式：互联网模式和托管机房模式

函数原型：

def OnStatus(self, status)

参数	解释
status(out)	快照数据状态，需要显式地调用 FreeMemory

3) IGMDOrderQueueSpi 接口

接收委托队列数据

OnMDOrderQueue 方法

函数功能：委托队列回调接口

适用运行模式：托管机房模式

函数原型：

def OnMDOrderQueue(self, order_queues, cnt)

参数	解释
order_queues(out)	委托队列数据 object，数据解析方式见 demo，需要显式地调用 FreeMemory
cnt(out)	委托队列数据个数

返回数据说明：

参考 **MDOrderQueue**

OnStatus 方法

函数功能：接收委托队列数据状态回调接口。

适用运行模式：托管机房模式

函数原型：

def OnStatus(self, status)

参数	解释
status(out)	委托队列数据状态，需要显式地调用 FreeMemory

4) IGMDTickExecutionSpi 接口

接收逐笔成交数据。

OnMDTickExecution 方法

函数功能：逐笔成交回调接口

适用运行模式：托管机房模式

函数原型：

def OnMDTickExecution(self, tick_execs, cnt)

参数	解释
tick_execs(out)	逐笔成交数据 object，数据解析方式见 demo，需要显式地调用 FreeMemory
cnt(out)	逐笔成交数据个数

返回数据说明：

参考 **MDTickExecution**

OnStatus 方法

函数功能：接收逐笔成交数据状态回调接口。

适用运行模式：托管机房模式

函数原型：

def OnStatus(self, status)

参数	解释
status(out)	逐笔成交数据状态，需要显式地调用 FreeMemory

5) OnMDTickOrder 接口

逐笔委托回调接口。

OnMDTickOrder 方法

函数功能：逐笔成交回调接口

适用运行模式：托管机房模式

函数原型：

def OnMDTickOrder(self, tick_ orders, cnt)

参数	解释
tick_orders(out)	逐笔委托数据 object，数据解析方式见 demo，需要显式地调用 FreeMemory
cnt(out)	逐笔委托数据个数

返回数据说明：

参考 **MDTickOrder**。

OnStatus 方法

函数功能：接收逐笔委托数据状态回调接口。

适用运行模式：托管机房模式

函数原型：

def OnStatus(self, status)

参数	解释
status(out)	逐笔委托数据状态，需要显式地调用 FreeMemory

6) IGMDCodeTableSpi 接口

接收代码表数据。

OnMDCodeTable 方法

函数功能：代码表回调接口

适用运行模式：托管机房模式

函数原型：

def OnMDCodeTable(self, code_tables, cnt)

参数	解释
code_tables(out)	代码表数据 object，数据解析方式见 demo，需要显式地调用 FreeMemory
cnt(out)	代码表数据个数

返回数据说明：

参考 **MDCodeTable**

OnStatus 方法

函数功能：接收代码表数据状态回调接口。

适用运行模式：托管机房模式

函数原型：

def OnStatus(self, status)

参数	解释
status(out)	代码表数据状态，需要显式地调用 FreeMemory

7) IGMDSecuritiesInfoSpi 接口

接收证券代码信息数据。

OnMDSecuritiesInfo 方法

函数功能：代码表回调接口

适用运行模式：互联网和托管机房模式

函数原型：

def OnMDSecuritiesInfo(self, code_tables, cnt)

参数	解释
code_tables(out)	证券代码信息数据 object，数据解析方式见 demo，需要显式地调用 FreeMemory
cnt(out)	证券代码信息数据个数

返回数据说明：

参考 MDCodeTableRecord

OnStatus 方法

函数功能：接收证券代码信息数据状态回调接口。

适用运行模式：互联网和托管机房模式

函数原型：

def OnStatus(self, status)

参数	解释
status(out)	证券代码信息数据状态，需要显式地调用 FreeMemory

8) IGMDExFactorSpi 接口

接收复权因子数据

OnMDExFactor 方法

函数功能：复权因子回调接口

适用运行模式：互联网和托管机房模式

函数原型：

def OnMDExFactor(self, ex_factor_tables, cnt)

参数	解释
ex_factor_tables(out)	复权因子数据 object，数据解析方式见 demo，需要显式地调用 FreeMemory
cnt(out)	复权因子数据个数

返回数据说明：

参考 MDExFactorTable

OnStatus 方法

函数功能：接收复权因子信息数据状态回调接口。
适用运行模式：互联网和托管机房模式
函数原型：

def OnStatus(self, status)

参数	解释
status(out)	复权因子数据状态，需要显式地调用 FreeMemory

9) IGMDFactorSpi 接口

接收加工因子数据

OnFactor 方法

函数功能：加工因子数据回调接口
适用运行模式：互联网和托管机房模式
函数原型：

def OnFactor(self, factors, cnt)

参数	解释
factors(out)	加工因子数据 object，数据解析方式见 demo，需要显式地调用 FreeMemory
cnt(out)	加工因子数据个数

返回数据说明：
参考 Factor

OnStatus 方法

函数功能：接收加工因子信息数据状态回调接口。
函数原型：

def OnStatus(self, status)

参数	解释
status(out)	加工因子数据状态，需要显式地调用 FreeMemory

10) IGMDThirdInfoSpi 接口

接收金融资讯数据

OnThirdInfo 方法

函数功能：金融资讯数据回调接口
适用运行模式：互联网和托管机房模式
函数原型：

def OnThirdInfo(self, data, cnt)

参数	解释

data(out)	金融资讯数据 object，数据解析方式见 demo，需要显式地调用 FreeMemory
cnt(out)	金融资讯数据个数

返回数据说明：

参考 ThirdInfoData

OnStatus 方法

函数功能：接收金融资讯信息数据状态回调接口。

适用运行模式：互联网和托管机房模式

函数原型：

def OnStatus(self, status)

参数	解释
status(out)	金融资讯数据状态，需要显式地调用 FreeMemory

3.5.7 回放数据方法

ReplayKline 方法

函数功能：回放 k 线数据接口

适用运行模式：托管机房模式

函数原型：

def IGMDApi_ReplayKline(pSpi, req_kline)

参数	解释
pSpi(in)	twg.IGMDHistorySpi 的派生类实例
req_kline(in)	设置 k 线信息

req_kline 属性定义如下：

名称	类型	说明
cq_flag	int	除权标志（0:不复权;1:向前复权;2:向后复权；默认 0）
qj_flag	int	全价标志（债券）（0：净价，1：全价）
auto_complete	int	自动补全（0:不补齐，1：补齐 默 1）
cyc_type	int	数据周期,参考 MDDatatype
begin_date	int	开始日期（必须为 yyyyMMdd）
end_date	int	结束日期（必须为 yyyyMMdd）
begin_time	int	开始时间（格式 HHmm）
end_time	int	结束时间（格式 HHmm）
replay_speed	int	返回倍速（暂不可用）
task_id	int	任务 id(任务编号,例如:1) 调用 GetTaskId 获取
req_items	object	请求数组 item 头指针,不得为空

req_item_cnt	int	请求回放代码数量

req_items 属性定义如下：

名称	类型	说明
market	int	市场类型,参考 MarketType，为 kNone 表示查询所有支持的市场(代码表目前只支持上交所、深交所与北交所)
security_code	str	证券代码,为空表示查询所有代码

返回数据说明：

参考 **MDKline**。

接口返回值：

返回错误码，参照 **ErrorCode**

ReplayRequest 方法

函数功能：回放 k 线数据接口

适用运行模式：托管机房模式

函数原型：

def IGMDApi_ReplayRequest(pSpi, req_replay)

参数	解释
pSpi(in)	twg.IGMDHistorySpi 的派生类实例
req_replay(in)	设置快照，逐笔成交回放信息

req_replay 属性定义如下：

名称	类型	说明
md_data_type	int	回放数据类型,参考 MDDatatype
begin_date	int	开始日期（必须为 yyyyMMdd）
end_date	int	结束日期（必须为 yyyyMMdd）
begin_time	int	开始时间（格式 HHmmssSSS）
end_time	int	结束时间（格式 HHmmssSSS）
replay_speed	int	返回倍速（暂不可用）
task_id	int	任务 id(任务编号,例如:1) 调用 GetTaskId 获取
req_items	object	请求数组 item 头指针,不得为空
req_item_cnt	int	请求回放代码数量

req_items 属性定义如下：

名称	类型	说明
market	int	市场类型,参考 **MarketType**，为 kNone 表示查询所有支持的市场(代码表目前只支持上交所、深交所与北交所)
security_code	str	证券代码,为空表示查询所有代码

返回数据说明：

参考 **MDSnapshotL2**、**MDIndexSnapshot**、**MDHKTSnapshot**、**MDOptionSnapshot**、**MDTickExecution**。

接口返回值：

返回错误码，参照 **ErrorCode**

CancelTask 方法

函数功能：回放 k 线数据接口

适用运行模式：托管机房模式

函数原型：

def IGMDApi_CancelTask(task_id)

参数	解释
task_id	任务 id(任务编号，例如:1) 调用 GetTaskId 获取

接口返回值：

返回 task_id

3.5.8 回放回调方法

目前是 tgw 中回放数据的回调基类。使用 tgw 时需要继承该类，并将派生类的实例传递给 IGMDHistorySpi::ReplayKline 和 IGMDHistorySpi::ReplayRequest 函数，供 tgw 回调使用。为保证程序正确运行，必须保证该实例生命周期长于 tgw。

（1） 继承所需的 tgw. IGMDHistorySpi 类，在派生类中重载回调函数；

（2） 派生类实例化；

（3） 派生类的实例传递给 tgw. IGMDHistorySpi. ReplayKline 和 tgw. IGMDHistorySpi. ReplayRequest 函数，供 tgw 回调使用；

（4） 派生类的实例生命周期长于 tgw，即派生类实例化必须在 init 初始化之前；

1) OnMDKline 方法

函数功能：接收 K 线回放数据回调

适用运行模式：托管机房模式

函数原型：

def OnMDKline(self, task_id, klines, cnt, kline_type);

参数：

参数	解释
task_id(out)	回放任务 id
kline(out)	K 线数据 object，数据解析方式见 demo， 需要显式地调用 FreeMemory
cnt(out)	K 线个数
kline_type(out)	回放数据类型，取值参考数据字典中 MDDatatype 中定义

返回数据说明：

参考 **MDKLine**

2) OnMDSnapshot 方法

函数功能：接收快照回放数据回调

适用运行模式：托管机房模式

函数原型：

def OnMDSnapshot(self, task_id, snapshots, cnt)

参数：

参数	解释
task_id(out)	回放任务 id
snapshots(out)	快照数据 object，数据解析方式见 demo， 需要显式地调用 FreeMemory
cnt(out)	快照数据个数

返回数据说明：

参考 MDSnapshotL2

3) OnMDTickExecution 方法

函数功能：接收逐笔成交回放数据回调

适用运行模式：托管机房模式

函数原型：

def OnMDTickExecution(self, task_id, ticks, cnt)

参数：

参数	解释
task_id(out)	回放任务 id
ticks(out)	逐笔成交数据 object，数据解析方式见 demo， 需要显式地调用 FreeMemory
cnt(out)	逐笔成交数据个数

返回数据说明：

参考 MDTickExecution

4) OnRspTaskStatus 方法

函数功能：接收回放任务状态

适用运行模式：托管机房模式

函数原型：

def OnRspTaskStatus(self, task_id, task_status)

参数：

参数	解释
task_id(out)	回放任务 id
task_status(out)	回放任务状态

task_status 结构体定义：

名称	类型	说明
task_id	int	任务 id (任务编号)
status	int	任务状态　0: 完成, 1: 失败, 2: 处理中, 3: 已取消
process_rate	int	任务进度(暂不支持)
error_code	int	错误码
error_msg_len	int	错误信息长度

error_msg	str	错误信息

3.6 开发 Demo 示例

SDK 安装包中提供 tgw_demo.py 开发示例可供参考：

```
# -*- coding: utf-8 -*-
from tgw import tgw
import time
import signal


# 订阅 spi
class IAMDSpiApp(tgw.IGMDSpi):
    def OnLog(self, level, log, len):
        print("tgw log: ", "level: ", level, end='')
        print("          log:      ", log)

    def OnLogon(self, data):
        print("tgw Logon information:    : ")
        print("api_mode : ", data.api_mode)
        print("logon json : ", data.logon_json)
        tgw.IGMDApi_FreeMemory(data)

    def OnMDSnapshot(self, data, cnt):
        print("Receive MDSnapshot Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(data, i)))

        # print(tgw.Tools_GetInt64DataByIndex(data.bid_price, 0)) #取出委托档位买一档价格(最多十档)
        # print(tgw.Tools_GetInt64DataByIndex(data.bid_volume, 0)) #取出委托档位买一档的委托量(最多十档)
        tgw.IGMDApi_FreeMemory(data)

    def OnMDIndexSnapshot(self, data, cnt):
        print("Receive MDIndexSnapshot Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(data, i)))
        tgw.IGMDApi_FreeMemory(data)

    def OnMDOptionSnapshot(self, data, cnt):
        print("Receive MDOptionSnapshot Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(data, i)))
            # print(tgw.Tools_GetInt64DataByIndex(data.bid_price, 0)) #取出委托档位买一档价格(最多五档)
            # print(tgw.Tools_GetInt64DataByIndex(data.bid_volume, 0)) #取出委托档位买一档的委托量(最多五档)
        tgw.IGMDApi_FreeMemory(data)

    def OnMDHKTSnapshot(self, data, cnt):
        print("Receive MDHKTSnapshot Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(data, i)))
            # print(tgw.Tools_GetInt64DataByIndex(data.bid_price, 0)) #取出委托档位买一档价格(最多五档)
            # print(tgw.Tools_GetInt64DataByIndex(data.bid_volume, 0)) #取出委托档位买一档的委托量(最多五档)
        tgw.IGMDApi_FreeMemory(data)

    def OnMDAfterHourFixedPriceSnapshot(self, data, cnt):
        print("Receive MDAfterHourFixedPriceSnapshot Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(data, i)))
        tgw.IGMDApi_FreeMemory(data)

    def OnMDCSIIndexSnapshot(self, data, cnt):
        print("Receive MDCSIIndexSnapshot Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(data, i)))
        tgw.IGMDApi_FreeMemory(data)
```

```
    def OnMDCnIndexSnapshot(self, data, cnt):
        print("Receive MDCnIndexSnapshot Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(data, i)))
        tgw.IGMDApi_FreeMemory(data)

    def OnMDHKTRealtimeLimit(self, data, cnt):
        print("Receive MDHKTRealtimeLimit Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(data, i)))
        tgw.IGMDApi_FreeMemory(data)

    def OnMDHKTProductStatus(self, data, cnt):
        print("Receive MDHKTProductStatus Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(data, i)))
        tgw.IGMDApi_FreeMemory(data)

    def OnMDHKTVCM(self, data, cnt):
        print("Receive MDHKTVCM Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(data, i)))
        tgw.IGMDApi_FreeMemory(data)

    def OnMDFutureSnapshot(self, data, cnt):
        print("Receive MDHKTVCM Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(data, i)))
        tgw.IGMDApi_FreeMemory(data)

    def OnKLine(self, kline, cnt, kline_type):
        print("Receive KLine Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(kline, i)))
        tgw.IGMDApi_FreeMemory(kline)

    def OnSnapshotDerive(self, snapshot_derive, cnt):
        print("Receive MDSnapshotDerive Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(snapshot_derive, i)))
        tgw.IGMDApi_FreeMemory(snapshot_derive)

    def OnFactor(self, factor):
        print("Receive Factor Info: ")
        print(" json_buf is: ", factor.json_buf)
        print(" data_size is: ", factor.data_size)
        tgw.IGMDApi_FreeMemory(factor)


# 回放 spi
class IReplayApp(tgw.IGMDHistorySpi):
    def OnMDSnapshot(self, task_id, snapshots, cnt):
        print("Receive stock snapshot Info: ")
        tgw.Tools_WriteReplaySnapshot(snapshots, cnt)
        tgw.IGMDApi_FreeMemory(snapshots)

    def OnMDTickExecution(self, task_id, ticks, cnt):
        print("Receive MDTickExecution Info: ")
        tgw.Tools_WriteReplayTickExecution(ticks, cnt)
        tgw.IGMDApi_FreeMemory(ticks)

    def OnMDKline(self, task_id, klines, cnt, kline_type):
        print("Receive OnMDKline Info: ")
        tgw.Tools_WriteReplayKline(klines, cnt, kline_type)
        tgw.IGMDApi_FreeMemory(klines)

    def OnRspTaskStatus(self, task_id, task_status):
        print("task_id is: ", task_status.task_id)
        print("status is: ", task_status.status)
        print("error_code is: ", task_status.error_code)
```

```
# 快照查询 spi
class IQuerySnapshotSpi(tgw.IGMDSnapshotSpi):
    def OnMDSnapshotL1(self, snapshots, cnt):
        print("Receive stock snapshot l1 Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(snapshots, i)))
            # print(tgw.Tools_GetInt64DataByIndex(snapshots.bid_price, 0)) #取出委托档位买一档价格(最多五档)
            # print(tgw.Tools_GetInt64DataByIndex(snapshots.bid_volume, 0)) #取出委托档位买一档的委托量(最多五
档)
        tgw.IGMDApi_FreeMemory(snapshots)

    def OnMDSnapshotL2(self, snapshots, cnt):
        print("Receive stock snapshot l2 Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(snapshots, i)))
            # print(tgw.Tools_GetInt64DataByIndex(snapshots.bid_price, 0)) #取出委托档位买一档价格(最多十档)
            # print(tgw.Tools_GetInt64DataByIndex(snapshots.bid_volume, 0)) #取出委托档位买一档的委托量(最多十
档)
        tgw.IGMDApi_FreeMemory(snapshots)

    def OnMDIndexSnapshot(self, index_snapshots, cnt):
        print("Receive index snapshot Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(index_snapshots, i)))
        tgw.IGMDApi_FreeMemory(index_snapshots)

    def OnMDHKTSnapshot(self, hk_snapshots, cnt):
        print("Receive hk snapshot Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(hk_snapshots, i)))
            # print(tgw.Tools_GetInt64DataByIndex(hk_snapshots.bid_price, 0)) #取出委托档位买一档价格(最多五档)
            # print(tgw.Tools_GetInt64DataByIndex(hk_snapshots.bid_volume, 0)) #取出委托档位买一档的委托量(最
多五档)
        tgw.IGMDApi_FreeMemory(hk_snapshots)

    def OnMDOptionSnapshot(self, opt_snapshots, cnt):
        print("Receive option snapshot Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(opt_snapshots, i)))
            # print(tgw.Tools_GetInt64DataByIndex(opt_snapshots.bid_price, 0)) #取出委托档位买一档价格(最多五
档)
            # print(tgw.Tools_GetInt64DataByIndex(opt_snapshots.bid_volume, 0)) #取出委托档位买一档的委托量(最
多五档)
        tgw.IGMDApi_FreeMemory(opt_snapshots)

    def OnMDFutureSnapshot(self, future_ticks, cnt):
        print("Receive future snapshot Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(future_ticks, i)))
            # print(tgw.Tools_GetInt64DataByIndex(future_ticks.bid_price, 0)) #取出委托档位买一档价格(最多五档)
            # print(tgw.Tools_GetInt64DataByIndex(future_ticks.bid_volume, 0)) #取出委托档位买一档的委托量(最多
五档)
        tgw.IGMDApi_FreeMemory(future_ticks)

    def OnStatus(self, status):
        print("Receive QuerySnapshotSpi Status : ")
        print("    error_code    =    ", status.error_code)
        print("    error_msg_len    =    ", status.error_msg_len)
        print("    error_msg    =    ", status.error_msg)
        print("    req_type    =    ", status.rsp_union_status.req_type)
        print("    security_code    =    ", status.rsp_union_status.security_code)
        print("    market_type    =    ", status.rsp_union_status.market_type)
        tgw.IGMDApi_FreeMemory(status)


# 逐笔委托查询 spi
class IQueryTickOrderSpi(tgw.IGMDTickOrderSpi):
    def OnMDTickOrder(self, tick_orders, cnt):
        print("Receive QueryTickOrderSpi Info: ")
        # 序列化输出数据
```

```
            for i in range(cnt):
                print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(tick_orders, i)))
            tgw.IGMDApi_FreeMemory(tick_orders)

        def OnStatus(self, status):
            print("Receive QueryTickOrderSpi Status : ")
            print("    error_code    =    ", status.error_code)
            print("    error_msg_len    =    ", status.error_msg_len)
            print("    error_msg    =    ", status.error_msg)
            print("    req_type    =    ", status.rsp_union_status.req_type)
            print("    security_code    =    ", status.rsp_union_status.security_code)
            print("    market_type    =    ", status.rsp_union_status.market_type)
            tgw.IGMDApi_FreeMemory(status)


# 逐笔成交 spi
class IQueryTickExecutionSpi(tgw.IGMDTickExecutionSpi):
        def OnMDTickExecution(self, tick_execs, cnt):
            print("Receive QueryTickExecutionSpi Info: ")
            # 序列化输出数据
            for i in range(cnt):
                print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(tick_execs, i)))
            tgw.IGMDApi_FreeMemory(tick_execs)

        def OnStatus(self, status):
            print("Receive QueryTickExecutionSpi Status : ")
            print("    error_code    =    ", status.error_code)
            print("    error_msg_len    =    ", status.error_msg_len)
            print("    error_msg    =    ", status.error_msg)
            print("    req_type    =    ", status.rsp_union_status.req_type)
            print("    security_code    =    ", status.rsp_union_status.security_code)
            print("    market_type    =    ", status.rsp_union_status.market_type)
            tgw.IGMDApi_FreeMemory(status)


# 委托队列 spi
class IQueryOrderQueueSpi(tgw.IGMDOrderQueueSpi):
        def OnMDOrderQueue(self, order_queues, cnt):
            print("Receive QueryOrderQueueSpi Info: ")
            # 序列化输出数据
            for i in range(cnt):
                print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(order_queues, i)))
                # print(tgw.Tools_GetInt64DataByIndex(data.volume, 0)) #取出订单明细一档(最多五十档)
            tgw.IGMDApi_FreeMemory(order_queues)

        def OnStatus(self, status):
            print("Receive QueryOrderQueueSpi Status : ")
            print("    error_code    =    ", status.error_code)
            print("    error_msg_len    =    ", status.error_msg_len)
            print("    error_msg    =    ", status.error_msg)
            print("    req_type    =    ", status.rsp_union_status.req_type)
            print("    security_code    =    ", status.rsp_union_status.security_code)
            print("    market_type    =    ", status.rsp_union_status.market_type)
            tgw.IGMDApi_FreeMemory(status)


# k 线查询 spi
class IQueryKlineSpi(tgw.IGMDKlineSpi):
        def OnMDKLine(self, klines, cnt, kline_type):
            print("Receive QueryKlineSpi Info: ")
            # 序列化输出数据
            for i in range(cnt):
                print('-----------------')
                print(tgw.Tools_GetDataByIndex(klines, i))
                print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(klines, i)))

            tgw.IGMDApi_FreeMemory(klines)

        def OnStatus(self, status):
            print("Receive QueryKlineSpi Status : ")
            print("    error_code    =    ", status.error_code)
            print("    error_msg_len    =    ", status.error_msg_len)
            print("    error_msg    =    ", status.error_msg)
            print("    req_type    =    ", status.rsp_union_status.req_type)
            print("    security_code    =    ", status.rsp_union_status.security_code)
            print("    market_type    =    ", status.rsp_union_status.market_type)
            tgw.IGMDApi_FreeMemory(status)
```

```
# 代码表查询 spi
class IQueryCodeTableSpi(tgw.IGMDCodeTableSpi):
    def OnMDCodeTable(self, code_tables, cnt):
        print("Receive QueryCodeTableSpi Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(code_tables, i)))
        tgw.IGMDApi_FreeMemory(code_tables)

    def OnStatus(self, status):
        print("Receive QueryCodeTableSpi Status : ")
        print("    error_code    =    ", status.error_code)
        print("    error_msg_len    =    ", status.error_msg_len)
        print("    error_msg    =    ", status.error_msg)
        print("    req_type    =    ", status.rsp_union_status.req_type)
        print("    security_code    =    ", status.rsp_union_status.security_code)
        print("    market_type    =    ", status.rsp_union_status.market_type)
        tgw.IGMDApi_FreeMemory(status)


# 证券代码信息查询 spi
class IQuerySecuritiesInfoSpi(tgw.IGMDSecuritiesInfoSpi):
    def OnMDSecuritiesInfo(self, code_tables, cnt):
        print("Receive QuerySecuritiesInfoSpi Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(code_tables, i)))
        tgw.IGMDApi_FreeMemory(code_tables)

    def OnStatus(self, status):
        print("Receive QuerySecuritiesInfoSpi Status : ")
        print("    error_code    =    ", status.error_code)
        print("    error_msg_len    =    ", status.error_msg_len)
        print("    error_msg    =    ", status.error_msg)

        for i in range(status.rsp_stockinfo_status.code_table_item_cnt):
            item = tgw.Tools_GetSubCodeTableItemByIndex(status.rsp_stockinfo_status.codes, i)
            print("    security_code    =    ", item.security_code)
            print("    market_type    =    ", item.market)
        tgw.IGMDApi_FreeMemory(status)


# 复权因子表信息查询 spi
class IQueryExFactorSpi(tgw.IGMDExFactorSpi):
    def OnMDExFactor(self, ex_factor_tables, cnt):
        print("Receive QueryExFactorSpi Info: ")
        # 序列化输出数据
        for i in range(cnt):
            print(tgw.Tools_Serialize(tgw.Tools_GetDataByIndex(ex_factor_tables, i)))
        tgw.IGMDApi_FreeMemory(ex_factor_tables)

    def OnStatus(self, status):
        print("Receive QueryExFactorSpi Status : ")
        print("    error_code    =    ", status.error_code)
        print("    error_msg_len    =    ", status.error_msg_len)
        print("    error_msg    =    ", status.error_msg)
        print("    req_type    =    ", status.rsp_union_status.req_type)
        print("    security_code    =    ", status.rsp_union_status.security_code)
        print("    market_type    =    ", status.rsp_union_status.market_type)
        tgw.IGMDApi_FreeMemory(status)


# 加工因子查询 spi
class IQueryFactorSpi(tgw.IGMDFactorSpi):
    def OnFactor(self, factors, cnt):
        print("Receive QueryFactorSpi Info: ")
        for i in range(cnt):
            data = tgw.Tools_GetDataByIndex(factors, i)
            print(" json_buf is: ", data.json_buf)
            print(" data_size is: ", data.data_size)
        tgw.Tools_FreeMemory(factors, cnt)  # 这里的查询因子释放必须使用这个接口，否则会内存泄露

    def OnStatus(self, status):
        print("Receive QueryFactorSpi Status : ")
        print("    factor_type    =    ", status.rsp_factor_status.factor_type)
        print("    factor_sub_type    =    ", status.rsp_factor_status.factor_sub_type)
        print("    factor_name    =    ", status.rsp_factor_status.factor_name)
        print("    error_code    =    ", status.error_code)
        print("    error_msg_len    =    ", status.error_msg_len)
```

```
            print("    error_msg    =    ", status.error_msg)
            tgw.IGMDApi_FreeMemory(status)


# 资讯数据查询 spi
class IQueryThirdInfoSpi(tgw.IGMDThirdInfoSpi):
    def OnThirdInfo(self, data, cnt):
            print("Receive QueryThirdInfoSpi Info: ")
            for i in range(cnt):
                print(" task_id is: ", data.task_id)
                print(" data_size is: ", data.data_size)
                print(" json_data is: ", data.json_data)
            tgw.IGMDApi_FreeMemory(data)

    def OnStatus(self, status):
            print("Receive QueryThirdInfoSpi Status : ")
            print("    error_code    =    ", status.error_code)
            print("    error_msg_len    =    ", status.error_msg_len)
            print("    error_msg    =    ", status.error_msg)
            print("    task_id    =    ", status.rsp_thirdinfo_status.task_id)
            tgw.IGMDApi_FreeMemory(status)


def GetMarkert(m):
    if m == "All":
            return tgw.MarketType.kNone
    elif m == "sz":
            return tgw.MarketType.kSZSE
    elif m == "sh":
            return tgw.MarketType.kSSE
    elif m == "shfe":
            return tgw.MarketType.kSHFE
    elif m == "cffex":
            return tgw.MarketType.kCFFEX
    elif m == "dce":
            return tgw.MarketType.kDCE
    elif m == "czce":
            return tgw.MarketType.kCZCE
    elif m == "ine":
            return tgw.MarketType.kINE
    elif m == "neeq":
            return tgw.MarketType.kNEEQ
    elif m == "bk":
            return tgw.MarketType.kBK


def GetCategory(m):
    if m == "All":
            return tgw.VarietyCategory.kNone
    elif m == "Stock":
            return tgw.VarietyCategory.kStock
    elif m == "Fund":
            return tgw.VarietyCategory.kFund
    elif m == "Bond":
            return tgw.VarietyCategory.kBond
    elif m == "Option":
            return tgw.VarietyCategory.kOption
    elif m == "Index":
            return tgw.VarietyCategory.kIndex
    elif m == "HKT":
            return tgw.VarietyCategory.kHKT
    elif m == "FutureOption":
            return tgw.VarietyCategory.kFutureOption
    elif m == "CFETSRMB":
            return tgw.VarietyCategory.kCFETSRMB
    elif m == "HKEx":
            return tgw.VarietyCategory.kHKEx
    elif m == "Others":
            return tgw.VarietyCategory.kOthers


def GetSubType(m):
    if m == "All":
            return tgw.SubscribeDataType.kNone
    elif m == "MD1MinKline":
            return tgw.SubscribeDataType.k1MinKline
    elif m == "MD3MinKline":
            return tgw.SubscribeDataType.k3MinKline
    elif m == "MD5MinKline":
            return tgw.SubscribeDataType.k5MinKline
```

```
    elif m == "MD10MinKline":
        return tgw.SubscribeDataType.k10MinKline
    elif m == "MD15MinKline":
        return tgw.SubscribeDataType.k15MinKline
    elif m == "MD30MinKline":
        return tgw.SubscribeDataType.k30MinKline
    elif m == "MD60MinKline":
        return tgw.SubscribeDataType.k60MinKline
    elif m == "MD120MinKline":
        return tgw.SubscribeDataType.k120MinKline
    elif m == "MDSnapshotDerive":
        return tgw.SubscribeDataType.kSnapshotDerive
    elif m == "MDSnapshot":
        return tgw.SubscribeDataType.kSnapshot
    elif m == "MDIndexSnapshot":
        return tgw.SubscribeDataType.kIndexSnapshot
    elif m == "MDOptionSnapshot":
        return tgw.SubscribeDataType.kOptionSnapshot
    elif m == "MDHKTSnapshot":
        return tgw.SubscribeDataType.kHKTSnapshot
    elif m == "MDAfterHourFixedPriceSnapshot":
        return tgw.SubscribeDataType.kAfterHourFixedPriceSnapshot
    elif m == "MDCSIIndexSnapshot":
        return tgw.SubscribeDataType.kCSIIndexSnapshot
    elif m == "MDCnIndexSnapshot":
        return tgw.SubscribeDataType.kCnIndexSnapshot
    elif m == "MDHKTRealtimeLimit":
        return tgw.SubscribeDataType.kHKTRealtimeLimit
    elif m == "MDHKTProductStatus":
        return tgw.SubscribeDataType.kHKTProductStatus
    elif m == "MDHKTVCM":
        return tgw.SubscribeDataType.kHKTVCM
    elif m == "MDFutureSnapshot":
        return tgw.SubscribeDataType.kFutureSnapshot


def GetCqFlag(f):
    if f == "1f":
        return 1
    elif f == "0f":
        return 0
    elif f == "2f":
        return 2
    else:
        return 0


def GetKlineType(f):
    if f == "1m":
        return tgw.MDDatatype.k1KLine
    elif f == "3m":
        return tgw.MDDatatype.k3KLine
    elif f == "5m":
        return tgw.MDDatatype.k5KLine
    elif f == "10m":
        return tgw.MDDatatype.k10KLine
    elif f == "15m":
        return tgw.MDDatatype.k15KLine
    elif f == "30m":
        return tgw.MDDatatype.k30KLine
    elif f == "60m":
        return tgw.MDDatatype.k60KLine
    elif f == "120m":
        return tgw.MDDatatype.k120KLine
    elif f == "1d":
        return tgw.MDDatatype.kDayKline
    elif f == "1w":
        return tgw.MDDatatype.kWeekKline
    elif f == "1M":
        return tgw.MDDatatype.kMonthKline
    elif f == "1s":
        return tgw.MDDatatype.kSeasonKline
    elif f == "1y":
        return tgw.MDDatatype.kYearKline


def HandleUpdatePassword():
    # 更新密码
    up_password_req = tgw.UpdatePassWordReq()
    up_password_req.username = "amd"
```

```
    up_password_req.old_password = "123456"
    up_password_req.new_password = "123"
    ec = tgw.IGMDApi_UpdatePassWord(up_password_req)
    if ec != tgw.ErrorCode.kSuccess:
        print("UpdatePassWord failed, error code is : ", ec)
    else:
        print("UpdatePassWord success")


def DealSub():
    sub_item = tgw.SubscribeItem()
    sub_item.market = GetMarkert("All")
    sub_item.flag = GetSubType("All")
    sub_item.category_type = GetCategory("All")
    sub_item.security_code = "000001"
    tgw.IGMDApi_Subscribe(sub_item, 1)  # 订阅全市场，全类型，全品种，全代码

    sub_factor_item = tgw.SubFactorItem()
    sub_factor_item.factor_type = "all"
    sub_factor_item.factor_sub_type = "all"
    sub_factor_item.factor_name = "all"

    tgw.IGMDApi_SubFactor(sub_factor_item, 1)


def DealQuery():
    # 查询 k 线
    req_kline = tgw.ReqKline()
    req_kline.security_code = "000001"
    req_kline.market_type = GetMarkert("sz")
    req_kline.cq_flag = 0
    req_kline.auto_complete = 1
    req_kline.cyc_type = GetKlineType("1m")
    req_kline.begin_date = 20230128
    req_kline.end_date = 20230308
    req_kline.begin_time = 930
    req_kline.end_time = 1700
    tgw.IGMDApi_QueryKline(spi_kline, req_kline)

    # 查询快照
    req_tick = tgw.ReqDefault()
    req_tick.market_type = GetMarkert("sz")
    req_tick.date = 20210611
    req_tick.begin_time = 93000000
    req_tick.end_time = 170000000
    req_tick.security_code = "000001"
    tgw.IGMDApi_QuerySnapshot(spi_snap, req_tick)

    # 查询逐笔委托
    req_tick_order = tgw.ReqDefault()
    req_tick_order.begin_time = 93000000
    req_tick_order.date = 20210611
    req_tick_order.end_time = 170000000
    req_tick_order.security_code = "000001"
    req_tick_order.market_type = GetMarkert("sz")
    tgw.IGMDApi_QueryTickOrder(spi_tick_order, req_tick_order)

    # 查询逐笔成交
    req_tick_execution = tgw.ReqDefault()
    req_tick_execution.begin_time = 93000000
    req_tick_execution.date = 20210611
    req_tick_execution.end_time = 170000000
    req_tick_execution.security_code = "000001"
    req_tick_execution.market_type = GetMarkert("sz")
    tgw.IGMDApi_QueryTickExecution(spi_tick_exec, req_tick_execution)

    # 查询委托队列
    req_order_queue = tgw.ReqDefault()
    req_order_queue.begin_time = 93000000
    req_order_queue.date = 20210611
    req_order_queue.end_time = 170000000
    req_order_queue.security_code = "000001"
    req_order_queue.market_type = GetMarkert("sz")
    tgw.IGMDApi_QueryOrderQueue(spi_order_queue, req_order_queue)

    # 查询代码表
    tgw.IGMDApi_QueryCodeTable(spi_code_table)

    # 查询证券代码信息
```

```
    item = tgw.SubCodeTableItem()
    item.market = GetMarkert("sz")
    item.security_code = "000001"
    tgw.IGMDApi_QuerySecuritiesInfo(spi_secur_info, item, 1)

    # 查询复权因子信息表
    tgw.IGMDApi_QueryExFactorTable(spi_ex_factor, "000001")

    # 查询因子
    req_factor = tgw.ReqFactor()
    req_factor.factor_type = "A"
    req_factor.factor_sub_type = "B"
    req_factor.factor_name = "C"
    req_factor.begin_date = 20210611
    req_factor.end_date = 20210611
    req_factor.begin_time = 93000000
    req_factor.end_time = 170000000
    tgw.IGMDApi_QueryFactor(spi_factor, req_factor)

    # 查询三方咨询
    task_id = tgw.IGMDApi_GetTaskID()
    tgw.IGMDApi_SetThirdInfoParam(task_id, "function_id", "01")
    tgw.IGMDApi_SetThirdInfoParam(task_id, "WD_CODE", "0000001")
    tgw.IGMDApi_QueryThirdInfo(spi_third_info, task_id)


def DealReplay():
    # 回放逐笔成交
    history_item1 = tgw.ReqHistoryItem()
    history_item1.market = tgw.MarketType.kSZSE
    history_item1.security_code = "000001"

    history_item11 = tgw.ReqHistoryItem()
    history_item11.market = tgw.MarketType.kSZSE
    history_item11.security_code = "000002"

    req_replay = tgw.ReqReplay()
    req_replay.begin_date = 20211228
    req_replay.end_date = 20211228
    req_replay.begin_time = 91500000
    req_replay.end_time = 103100000
    req_replay.task_id = tgw.IGMDApi_GetTaskID()
    # 回放两只代码
    req_replay.req_item_cnt = 2
    req_replay.req_items = tgw.Tools_CreateReqHistoryItem(2)
    tgw.Tools_SetReqHistoryItem(req_replay.req_items, 0, history_item1)
    tgw.Tools_SetReqHistoryItem(req_replay.req_items, 1, history_item11)

    req_replay.md_data_type = tgw.MDDatatype.kTickExecution
    tgw.IGMDApi_ReplayRequest(spi_replay, req_replay)

    # 回放快照
    history_item2 = tgw.ReqHistoryItem()
    history_item2.market = tgw.MarketType.kSZSE
    history_item2.security_code = "000001"

    req_replay2 = tgw.ReqReplay()
    req_replay2.begin_date = 20211228
    req_replay2.end_date = 20211228
    req_replay2.begin_time = 91500000
    req_replay2.end_time = 103100000
    req_replay2.task_id = tgw.IGMDApi_GetTaskID()
    req_replay2.req_item_cnt = 1
    req_replay2.req_items = history_item2
    req_replay2.md_data_type = tgw.MDDatatype.kSnapshot
    tgw.IGMDApi_ReplayRequest(spi_replay, req_replay2)

    # 回放 k 线
    history_item3 = tgw.ReqHistoryItem()
    history_item3.market = tgw.MarketType.kSZSE
    history_item3.security_code = "000001"

    req_k = tgw.ReqReplayKline()
    req_k.begin_date = 20211228
    req_k.begin_time = 930
    req_k.cq_flag = 0
    req_k.cyc_type = 1
    req_k.auto_complete = 1
    req_k.end_date = 20211228
```

```
    req_k.end_time = 1032
    req_k.task_id = tgw.IGMDApi_GetTaskID()
    req_k.req_item_cnt = 1
    req_k.req_items = history_item3
    tgw.IGMDApi_ReplayKline(spi_replay, req_k)


def GetChannelMode(mode):
    ret_mode = 0
    if mode == "QTCP":
        ret_mode = ret_mode | tgw.ColocatChannelMode.kQTCP
    elif mode == "TCP":
        ret_mode = ret_mode | tgw.ColocatChannelMode.kTCP
    elif mode == "RTCP":
        ret_mode = ret_mode | tgw.ColocatChannelMode.kRTCP
    return ret_mode


def Init():
    cfg = tgw.Cfg()

    # 服务器地址配置
    cfg.server_vip = "10.4.47.21"
    cfg.server_port = 8600
    # 用户登录账号配置
    cfg.username = "test000111"  # 账号
    cfg.password = "123456"  # 密码
    # 运行模式配置
    # api_mode = tgw.ApiMode.kColocationMode # 设置 api 模式 托管机房模式
    api_mode = tgw.ApiMode.kInternetMode  # 设置 api 模式 互联网模式
    if (api_mode == tgw.ApiMode.kColocationMode):
        cfg.coloca_cfg.channel_mode = tgw.ColocatChannelMode.kQTCP  # tcp 查询模式
        cfg.coloca_cfg.qtcp_channel_thread = 2
        cfg.coloca_cfg.qtcp_max_req_cnt = 1000

    # 初始化返回错误码，完成登录验证、运行模式设置、传实例到订阅方法三个功能
    error_code = tgw.IGMDApi_Init(spi, cfg, api_mode)
    # 如初始化失败，退出流程
    if error_code != tgw.ErrorCode.kSuccess:
        print("Init tgw failed")
        tgw.IGMDApi_Release()
        exit(-1)


def CtrlC(signum, frame):
    print("bey bey")
    global g_is_running
    g_is_running = False


if __name__ == "__main__":
    signal.signal(signal.SIGINT, CtrlC)
    signal.signal(signal.SIGTERM, CtrlC)

    g_is_running = True
    # ---------订阅 spi 实例---------
    spi = IAMDSpiApp()

    # ---------查询 spi 实例---------
    # k 线查询 spi 实例
    spi_kline = IQueryKlineSpi()
    # 快照查询 spi 实例
    spi_snap = IQuerySnapshotSpi()
    # 逐笔委托查询 spi 实例
    spi_tick_order = IQueryTickOrderSpi()
    # 逐笔成交 spi 实例
    spi_tick_exec = IQueryTickExecutionSpi()
    # 委托队列 spi 实例
    spi_order_queue = IQueryOrderQueueSpi()
    # 代码表查询 spi 实例
    spi_code_table = IQueryCodeTableSpi()
    # 证券代码信息查询 spi 实例
    spi_secur_info = IQuerySecuritiesInfoSpi()
    # 复权因子表信息查询 spi 实例
    spi_ex_factor = IQueryExFactorSpi()
    # 加工因子查询 spi 实例
    spi_factor = IQueryFactorSpi()
```

```
    # 资讯数据查询 spi 实例
    spi_third_info = IQueryThirdInfoSpi()

    # ---------回放 spi 实例---------
    spi_replay = IReplayApp()

    Init()

    time.sleep(2)
    # 修改密码
    HandleUpdatePassword()
    # 订阅接口
    DealSub()
    # 查询接口
    DealQuery()
    # 回放接口
    DealReplay()
    while True:
        try:
            if g_is_running != True:
                break
        except Exception as e:
            print(str(e))
        time.sleep(1)

    tgw.IGMDApi_Release()
```

3.7 演示示例（获取和保存数据）

本演示示例以 python3.6 版本为例。SDK 提供 tgw_test 标准演示程序，通过 tgw.json 配置订阅信息后，运行脚本 run_test36.sh 获取订阅数据并通过所配置的路径，落地 csv 文件供查看，以下是程序的命令参数说明和示例。

tgw_test 程序用于指引用户了解 API 工作流程和使用步骤，也供开发人员以及测试人员进行功能验证和测试分析。

3.7.1 修改参数

etc/test.json 下面的参数解析如下

```
{
    "ApiMode": 1,                       //API 模式，1 托管机房，2 互联网
    "CsvFileDir": "./data",             //csv 落地文件路径
    "ServerVip": "127.0.0.1",           // Nginx 地址
    "ServerPort": 9200,                 // Nginx 端口
    "UserName": "amd",                  // 用户名
    "Password": "123456",               // 密码
    "ColocQTcpMaxReqCnt": 100,          // 托管机房 tcp 查询通道最大请求数
    "ColocQTcpChannelThread": 2,        // 托管机房 tcp 查询通道消费线程数
    "ColocQTcpReqTimeOut": 1,           // 托管机房 tcp 查询通道查询超时时间设置
    "ColocChannelMode": ["TCP"],        // 支持：TCP、QTCP、RTCP 模式，可进行组合
                                           例如["QTCP","TCP"] 表示同时开启 tcp 和 qtcp
    "Subscribe":[       // 订阅配置
        {
            "Enable": false,            // 开关
            "security_code":"All",      // 全部代码
```

```
            "market_type":"All",        // 全部市场
            "data_type":"All",          // 全部类型
            "category_type": "All"      // 全部品种类别
        },
        {
            "Enable": false,                        //开关
            "security_code":"000002",               //代码
            "market_type":"sz",                     //市场，取值如下 sz(深交所)，bkzs(板块)，
                                                     sh(上交所)，shfe(上期所)，cffex(中 金所) dce(大商所)，
                                                     czce(郑商所)，ine(上海国际能源交易中心)
            "data_type":"MDSnapshotDerive",   // 类型，取值如下：MD1MinKline, MD3MinKline, MD5MinKline,
                                                 MD10MinKline, MD15MinKline, MD30MinKline,
                                                 MD60MinKline, MD120MinKline, MDSnapshotDerive,
                                                 MDSnapshot, MDIndexSnapshot, MDOptionSnapshot,
                                                 MDHKTSnapshot, MDAfterHourFixedPriceSnapshot,
                                                 MDCSIIndexSnapshot, MDCnIndexSnapshot,
                                                 MDHKTRealtimeLimit, MDHKTProductStatus, MDHKTVCM,
                                                 MDFutureSnapshot
            "category_type": "All"              // 品种类别，取值如下 // Stock, Fund, Bond, Option,
                                                Index, HKT, FutureOption, CFETSRMB, HKEx, Others
        }
    ],
    "SubFactor":[      //订阅因子配置
        {
            "Enable": false,          // 开关
            "factor_type":"all",        // 因子类型
            "factor_sub_type":"all",    // 因子子类型
            "factor_name":"all"         // 因子名称
        }
    ],
    "QueryDefault":[                    // 查询逐笔委托，快照，逐笔成交，委托队列配置
        {
            "Enable": true,           // 开关
            "type": "GetOrder",       // 查询类型，取值如下：快照：GetSnapshot；逐笔成交：GetTickExecution，
                                        委托队列：GetOrderQueue
            "security_code":"000001.sz",   // 代码
            "date": 20220419,         // 日期
            "begin_time": 80000000, // 开始时间
            "end_time": 113000000   // 结束时间
        },
        {
            "Enable": true,
            "type": "GetSnapshot",
            "security_code":"000001.sz",
```

```
            "date": 20220419,
            "begin_time": 80000000,
            "end_time": 113000000
        },
        {
            "Enable": true,
            "type": "GetTickExecution",
            "security_code":"000001.sz",
            "date": 20220419,
            "begin_time": 80000000,
            "end_time": 113000000
        },
        {
            "Enable": true,
            "type": "GetOrderQueue",
            "security_code":"000001.sz",
            "date": 20220419,
            "begin_time": 80000000,
            "end_time": 113000000
        }
    ],
    "QueryKLine":[      // 查询 k 线配置
        {
            "Enable": true,      //开关型
            "type":"10m",      // k 线类型：1m、3m、5m、10m、15m、30m、60m、120m、1d、1w、1M、 1s、1
            "auto_complete": 1,      // 自动补全：0：不补全，1：补全
            "cq_flag": "2f",            // 复权：（0：不复权;1：向前复权;2：向后复权）
            "security_code":"000003.sz",      //代 码
            "begin_date": 20220419,      // 开始日期
            "end_date": 20220419,          // 结束日期
            "begin_time": 930,              // 开始时间
            "end_time": 1130                // 结束时间
        }
    ],
    "QuerySecurInfo":[      // 查询证券代码信息配
        {
            "Enable": true,
            "security_code":"000001.sz"
        }
    ],
    "QueryCodeTable":{      // 查询代码
        "Enable": true
    },
    "QueryExFactorTable":[      // 查询复权因子表信息配置
```

```
        {
            "Enable": true,
            "security_code":"000001"
        }
    ],
    "QueryFactor":[      //查询加工因子配置
        {
            "Enable": true,
            "factor_type":"A",              // 因子类型
            "factor_sub_type":"B",      // 因子子类型
            "factor_name":"C",              // 因子名称
            "begin_date": 20220419,     // 开始日期
            "end_date": 20220419,        // 结束日期
            "begin_time": 80000000,     // 开始时间
            "end_time": 113000000       // 结束时间
        }
    ],
    "QueryThirdInfo":       // 查询金融资讯数据配置
    {
        "Enable": true,
        "item":[       // 查询条件
            {
                "key" : "function_id",       // key
                "value" : "1"       // value
            },
            {
                "key" : "name",
                "value": "xiaoming"
            }
        ]
    },
    "ReplaySnapshot":[       // 回放快照配置
        {
            "Enable": false,
            "security_code":["000001.sz"],
            "begin_date": 20220419,
            "end_date": 20220419,
            "begin_time": 80000000,
            "end_time": 113000000
        }
    ],
    "ReplayTickExection":[       // 回放逐笔成交配置
        {
            "Enable": false,
```

```
                "security_code":["000001.sz"],
                "begin_date": 20220419,
                "end_date": 20220419,
                "begin_time": 80000000,
                "end_time": 113000000
            }
        ],
        "ReplayKLine":[      // 回放 k 线配置
            {
                "Enable": false,
                "type":"10m",       // k 线类型
                "auto_complete": 1,     // 自动补全
                "cq_flag": "2f",       // 复权
                "security_code":["000003.sz"],
                "begin_date": 20220419,
                "end_date": 20220419,
                "begin_time": 930,
                "end_time": 1130
            }
        ]
}
```

3.7.2 运行启动脚本

- Linux 版本 SDK 运行前需要启动位于 SDK 根目录 python3.6 下的 run_test36.sh 脚本。终端中输入./ run_test36.sh 启动运行。
- Windows 版本 SDK 运行前需要启位于 SDK 根目录 python3.6 下的 run_test36.bat 脚本。cmd 命令框中输入 run_test36.bat 启动运行。

3.7.3 查看订阅数据 csv 文件

在设置的 csv 路径下查看订阅接收到的行情数据文件，tgw_test 按照数据类型生成单独的 csv 文件。

```
-rw-rw-r--. 1 guosx guosx     314 7月    5 09:56 10MinKline.csv
-rw-rw-r--. 1 guosx guosx       0 7月    5 09:56 120MinKline.csv
-rw-rw-r--. 1 guosx guosx     226 7月    5 09:56 15MinKline.csv
-rw-rw-r--. 1 guosx guosx    6903 7月    5 09:56 1MinKline.csv
-rw-rw-r--. 1 guosx guosx       0 7月    5 09:56 30MinKline.csv
-rw-rw-r--. 1 guosx guosx     842 7月    5 09:56 3MinKline.csv
-rw-rw-r--. 1 guosx guosx     578 7月    5 09:56 5MinKline.csv
-rw-rw-r--. 1 guosx guosx       0 7月    5 09:56 60MinKline.csv
-rw-rw-r--. 1 guosx guosx    4472 7月    5 09:56 AfterHourFixPriceSnapshot.csv
-rw-rw-r--. 1 guosx guosx    2288 7月    5 09:56 CnIndexSnapshot.csv
-rw-rw-r--. 1 guosx guosx       0 7月    5 09:56 CSIIndexSnapshot.csv
-rw-rw-r--. 1 guosx guosx       0 7月    5 09:56 Factor.csv
-rw-rw-r--. 1 guosx guosx       0 7月    5 09:56 HKTProduStatus.csv
-rw-rw-r--. 1 guosx guosx     196 7月    5 09:56 HKTRealLimit.csv
-rw-rw-r--. 1 guosx guosx   61496 7月    5 09:56 HKTSnapshot.csv
-rw-rw-r--. 1 guosx guosx       0 7月    5 09:56 HKTVCM.csv
-rw-rw-r--. 1 guosx guosx  122964 7月    5 09:56 IndexSnapshot.csv
-rw-rw-r--. 1 guosx guosx  735302 7月    5 09:56 OptionSnapshot.csv
-rw-rw-r--. 1 guosx guosx 1315455 7月    5 09:56 SnapshotDerive.csv
-rw-rw-r--. 1 guosx guosx 2981066 7月    5 09:56 StockSnapshot.csv
[guosx@localhost subscribe]$ pwd
/home/guosx/4.0.0/mdga4.0.0/cplusplus/csvfile/subscribe
[guosx@localhost subscribe]$
```

3.8 开发注意事项

1) 初始化成功则可执行订阅或者取消订阅操作或进行查询，初始化失败则调用 Release 函数释放资源后退出。
2) 订阅时调用接口 Subscribe 或 SubFactor，若账号不限制订阅数则可以填写成订阅所有市场、类型、证券代码。若有限制则只能单市场，单证券代码，数据类型不受限制），需要连续订阅多只，则可参照 demo 示例，使用数组入参的方式。
3) 订阅成功后，对应数据将回调到提供的接口，如 OnMDSnapshot()方法将接收到现货快照数据，具体数据类型对应不同接口，用户可选择实现或者不处理，在完成对应处理后，必须调用 FreeMemory()方法释放传入的指针内存，否则将造成内存泄漏。
4) 取消订阅的流程、规则，与订阅流程、规则类似，取消后将不再接收该类型的数据，取消即时生效但数据为异步回调，取消前已收到的数据仍然会接收。
5) 查询接口使用时请先判断返回错误码，若不为成功则出参无数据，若成功则根据 QueryData 中数据类型进行指定处理（即将指针转成对应的结构体进行解析处理，查询和推送结构相同），使用后需要对 QueryData 中的 msg 指针调用 FreeMemory()方法释放内存，以避免内存泄漏
6) 调用查询接口时有最大并发量限制，超过最大并发量则等待部分任务完成，未超过则发起查询请求等待结果返回。
7) 查询成功则针对数据类型进行指定的指针转换解析数据，使用完成后调用 FreeMemory 释放内存，否则将造成内存泄漏。
8) 登录和初始化不冲突，初始化成功并不一定登陆成功，登录失败将有对应错误日志提示（OnLog 函数回调时），用户可参照下面的错误码 JsonStatus，排查对应原因

返回值 status 定义：

错误码	说明
-100	失败
-99	ums 鉴权超时
-98	ums 鉴权超过最大连接数
-97	ums 鉴权用户不存在

-96	非法请求
-95	账号/密码错误
-94	账号过期
-93	订阅失败
-92	取消订阅失败
-91	查询请求超过最大请求数
-90	该数据类型无查询权限
-89	查询请求超时
-88	未知的错误类型
-87	Token 校验失败
-86	非白名单用户 ip 或者 mac
-85	服务端 redis 信息写入失败
-84	周流量使用达到上限
0	成功

9) 编译运行当前 demo 没有问题，但是如果用户移动了默认的目录层级或者自行引入了开发所需的其他三方库，可能造成编译运行的问题，用户需自行解决。
10) 如果运行报证书认证失败，请运行根目录下设置环境变量脚本 tgw_install。
11) 如启动脚本报错 command not found，需要根据本身环境配置更改脚本的执行命令，原因如下：如果安装的 python3.6 版本对应执行命令仍叫 python，将会报错，因为脚本执行命令使用了带后缀名称 python3.6，主要是为了防止环境中同时存在同名的其他 python 版本，导致运行时未正确使用对应 3.6 版本出现未知问题。

4. 公共数据字典

所有公共数据字典的的定义都在 **C++/include/tgw_datatype.h** 下面定义。

4.1 事件级别(EventLevel)

事件级别	数值	说明
kInfo	=1	普通事件
kWarn	=2	告警事件
kError	=3	错误事件，比较严重,需要介入处理

4.2 错误码(ErrorCode)

错误码	数值	说明
kFailure	-100	失败
kUnInited	-99	未初始化
kNullSpi	-98	空指针
kParamIllegal	-97	参数非法
kNetError	-96	网络异常
kPermissionError	-95	数据无权限
kLogonFailed	-94	未登录
kAllocateMemoryFailed	-93	分配内存失败
kChannelError	-92	通道错误
kOverLoad	-91	hqs 任务队列溢出
kLogoned	-90	账号已登录
kHqsError	-89	HQS 系统错误
kNonQueryTimePeriod	-88	非查询时间段(非查询时间段不支持查询)
kDbAndCodeTableNoCode	-87	数据库和代码表中没有指定的代码
kIllegalMode	-86	api 模式非法
kThreadBusy	-85	超过最大可用线程资源
kParseDataError	-84	数据解析出错
kTimeout	-83	获取数据超时
kFlowOverLimit	-82	周流量耗尽
kCodeTableCacheNotAvailable	-81	代码表缓存不可用
kOverMaxSubLimit	-80	超过最大订阅限制
kLostConnection	-79	丢失连接
kOverMaxQueryLimit	-78	超过最大查询数（含代码表）
kFunctionIdNull	-77	金融资讯数据查询未设置功能号
kDataEmpty	-76	数据为空

kUserNotExist	-75	用户不存在
kVerifyFailure	-74	账号/密码错误
kApiInterfaceUsing	-73	api 接口不能同时多次调用
kSuccess	0	成功

4.3 回放请求任务状态(HistoryTaskStatus)

数据类型	数据名称	数值	说明
int	kSuccess	0	任务成功完成
int	kFailed	1	任务执行失败
int	kTaskCancel	2	任务取消
int	kTaskWaiting	3	任务执行中
int	kTaskTimeOut	4	回放任务超时

4.4 数据类型(MDDatatype)

类型名	数值	说明
k1KLine	=10000	1 分钟 K 线
k3KLine	=10001	3 分钟 K 线
k5KLine	=10002	5 分钟 K 线
k10KLine	=10003	10 分钟 K 线
k15KLine	=10004	15 分钟 K 线
k30KLine	=10005	30 分钟 K 线
k60KLine	=10006	60 分钟 K 线
k120KLine	=10007	120 分钟 K 线
kDayKline	=10008	日 K 线
kWeekKline	=10009	周 K 线
kMonthKline	=10010	月 K 线
kSeasonKline	=10011	季 K 线
kYearKline	=10012	年 K 线
kTickExecution	=10013	逐笔成交
kSnapshot	=10014	快照

4.5 日志输出级别(LogLevel)

日志输出级别	数值	说明
kTrace	=0	跟踪级别日志
kDebug	=1	调试级别日志
kInfo	=2	普通级别日志

kWarn	=3	警告级别日志
kError	=4	错误级别日志
kFatal	=5	致命级别日志

4.6 市场类型定义(MarketType)

市场类型	数值	说明
kNone	=0	表示全市场
kNEEQ	=2	北交所
kSHFE	=3	上期所
kCFFEX	=4	中金所
kDCE	=5	大商所
kCZCE	=6	郑商所
kINE	=7	上海国际能源交易中心
kSSE	=101	上交所
kSZSE	=102	深交所
kHKEx	=103	港交所
kBK	=201	板块
kMax	=255	市场类型最大值

4.7 API 通道模式(ApiMode)

数据类型	数据名称	数值	说明
int	kColocationMode	1	托管机房模式
int	kInternetMode	2	互联网模式

4.8 托管机房模式通道(ColocatChannelMode)

数据类型	数据名称	数值	说明
int	kTCP	0x0000000200000000	TCP 推送获取数据
int	kQTCP	0x0000000800000000	TCP 查询获取数据
int	kRTCP	0x0000001000000000	TCP 回放获取数据

4.9 订阅数据类型(SubscribeDataType)

数据类型	数据名称	数值	说明	适用模式
int	kNone	0	订阅全部数据	托管机房模式

				互联网模式
int	k1MinKline	1	订阅 1 分钟 k 线数据	托管机房模式 互联网模式
int	k3MinKline	2	订阅 3 分钟 k 线数据	托管机房模式 互联网模式
int	k5MinKline	3	订阅 5 分钟 k 线数据	托管机房模式 互联网模式
int	k10MinKline	4	订阅 10 分钟 k 线数据	托管机房模式 互联网模式
int	k15MinKline	5	订阅 15 分钟 k 线数据	托管机房模式 互联网模式
int	k30MinKline	6	订阅 30 分钟 k 线数据	托管机房模式 互联网模式
int	k60MinKline	7	订阅 60 分钟 k 线数据	托管机房模式 互联网模式
int	k120MinKline	8	订阅 120 分钟 k 线数据	托管机房模式 互联网模式
int	kSnapshotDerive	9	订阅快照衍生数据	托管机房模式 互联网模式
int	kSnapshot	10	订阅现货快照数据	互联网模式
int	kOptionSnapshot	11	订阅期权快照数据	互联网模式
int	kHKTSnapshot	12	订阅港股快照数据	互联网模式
int	kIndexSnapshot	13	订阅指数快照数据	互联网模式
int	kAfterHourFixedPriceSnapshot	14	订阅盘后快照数据	互联网模式
int	kCSIIndexSnapshot	15	订阅中证指数数据	互联网模式
int	kCnIndexSnapshot	16	订阅国证指数快照数据	互联网模式
int	kHKTRealtimeLimit	17	订阅港股通实时额度数据	互联网模式
int	kHKTProductStatus	18	订阅港股通产品状态数据	互联网模式
int	kHKTVCM	19	订阅港股 VCM 数据	互联网模式
int	kFutureSnapshot	20	订阅期货数据	互联网模式

4.10 品种类型定义(VarietyCategory)

数据类型	数据名称	数值	说明
int	kNone	0	None
int	kStock	1	股票
int	kFund	2	基金
int	kBond	3	债券
int	kOption	4	期权
int	kIndex	5	指数
int	kHKT	6	港股通

int	kFutureOption	7	期货期权
int	kCFETSRMB	8	银行间本币交易产品
int	kHKEx	9	港股
int	kOthers	255	其他

5. 行情数据字典

所有行情数据字典的的定义都在 C++/include/tgw_struct.h 下面定义。

5.1 K 线(MDKLine)

数据类型	字段名称	说明
int	market_type	市场类型
str	security_code	证券代码
int	orig_time	时间（YYYYMMDDHHMMSSsss)
int	kline_time	k 线时间
int	open_price	开盘价，实际值需除以 1000000
int	high_price	最高价，实际值需除以 1000000
int	low_price	最低价，实际值需除以 1000000
int	close_price	最新价，实际值需除以 1000000
int	volume_trade	成交量，无倍数放大
int	value_trade	成交金额，无倍数放大

5.2 现货衍生(MDSnapshotDerive)

数据类型	字段名称	说明
int	market_type	市场类型
str	security_code	证券代码
int	orig_time	时间（为 YYYYMMDDHHMMSSsss）
uint	average_price	均价，实际值需除以 1000000
uint	circulation_value	流通市值，实际值需除以 1000000
uint	total_value	总市值，实际值需除以 1000000
uint	initiative_buy_volume	外盘，实际值需除以 100
uint	initiative_sell_volume	内盘，实际值需除以 100
ustr	turnover_rate	换手率，实际值需除以 100000
str	volume_ratio	量比，实际值需除以 100000
str	ask_bid_ratio	委比，实际值需除以 100000
ustr	amplitude	振幅，实际值需除以 100000
str	PE_static	静态市盈率，实际值需除以 100
str	PE_dynamic	动态市盈率，实际值需除以 100
str	PE_TTM	最近 4 季度滚动市盈率，实际值需除以 100
str	PB	市净率，实际值需除以 100
int	entrustment_diff	委差

str	initiative_flag	内外盘标记(B/S)

5.3 加工因子(Factor)

数据类型	字段名称	说明
ustr	data_size	json 数据的大小
str*	json_buf	json 结构

5.4 L1 现货快照(MDSnapshotL1)

数据类型	字段名称	说明
int	market_type	市场类型
str	security_code	证券代码
int	variety_category	品种类别（参照描述）
int	orig_time	交易所行情数据时间（YYYYMMDDHHMMSSsss）
str	trading_phase_code	交易阶段代码
int	pre_close_price	昨收价，实际值需除以 1000000
int	open_price	开盘价，实际值需除以 1000000
int	high_price	最高价，实际值需除以 1000000
int	low_price	最低价，实际值需除以 1000000
int	last_price	最新价，实际值需除以 1000000
int	close_price	收盘价，实际值需除以 1000000，北交所为 0
Object	bid_price	tgw.Tools_GetInt64DataByIndex(data.bid_price, 0) #取出委托档位买一档价格(最多 5 档) 申买价，实际值需除以 1000000
Object	bid_volume	tgw.Tools_GetInt64DataByIndex(data. bid_volume, 0) #取出委托档位买一档量(最多 5 档) 申买量，实际值需除以 100
Object	offer_price	tgw.Tools_GetInt64DataByIndex(data. offer_price, 0) #取出委托档位卖一档价格(最多 5 档) 申卖价，实际值需除以 1000000
Object	offer_volume	tgw.Tools_GetInt64DataByIndex(data. offer_volume, 0) #取出委托档位卖一档量(最多 5 档) 申卖量，实际值需除以 100
int	num_trades	成交笔数
int	total_volume_trade	成交总量，实际值需除以 100
int	total_value_trade	成交总金额，实际值需除以 100000
int	IOPV	IOPV 净值估产（仅基金品种有效），实际值需除以 1000000，北交所为 0
int	high_limited	涨停价，实际值需除以 1000000

int	low_limited	跌停价，实际值需除以 1000000

5.5 指数快照(MDIndexSnapshot)

数据类型	字段名称	说明
int	market_type	市场类型
str	security_code	证券代码
str	orig_time	交易所行情数据时间（YYYYMMDDHHMMSSsss)
str	trading_phase_code	产品实时阶段及标志（仅深圳有效）
int	pre_close_index	前收盘价，实际值需除以 1000000
int	open_index	今开盘价，实际值需除以 1000000
int	high_index	最高价，实际值需除以 1000000
int	low_index	最低价，实际值需除以 1000000
int	last_index	最新价，实际值需除以 1000000
int	close_index	收盘价（仅上海有效），实际值需除以 1000000
int	total_volume_trade	参与计算相应指数的交易数量，实际值需除以 100（上交所:手，深交所:张）
int	total_value_trade	参与计算相应指数的成交金额，实际值需除以 100000

5.6 期权快照(MDOptionSnapshot)

数据类型	字段名称	说明
int	market_type	市场类型
str	security_code	期权代码
int	orig_time	交易所行情数据时间（YYYYMMDDHHMMSSsss)
str	trading_phase_code	产品实时阶段及标志
int	total_long_position	总持仓量，实际值需除以 100
int	total_volume_trade	总成交数，实际值需除以 100
int	total_value_trade	总成交额，实际值需除以 100000
int	pre_settle_price	昨结算价（仅上海有效），实际值需除以 1000000
int	pre_close_price	昨收盘价，实际值需除以 1000000
int	open_price	今开盘价，实际值需除以 1000000
int	auction_price	动态参考价（波动性中断参考价，仅上海有效），实际值

		需除以 1000000
int	auction_volume	虚拟匹配数量（仅上海有效），实际值需除以 100
int	high_price	最高价，实际值需除以 1000000
int	low_price	最低价，实际值需除以 1000000
int	last_price	最新价，实际值需除以 1000000
int	close_price	今收盘价，实际值需除以 1000000
int	high_limited	涨停价，实际值需除以 1000000
int	low_limited	跌停价，实际值需除以 1000000
Object	bid_price	tgw.Tools_GetInt64DataByIndex(data.bid_price, 0) #取出委托档位买一档价格(最多 5 档) 申买价，实际值需除以 1000000
Object	bid_volume	tgw.Tools_GetInt64DataByIndex(data. bid_volume, 0) #取出委托档位买一档量(最多 5 档) 申买量，实际值需除以 100
Object	offer_price	tgw.Tools_GetInt64DataByIndex(data. offer_price, 0) #取出委托档位卖一档价格(最多 5 档) 申卖价，实际值需除以 1000000
Object	offer_volume	tgw.Tools_GetInt64DataByIndex(data. offer_volume, 0) #取出委托档位卖一档量(最多 5 档) 申卖量，实际值需除以 100
int	settle_price	今日结算价（仅上海有效），实际值需除以 1000000
int	ref_price	参考价（仅深圳有效），实际值需除以 1000000
str	contract_type	合约类别
int	expire_date	到期日
str	underlying_security_cod	标的代码
int	exercise_price	行权价，实际值需除以 1000000

5.7 港股通快照(MDHKTSnapshot)

数据类型	字段名称	说明
int	market_type	市场类型
str	security_code	港股代码
int	orig_time	交易所行情数据时间（YYYYMMDDHHMMSSsss)
str	trading_phase_code	产品实时阶段及标志
int	total_volume_trade	总成交数，实际值需除以 100
int	total_value_trade	总成交额，实际值需除以 100000
int	pre_close_price	昨收价，实际值需除以 1000000
int	nominal_price	按盘价，实际值需除以 1000000
int	high_price	最高价，实际值需除以 1000000
int	low_price	最低价，实际值需除以 1000000

int	last_price	最新价，实际值需除以 1000000
Object	bid_price	申买价， tgw.Tools_GetInt64DataByIndex(data.bid_price, 0) #取出委托档位买一档价格(连续竞价下为一档，集中竞价阶段可能有两档)实际值需除以 1000000
Object	bid_volume	申买量， tgw.Tools_GetInt64DataByIndex(data. bid_volume, 0) #取出委托档位买一档量(正常情况下为一档，集中竞价阶段可能有两档)实际值需除以 100
Object	offer_price	申卖价， tgw.Tools_GetInt64DataByIndex(data. offer_price, 0) (正常情况下为一档，集中竞价阶段可能有两档)实际值需除以 1000000
Object	offer_volume	申买量， tgw.Tools_GetInt64DataByIndex(data. offer_volume, 0) (正常情况下为一档，集中竞价阶段可能有两档)实际值需除以 100
int	ref_price	参考价
int	high_limited	冷静期价格上限
int	low_limited	冷静期价格下限
int	bid_price_limit_up	买盘上限价，实际值需除以 1000000（仅深圳有效）
int	bid_price_limit_down	买盘下限价，实际值需除以 1000000（仅深圳有效）
int	offer_price_limit_up	卖盘上限价，实际值需除以 1000000（仅深圳有效）
int	offer_price_limit_do wn	卖盘下限价，实际值需除以 1000000（仅深圳有效）

5.8 期货快照(MDFutureSnapshot)

数据类型	字段名称	说明
int	market_type	市场类型
str	security_code	合约代码
int	orig_time	交易日 YYYYMMDDHHMMSSsss (ActionDay+UpdateTime+UpdateMillisec)
str	action_day	业务日期
int	last_price	最新价，实际值需除以 1000000
int	pre_settle_price	上次结算价，实际值需除以 1000000
int	pre_close_price	昨收价，实际值需除以 1000000
int	pre_open_interest	昨持仓量，实际值需除以 100
int	open_price	开盘价，实际值需除以 1000000
int	high_price	最高价，实际值需除以 1000000
int	low_price	最低价，实际值需除以 1000000

int	total_volume_trade	数量，实际值需除以 100
int	total_value_trade	总成交金额，实际值需除以 100000
int	open_interest	持仓量，实际值需除以 100
int	close_price	今收盘，实际值需除以 1000000
int	settle_price	本次结算价，实际值需除以 1000000
int	high_limited	涨停板价，实际值需除以 1000000
int	low_limited	跌停板价，实际值需除以 1000000
int	pre_delta	昨虚实度，实际值需除以 1000000
int	curr_delta	今虚实度，实际值需除以 1000000
Object	bid_price	tgw.Tools_GetInt64DataByIndex(data.bid_price, 0) #取出委托档位买一档价格(最多 5 档) 申买价，实际值需除以 1000000
Object	bid_volume	tgw.Tools_GetInt64DataByIndex(data.bid_volume, 0) #取出委托档位买一档量(最多 5 档) 申买量，实际值需除以 100
Object	offer_price	tgw.Tools_GetInt64DataByIndex(data.offer_price, 0) #取出委托档位卖一档价格(最多 5 档) 申卖价，实际值需除以 1000000
Object	offer_volume	tgw.Tools_GetInt64DataByIndex(data.offer_volume, 0) #取出委托档位卖一档量(最多 5 档) 申卖量，实际值需除以 100
int	average_price	当日均价，实际值需除以 1000000
str	trading_day	交易日期
int	variety_category	品种类别（暂不可用）
int	latest_volume_trade	最新成交量，实际值需除以 100（此字段仅托管机房模式有效）
int	init_volume_trade	初始持仓量，实际值需除以 100（此字段仅托管机房模式有效）
int	change_volume_trade	持仓量变化，实际值需除以 100（此字段仅托管机房模式有效）
int	bid_imply_volume	申买推导量，实际值需除以 100（此字段仅托管机房模式有效）
int	offer_imply_volume	申卖推导量，实际值需除以 100（此字段仅托管机房模式有效）
int	total_bid_volume_trade	总买入量，实际值需除以 100（此字段仅托管机房模式有效）
int	total_ask_volume_trade	总卖出量，实际值需除以 100（此字段仅托管机房模式有效）
str	exchange_inst_id	合约在交易所的代码

5.9 盘后定价交易快照(MDAfterHourFixedPriceSnapshot)

数据类型	字段名称	说明
int	market_type	市场类型
str	security_code	证券代码
int	variety_category	品种类别
int	orig_time	交易所行情数据时间（为 YYYYMMDDHHMMSSsss）
str	trading_phase_code	交易阶段代码
int	close_price	今日收盘价（仅上海有效），实际值需除以 1000000
int	bid_price	申买价，实际值需除以 1000000
int	bid_volume	申买量，实际值需除以 100
int	offer_price	申卖价，实际值需除以 1000000
int	offer_volume	申卖量，实际值需除以 100
int	pre_close_price	昨收价，实际值需除以 1000000
int	num_trades	成交笔数
int	total_volume_trade	成交总量，实际值需除以 100
int	total_value_trade	成交总金额，实际值需除以 100000

5.10 中证指数行情(MDCSIIndexSnapshot)

数据类型	字段名称	说明
int	market_type	市场类型
int	index_market	指数市场
str	security_code	证券代码
int	orig_time	交易所行情数据时间（YYYYMMDDHHMMSSsss)
int	last_index	最新指数，实际值需除以 1000000
int	open_index	今开盘指数，实际值需除以 1000000
int	high_index	最高指数，实际值需除以 1000000
int	low_index	最低指数，实际值需除以 1000000
int	close_index	收盘指数，实际值需除以 1000000
int	pre_close_index	前收盘指数，实际值需除以 1000000
int	change	涨跌，实际值需除以 1000000
int	ratio_of_change	涨跌幅，实际值需除以 1000000
int	total_volume_trade	成交量，实际值需除以 100
int	total_value_trade	成交金额，实际值需除以 100000(单位为万

		元)
int	exchange_rate	汇率，实际值需除以 100000000
str	currency_symbol	币种标志（0-人民币 1-港币 2-美元 3-台币 4-日元）

5.11国证指数快照(MDCnIndexSnapshot)

数据类型	字段名称	说明
int	market_type	市场类型
str	security_code	证券代码
int	orig_time	交易所行情数据时间(YYYYMMDDHHMMSSsss)
str	trading_phase_code	产品实时阶段及标志
int	pre_close_index	前收盘指数，实际值需除以 1000000
int	open_index	今开盘指数，实际值需除以 1000000
int	high_index	最高指数，实际值需除以 1000000
int	low_index	最低指数，实际值需除以 1000000
int	last_index	最新指数，实际值需除以 1000000
int	close_index	收盘指数，实际值需除以 1000000
int	total_volume_trade	参与计算相应指数的交易数量 N18(2)，实际值需除以 100
int	total_value_trade	参与计算相应指数的成交金额 N18(2)，实际值需除以 100000

5.12港股通实时额度(MDHKTRealtimeLimit)

数据类型	字段名称	说明
int	market_type	市场类型
int	orig_time	交易所行情数据时间（YYYYMMDDHHMMSSsss)
int	threshold_amount	每日初始额度，单位人民币元，实际值需除以 100000
int	pos_amt	日中剩余额度，单位人民币元，实际值需除以 100000
str	amount_status	额度状态(1-额度用完或其他原因全市场禁止买入 2-额度可用)
str	mkt_status	上交所港股通市场状态(上交所独有，来源于上交所文件行情)

5.13港股通产品状态(MDHKTProductStatus)

数据类型	字段名称	说明
int	market_type	市场类型
str	security_code	证券代码
int	orig_time	交易所行情数据时间(YYYYMMDDHHMMSSsss)
str	trading_status1	证券交易状态（整手订单）
str	trading_status2	证券交易状态（零股订单）

5.14港股 VCM(MDHKTVCM)

数据类型	字段名称	说明
int	market_type	市场类型
str	security_code	港股代码
int	orig_time	交易所行情数据时间(YYYYMMDDHHMMSSsss)
int	start_time	市调机制开始时间
int	end_time	市调机制结束时间
int	ref_price	市调机制参考价格，实际值需除以 1000000
int	low_price	市调机制最低价格，实际值需除以 1000000
int	high_price	市调机制最高价格，实际值需除以 1000000

5.15L2 现货快照(MDSnapshotL2)

数据类型	字段名称	说明
int	market_type	市场类型
str	security_code	证券代码
int	orig_time	时间（为 YYYYMMDDHHMMSSsss）
str	trading_phase_code	交易阶段代码
int	pre_close_price	昨收价，实际值需除以 1000000
int	open_price	开盘价，实际值需除以 1000000
int	high_price	最高价，实际值需除以 1000000
int	low_price	最低价，实际值需除以 1000000
int	last_price	最新价，实际值需除以 1000000
int	close_price	收盘价（仅上海有效），实际值需除以 1000000
Object	bid_price	tgw.Tools_GetInt64DataByIndex(data.bid_price, 0) #取出委托档位买一档价格(最多 10 档)

		申买价，实际值需除以 1000000
Object	bid_volume	tgw.Tools_GetInt64DataByIndex(data. bid_volume, 0) #取出委托档位买一档量(最多 10 档) 申买量，实际值需除以 100
Object	offer_price	tgw.Tools_GetInt64DataByIndex(data. offer_price, 0) #取出委托档位卖一档价格(最多 10 档) 申卖价，实际值需除以 1000000
Object	offer_volume	tgw.Tools_GetInt64DataByIndex(data. offer_volume, 0) #取出委托档位卖一档量(最多 10 档) 申卖量，实际值需除以 100
int	num_trades	成交笔数
int	total_volume_trade	成交总量，实际值需除以 100
int	total_value_trade	成交总金额，实际值需除以 100000
int	total_bid_volume	委托买入总量，实际值需除以 100
int	total_offer_volume	委托卖出总量，实际值需除以 100
int	weighted_avg_bid_price	加权平均为委买价格，实际值需除以 1000000
int	weighted_avg_offer_price	加权平均为委卖价格，实际值需除以 1000000
int	IOPV	IOPV 净值估产，实际值需除以 1000000
int	yield_to_maturity	到期收益率，实际值需除以 1000
int	high_limited	涨停价，实际值需除以 1000000
int	low_limited	跌停价，实际值需除以 1000000
int	price_earning_ratio1	市盈率 1，实际值需除以 1000000
int	price_earning_ratio2	市盈率 2，实际值需除以 1000000
int	change1	升跌 1（对比昨收价），实际值需除以 1000000
int	change2	升跌 2（对比上一笔），实际值需除以 1000000

5.16现货逐笔成交(MDTickExecution)

数据类型	字段名称	说明
str	market_type	市场类型
str	security_code	证券代码
int	exec_time	时间(YYYYMMDDHHMMSSsss)
str	channel_no	频道号
int	appl_seq_num	频道编号
int	exec_price	成交价格(类型:价格)
int	exec_volume	成交数量(类型:数量)
int	value_trade	成交金额(类型:金额)
int	bid_appl_seq_num	买方委托索引
int	offer_appl_seq_num	卖方委托索引

int	side	买卖方向(仅上海有效 B-外盘,主动买 S-内盘,主动卖 N-未知)
int	exec_type	成交类型(深圳:4-撤销 F-成交,上海:F-成交)
str	md_stream_id	行情类别(仅深圳有效)
int	biz_index	业务序号
int	variety_category	品种类别(此字段暂不可用)

5.17逐笔委托(MDTickOrder)

数据类型	字段名称	说明
int	market_type	市场类型
str	security_code	证券代码
int	appl_seq_num	消息记录号
str	channel_no	频道编号
int	order_time	委托时间（YYYYMMDDHHMMSSsss）
int	order_price	委托价格
int	order_volume	委托数量
int	side	买卖方向 深圳市场:(1-买 2-卖 G-借入 F-出借) 上海市场:(B:买单,S:卖单)
int	order_type	订单类别
str	md_stream_id[ConstField::kMDStreamIDLen]	行情类别
str	product_status[ConstField::kTradingPhaseCodeLen]	产品状态(仅上海有效)
int	orig_order_no	原始订单号
int	biz_index	业务序号

5.18委托队列(MDOrderQueue)

数据类型	字段名称	说明
int	market_type	市场类型
str	security_code	证券代码
int	order_time	委托时间 YYYYMMDDHHMMSSsss
int	side	买卖方向(B-买 S-卖)
int	order_price	委托价格
int	order_volume	委托数量
str	num_of_orders	总委托笔数
str	items	明细个数
int	volume[50]	订单明细
str	channel_no	频道代码

str	md_stream_id	行情类别

5.19代码表(MDCodeTable)

数据类型	字段名称	说明
str	security_code	交易所证券代码
str	symbol	证券简称
str	english_name	英文简称
int	market_type	市场类型
str	security_type	证券类别
str	currency	币种

说明：此数据结构为 IGMDCodeTableSpi 中 OnMDCodeTable 数据回调接口所用。

5.20复权因子表(MDExFactorTable)

数据类型	字段名称	说明
str	inner_code	证券内部代码
str	security_code	证券代码
ustr	ex_date	除权除息日（为 yyyyMMdd）
double	ex_factor	复权因子 N38(18)
double	cum_factor	累计复权因子 N38(18)

5.21个股基础信息(MDStockInfo)

数据类型	字段名称	说明
str	security_code	交易所证券代码
str	symbol	证券简称
int	market_type	市场类型
str	security_type	证券类别
str	currency	币种
str	security_status	证券状态 kSecurityStatusLen=24
uint	pre_close_price	昨收价
uint	total_shares	总股本
uint	flow_shares	流通股本
int	noprofit	是否盈利 Y：是，未盈利 N：否，已盈利
int	weighted_voting_rights	是否存在投票权差异 Y.是 N.否
int	registration_flag	是否注册制 Y.是 N.否
double	eps	每股收益
double	eps_cell	预计每股收益

double	net_profit_ttm	净利润（TTM）
double	net_profit	净利润
double	net_asset	净资产
double	net_profit_recent_annual	净利润
double	net_profit_first_quarter	净资产

5.22代码表(MDCodeTableRecord)

数据类型	字段名称	说明
str	security_code	证券代码
int	market_type	证券市场
str	symbol	简称
str	english_name	英文名
str	security_type	证券子类别
str	currency	币种(CNY:人民币,HKD:港币,USD:美元,AUD:澳币,CAD:加币,JPY:日圆,SGD:新加坡币,GBP:英镑,EUR:欧元,TWD:新台币,Other:其他)
int	variety_category	证券类别
int	pre_close_price	昨收价(类型:价格)
str	underlying_security_id	标的代码(仅期权/权证/期货期权有效)
str	contract_type	合约类别(仅期权/期货期权有效)
int	exercise_price	行权价(仅期权/期货期权有效,类型:价格)
ustr	expire_date	到期日(仅期权/期货期权有效)
int	high_limited	涨停价(类型:价格)
int	low_limited	跌停价(类型:价格)
str	security_status	产品状态标志
int	price_tick	最小价格变动单位(类型:价格)
int	buy_qty_unit	限价买数量单位(类型:数量)
int	sell_qty_unit	限价卖数量单位(类型:数量)
int	market_buy_qty_unit	市价买数量单位(类型:数量)
int	market_sell_qty_unit	市价卖数量单位(类型:数量)
int	buy_qty_lower_limit	限价买数量下限(类型:数量)
int	buy_qty_upper_limit	限价买数量上限(类型:数量)
int	sell_qty_lower_limit	限价卖数量下限(类型:数量)
int	sell_qty_upper_limit	限价卖数量上限(类型:数量)
int	market_buy_qty_lower_limit	市价买数量下限(类型:数量)
int	market_buy_qty_upper_limit	市价买数量上限(类型:数量)
int	market_sell_qty_lower_limit	市价卖数量下限(类型:数量)
int	market_sell_qty_upper_limit	市价卖数量上限(类型:数量)
ustr	list_day	上市日期

int	par_value	面值(类型:价格)
int	outstanding_share	总发行量(上交所不支持,类型:数量)
int	public_float_share_quantity	流通股数(上交所不支持,类型:数量)
int	contract_multiplier	对回购标准券折算率(类型:比例)
str	regular_share[ConstField::RegularShare]	对应回购标准券(仅深交所)
int	interest	应计利息(类型:汇率)
int	coupon_rate	票面年利率(类型:比例)
str	product_code[ConstField::kFutureSecurityCodeLen]	期货品种产品代码(仅期货期权有效)
ustr	delivery_year	交割年份(仅期货期权有效)
ustr	delivery_month	交割月份(仅期货期权有效)
ustr	create_date	创建日期(仅期货期权有效)
ustr	start_deliv_date	开始交割日(仅期货期权有效)
ustr	end_deliv_date	结束交割日(仅期货期权有效)
ustr	position_type	持仓类型(仅期货期权有效)

说明：此数据结构为 IGMDStockInfoSpi 中 OnMDStockInfo 数据回调接口所用。

5.23金融资讯数据(ThirdInfoData)

数据类型	字段名称	说明
uint	task_id	任务 id
uint	data_size	数据大小
str*	json_data	数据 json 串

6. 补充字段取值说明

6.1 证券代码表 security_type

深交所:
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:跨市场股票 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:杠杆 ETF （预留）
22: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
36:主板、中小板存托凭证
37:创业板存托凭证

港股:
EQTY:股本
TRST:信托
WRNT:权证
BOND:债券
BWRT:一篮子权证

上交所：
GBF:国债
GBZ:无息国债
DST:国债分销（仅用于分销阶段）
DVP:公司债（地方债）分销
CBF:企业债券
CCF:可转换企业债券
CPF:公司债券（或地方债券）
FBF:金融机构发行债券
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
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）
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
OEQ:其它股票
CIW:企业发行权证
COV:备兑权证
FEQ:个股期货
FBD:债券期货
OFT:其它期货
AMP:集合资产管理计划
WIT:国债预发行
LOF:LOF 基金
OPS:公开发行优先股
PPS:非公开发行优先股
QRP:报价回购
CMD:控制指令（中登身份认证密码服务产品复用 CMD 证券子类别）

6.2 证券代码表的 currency 取值

CNY: 人民币
HKD: 港币
USD: 美元
AUD: 澳币
CAD: 加币
JPY: 日圆
SGD: 新加坡币
GBP: 英 镑
EUR: 欧元

6.3 交易阶段代码取值

**********上海现货快照交易状态**********
该字段为 8 位字符数组,左起每位表示特定的含义,无定义则填空格。
第 0 位: ‘S’表示启动(开市前)时段,‘C’表示开盘集合竞价时段,‘T’表示连续交易时段,‘E’表示闭市时段,‘P’表示产品停牌。
第 1 位: ‘0’表示此产品不可正常交易,‘1’表示此产品可正常交易。
第 2 位: ‘0’表示未上市,‘1’表示已上市。
第 3 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。

**********深圳现货快照交易状态**********
第 0 位: ‘S’= 启动(开市前)‘O’= 开盘集合竞价‘T’= 连续竞价‘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。交易阶段代码

6.4 Security_status

1:停牌
2:除权
3:除息
4:ST
5:*ST
6:上市首日
7:公司再融资
8:恢复上市首日
9:网络投票
10:退市整理期
12:增发股份上市
13:合约调整
14:暂停上市后协议转让
15:实施双转单调整
16:特定债券转让
17:上市初期

上交所，参考参考 IS101_上海证券交易所竞价撮合平台市场参与者接口规格说明书 1.47 版_20200703 .docx
该字段为 20 位字符串，每位表示含义如下，无定义则填空格。
该字段为 20 位字符串，每位表示含义如下，无定义则填空格。
第 1 位对应：‘N’表示首日上市。
第 2 位对应：‘D’表示除权。
第 3 位对应：‘R’表示除息。
第 4 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 5 位不启用。
第 6 位对应：‘L’表示债券投资者适当性要求类，’M’表示债券机构投资者适当性要求类。
第 7 位对应：‘F’表示 15:00 闭市的产品，对应 mktdt00.txt 行情文件中市场行情状态第 4 位闭市标志，‘S’表示 15:30 闭市的产品，对应 mktdt00.txt 行情文件中市场行情状态第 5 位闭市标志，为空表示非竞价撮合平台挂牌产品，无意义。

6.5 K 线算法说明

前推算法举例：9:31 的 1 分钟 K 线，计算的是 9:30:00.000~9:30:59.999 期间的 K 线。 9:40 的 5 分钟 K 线，计算的是 9:35:00.000~9:39:59.999 期间的 K 线。