目录

C# SDK 文档

        - Symbol - 标的交易信息
        - TradeDate - 年度交易日历
        - TradingSession - 交易时段
        - ContractExpireRestDays - 合约到期剩余天数
        - IndustryCategory - 行业分类
        - IndustryConstituent - 行业成分股
        - SymbolIndustry - 股票所属行业
        - SectorCategory - 板块分类
        - SectorConstituent - 板块成分股
        - SymbolSector - 股票所属板块
        - IndexConstituent - 指数成分股
        - StockDividend - 股票分红送股信息
        - StockRation - 股票配股信息
        - AdjFactor - 股票复权因子
        - ShareholderNum - 股东户数
        - Shareholder - 十大股东
        - ShareChange - 股本变动
        - FundamentalsBalance - 资产负债表
        - FundamentalsCashflow - 利润表
        - FundamentalsIncome - 现金流量表
        - FinancePrime - 财务主要指标
        - FinanceDeriv - 财务衍生指标
        - DailyValuation - 交易衍生指标-估值类
        - DailyMktvalue - 交易衍生指标-市值类
        - DailyBasic - 交易衍生指标-基础类
        - ContinuousContractsInfo - 期货连续合约映射
        - FutContractInfo - 期货标准品种信息
        - FutTransactionRanking - 期货每日成交持仓排名
        - WarehouseReceiptInfo - 期货仓单数据
        - TransactionRankingInfo - 期货每日成交持仓排名
        - EtfConstituents - ETF基金成分股
        - PortfolioStockInfo - 基金资产组合（股票投资组合）
        - PortfolioBondInfo - 基金资产组合（债券投资组合）
        - PortfolioFundInfo - 基金资产组合（基金投资组合）
        - NetValueInfo - 基金净值
        - FndAdjFactorInfo - 基金复权因子
        - FndDividendInfo - 基金分红信息
        - SplitInfo - 基金拆分折算信息
        - ConversionPrice - 可转债转股价变动信息
        - CallInfo - 可转债赎回信息
        - PutInfo - 可转债回售信息
        - AmountChange - 可转债剩余规模变动
        - OptContractInfo - 期权标的基础信息
        - RiskValueInfo - 期权波动率
        - GetSymbolsByInAtOutReq - 期权档位合约信息
    - 交易类
        - Account - 账户结构
        - AccountStatus - 账户状态结构

指引

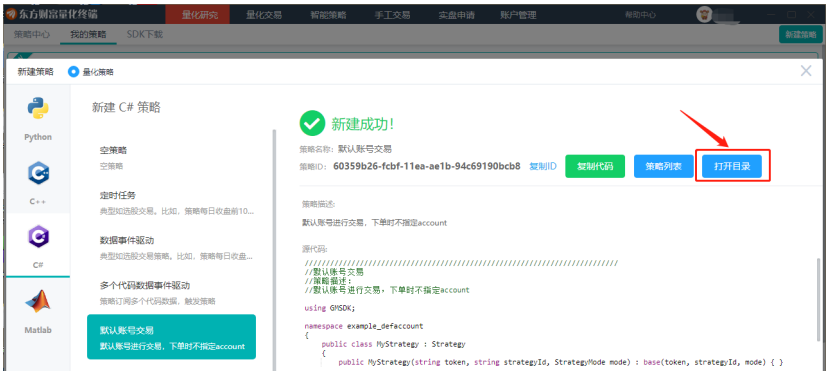
快速新建策略

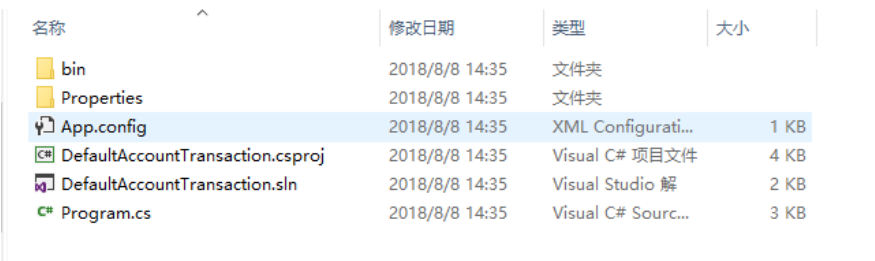
- 下载东方财富量化终端
- 打开终端后，登陆东方财富量化账号点击研究策略，新建策略 或者点击右上角新建策略

- 新建一个典型默认账户交易策略 新建C#的默认账户交易策略

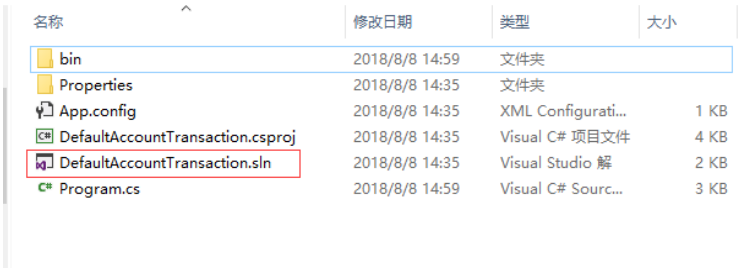
编译策略

- 打开新建策略文件目录 策略文件目录内容可以拷贝到本地其他盘符也可以进行编译生成

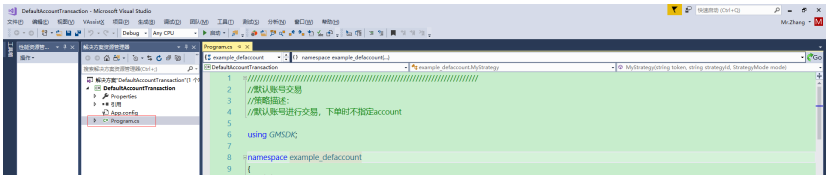




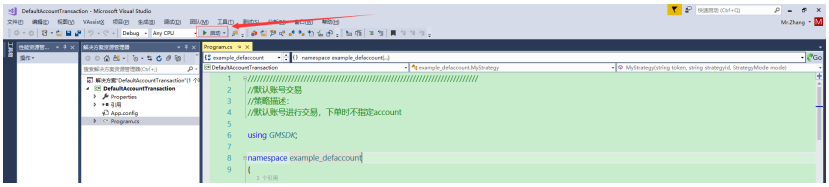
- 打开工程文件 sln 文件 需要用 **visual studio** 打开工程文件 (注意：**visual studio 2013**及以下版本需安装**.net framework 4.5.2**)



- 编写策略 打开Program.cs文件，可进行策略编辑



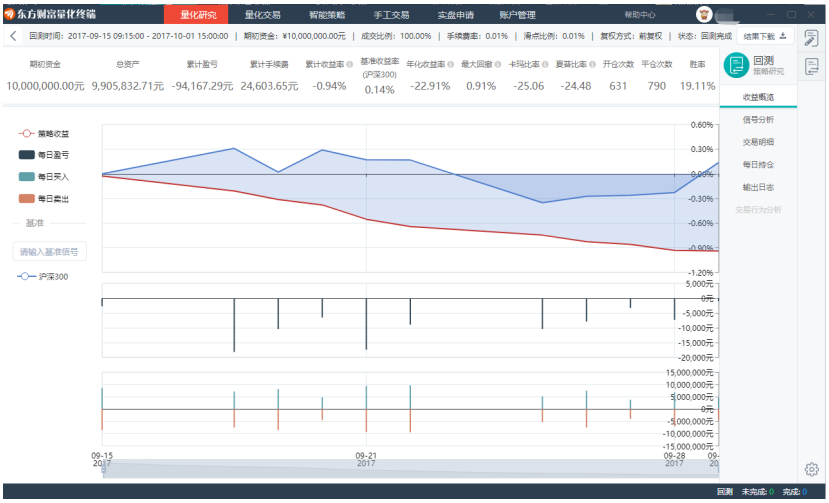
编译并运行策略



- 查看运行结果 东方财富量化客户端中关闭新建策略窗口并打开回测结果列表

查看回测结果

回测相关数据指标



获取 SDK

- 通过网页下载 sdk安装包: C# SDK

- 或者通过 NuGet 方式安装 SDK
  - 方式一：visual studio
  - 项目->管理 NuGet 程序包->浏览
  - 搜索 gmsdk-net, `32位` 选择**gmsdk-net-x86**, `64位` 选择**gmsdk-net-x64**
  - 选择最新版，安装
  - 方式二：程序包管理器控制台

    32 位

    ```
    PM> Install-Package gmsdk-net-x86
    ```

    64位

    ```
    PM> Install-Package gmsdk-net-x64
    ```

    注意： 根据策略选择32位或64位版本 `切勿混装`

建立我们第一个策略

- 打开Visual Studio新建空白工程并新建源码文件
- 工程中引用 **gmsdk-net.dll**
- 引入命名空间：GMSDK

```
using GMSDK;
```

- 将 **gmsdk.dll**, **protobuf-net.dll**放到策略执行文件所在目录

策略应该是这样的

- 继承策略基类
- 重改关注事件
- 在OnInit里订阅行情，初始化
- 在main方法中实例化一个派生类对像
- 设置token,策略id,和mode
- 开始运行

继承策略基类

```
public class MyStrategy: Strategy
{
    public MyStrategy(string token, string strategyId, StrategyMode mode) : base(toker
}
```

重改关注事件

```
public class MyStrategy: Strategy
{
    public MyStrategy(string token, string strategyId, StrategyMode mode) : base(toker

    //重写OnInit事件，进行策略开发
    public override void OnInit()
    {
        Console.WriteLine("OnInit");
        return;
    }
}
```

在**OnInit**里订阅行情，初始化

```
class MyStrategy :public Strategy
{
    public MyStrategy(string token, string strategyId, StrategyMode mode) : base(toker

    //重写OnInit事件，进行策略开发
    public override void OnInit()
    {
        Console.WriteLine("OnInit");
        Subscribe("SHSE.600000", "tick");
        return;
    }
}
```

在**main**里实例化一个派生类对像

1. 获取token：打开客户端->点击右上角用户头像 -> 系统设置 -> 复制token
2. 获取策略id：打开客户端->策略研究->右上角新建策略->新建C#策略->复制策略ID
3. 策略模式：
   - MODE_LIVE
   - MODE_BACKTEST

```
MyStrategy s("27cbdfd8cd9b86dea554a5612baa4a8eee51af79", "536f1097-8b27-11e8-b6af-94c6
```

开始运行

```
s.Run();
```

订阅行情策略示例

源文件

```
using GMSDK;

namespace example
{
    public class MyStrategy : Strategy
    {
        public MyStrategy(string token, string strategyId, StrategyMode mode) : base(t

        //重写OnInit事件，进行策略开发
        public override void OnInit()
        {
            System.Console.WriteLine("OnInit");
            //订阅行情数据
            Subscribe("SHSE.600000", "tick");
            return;
        }

        public override void OnTick(Tick tick)
        {
            System.Console.WriteLine("{0,-50}{1}", "代码", tick.symbol);
            System.Console.WriteLine("{0,-50}{1}", "时间", tick.createdAt);
            System.Console.WriteLine("{0,-50}{1}", "最新价", tick.price);
            System.Console.WriteLine("{0,-50}{1}", "开盘价", tick.open);
            System.Console.WriteLine("{0,-50}{1}", "最高价", tick.high);
            System.Console.WriteLine("{0,-50}{1}", "最低价", tick.low);
            System.Console.WriteLine("{0,-50}{1}", "成交总量", tick.cumVolume);
            System.Console.WriteLine("{0,-50}{1}", "成交总额/最新成交额,累计值", tick.cu
            System.Console.WriteLine("{0,-50}{1}", "合约持仓量(期)，累计值", tick.cumPo
            System.Console.WriteLine("{0,-50}{1}", "瞬时成交额", tick.lastAmount);
            System.Console.WriteLine("{0,-50}{1}", "瞬时成交量", tick.lastVolume);
            System.Console.WriteLine("{0,-50}{1}", "(保留)交易类型，对应多开，多平等类型
            System.Console.WriteLine("{0,-50}{1}", "一档委买价", tick.quotes[0].bidPric
            System.Console.WriteLine("{0,-50}{1}", "一档委买量", tick.quotes[0].bidVolu
            System.Console.WriteLine("{0,-50}{1}", "一档委卖价", tick.quotes[0].askPric
            System.Console.WriteLine("{0,-50}{1}", "一档委卖量", tick.quotes[0].askVolu
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            MyStrategy s = new MyStrategy("27cbdfd8cd9b86dea554a5612baa4a8eee51af79",
            s.SetBacktestConfig("2017-07-25 8:20:00", "2018-07-25 17:30:00");
            s.Run();
            System.Console.WriteLine("回测完成！");
            System.Console.Read();
        }
    }
}
```

典型场景

空策略

```
////////////////////////////////////////////////////////////////////////
//空策略
using GMSDK;

namespace example
{
    public class MyStrategy : Strategy
    {
        public MyStrategy(string token, string strategyId, StrategyMode mode) : base(t

        //重写OnInit事件，进行策略开发
        public override void OnInit()
        {
            System.Console.WriteLine("OnInit");
            //订阅行情数据
            Subscribe("SHSE.600000", "tick");
            return;
        }

        public override void OnTick(Tick tick)
        {
            System.Console.WriteLine("{0,-50}{1}", "代码", tick.symbol);
            System.Console.WriteLine("{0,-50}{1}", "时间", tick.createdAt);
            System.Console.WriteLine("{0,-50}{1}", "最新价", tick.price);
            System.Console.WriteLine("{0,-50}{1}", "开盘价", tick.open);
            System.Console.WriteLine("{0,-50}{1}", "最高价", tick.high);
            System.Console.WriteLine("{0,-50}{1}", "最低价", tick.low);
            System.Console.WriteLine("{0,-50}{1}", "成交总量", tick.cumVolume);
            System.Console.WriteLine("{0,-50}{1}", "成交总额/最新成交额,累计值", tick.cu
            System.Console.WriteLine("{0,-50}{1}", "合约持仓量(期)，累计值", tick.cumPo
            System.Console.WriteLine("{0,-50}{1}", "瞬时成交额", tick.lastAmount);
            System.Console.WriteLine("{0,-50}{1}", "瞬时成交量", tick.lastVolume);
            System.Console.WriteLine("{0,-50}{1}", "(保留)交易类型，对应多开，多平等类型
            System.Console.WriteLine("{0,-50}{1}", "一档委买价", tick.quotes[0].bidPric
            System.Console.WriteLine("{0,-50}{1}", "一档委买量", tick.quotes[0].bidVolu
            System.Console.WriteLine("{0,-50}{1}", "一档委卖价", tick.quotes[0].askPric
            System.Console.WriteLine("{0,-50}{1}", "一档委卖量", tick.quotes[0].askVolu
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            MyStrategy s = new MyStrategy("27cbdfd8cd9b86dea554a5612baa4a8eee51af79",
            s.SetBacktestConfig("2017-07-25 8:20:00", "2018-07-25 17:30:00");
            s.Run();
            System.Console.WriteLine("回测完成！");
            System.Console.Read();
        }


    }
}
```

定时任务

```
////////////////////////////////////////////////////////////////////////////
//定时任务
//策略描述:
    典型如选股交易。比如，策略每日收盘前10分钟执行：选股->决策逻辑->交易->退出。可能无需订阅实

using GMSDK;

namespace example_schedule
{
    public class MyStrategy : Strategy
    {
        public MyStrategy(string token, string strategyId, StrategyMode mode) : base(t

        //重写OnInit事件，进行策略开发
        public override void OnInit()
        {
            System.Console.WriteLine("OnInit");

            //设置定时任务
            Schedule("1d", "13:24:00");
            return;
        }

        //定时任务触发事件
        public override void OnSchedule(string data_rule, string timeRule)
        {
            //购买200股浦发银行股票
            GMData<Order> o = OrderValue("SHSE.600000", 200, OrderSide.OrderSide_Buy,
            if (o.status == 0)   //该判断仅表示函数调用无异常
            {
                //
            }
        }

        //回测完成后收到绩效报告
        public override void OnBacktestFinished(Indicator indicator)
        {
            System.Console.WriteLine("OnBacktestFinished: ");
            System.Console.WriteLine("{0,-50}{1}", "账号ID: ", indicator.accountId);
            System.Console.WriteLine("{0,-50}{1}", "累计收益率: ", indicator.pnlRatio);
            System.Console.WriteLine("{0,-50}{1}", "年化收益率: ", indicator.pnlRatioAn
            System.Console.WriteLine("{0,-50}{1}", "夏普比率: ", indicator.sharpRatio);
            System.Console.WriteLine("{0,-50}{1}", "最大回撤: ", indicator.maxDrawdown)
            System.Console.WriteLine("{0,-50}{1}", "风险比率: ", indicator.riskRatio);
            System.Console.WriteLine("{0,-50}{1}", "开仓次数: ", indicator.openCount);
            System.Console.WriteLine("{0,-50}{1}", "平仓次数: ", indicator.closeCount);
            System.Console.WriteLine("{0,-50}{1}", "盈利次数: ", indicator.winCount);
            System.Console.WriteLine("{0,-50}{1}", "亏损次数: ", indicator.loseCount);
            System.Console.WriteLine("{0,-50}{1}", "胜率: ", indicator.winRatio);
            System.Console.WriteLine("{0,-50}{1}", "指标创建时间: ", indicator.createdA
            System.Console.WriteLine("{0,-50}{1}", "指标变更时间: ", indicator.updatedA
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            MyStrategy s = new MyStrategy("27cbdfd8cd9b86dea554a5612baa4a8eee51af79",
            s.SetBacktestConfig("2016-07-11 17:20:00", "2017-07-11 17:30:00");
            s.Run();
            System.Console.WriteLine("回测完成! ");
            System.Console.Read();
        }
    }
}
```

数据事件驱动

```
//////////////////////////////////////////////////////////////////////////
//数据事件驱动
//策略描述：
//典型如选股交易策略。比如，策略每日收盘前10分钟执行：选股->决策逻辑->交易->退出。可能无需订阅

using GMSDK;

namespace example_dataevent
{
    public class MyStrategy : Strategy
    {
        public MyStrategy(string token, string strategyId, StrategyMode mode) : base(t

        //重写OnInit事件， 进行策略开发
        public override void OnInit()
        {
            System.Console.WriteLine("OnInit");

            //订阅浦发银行，bar频率为一天
            Subscribe("SHSE.600000", "1d");
            return;
        }

        //重写OnBar事件
        public override void OnBar(Bar bar)
        {
            System.Console.WriteLine("OnBar：");
            System.Console.WriteLine("{0, -50}{1}", "代码：", bar.symbol);
            System.Console.WriteLine("{0, -50}{1}", "bar的开始时间：", bar.bob);
            System.Console.WriteLine("{0, -50}{1}", "bar的结束时间：", bar.eob);
            System.Console.WriteLine("{0, -50}{1}", "开盘价：", bar.open);
            System.Console.WriteLine("{0, -50}{1}", "收盘价：", bar.close);
            System.Console.WriteLine("{0, -50}{1}", "最高价：", bar.high);
            System.Console.WriteLine("{0, -50}{1}", "最低价：", bar.low);
            System.Console.WriteLine("{0, -50}{1}", "成交量：", bar.volume);
            System.Console.WriteLine("{0, -50}{1}", "成交金额：", bar.amount);
            System.Console.WriteLine("{0, -50}{1}", "前收盘价：", bar.preClose);
            System.Console.WriteLine("{0, -50}{1}", "持仓量：", bar.position);
            System.Console.WriteLine("{0, -50}{1}", "bar频度：", bar.frequency);
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            MyStrategy s = new MyStrategy("27cbdfd8cd9b86dea554a5612baa4a8eee51af79",
            s.SetBacktestConfig("2016-07-11 17:20:00", "2017-07-11 17:30:00");
            s.Run();
            System.Console.WriteLine("回测完成！");
            System.Console.Read();
        }
    }
}
```

默认交易账号

```
////////////////////////////////////////////////////////////////////////
//默认账号交易
//策略描述:
//默认账号进行交易，下单时不指定account

using GMSDK;

namespace example_defaccount
{
    public class MyStrategy : Strategy
    {
        public MyStrategy(string token, string strategyId, StrategyMode mode) : base(t

        //重写OnInit事件，进行策略开发
        public override void OnInit()
        {
            System.Console.WriteLine("OnInit");
            Subscribe("SHSE.600000,SZSE.000001", "1d");

            return;
        }

        //重写OnBar事件
        public override void OnBar(Bar bar)
        {
            //不指定account 使用默认账户下单
            OrderVolume(bar.symbol, 200, OrderSide.OrderSide_Buy, OrderType.OrderType_

        }

        //回测完成后收到绩效报告
        public override void OnBacktestFinished(Indicator indicator)
        {
            System.Console.WriteLine("OnBacktestFinished: ");
            System.Console.WriteLine("{0,-50}{1}", "账号ID: ", indicator.accountId);
            System.Console.WriteLine("{0,-50}{1}", "累计收益率: ", indicator.pnlRatio);
            System.Console.WriteLine("{0,-50}{1}", "年化收益率: ", indicator.pnlRatioAr
            System.Console.WriteLine("{0,-50}{1}", "夏普比率: ", indicator.sharpRatio);
            System.Console.WriteLine("{0,-50}{1}", "最大回撤: ", indicator.maxDrawdown)
            System.Console.WriteLine("{0,-50}{1}", "风险比率: ", indicator.riskRatio);
            System.Console.WriteLine("{0,-50}{1}", "开仓次数: ", indicator.openCount);
            System.Console.WriteLine("{0,-50}{1}", "平仓次数: ", indicator.closeCount);
            System.Console.WriteLine("{0,-50}{1}", "盈利次数: ", indicator.winCount);
            System.Console.WriteLine("{0,-50}{1}", "亏损次数: ", indicator.loseCount);
            System.Console.WriteLine("{0,-50}{1}", "胜率: ", indicator.winRatio);
            System.Console.WriteLine("{0,-50}{1}", "指标创建时间: ", indicator.createdA
            System.Console.WriteLine("{0,-50}{1}", "指标变更时间: ", indicator.updatedA
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            MyStrategy s = new MyStrategy("27cbdfd8cd9b86dea554a5612baa4a8eee51af79",
            s.SetBacktestConfig("2016-07-11 17:20:00", "2017-07-11 17:30:00");
            s.Run();
            System.Console.WriteLine("回测完成! ");
            System.Console.Read();
        }
    }
}
```

显示指定交易账号

```
////////////////////////////////////////////////////////////////////////////
//显示指定交易账号
//策略描述：
//下单时指定交易账号，account参数传账号id或者账号标题


using GMSDK;

namespace example_defaccount
{
    public class MyStrategy : Strategy
    {
        public MyStrategy(string token, string strategyId, StrategyMode mode) : base(t

        //重写OnInit事件，进行策略开发
        public override void OnInit()
        {
            System.Console.WriteLine("OnInit");
            Subscribe("SHSE.600000,SZSE.000001", "1d");

            return;
        }

        //重写OnBar事件
        public override void OnBar(Bar bar)
        {
            //不指定account 使用默认账户下单
            OrderVolume(bar.symbol, 200, OrderSide.OrderSide_Buy, OrderType.OrderType_

        }

        //回测完成后收到绩效报告
        public override void OnBacktestFinished(Indicator indicator)
        {
            System.Console.WriteLine("OnIndicator: ");
            System.Console.WriteLine("{0,-50}{1}", "账号ID: ", indicator.accountId);
            System.Console.WriteLine("{0,-50}{1}", "累计收益率: ", indicator.pnlRatio);
            System.Console.WriteLine("{0,-50}{1}", "年化收益率: ", indicator.pnlRatioAr
            System.Console.WriteLine("{0,-50}{1}", "夏普比率: ", indicator.sharpRatio);
            System.Console.WriteLine("{0,-50}{1}", "最大回撤: ", indicator.maxDrawdown)
            System.Console.WriteLine("{0,-50}{1}", "风险比率: ", indicator.riskRatio);
            System.Console.WriteLine("{0,-50}{1}", "开仓次数: ", indicator.openCount);
            System.Console.WriteLine("{0,-50}{1}", "平仓次数: ", indicator.closeCount);
            System.Console.WriteLine("{0,-50}{1}", "盈利次数: ", indicator.winCount);
            System.Console.WriteLine("{0,-50}{1}", "亏损次数: ", indicator.loseCount);
            System.Console.WriteLine("{0,-50}{1}", "胜率: ", indicator.winRatio);
            System.Console.WriteLine("{0,-50}{1}", "指标创建时间: ", indicator.createdA
            System.Console.WriteLine("{0,-50}{1}", "指标变更时间: ", indicator.updatedA
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            MyStrategy s = new MyStrategy("27cbdfd8cd9b86dea554a5612baa4a8eee51af79",
            s.SetBacktestConfig("2016-07-11 17:20:00", "2017-07-11 17:30:00");
            s.Run();
            System.Console.WriteLine("回测完成! ");
            System.Console.Read();
        }
    }
}
```

模式选择

```
////////////////////////////////////////////////////////////////////////
//模式选择
//策略描述:
//策略支持两种运行模式，实时模式和回测模式，用户需要在运行策略时选择模式，实例化策略参数mode=

using GMSDK;

namespace example
{
    public class MyStrategy : Strategy
    {
        public MyStrategy(string token, string strategyId, StrategyMode mode) : base(t

        //重写OnInit事件，进行策略开发
        public override void OnInit()
        {
            System.Console.WriteLine("OnInit");
            //订阅行情数据
            Subscribe("SHSE.600000", "tick");
            return;
        }

        public override void OnTick(Tick tick)
        {
            System.Console.WriteLine("{0,-50}{1}", "代码", tick.symbol);
            System.Console.WriteLine("{0,-50}{1}", "时间", tick.createdAt);
            System.Console.WriteLine("{0,-50}{1}", "最新价", tick.price);
            System.Console.WriteLine("{0,-50}{1}", "开盘价", tick.open);
            System.Console.WriteLine("{0,-50}{1}", "最高价", tick.high);
            System.Console.WriteLine("{0,-50}{1}", "最低价", tick.low);
            System.Console.WriteLine("{0,-50}{1}", "成交总量", tick.cumVolume);
            System.Console.WriteLine("{0,-50}{1}", "成交总额/最新成交额,累计值", tick.cu
            System.Console.WriteLine("{0,-50}{1}", "合约持仓量(期)，累计值", tick.cumPo
            System.Console.WriteLine("{0,-50}{1}", "瞬时成交额", tick.lastAmount);
            System.Console.WriteLine("{0,-50}{1}", "瞬时成交量", tick.lastVolume);
            System.Console.WriteLine("{0,-50}{1}", "(保留)交易类型，对应多开，多平等类型
            System.Console.WriteLine("{0,-50}{1}", "一档委买价", tick.quotes[0].bidPric
            System.Console.WriteLine("{0,-50}{1}", "一档委买量", tick.quotes[0].bidVolu
            System.Console.WriteLine("{0,-50}{1}", "一档委卖价", tick.quotes[0].askPric
            System.Console.WriteLine("{0,-50}{1}", "一档委卖量", tick.quotes[0].askVolu
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            MyStrategy s = new MyStrategy("27cbdfd8cd9b86dea554a5612baa4a8eee51af79",
            s.Run();
            System.Console.Read();
        }
    }
}
```

数据研究

```
////////////////////////////////////////////////////////////////////////
//数据研究
//策略描述：
//无需实时数据驱动策略，无需交易下单，只是取数据的场景

using GMSDK;

namespace example_datares
{
    public class MyStrategy : Strategy
    {
        public MyStrategy(string token, string strategyId, StrategyMode mode) : base(t

        //重写OnInit事件，进行策略开发
        public override void OnInit()
        {
            System.Console.WriteLine("OnInit");

            GMDataList<Tick> ht = GMApi.HistoryTicks("SZSE.000002", "2017-07-11 10:20:
            if (ht.status == 0)
            {
                foreach (var tick in ht.data)
                {
                    System.Console.WriteLine("{0,-50}{1}", "代码", tick.symbol);
                    System.Console.WriteLine("{0,-50}{1}", "时间", tick.createdAt);
                    System.Console.WriteLine("{0,-50}{1}", "最新价", tick.price);
                    System.Console.WriteLine("{0,-50}{1}", "开盘价", tick.open);
                    System.Console.WriteLine("{0,-50}{1}", "最高价", tick.high);
                    System.Console.WriteLine("{0,-50}{1}", "最低价", tick.low);
                    System.Console.WriteLine("{0,-50}{1}", "成交总量", tick.cumVolume)
                    System.Console.WriteLine("{0,-50}{1}", "成交总额/最新成交额,累计值",
                    System.Console.WriteLine("{0,-50}{1}", "合约持仓量(期)，累计值", ti
                    System.Console.WriteLine("{0,-50}{1}", "瞬时成交额", tick.lastAmour
                    System.Console.WriteLine("{0,-50}{1}", "瞬时成交量", tick.lastVolum
                    System.Console.WriteLine("{0,-50}{1}", "(保留)交易类型，对应多开，多
                    System.Console.WriteLine("{0,-50}{1}", "一档委买价", tick.quotes[0
                    System.Console.WriteLine("{0,-50}{1}", "一档委买量", tick.quotes[0
                    System.Console.WriteLine("{0,-50}{1}", "一档委卖价", tick.quotes[0
                    System.Console.WriteLine("{0,-50}{1}", "一档委卖量", tick.quotes[0
                }
            }
            return;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            MyStrategy s = new MyStrategy("27cbdfd8cd9b86dea554a5612baa4a8eee51af79",
            s.SetBacktestConfig("2017-07-25 8:20:00", "2018-07-25 17:30:00");
            s.Run();
            System.Console.Read();
        }
    }
}
```

重要概念

symbol - 代码标识

东方财富量化代码(**symbol**)是东方财富量化平台用于唯一标识交易标的代码,

格式为：**交易所代码.交易标代码**, 比如深圳平安的**symbol** 示例： `SZSE.000001`

交易所代码

目前东方财富量化支持国内的2个交易所, 各交易所的代码缩写如下：

市场中文名	市场代码
上交所	SHSE
深交所	SZSE

交易标的代码

交易表代码是指交易所给出的交易标的代码, 包括股票, 期货, 期权, 指数, 基金等代码。

具体的代码请参考交易所的给出的证券代码定义

策略运行模式

策略支持两种运行模式, 实时模式和回测模式,用户需要在运行策略时选择模式.

实时模式

订阅行情服务器推送的实时行情，也就是交易所的实时行情，只在交易时段提供。

回测模式

订阅指定时段、指定交易代码、指定数据类型的行情，行情服务器将按指定条件全速回放对应的行情数据。适用的场景是策略回测阶段，快速验证策略的绩效是否符合预期。

策略基类

基类原型

策略类简介

策略类集成了行情、交易和事件的接口，用户的策略都从此类继承实现自己的业务逻辑。每个进程只能实例化一个策略类对象。

策略类定义

```
public class Strategy
{
    //策略基类构造函数
    //token:
    //strategy_id: 策略ID
    //mode: 运行模式
    public Strategy(string token, string strategyId, int mode);


//====================================策略参数类函数================================
    //添加参数
    public int AddParameters(Parameter param);

    //删除参数
    public int DelParameters(string keys);

    //获取标的
    public GMDataList<string> GetSymbols();

    //设置标的
    public int SetSymbols(string symbols);

    //设置回测参数
       public int SetBacktestConfig(string startTime, string endTime, double initialCa

    //设置参数
    public int SetParameters(List<Parameter> parameters);


//=====================================交易函数==================================
    //查询交易账号
    public GMDataList<Account> GetAccounts();

    //查询资金
    public GMDataList<Cash> GetCash(string account = null);

    //查询成交
    public GMDataList<ExecRpt> GetExecutionReports(string account = null);

    //查询委托
    public GMDataList<Order> GetOrders(string account = null);

    //查询持仓
    public GMDataList<Position> GetPosition(string account = null);

    //查询未结委托
    public GMDataList<Order> GetUnfinishedOrders(string account = null);

    //委托撤单,
    public int OrderCancel(string clOrdIds);

    //撤销所有委托
    public void OrderCancelAll();

    //平当前所有可平持仓
    public GMDataList<Order> OrderCloseAll();

    //按总资产指定比例委托
    public GMData<Order> OrderPercent(string symbol, double percent, int side, int ord

    //调仓到目标持仓比例（总资产的比例）
    public GMData<Order> OrderTargetPercent(string symbol, double percent, int positio

    //调仓到目标持仓额
    public GMData<Order> OrderTargetValue(string symbol, double value, int positionSid
```

```
    //调仓到目标持仓量
    public GMData<Order> OrderTargetVolume(string symbol, int volume, int positionSide

    //按指定价值委托
    public GMData<Order> OrderValue(string symbol, double value, int side, int orderTy

    //按指定量委托
    public GMData<Order> OrderVolume(string symbol, int volume, OrderSide side, OrderT


//====================================基础函数================================
    //运行策略
    public int Run();

    //定时任务
    public int Schedule(string dataRule, string timeRule);

    //停止策略
    public int Stop();

    //当前事件
    public long Now();

    //设置token
    public int SetToken(string token);

    //设置运行模式
    public int SetMode(StrategyMode mode);

    //设置策略ID
    public int SetStrategyId(string strategyId);

    //查询指定账户状态
    public AccountStatus GetAccountStatus(string accountId);

    //查询所有账户状态
    public List<AccountStatus> GetAccountStatus()

//====================================数据函数====================================
    //订阅行情
    public int Subscribe(string symbols, string frequency, bool unsubscribePrevious =

    //退订行情
    public int Unsubscribe(string symbols, string frequency);


//====================================事件函数====================================
    //初始化完成
    public virtual void OnInit();

    //实盘账号状态变化
    public virtual void OnAccountStatus(AccountStatus accountStatus);

    //收到bar行情
    public virtual void OnBar(Bar bar);

    //cash发生变化
    public virtual void OnCashStatus(Cash order);

    //错误事件
    public virtual void OnError(int errorCode, string errorMsg);

    //执行回报
    public virtual void OnExecutionReport(ExecRpt rpt);
```

```
    //回测结束
    public virtual void OnBacktestFinished(Indicator indicator);

    //数据库已连接
    public virtual void OnMarketDataConnected();

    //数据库断开
    public virtual void OnMarketDataDisconnected();

    //委托发生变化
    public virtual void OnOrderStatus(Order order);

    //运行时参数发生变化
    public virtual void OnParameter(List<Parameter> param);

    //position发生变化
    public virtual void OnPosition(Position position);

    //定时任务触发
    public virtual void OnSchedule(string dataRule, string timeRule);

    //策略结束
    public virtual void OnStop();

    //收到tick行情
    public virtual void OnTick(Tick tick);

    //交易已连接
    public virtual void OnTradeDataConnected();

    //交易断开
    public virtual void OnTradeDataDisconnected();
}
```

策略基类

基本成员函数

Strategy - 构造函数

构造策略对象。

函数原型:

```
Strategy();
Strategy(string token, string strategyId, StrategyMode mode);
```

参数：

参数名	类型	说明
token	string	系统权限密钥,可在终端系统设置-密钥管理中生成
strategyId	string	策略ID,在终端中获取
mode	StrategyMode	策略模式，参见 `enum StrategyMode`

注意事项：

- 一个进程只能构造一个策略对象。

Run - 运行策略

运行策略。只有调用Run后，才会驱动所有的事件，如行情接入与交易事件。

函数原型:

```
int Run();
```

参数：

参数名	类型	说明
返回值	int	如果策略正常退出返回0， 非正常退出返回错误码

注意事项：

调用Run会阻塞线程，策略进入事件驱动状态，所以所有初始操作（如读配置文件，分配缓冲区等）都应该在Run之前完成，如果run退出，意味着策略运行结束，整个进程应该就此退出。

Stop - 停止策略

用于停止策略， 也就是如果调用Run()之后， 在某个事件响应中调用Stop, 这是Run就是退出，并返回0。

函数原型:

```
void Stop();
```

SetToken - 设置用户token

函数原型:

```
int SetToken(string token)
```

参数:

参数名	类型	说明
token	string	系统权限密钥,可在终端系统设置-密钥管理中生成

注意事项: 不管是从构造函数传入还成员函数传入，`token`，`strategyId`，`mode` 都是必须要设置的参数。

SetMode - 设置运行模式

函数原型

```
int SetMode(StrategyMode mode)
```

参数:

参数名	类型	说明
mode	StrategyMode	策略运行模式，参见 `StrategyMode`

注意事项: 不管是从构造函数传入还成员函数传入，`token`，`strategyId`，`mode` 都是必须要设置的参数

SetStrategyId - 设置策略ID

函数原型:

```
int SetStrategyId(string strategyId)
```

注意事项: 不管是从构造函数传入还成员函数传入，`token`，`strategyId`，`mode` 都是必须要设置的参数

参数:

参数名	类型	说明
strategyId	string	策略ID,在终端中获取

GetAccountStatus - 获取策略所有账户状态

函数原型:

```
List<AccountStatus> GetAccountStatus()
```

参数：

参数名	类型	说明
返回值	List	AccountStatus 列表

示例：

```
//获取当策略所有账户状态
var status_l = GetAccountStatus();
//遍历账户
foreach (var status in status_l)
{
    //打印 AccountStatus 字段
    System.Console.WriteLine("accountId: {0}, accountName: {1}, state: {2}, errorCode:
}
```

GetAccountStatus - 获取指定账户状态

函数原型：

```
AccountStatus GetAccountStatus(string accountId)
```

参数：

参数名	类型	说明
返回值	AccountStatus	账户状态结构

Schedule - 预设定时任务

在指定时间自动执行策略算法, 通常用于选股类型策略。Schedule一般在OnInit中调用。如果Schedule预设成功，那么达成预设时间条件时，OnSchedule会被调用，并在OnSchedule的参数中返回设置的 `dataRule` 和 `timeRule` 。Schedule可以调用多次，设置多个不同定时任务。

函数原型：

```
int Schedule(string dataRule, string timeRule);
```

参数：

参数名	类型	说明
dataRule	string	n + 时间单位， 可选’d/w/m’ 表示n天/n周/n月
timeRule	string	执行算法的具体时间 (%H:%M:%S 格式)
返回值	int	预设成功返回0， 预设失败返回错误码

示例：

```
#每天的19:06:20执行
Schedule(dateRule="1d", timeRule="19:06:20")

#每月的第一个交易日的09:40:00执行
Schedule(dateRule="1m", time_rule="9:40:00")
```

注意事项：

- 现在 `dataRule` 暂只支持 1d,1w,1m， 任意n后续会支持。

Now - 获取当前时间

函数原型：

```
DateTime Now();
```

参数：

参数名	类型	说明
返回值	DateTime	当前时间

注意事项：

- 实时模式下，返回当前的系统时间。回测模式下，返回当前的回测时间点。格式是DateTime。

SetBacktestConfig - 设置回测参数

如果mode设置为回测模式，则在调用Run之前，需要先设置本函数设置回测参数。在实时模式下，该调用被忽略。

函数原型：

```
int SetBacktestConfig(
    string startTime,
    string endTime,
    double initialCash = 1000000,
    double transactionRatio = 1,
    double commissionRatio = 0,
    double slippageRatio = 0,
    Adjust    adjust = 0,
    int    checkCache = 1
);
```

参数：

参数名	类型	说明
startTime	string	回测开始时间 (%Y-%m-%d %H:%M:%S格式)
endTime	string	回测结束时间 (%Y-%m-%d %H:%M:%S格式)
initialCash	double	回测初始资金, 默认1000000
transactionRatio	double	回测成交比例, 默认1.0, 即下单100%成交
commissionRatio	double	回测佣金比例, 默认0
slippageRatio	double	回测滑点比例, 默认0
adjust	Adjust	复权方式，参见 `enum Adjust`
checkCache	int	回测是否使用缓存：1 - 使用， 0 - 不使用；默认使用

注意： startTime和endTime中月,日,时,分,秒均可以只输入个位数，例: `"2016-6-7 9:55:0"` 或 `"2017-8-1 14:6:0"` ,但若对应位置为零，则0不可被省略,比如不能输入 `"2017-8-1 14:6: "`

行情成员函数

subscribe - 订阅行情

订阅行情推送，实时模式下订阅实时行情推送，回测模式下订阅历史行情推送。订阅tick会触发OnTick回调，订阅bar则触发OnBar回调。

函数原型:

```
int Subscribe(string symbols, string frequency, bool unsubscribePrevious = false);
```

参数：

参数名	类型	说明
symbols	string	订阅标的代码列表, 字符串格式，如有多个代码, 中间用 , (英文逗号) 隔开
frequency	string	频率, 支持 'tick', '60s', '300s', '900s' 等, 默认'1d', 详情见股票行情数据
unsubscribePrevious	bool	是否取消过去订阅的symbols, 默认false不取消, 输入true则取消所有原来的订阅。
返回值	int	订阅成功返回0， 订阅失败返回错误码

示例：

```
//订阅 SHSE.600000和 SZSE.000001 两个标的的tick行情
Subscribe(symbols="SHSE.600000,SHSE.600004", frequency="tick");

//订阅 SHSE.600000和 SZSE.000001 两个标的的1分钟bar
Subscribe(symbols="SHSE.600000,SHSE.600004", frequency="60s");
```

unsubscribe - 退订行情

退订已经订阅行情推送， 与Subscribe作用相返。

函数原型:

```
int unsubscribe(string symbols, string frequency);
```

参数：

参数名	类型	说明
symbols	string	退订标的代码列表, 字符串格式，如有多个代码, 中间用 , (英文逗号) 隔开
frequency	string	频率, 支持 'tick', '60s', '300s', '900s' 等, 默认'1d', 详情见股票行情数据
返回值	int	退订成功返回0， 退订失败返回错误码

示例：

```
//退订 SHSE.600000和 SZSE.000001 两个标的的tick行情
Unsubscribe(symbols="SHSE.600000,SHSE.600004", frequency="tick");
```

交易成员函数

GetAccounts - 查询交易账号

用于查询交易账号配置信息。多用于实盘时，策略同时关联多个交易账号的时候，获取所有交易账号的信息，所返回的账号id(`accounId`)用于后续各个交易api的入参， 即指定操作某个交易账户。 如果关联的交易账号只有一个， 一般用不到此函数。

函数原型:

```
GMDataList<Account> GetAccounts();
```

参数：

参数名	类型	说明
返回值	`GMDataList<Account>`	一个GMDataList结构

orderVolume - 按指定量委托

按指定量委托, 如果调用成功，后续委托单状态变化将会触发on_order_status回调。

函数原型:

```
GMData<Order> OrderVolume(string symbol, int volume, OrderSide side, OrderType orderTy
```

参数：

参数名	类型	说明
symbol	string	标的代码，只能单个标的
volume	int	委托数量
side	OrderSide	委托方向 参见 `enum OrderSide`
orderType	OrderType	委托类型 参见 `enum OrderType`
positionEffect	PositionEffect	开平类型 参见 `enum PositionSide`
price	double	委托价格（限价委托时的委托价，沪市市价委托的市价保护价，多位小数时会四舍五入，股票只保留两位，基金只保留三位）
account	string	实盘账号id,关联多实盘账号时填写，可以从 GetAccounts获取，也可以从终端实盘账号配置里拷贝。如果策略只关联一个账号，可以设置为null
返回值	Order	一个Order结构, 如果函数调用失败，Order.status 值为 `OrderStatus_Rejected` , Order.ordRejReasonDetail 为错误原因描述, 其它情况表示函数调用成功，Order.clOrdId 为本次委托的标识，可用于追溯订单状态或撤单

示例：

```
//以11块的价格限价买入10000股浦发银行
GMData<Order> o = orderVolume("SHSE.600000", 10000, OrderSide.OrderSide_Buy, OrderType
```

注意：

1. 仅支持一个标的代码，若交易代码输入有误，终端会拒绝此单，并显示 `委托代码不正确` 。

2. 若下单数量输入有误，终端会拒绝此单，并显示 `委托量不正确` 。股票买入最小单位为 `100` ，卖出最小单位为 `1` ,如存在不足100股的持仓一次性卖出;期货买卖最小单位为 `1` ， `向下取整` 。

3. 若仓位不足，终端会拒绝此单，显示 `仓位不足` 。平仓时股票默认 `平昨仓` ，期货默认 `平今仓` 。应研究需要， `股票也支持卖空操作` 。

4. OrderType优先级高于price,若指定OrderTpye_Market下市价单，使用价格为最新一个tick中的最新价，price参数失效。则price参数失效。若OrderTpye_Limit限价单，仿真模式价格错误，终端拒绝此单，显示委托价格错误， `回测模式下对价格无限制` 。

5. 函数调用成功并不意味着委托已经成功，只是意味委托单已经成功发出去， 委托是否成功根据OnOrderStatus，或 GetOrder来判断。

OrderValue - 按指定价值委托

按指定价值委托, 如果调用成功，后续委托单状态变化将会触发OnOrderStatus回调。

函数原型:

```
GMData<Order> OrderValue(string symbol, double value, OrderSide side, OrderType orderT
```

参数：

参数名	类型	说明
symbol	string	标的代码，只能单个标的
value	int	股票价值
side	OrderSide	委托方向 参见 `enum OrderSide`
orderType	OrderType	委托类型 参见 `enum OrderType`
positionEffect	PositionEffect	开平类型 参见 `enum PositionSide`
price	double	委托价格（限价委托时的委托价，沪市市价委托的市价保护价，多位小数时会四舍五入，股票只保留两位，基金只保留三位）
account	string	实盘账号id,关联多实盘账号时填写，可以从 GetAccounts获取，也可以从终端实盘账号配置里拷贝。如果策略只关联一个账号，可以设置为null
返回值	Order	一个Order结构, 如果函数调用失败，Order.status 值为 `OrderStatus_Rejected` , Order.ordRejReasonDetail 为错误原因描述, 其它情况表示函数调用成功，Order.clOrdId 为本次委托的标识，可用于追溯订单状态或撤单

示例：

```
//下限价单，以11元每股的价格买入价值为100000元的SHSE.600000，根据volume = value / price,计
GMData<Order> o = order_value("SHSE.600000", 100000, OrderSide.OrderSide_Buy, OrderTyp
```

注意：

1. 仅支持一个标的代码，若交易代码输入有误，终端会拒绝此单，并显示 `委托代码不正确` 。

2. 根据指定价值计算购买标的数量，即 `value/price` 。股票买卖最小单位为 `100` ，不足100部分 `向下取整` ，如存在不足100的持仓一次性卖出;期货买卖最小单位为 `1` ， `向下取整` 。

3. 若仓位不足，终端会拒绝此单，显示 `仓位不足` 。平仓时股票默认 `平昨仓` ，期货默认 `平今仓` 。应研究需要， `股票也支持卖空操作` 。

4. OrderType优先级高于price,若指定OrderTpye_Market下市价单，使用价格为最新一个tick中的最新价，price参数失效。则price参数失效。若OrderTpye_Limit限价单，仿真模式价格错误，终端拒绝此单，显示委托价格错误，`回测模式下对价格无限制`。

5. 函数调用成功并不意味着委托已经成功，只是意味委托单已经成功发出去， 委托是否成功根据OnOrderStatus，或 GetOrder来判断。

OrderPercent - 按总资产指定比例委托

按总资产指定比例委托, 如果调用成功，后续委托单状态变化将会触发OnOrderStatus回调。

函数原型:

```
GMData<Order> OrderPercent(string symbol, double percent, OrderSide side, OrderType or
```

参数：

参数名	类型	说明
symbol	string	标的代码，只能单个标的
percent	double	委托占总资产比例
side	OrderSide	委托方向 参见 `enum OrderSide`
orderType	OrderType	委托类型 参见 `enum OrderType`
positionEffect	PositionEffect	开平类型 参见 `enum PositionSide`
price	double	委托价格（限价委托时的委托价，沪市市价委托的市价保护价，多位小数时会四舍五入，股票只保留两位，基金只保留三位）
account	string	实盘账号id,关联多实盘账号时填写，可以从 GetAccounts获取，也可以从终端实盘账号配置里拷贝。如果策略只关联一个账号，可以设置为null
返回值	Order	一个Order结构, 如果函数调用失败，Order.status 值为 `OrderStatus_Rejected` ，Order.ordRejReasonDetail 为错误原因描述, 其它情况表示函数调用成功，Order.clOrdId 为本次委托的标识，可用于追溯订单状态或撤单

示例：

```
//当前总资产为1000000。下限价单，以11元每股的价格买入SHSE.600000,期望买入比例占总资产的10%,
GMData<Order> o = OrderPercent("SHSE.600000", 0.1, OrderSide.OrderSide_Buy, OrderType.
```

注意：

1. 仅支持一个标的代码，若交易代码输入有误，终端会拒绝此单，并显示 `委托代码不正确` 。

2. 根据指定比例计算购买标的数量,即 `(nav*precent)/price` ,股票买卖最小单位为 `100` ，不足100部分 `向下取整` ，如存在不足100的持仓一次性卖出;期货买卖最小单位为 `1` ， `向下取整` 。

3. 若仓位不足，终端会拒绝此单，显示 `仓位不足` 。平仓时股票默认 `平昨仓` ，期货默认 `平今仓` 。应研究需要， `股票也支持卖空操作` 。

4. OrderType优先级高于price,若指定OrderTpye_Market下市价单，使用价格为最新一个tick中的最新价，price参数失效。则price参数失效。若OrderTpye_Limit限价单，仿真模式价格错误，终端拒绝此单，显示委托价格错误， `回测模式下对价格无限制` 。

5. 函数调用成功并不意味着委托已经成功，只是意味委托单已经成功发出去， 委托是否成功根据OnOrderStatus，或 GetOrder来判断。

OrderTargetVolume - 调仓到目标持仓量

调仓到目标持仓量, 如果调用成功，后续委托单状态变化将会触发OnOrderStatus回调。

函数原型:

```
GMData<Order> OrderTargetVolume(string symbol, int volume, OrderSide positionSide, Orc
```

参数：

参数名	类型	说明
symbol	string	标的代码，只能单个标的
volume	int	期望的最终数量
positionSide	PositionSide	持仓方向 参见 `enum PositionSide`)
orderType	OrderType	委托类型 参见 `enum OrderType`
price	double	委托价格（限价委托时的委托价，沪市市价委托的市价保护价，多位小数时会四舍五入，股票只保留两位，基金只保留三位）
account	string	实盘账号id,关联多实盘账号时填写，可以从 GetAccounts获取，也可以从终端实盘账号配置里拷贝。如果策略只关联一个账号，可以设置为null
返回值	Order	一个Order结构, 如果函数调用失败，Order.status 值为 `OrderStatus_Rejected` , Order.ordRejReasonDetail 为错误原因描述, 其它情况表示函数调用成功，Order.clOrdId 为本次委托的标识，可用于追溯订单状态或撤单

示例：

```
//当前SHSE.600000多方向持仓量为0，期望持仓量为10000，下单量为期望持仓量 - 当前持仓量 = 1000
GMData<Order> o = OrderTargetVolume("SHSE.600000", 10000, PositionSide.PositionSide_Lo
```

```
//600000浦发银行需要全部卖出时，volume设置为0，position_side设置为PositionSide_Long，表示
GMData<Order> o = OrderTargetVolume("SHSE.600000", 0, PositionSide.PositionSide_Long,
```

注意：

1. 仅支持一个标的代码，若交易代码输入有误，终端会拒绝此单，并显示 `委托代码不正确` 。

2. 根据目标数量计算下单数量，系统判断开平仓类型。若下单数量有误，终端拒绝此单，并显示 `委托量不正确` 。若实际需要买入数量为0，则订单会被拒绝， `终端无显示，无回报` 。股票买卖最小单位为 `100` ，不足100部分 `向下取整` ，如存在不足100的持仓一次性卖出;期货买卖最小单位为 `1` ， `向下取整` 。

3. 若仓位不足，终端会拒绝此单，显示 `仓位不足` 。平仓时股票默认 `平昨仓` ，期货默认 `平今仓` 。应研究需要， `股票也支持卖空操作` 。

4. OrderType优先级高于price,若指定OrderTpye_Market下市价单，使用价格为最新一个tick中的最新价，price参数失效。则price参数失效。若OrderTpye_Limit限价单，仿真模式价格错误，终端拒绝此单，显示委托价格错误， `回测模式下对价格无限制` 。

5. 函数调用成功并不意味着委托已经成功，只是意味委托单已经成功发出去， 委托是否成功根据OnOrderStatus，或 GetOrder来判断。

6. 股票交易position_side 仅支持设置多方向。

OrderTargetValue - 调仓到目标持仓额

调仓到目标持仓额, 如果调用成功，后续委托单状态变化将会触发OnOrderStatus回调。

函数原型:

```
GMData<Order> OrderTargetValue(string symbol, double value, PositionSide positionSide,
```

参数：

参数名	类型	说明
symbol	string	标的代码，只能单个标的
value	int	期望的股票最终价值
positionSide	PositionSide	持仓方向 参见 `enum PositionSide`)
orderType	OrderType	委托类型 参见 `enum OrderType`
price	double	委托价格（限价委托时的委托价，沪市市价委托的市价保护价，多位小数时会四舍五入，股票只保留两位，基金只保留三位）
account	string	实盘账号id,关联多实盘账号时填写，可以从 get_accounts获取，也可以从终端实盘账号配置里拷贝。如果策略只关联一个账号，可以设置为null
返回值	Order	一个Order结构, 如果函数调用失败，Order.status 值为 `OrderStatus_Rejected` , Order.ordRejReasonDetail 为错误原因描述, 其它情况表示函数调用成功，Order.clOrdId 为本次委托的标识，可用于追溯订单状态或撤单

示例：

```
//当前SHSE.600000多方向当前持仓量为0，目标持有价值为100000的该股票，根据value / price 计算
GMData<Order> o = OrderTargetValue("SHSE.600000", 100000, PositionSide.PositionSide_Lo
```

```
//600000浦发银行需要全部卖出时，value设置为0，position_side设置为PositionSide_Long，表示把
GMData<Order> o = OrderTargetValue("SHSE.600000", 0, PositionSide.PositionSide_Long, C
```

注意：

1. 仅支持一个标的代码，若交易代码输入有误，终端会拒绝此单，并显示 `委托代码不正确` 。

2. 根据目标数量计算下单数量，系统判断开平仓类型。若下单数量有误，终端拒绝此单，并显示 `委托量不正确` 。若实际需要买入数量为0，则订单会被拒绝， `终端无显示，无回报` 。股票买卖最小单位为 `100` ，不足100部分 `向下取整` ，如存在不足100的持仓一次性卖出;期货买卖最小单位为 `1` ， `向下取整` 。

3. 若仓位不足，终端会拒绝此单，显示 `仓位不足` 。平仓时股票默认 `平昨仓` ，期货默认 `平今仓` 。应研究需要， `股票也支持卖空操作` 。

4. OrderType优先级高于price,若指定OrderTpye_Market下市价单，使用价格为最新一个tick中的最新价，price参数失效。则price参数失效。若OrderTpye_Limit限价单，仿真模式价格错误，终端拒绝此单，显示委托价格错误， `回测模式下对价格无限制` 。

5. 函数调用成功并不意味着委托已经成功，只是意味委托单已经成功发出去， 委托是否成功根据OnOrderStatus，或 GetOrder来判断。

6. 股票交易position_side 仅支持设置多方向。

OrderTargetPercent - 调仓到目标持仓比例（总资产的比例）

调仓到目标持仓比例（总资产的比例）, 如果调用成功，后续委托单状态变化将会触发OnOrderStatus回调。

函数原型:

```
GMData<Order> OrderTargetPercent(string symbol, double percent, PositionSide positionS
```

参数：

参数名	类型	说明
symbol	string	标的代码，只能单个标的
percent	double	期望的最终占总资产比例
positionSide	PositionSide	持仓方向 参见 `enum PositionSide`)
orderType	OrderType	委托类型 参见 `enum OrderType`
price	double	委托价格（限价委托时的委托价，沪市市价委托的市价保护价，多位小数时会四舍五入，股票只保留两位，基金只保留三位）
account	string	实盘账号id,关联多实盘账号时填写，可以从 get_accounts获取，也可以从终端实盘账号配置里拷贝。如果策略只关联一个账号，可以设置为null
返回值	Order	一个Order结构, 如果函数调用失败，Order.status 值为 `OrderStatus_Rejected` , Order.ordRejReasonDetail 为错误原因描述, 其它情况表示函数调用成功，Order.clOrdId 为本次委托的标识，可用于追溯订单状态或撤单

示例：

```
//当前总资产价值为1000000，目标为以11元每股的价格买入SHSE.600000的价值占总资产的10%，根据vo

GMData<Order> o = OrderTargetPercent("SHSE.600000", 0.1, PositionSide.PositionSide_Lor
```

```
//600000浦发银行需要全部卖出时，percent设置为0，position_side设置为PositionSide_Long，表示

GMData<Order> o = OrderTargetPercent("SHSE.600000", 0, PositionSide.PositionSide_Long,
```

注意：

1. 仅支持一个标的代码，若交易代码输入有误，终端会拒绝此单，并显示 `委托代码不正确` 。

2. 根据目标比例计算下单数量，为占 `总资产(nav)` 比例，系统判断开平仓类型。若下单数量有误，终端拒绝此单，并显示 `委托量不正确` 。若实际需要买入数量为0，则本地拒绝此单， `终端无显示，无回报` 。股票买卖最小单位为 `100` ，不足100部分 `向下取整` ，如存在不足100的持仓一次性卖出;期货买卖最小单位为 `1` ， `向下取整` 。

3. 若仓位不足，终端会拒绝此单，显示 `仓位不足` 。平仓时股票默认 `平昨仓` ，期货默认 `平今仓` 。应研究需要， `股票也支持卖空操作` 。

4. OrderType优先级高于price,若指定OrderTpye_Market下市价单，使用价格为最新一个tick中的最新价，price参数失效。则price参数失效。若OrderTpye_Limit限价单，仿真模式价格错误，终端拒绝此单，显示委托价格错误， `回测模式下对价格无限制` 。

5. 函数调用成功并不意味着委托已经成功，只是意味委托单已经成功发出去， 委托是否成功根据OnOrderStatus，或 GetOrder来判断。

6. 股票交易position_side 仅支持设置多方向。

OrderCloseAll - 平当前所有可平持仓

平当前所有可平持仓, 如果调用成功，后续委托单状态变化将会触发OnOrderStatus回调

函数原型:

```
GMDataList<Order> OrderCloseAll()
```

参数：

参数名	类型	说明
返回值	`GMDataList<order>`	一个 `GMDataList<order>` 对象

OrderCancel - 委托撤单

撤销单个委托单, 如果调用成功，后续委托单状态变化将会触发OnOrderStatus回调

函数原型:

```
int OrderCancel(string clOrdIds, string account = null)
```

参数：

参数名	类型	说明
clOrdIds	string	委托单的客户id, 可以在下单或查单时获得
account	string	实盘账号id, 关联多实盘账号时填写，可以从GetAccounts获取，也可以从终端实盘账号配置里拷贝。如果策略只关联一个账号，可以设置为null
返回值	int	成功返回0， 失败返回错误码

OrderCancelAll - 撤销所有委托

撤销所有委托, 如果调用成功，后续委托单状态变化将会触发OnOrderStatus回调

函数原型:

```
int OrderCancelAll();
```

参数：

参数名	类型	说明
返回值	int	成功返回0， 失败返回错误码

GetOrders - 查询所有委托

查询所有委托单

函数原型:

```
GMDataList<Order> GetOrders(string account = null)
```

参数：

参数名	类型	说明
account	string	账号ID `accountId`, 如果输入为null, 则返回所有账号的委托
返回值	`GMDataList<Order>`	一个 `GMDataList<order>` 对象

GetUnfinishedOrders - 查询未结委托

查询所有未结委托

函数原型:

```
GMDataList<Order> GetUnfinishedOrders(string account = null)
```

参数：

参数名	类型	说明
account	string	账号ID `accountId`, 如果输入为null, 则返回所有账号的委托
返回值	`GMDataList<Order>`	一个 `GMDataList<order>` 对象

GetExecutionReports - 查询成交

查询所有成交

函数原型:

```
GMDataList<ExecRpt> GetExecutionReports(string account = null)
```

参数：

参数名	类型	说明
account	string	账号ID `accountId`，如果输入为null，则返回所有账号的成交
返回值	`GMDataList<ExecRpt>`	一个 `GMDataList<ExecRpt>` 对象

GetCash - 查询资金

查询资金

函数原型：

```
GMDataList<Cash> GetCash(string account = null)
```

参数：

参数名	类型	说明
account	string	账号ID `accountId`，如果输入为NULL，则返回所有账号的资金
返回值	`GMDataList<Cash>`	一个 `GMDataList<Cash>` 对象

GetPosition - 查询持仓

查询所有持仓

函数原型：

```
GMDataList<Position> GetPosition(string account = null)
```

参数：

参数名	类型	说明
account	string	账号ID `accountId`，如果输入为null，则返回所有账号的持仓
返回值	`GMDataList<Position>`	一个 `GMDataList<Position>` 对象

动态参数成员函数

AddParameters - 添加参数

添加动态参数， 添加成功后， 参数将在终端上显示。

函数原型:

```
int AddParameters(List<Parameter> parameters)
int AddParameters(Parameter parameter)
```

参数：

参数名	类型	说明
parameters	List	`Parameter` 列表
parameter	Parameter	一个 `Parameter` 对象
返回值	int	成功返回0， 失败返回错误码

DelParameters - 删除参数

删除动态参数

函数原型:

```
int DelParameters(string keys)
```

参数：

参数名	类型	说明
keys	string	对应参数的键值, 多个参数使用 `,` 间隔， 如 `key1,key2,...`
返回值	int	成功返回0， 失败返回错误码

SetParameters - 设置参数

设置参数值

函数原型:

```
int SetParameters(List<Parameter> parameters)
int SetParameters(Parameter parameter)
```

参数：

参数名	类型	说明
parameters	List	`Parameter` 列表
parameter	Parameter	`Parameter` 对象
返回值	int	成功返回0， 失败返回错误码

GetParameters - 获取参数

获取参数值

函数原型:

```
GMDataList<Parameter> GetParameters()
```

参数：

参数名	类型	说明
返回值	GMDataList	一个 `GMDataList` 对象

SetSymbols - 设置标的

设置交易标的， 设置成功后， 标的将在终端上显示。

函数原型:

```
int SetSymbols(string symbols);
```

参数：

参数名	类型	说明
symbols	string	symbol列表，逗号分隔
返回值	int	成功返回0， 失败返回错误码

GetSymbols - 获取标的

获取交易标的

函数原型:

```
GMDataList<string> GetSymbols()
```

参数：

参数名	类型	说明
返回值	GMDataList	一个 `GMDataList` 对象

事件成员函数

OnInit - 初始化完成

sdk初始化完成时触发, 用户可以改写此成员函数，在些订阅行情，提取历史数据等初始化操作。

函数原型:

```
virtual void OnInit();
```

OnTick - 收到Tick行情

收到Tick行情时触发

函数原型:

```
virtual void OnTick(Tick tick);
```

参数：

参数名	类型	说明
tick	Tick	收到的Tick行情

OnBar - 收到bar行情

收到bar行情时触发

函数原型:

```
virtual void OnBar(Bar bar);
```

参数：

参数名	类型	说明
bar	List	收到的Bar行情

OnOrderStatus - 委托变化

委托变化时触发

函数原型:

```
virtual void OnOrderStatus(Order order);
```

参数：

参数名	类型	说明
order	Order	发生变化的委托

OnExecutionReport - 执行回报

收到回报时触发

函数原型:

```
virtual void OnExecutionReport(ExecRpt rpt);
```

参数:

参数名	类型	说明
rpt	ExecRpt	收到的回报

OnParameter - 参数变化

参数变化时触发, 一般是终端修了动态参数

函数原型:

```
virtual void OnParameter(List<Parameter> param);
```

参数:

参数名	类型	说明
param	List	变化的参数

OnSchedule - 定时任务触发

预设任务时间条件符合时触发

函数原型:

```
virtual void OnSchedule(string dataRule, string timeRule);
```

参数:

参数名	类型	说明
dataRule	string	设置的 dataRule
timeRule	string	设置的 timeRule

OnBacktestFinished - 回测完成后收到绩效报告

回测完成后收到绩效报告时触发

函数原型:

```
virtual void OnBacktestFinished(Indicator indicator);
```

参数：

参数名	类型	说明
dataRule	Indicator	设置的 dataRule

OnAccountStatus - 实盘账号状态变化

实盘账号状态变化时触发, 比如实盘账号登录，退出登录等

函数原型：

```
virtual void OnAccountStatus(AccountStatus accountStatus);
```

参数：

参数名	类型	说明
accountStatus	AccountStatus	对应变化的账号

OnError - 错误产生

有错误产生时触发, 比如网络断开。

函数原型：

```
virtual void OnError(int errorCode, string errorMsg);
```

参数：

参数名	类型	说明
errorCode	int	错误码
errorMsg	string	错误信息

OnStop - 收到策略停止信号

终端点击停止策略时触发

函数原型：

```
virtual void OnStop();
```

OnMarketDataConnected - 数据服务已经连接上

数据服务已经连接时触发

函数原型：

```
virtual void OnMarketDataConnected();
```

OnTradeDataConnected - 交易已经连接上

交易已经连接时触发

函数原型:

```
virtual void OnTradeDataConnected();
```

OnMarketDataDisconnected - 数据连接断开了

数据连接断开时触发

函数原型:

```
virtual void OnMarketDataDisconnected();
```

OnTradeDataDisconnected - 交易连接断开了

交易连接断开时触发

函数原型:

```
virtual void OnTradeDataDisconnected();
```

数据查询函数

行情数据查询函数（免费）

GMApi 静态方法

Current - 查询当前行情快照

查询当前行情快照，返回tick数据。回测时，返回回测时间点的tick数据

函数原型:

```
static GMDataList<Tick> Current(string symbols);
```

参数：

参数名	类型	说明
symbols	string	查询代码，如有多个代码, 中间用 `,` (英文逗号) 隔开
返回值	GMDataList	一个 `GMDataList<Tick>` 对象

示例：

```
static GMDataList<Tick> currentData = GMApi.Current("SZSE.000001,SZSE.000002");
```

注意：

startTime和endTime中月,日,时,分,秒均可以只输入个位数,例: `"2010-7-8 9:40:0"` 或 `"2017-7-30 12:3:0"` ,但若对应位置为零，则 `0` 不可被省略,如不可输入 `"2017-7-30 12:3: "`

HistoryTicks - 查询历史Tick行情

函数原型:

```
static GMDataList<Tick> HistoryTicks(string symbols, string startTime, string endTime,
```

参数：

参数名	类型	说明
symbols	string	标的代码,若要获取多个标的的历史数据，可以采用中间用 `,` (英文逗号) 隔开的方法
startTime	string	开始时间 (%Y-%m-%d %H:%M:%S 格式),其中日线包含start_time数据, 非日线不包含start_time数据
endTime	string	结束时间 (%Y-%m-%d %H:%M:%S 格式),
adjust	Adjust	复权方式，参见 `enum Adjust`
adjustEndTime	string	复权基点时间, 默认当前时间
skipSuspended	bool	是否跳过停牌, 默认跳过
fillMissing	string	填充方式, `None` - 不填充, `'NaN'` - 用空值填充, `'Last'` - 用上一个值填充, 默认None
返回值	GMDataList	一个 `GMDataList<Tick>` 对象

示例：

```
GMDataList<Tick> historyTick = GMApi.HistoryTicks("SHSE.000300", "2010-07-28 08:00:00"
```

注意：

1.startTime和endTime中月,日,时,分,秒均可以只输入个位数,例: `"2010-7-8 9:40:0"` 或 `"2017-7-30 12:3:0"` ,但若对应位置为零，则 `0` 不可被省略,如不可输入 `"2017-7-30 12:3: "`

2. skipSuspended 和 fillMissing 参数暂不支持

HistoryBars - 查询历史Bar行情

函数原型:

```
static GMDataList<Bar> HistoryBars(string symbols, string frequency, string startTime,
```

参数：

参数名	类型	说明
symbols	string	标的代码,若要获取多个标的的历史数据，可以采用中间用 `,` (英文逗号) 隔开的方法
frequency	string	频率, 支持 '1d', '60s'等, 默认 '1d', 详情见订阅频率
startTime	string	开始时间 (%Y-%m-%d %H:%M:%S 格式),其中日线包含start_time数据, 非日线不包含start_time数据
endTime	string	结束时间 (%Y-%m-%d %H:%M:%S 格式),
adjust	Adjust	复权方式，参见 `enum Adjust`
adjustEndTime	string	复权基点时间, 默认当前时间
skipSuspended	bool	是否跳过停牌, 默认跳过
fillMissing	string	填充方式，`None` - 不填充，`'NaN'` - 用空值填充，`'Last'` - 用上一个值填充, 默认None
返回值	GMDataList	一个 `GMDataList<Bar>` 对象

示例：

```
//获取1分钟的bar
GMDataList<Bar> historyBar = GMApi.HistoryBars("SHSE.000300", "60s", "2010-07-28 08:00

//获取60分钟的bar
GMDataList<Bar> historyBar = GMApi.HistoryBars("SHSE.000300", "3600s", "2010-07-28 08:

//获取日线
GMDataList<Bar> historyBar = GMApi.HistoryBars("SHSE.000300", "1d", "2010-07-28 08:00:
```

注意：

1.startTime和endTime中月,日,时,分,秒均可以只输入个位数,例: `"2010-7-8 9:40:0"` 或 `"2017-7-30 12:3:0"` ,但若对应位置为零，则 `0` 不可被省略,如不可输入 `"2017-7-30 12:3: "`

2. skipSuspended 和 fillMissing 参数暂不支持

HistoryTicksN - 查询最新n条Tick行情

函数原型:

```
static GMDataList<Tick> HistoryTicksN(string symbols, int n, string endTime = null, Ac
```

参数：

参数名	类型	说明
symbols	string	标的代码(只允许单个标的的代码字符串)
n	int	获取行情的数量
endTime	string	从指定时间开始， `往前` 取行情, 如果为NULL, 那么为当前时间开始（回测模式下为当前回测时间点）
adjust	Adjust	复权方式，参见 `enum Adjust`
adjustEndTime	string	复权基点时间, 默认当前时间
skipSuspended	bool	是否跳过停牌, 默认跳过
fillMissing	string	填充方式, `None` - 不填充, `'NaN'` - 用空值填充, `'Last'` - 用上一个值填充, 默认None
返回值	GMDataList	一个 `GMDataList<Tick>` 对象

示例：

```
//获取沪深300最新10条tick
GMDataList<Tick> ticks = GMApi.HistoryTicksN("SHSE.000300", 10);
```

注意：

1.endTime中月,日,时,分,秒均可以只输入个位数,例: `"2010-7-8 9:40:0"` 或 `"2017-7-30 12:3:0"` ,但若对应位置为零，则 `0` 不可被省略,如不可输入 `"2017-7-30 12:3: "`

2. skipSuspended 和 fillMissing 参数暂不支持

HistoryBarsN - 查询最新n条Bar行情

函数原型:

```
static GMDataList<Bar> HistoryBarsN(string symbols, string frequency, int n, string er
```

参数：

参数名	类型	说明
symbols	string	标的代码(只允许单个标的的代码字符串)
frequency	string	频率, 支持 '1d', '60s'等, 默认 '1d', 详情见订阅频率
n	int	获取行情的数量
endTime	string	从指定时间开始， 往前 取行情, 如果为NULL, 那么为当前时间开始（回测模式下为当前回测时间点）
adjust	Adjust	复权方式，参见 `enum Adjust`
adjustEndTime	string	复权基点时间, 默认当前时间
skipSuspended	bool	是否跳过停牌, 默认跳过
fillMissing	string	填充方式，`None` - 不填充，`'NaN'` - 用空值填充，`'Last'` - 用上一个值填充, 默认None
返回值	GMDataList	一个 `GMDataList<Bar>` 对象

示例：

```
//获取沪深300最新10条1分钟bar
GMDataList<Bar> bars = GMApi.HistoryBarsN("SHSE.000300", "60s", 10);
```

注意：

1.endTime中月,日,时,分,秒均可以只输入个位数,例: `"2010-7-8 9:40:0"` 或 `"2017-7-30 12:3:0"` ,但若对应位置为零，则 `0` 不可被省略,如不可输入 `"2017-7-30 12:3: "`

2. skipSuspended 和 fillMissing 参数暂不支持

通用数据函数（免费）

SetToken - 设置token

函数原型

```
static int SetToken(string token);
```

参数

参数名	类型	说明
token	string	改参数通过 客户端->系统设置->(秘钥管理)token 获得
返回值	int	0 成功， 其他表示错误码

示例

```
GMApi.SetToken("27cbdfd8cd9b86dea554a5612baa4a8eee");
```

SetAddr - 设置终端服务地址

设置终端服务地址, 默认本地地址, 可不填

函数原型：

```
void SetAddr(string addr)
```

参数:

参数名	类型	说明
addr	string	ip+终端端口，如”127.0.0.1:7001”

注意： **1.**如运行策略与终端不在同一设备上，必须调用此方法，`7001` 为终端端口
2.不关注策略运行结果，可不启动终端并设置终端服务地址为
`discovery.myquant.cn:7061`

GetSymbolInfos - 查询标的基本信息

获取指定(范围)交易标的基本信息，与时间无关.

函数原型：

```
public static GMDataList<SymbolInfo> GetSymbolInfos(int secType1, int secType2 = 0, st
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
secType1	int	证券品种大类	Y	无	指定一种证券大类，只能输入一个. 证券大类 secType1 清单 1010: 股票， 1020: 基金，1030: 债券， 1040: 期货，1050: 期权， 1060: 指数，1070：板块.
secType2	int	证券品种细类	N	0	指定一种证券细类，只能输入一个. 默认 `0` 表示不区分细类，即证券大类下所有细类. 证券细类见 secType2 清单 - 股票 101001:A 股，101002:B 股，101003:存托凭证 - 基金 102001:ETF，102002:LOF，102005:FOF - 债券 103001:可转债，103008:回购 - 期货 104001:股指期货，104003:商品期货，104006:国债期货 - 期权 105001:股票期权，105002:指数期权，105003:商品期权 - 指数 106001:股票指数，106002:基金指数，106003:债券指数，106004:期货指数 - 板块：107001:概念板块
exchanges	string	交易所代码	N	null	输入交易所代码，可输入多个. 采用 string 格式时，多个交易所代码必须用英文逗号分割，如： `"SHSE,SZSE"` 默认 `null` 表示所有交易所. 交易所代码清单 SHSE:上海证券交易所，SZSE:深圳证券交易所， CFFEX:中金所，SHFE:上期所，DCE:大商所，CZCE:郑商所， INE:上海国际能源交易中心，GFEX:广期所
symbols	string	标的代码	N	null	输入标的代码，可输入多个. 采用 string 格式时，多个标的代码必须用英文逗号分割，如： `"SHSE.600008,SZSE.000002"` 默认 `null` 表示所有标的.

返回值：

`SymbolInfo` 结构列表，参见 `SymbolInfo` 定义与 `GMDataList` 类的用法。

示例：

```
GetSymbolInfos(1010, 0, null, "SHSE.600008,SZSE.000002")
```

注意：

1. `secType1` 为必填参数，即一次只能查询一个品种的标的基本信息。

2. 查询的标的信息根据参列表合 `secType1, secType2, exchanges, symbols` 取交集，若输入参数之间出现任何矛盾（换句话说，所有的参数限制出满足要求的交集为空)，则返回 `空`
3. 若输入包含无效标的代码 `symbols` ，则返回只包含有效标的代码对应的数据。
4. 参列表合示例：

查询以下范围 symbol 的基本信息	secType1	secType2	exchanges	symbols
查询指定股票	1010	0	null	'SHSE.600008,SZ
查询 A 股股票	1010	101001	null	null
查询深交所股票	1010	0	'SZSE'	null
查询 ETF	1020	102001	null	null
查询上交所 LOF	1020	102002	'SHSE'	null
查询可转债	1030	103001	null	null
查询深交所可转债	1030	103001	'SZSE'	null
查询股指期货	1040	104001	null	null
查询商品期货	1040	104003	null	null
查询郑商所和大商所期货	1040	0	'CZCE,DCE'	null
查询股票期权	1050	105001	null	null
查询上交所股票期权	1050	105001	'SHSE'	null
查询指数期权	1050	105002	null	null
查询商品期权	1050	105003	null	null
查询上期所商品期权	105003	0	'SHFE'	null
查询股票指数	1060	106001	null	null

GetSymbols - 查询指定交易日多标的交易信息

获取指定交易日多个标的交易信息，包括基本信息及日度数据.

函数原型：

```
public static GMDataList<SymbolContent> GetSymbols(int secType1, int secType2 = 0, str
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
secType1	int	证券品种大类	Y	无	指定一种证券大类，只能 一个. 证券大类 secType 单 1010: 股票， 1020: 1030: 债券， 1040: 期 1050: 期权， 1060: 指数 1070：板块.
secType2	int	证券品种细类	N	0	指定一种证券细类，只能 一个. 默认 `0` 表示不区分 类，即证券大类下所有细 证券细类见 secType2 说 股票 101001:A 股，101 股，101003:存托凭证 - 102001:ETF，102002:I 102005:FOF - 债券 103 可转债，103008:回购 - 104001:股指期货，104 商品期货，104006:国债 - 期权 105001:股票期权 105002:指数期权，105 商品期权 - 指数 106001 指数，106002:基金指数 106003:债券指数，106 期货指数 - 板块：1070(念板块
exchanges	string	交易所代码	N	null	输入交易所代码，可输入 个. 采用 string 格式时， 交易所代码必须用英文逗 割，如： `"SHSE,SZSE"` 默 认 `null` 表示所有交易所 所代码清单 SHSE:上海 交易所，SZSE:深圳证券 所， CFFEX:中金所，S 上期所，DCE:大商所， CZCE:郑商所， INE:上 际能源交易中心，GFE 期所
symbols	string	标的代码	N	null	输入标的代码，可输入多 采用 string 格式时，多个 代码必须用英文逗号分割 如： `"SHSE.600008,SZSE.00` 默认 `null` 表示所有标的
skipSuspended	bool	跳过停牌	N	true	是否跳过全天停牌，默 认 `true` 跳过
skipSt	bool	跳过ST	N	true	是否跳过包含 ST 的股 票： `ST`, `ST`, `SST`, `SST` , 默 认 `true` 跳过
tradeDate	string	交易日期	N	null	交易日期，%Y-%m-%d 式，默认 `null` 取最新截 含退市标的)

返回值：

`SymbolContent` 结构列表，参见 `SymbolContent` 定义与 `GMDataList` 类的用法。

示例：

```
GetSymbols(secType1=1010, 0, null, "SHSE.600008,SZSE.000002")
```

注意：

1. `secType1` 为必填参数，即一次只能查询一个品种的标的最新交易日信息。
2. 查询的标的信息根据参列表合 `secType1, secType2, exchanges, symbols` 取交集，若输入参数之间出现任何矛盾（换句话说，所有的参数限制出满足要求的交集为空），则返回 `空`
3. 若输入包含无效标的代码 `symbols` ，则返回只包含有效标的代码对应的数据。
4. 获取全 A 股票代码示例 `GetSymbols(1010, 101001)`
5. 可转债的到期日(退市日期)为 `delistedDate` ，转股价值为 `转股价值 = 转股数股价 = (100/可转债转股价) 股价`

GetHistorySymbol - 查询指定标的多日交易信息

获取指定标的多个历史交易日的交易信息，包括基本信息及日度数据.

函数原型：

```
public static GMDataList<SymbolContent> GetHistorySymbol(string symbol, string startDa
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	标的代码	Y	无	输入标的代码，只能输入一个.
startDate	string	开始时间	Y	null	开始时间日期，%Y-%m-%d 格式.
endDate	string	结束时间	Y	null	结束时间日期，%Y-%m-%d 格式.

返回值：

`SymbolContent` 结构列表，参见 `SymbolContent` 定义与 `GMDataList` 类的用法。

示例：

```
GetHistorySymbol("SZSE.000002", "2022-09-01", "2022-09-30")
```

注意：

1. 若输入包含无效标的代码 `symbol` ，则返回只包含有效标的代码对应的数据。
2. 停牌时且股票发生除权除息，涨停价和跌停价可能有误差。

3. 对每个标的，数据根据 `tradeDate` 的升序进行排序。
4. `startDate` 和 `endDate`，例: `'2010-7-8'或'2017-7-30'`
5. 当 `startDate` 大于或者等于 `endDate` 时, 取指定时间段的数据,当 `startDate` > `endDate` 时, 返回报错
6. 可转债的到期日(退市日期) `delistedDate`，转股价值为 `转股价值 = 转股数股价 = (100/可转债转股价) 股价`

GetTradingDatesByYear - 查询年度交易日历

查询一个交易所的指定年份的交易日历.

函数原型：

```
public static GMDataList<TradingDateContent> GetTradingDatesByYear(string exchange, ir
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
exchange	string	交易所代码	Y	无	只能填写一个交易所代码 交易所代码清单: SHSE:上海证券交易所，SZSE:深圳证券交易所，CFFEX:中金所，SHFE:上期所，DCE:大商所，CZCE:郑商所，INE:上海国际能源交易中心，GFEX:广期所
startYear	int	开始年份	Y	无	查询交易日历开始年份（含），yyyy 格式
endYear	int	结束年份	Y	无	查询交易日历结束年份（含），yyyy 格式

返回值:

`TradingDateContent` 结构列表，参见 `TradingDateContent` 定义与 `GMDataList` 类的用法。

示例：

```
GetTradingDatesByYear("SHSE", 2020, 2023)
```

注意：

1. `exchange` 参数仅支持输入单个交易所代码，若代码错误，会报错
2. 开始年份必须不晚于结束年份，否则返回 `空`

GetTradingSession - 查询交易时段

查询一个标的所属品种交易时间段.

函数原型：

```
public static GMDataList<TradingSession> GetTradingSession(string symbols);
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbols	string	标的代码	Y	无	输入标的代码，可输入多个. 多个标的代码必须用英文逗号分割， 如： `"SHSE.600008,SZSE.000002"`

返回值：

`TradingSession` 结构列表，参见 `TradingSession` 定义与 `GMDataList` 类的用法。

示例：

```
GetTradingSession("SHFE.au2306")
```

注意：

1. 如果输入不存在的合约代码 symbol，会报错提示"该合约[symbol]不存在"。

GetContractExpireRestDays - 查询合约到期剩余天数

查询期货合约、期权合约、可转债的到期剩余天数。

函数原型：

```
public static GMDataList<ContractExpireRestDays> GetContractExpireRestDays(string symb
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbols	string	标的代码	Y	无	输入标的代码，可输入多个.多个标的代码必须用英文逗号分割， 如： `"CFFEX.IF2212,CFFEX.IC221`
startDate	string	开始日期	Y	无	%Y-%m-%d 格式，不早于合约上市日
endDate	string	结束日期	Y	无	%Y-%m-%d 格式，不早于指定的开始日期，否则返回报错
tradeFlag	bool	交易日	N	false	是否需要按交易日计算，默认 `false` 按自然日计算，则返回到期剩余自然日天数; 设置为 `true` 按交易日计算，则返回到期剩余交易日天数

返回值：

`ContractExpireRestDays` 结构列表，参见 `ContractExpireRestDays` 定义与 `GMDataList` 类的用法。

示例：

```
GetContractExpireRestDays("CFFEX.IM2212", "2022-12-12", "2022-12-16", true)
```

注意：

1. 参数 `startDate` 和 `endDate` 必须是yyyy-mm-dd字符串格式
2. 在到期日当天，到期剩余天数为 0。正数表示距离到期日的剩余天数，0 表示到期日当天，负数表示距离到期日已经过去的天数。
3. 如果输入不存在的合约代码 `symbol` ，会报错提示"该合约[symbol]不存在"。
4. 如果输入的合约代码 `symbol` 在时间段内的某个日期未上市，在该日期的到期剩余天数返回空。
5. 用于剩余天数计算的到期日是最后交易日。

股票财务数据及基础数据函数（免费）

StkGetIndexConstituents - 查询指数成分股

查询指定指数在最新交易日的成分股和权重

函数原型：

```
public static GMDataList<IndexConstituent> StkGetIndexConstituents(string symbol, stri
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
index	string	指数代码	Y	无	必填，只能输入一个指数，如：`'SHSE.000905'`
tradeDate	string	交易日期	N	null	交易日期，%Y-%m-%d 格式， 默认 `null` 为最新交易日

返回值：

`IndexConstituent` 结构列表，参见 `IndexConstituent` 定义与 `GMDataList` 类的用法。

示例：

```
StkGetIndexConstituents("SHSE.000300")
```

注意：

1. 数据日频更新，在交易日约 20 点更新当日数据。如果当日数据尚未更新，调用时不指定 `tradeDate` 会返回前一交易日的成分数据，调用时指定 `tradeDate` 为当日会返回空。
2. `tradeDate` 输入非交易日，会返回空。 `tradeDate` 出入的日期格式错误，会报错。
3. 指数列表参考

StkGetFundamentalsBalance - 查询资产负债表数据

查询指定时间段某一股票所属上市公司的资产负债表数据

函数原型：

```
public static GMDataList<FundamentalsBalance> StkGetFundamentalsBalance(string symbol,
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	股票代码	Y	无	必填，只能填一个股票标的
fields	string	返回字段	Y	无	指定需要返回的财务字段，如有多个字段，中间用英文逗号分隔
rptType	int	报表类型	N	0	按报告期查询可指定以下报表类型： 1-一季度报 6-中报 9-前三季报 12-年报 默认 `0` 为不限
dataType	int	数据类型	N	0	在发布原始财务报告以后，上市公司可能会对数据进行修正。 101-合并原始 102-合并调整 201-母公司原始 202-母公司调整 默认 `0` 返回当期合并调整，如果没有调整返回合并原始
startDate	string	开始时间	N	null	开始时间，时间类型为报告日期，%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间，时间类型为报告日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
pubDate	string	发布日期	若数据类型选择合并原始(`dataType=101`)，则返回原始发布的发布日期 若数据类型选择合并调整(`dataType=102`)，则返回调整后最新发布日期 若数据类型选择母公司原始(`dataType=201`)，则返回母公司原始发布的发布日期 若数据类型选择母公司调整(`dataType=202`)，则返回母公司调整后最新发布日期
rptDate	string	报告日期	报告截止日期，财报统计的最后一天 在指定时间段[开始时间,结束时间]内的报告截止日期
rptType	int	报表类型	返回数据的报表类型：1-一季度报, 6-中报, 9-前三季报, 12-年报
dataType	int	数据类型	返回数据的数据类型：101-合并原始, 102-合并调整, 201-母公司原始, 202-母公司调整
[fields]	float	财务字段数据	指定返回 `fields` 字段的数值. 支持的字段名请参考 资产负债表

示例：

```
StkGetFundamentalsBalance("SHSE.600000", "cash_bal_cb")
```

注意：

1. 当 `startDate` == `endDate` 时，取离 `endDate` 最近报告日期的一条数据，

当 `startDat` < `endDate` 时，取指定时间段的数据，

当 `startDate` > `endDate` 时，返回报错。

1. 若在指定历史时间段内，有多个同一类型报表（如不同年份的一季度报表），将按照报告日期顺序返回。
2. 如果 `fields` 参数的财务字段填写不正确，或填写空字段，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

资产负债表

字段名	类型	中文名称	量纲	说明
流动资产(资产)				
cash_bal_cb	float	现金及存放中央银行款项	元	银行
dpst_ob	float	存放同业款项	元	银行
mny_cptl	float	货币资金	元	
cust_cred_dpst	float	客户信用资金存款	元	证券
cust_dpst	float	客户资金存款	元	证券
pm	float	贵金属	元	银行
bal_clr	float	结算备付金	元	
cust_rsv	float	客户备付金	元	证券
ln_to_ob	float	拆出资金	元	
fair_val_fin_ast	float	以公允价值计量且其变动计入当期损益的金融资产	元	
ppay	float	预付款项	元	
fin_out	float	融出资金	元	
trd_fin_ast	float	交易性金融资产	元	
deriv_fin_ast	float	衍生金融资产	元	
note_acct_rcv	float	应收票据及应收账款	元	
note_rcv	float	应收票据	元	
acct_rcv	float	应收账款	元	
acct_rcv_fin	float	应收款项融资	元	
int_rcv	float	应收利息	元	
dvd_rcv	float	应收股利	元	
oth_rcv	float	其他应收款	元	
in_prem_rcv	float	应收保费	元	
rin_acct_rcv	float	应收分保账款	元	
rin_rsv_rcv	float	应收分保合同准备金	元	保险
rcv_un_prem_rin_rsv	float	应收分保未到期责任准备金	元	
rcv_clm_rin_rsv	float	应收分保未决赔偿准备金	元	保险
rcv_li_rin_rsv	float	应收分保寿险责任准备金	元	保险
rcv_lt_hi_rin_rsv	float	应收分保长期健康险责任准备金	元	保险

字段名	类型	中文名称	量纲	说明
ph_plge_ln	float	保户质押贷款	元	保险
ttl_oth_rcv	float	其他应收款合计	元	
rfd_dpst	float	存出保证金	元	证券、保险
term_dpst	float	定期存款	元	保险
pur_resell_fin	float	买入返售金融资产	元	
aval_sale_fin	float	可供出售金融资产	元	
htm_inv	float	持有至到期投资	元	
hold_for_sale	float	持有待售资产	元	
acct_rcv_inv	float	应收款项类投资	元	保险
invt	float	存货	元	
contr_ast	float	合同资产	元	
ncur_ast_one_y	float	一年内到期的非流动资产	元	
oth_cur_ast	float	其他流动资产	元	
cur_ast_oth_item	float	流动资产其他项目	元	
ttl_cur_ast	float	流动资产合计	元	
非流动资产(资产)				
loan_adv	float	发放委托贷款及垫款	元	
cred_inv	float	债权投资	元	
oth_cred_inv	float	其他债权投资	元	
lt_rcv	float	长期应收款	元	
lt_eqy_inv	float	长期股权投资	元	
oth_eqy_inv	float	其他权益工具投资	元	
rfd_cap_guar_dpst	float	存出资本保证金	元	保险
oth_ncur_fin_ast	float	其他非流动金融资产	元	
amor_cos_fin_ast_ncur	float	以摊余成本计量的金融资产（非流动）	元	
fair_val_oth_inc_ncur	float	以公允价值计量且其变动计入其他综合收益的金融资产（非流动）	元	
inv_prop	float	投资性房地产	元	
fix_ast	float	固定资产	元	
const_prog	float	在建工程	元	

字段名	类型	中文名称	量纲	说明
const_matl	float	工程物资	元	
fix_ast_dlpl	float	固定资产清理	元	
cptl_bio_ast	float	生产性生物资产	元	
oil_gas_ast	float	油气资产	元	
rig_ast	float	使用权资产	元	
intg_ast	float	无形资产	元	
trd_seat_fee	float	交易席位费	元	证券
dev_exp	float	开发支出	元	
gw	float	商誉	元	
lt_ppay_exp	float	长期待摊费用	元	
dfr_tax_ast	float	递延所得税资产	元	
oth_ncur_ast	float	其他非流动资产	元	
ncur_ast_oth_item	float	非流动资产其他项目	元	
ttl_ncur_ast	float	非流动资产合计	元	
oth_ast	float	其他资产	元	银行、证券、保险
ast_oth_item	float	资产其他项目	元	
ind_acct_ast	float	独立账户资产	元	保险
ttl_ast	float	资产总计	元	
流动负债(负债)				
brw_cb	float	向中央银行借款	元	
dpst_ob_fin_inst	float	同业和其他金融机构存放款项	元	银行、保险
ln_fm_ob	float	拆入资金	元	
fair_val_fin_liab	float	以公允价值计量且其变动计入当期损益的金融负债	元	
sht_ln	float	短期借款	元	
adv_acct	float	预收款项	元	
contr_liab	float	合同负债	元	
trd_fin_liab	float	交易性金融负债	元	
deriv_fin_liab	float	衍生金融负债	元	

字段名	类型	中文名称	量纲	说明
sell_repo_ast	float	卖出回购金融资产款	元	
cust_bnk_dpst	float	吸收存款	元	银行、保险
dpst_cb_note_pay	float	存款证及应付票据	元	银行
dpst_cb	float	存款证	元	银行
acct_rcv_adv	float	预收账款	元	保险
in_prem_rcv_adv	float	预收保费	元	保险
fee_pay	float	应付手续费及佣金	元	
note_acct_pay	float	应付票据及应付账款	元	
stlf_pay	float	应付短期融资款	元	
note_pay	float	应付票据	元	
acct_pay	float	应付账款	元	
rin_acct_pay	float	应付分保账款	元	
emp_comp_pay	float	应付职工薪酬	元	
tax_pay	float	应交税费	元	
int_pay	float	应付利息	元	
dvd_pay	float	应付股利	元	
ph_dvd_pay	float	应付保单红利	元	保险
indem_pay	float	应付赔付款	元	保险
oth_pay	float	其他应付款	元	
ttl_oth_pay	float	其他应付款合计	元	
ph_dpst_inv	float	保户储金及投资款	元	保险
in_contr_rsv	float	保险合同准备金	元	保险
un_prem_rsv	float	未到期责任准备金	元	保险
clm_rin_rsv	float	未决赔款准备金	元	保险
li_liab_rsv	float	寿险责任准备金	元	保险
lt_hi_liab_rsv	float	长期健康险责任准备金	元	保险
cust_bnk_dpst_fin	float	吸收存款及同业存放	元	
inter_pay	float	内部应付款	元	
agy_secu_trd	float	代理买卖证券款	元	
agy_secu_uw	float	代理承销证券款	元	
sht_bnd_pay	float	应付短期债券	元	

字段名	类型	中文名称	量纲	说明
est_cur_liab	float	预计流动负债	元	
liab_hold_for_sale	float	持有待售负债	元	
ncur_liab_one_y	float	一年内到期的非流动负债	元	
oth_cur_liab	float	其他流动负债	元	
cur_liab_oth_item	float	流动负债其他项目	元	
ttl_cur_liab	float	流动负债合计	元	
非流动负债（负债）				
lt_ln	float	长期借款	元	
lt_pay	float	长期应付款	元	
leas_liab	float	租赁负债		
dfr_inc	float	递延收益	元	
dfr_tax_liab	float	递延所得税负债	元	
bnd_pay	float	应付债券	元	
bnd_pay_pbd	float	其中:永续债	元	
bnd_pay_pfd	float	其中:优先股	元	
oth_ncur_liab	float	其他非流动负债	元	
spcl_pay	float	专项应付款	元	
ncur_liab_oth_item	float	非流动负债其他项目	元	
lt_emp_comp_pay	float	长期应付职工薪酬	元	
est_liab	float	预计负债	元	
oth_liab	float	其他负债	元	银行、证券、保险
liab_oth_item	float	负债其他项目	元	银行、证券、保险
ttl_ncur_liab	float	非流动负债合计	元	
ind_acct_liab	float	独立账户负债	元	保险
ttl_liab	float	负债合计	元	
所有者权益(或股东权益)				
paid_in_cptl	float	实收资本（或股本）	元	

字段名	类型	中文名称	量纲	说明
oth_eqy	float	其他权益工具	元	
oth_eqy_pfd	float	其中:优先股	元	
oth_eqy_pbd	float	其中:永续债	元	
oth_eqy_oth	float	其中:其他权益工具	元	
cptl_rsv	float	资本公积	元	
treas_shr	float	库存股	元	
oth_comp_inc	float	其他综合收益	元	
spcl_rsv	float	专项储备	元	
sur_rsv	float	盈余公积	元	
rsv_ord_rsk	float	一般风险准备	元	
trd_risk_rsv	float	交易风险准备	元	证券
ret_prof	float	未分配利润	元	
sugg_dvd	float	建议分派股利	元	银行
eqy_pcom_oth_item	float	归属于母公司股东权益其他项目	元	
ttl_eqy_pcom	float	归属于母公司股东权益合计	元	
min_sheqy	float	少数股东权益	元	
sheqy_oth_item	float	股东权益其他项目	元	
ttl_eqy	float	股东权益合计	元	
ttl_liab_eqy	float	负债和股东权益合计	元	

StkGetFundamentalsCashflow - 查询现金流量表数据

查询指定时间段某一股票所属上市公司的现金流量表数据

函数原型：

```
public static GMDataList<FundamentalsCashflow> StkGetFundamentalsCashflow(string symbo
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	股票代码	Y	无	必填，只能填一个股票标的
fields	string	返回字段	Y	无	指定需要返回的财务字段，如有多个字段，中间用英文逗号分隔
rptType	int	报表类型	N	0	按报告期查询可指定以下报表类型： 1-一季度报 6-中报 9-前三季报 12-年报 默认 `0` 为不限
dataType	int	数据类型	N	0	在发布原始财务报告以后，上市公司可能会对数据进行修正。 101-合并原始 102-合并调整 201-母公司原始 202-母公司调整 默认 `0` 返回当期合并调整，如果没有调整返回合并原始
startDate	string	开始时间	N	null	开始时间，时间类型为报告日期，%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间，时间类型为报告日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
pubDate	string	发布日期	若数据类型选择合并原始(`dataType=101`)，则返回原始发布的发布日期 若数据类型选择合并调整(`dataType=102`)，则返回调整后最新发布日期 若数据类型选择母公司原始(`dataType=201`)，则返回母公司原始发布的发布日期 若数据类型选择母公司调整(`dataType=202`)，则返回母公司调整后最新发布日期
rptDate	string	报告日期	报告截止日期，财报统计的最后一天 在指定时间段[开始时间,结束时间]内的报告截止日期
rptType	int	报表类型	返回数据的报表类型：1-一季度报, 6-中报, 9-前三季报, 12-年报
dataType	int	数据类型	返回数据的数据类型：101-合并原始, 102-合并调整, 201-母公司原始, 202-母公司调整
[fields]	float	财务字段数据	指定返回 `fields` 字段的数值. 支持的字段名请参考 现金流量表

示例：

```
StkGetFundamentalsCashflow("SHSE.600000", "cash_pay_fee")
```

注意：

1. 当 `startDate` == `endDate` 时，取离 `endDate` 最近报告日期的一条数据，

当 `startDat` < `endDate` 时，取指定时间段的数据，

当 `startDate` > `endDate` 时，返回报错。

1. 若在指定历史时间段内，有多个同一类型报表（如不同年份的一季度报表），将按照报告日期顺序返回。
2. 如果 `fields` 参数的财务字段填写不正确，或填写空字段，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

现金流量表

字段名	类型	中文名称	量纲	说明
一、经营活动产生的现金流量				
cash_rcv_sale	float	销售商品、提供劳务收到的现金	元	
net_incr_cust_dpst_ob	float	客户存款和同业存放款项净增加额	元	
net_incr_cust_dpst	float	客户存款净增加额	元	银行
net_incr_dpst_ob	float	同业及其他金融机构存放款项净增加额	元	银行
net_incr_brw_cb	float	向中央银行借款净增加额	元	
net_incr_ln_fm_oth	float	向其他金融机构拆入资金净增加额	元	
cash_rcv_orig_in	float	收到原保险合同保费取得的现金	元	
net_cash_rcv_rin_biz	float	收到再保险业务现金净额	元	
net_incr_ph_dpst_inv	float	保户储金及投资款净增加额	元	
net_decrdpst_cb_ob	float	存放中央银行和同业款项及其他金融机构净减少额	元	银行、保险
net_decr_cb	float	存放中央银行款项净减少额	元	银行
net_decr_ob_fin_inst	float	存放同业及其他金融机构款项净减少额	元	银行
net_cert_dpst	float	存款证净额	元	银行
net_decr_trd_fin	float	交易性金融资产净减少额	元	银行
net_incr_trd_liab	float	交易性金融负债净增加额	元	银行
cash_rcv_int_fee	float	收取利息、手续费及佣金的现金	元	
cash_rcv_int	float	其中：收取利息的现金	元	银行
cash_rcv_fee	float	收取手续费及佣金的现金	元	银行
net_incr_lnfm_sell_repo	float	拆入资金及卖出回购金融资产款净增加额	元	银行
net_incr_ln_fm	float	拆入资金净增加额	元	

字段名	类型	中文名称	量纲	说明
net_incr_sell_repo	float	卖出回购金融资产款净增加额	元	银行保险
net_decr_lnto_pur_resell	float	拆出资金及买入返售金融资产净减少额	元	银行
net_decr_ln_cptl	float	拆出资金净减少额	元	银行、保险
net_dect_pur_resell	float	买入返售金融资产净减少额	元	银行、保险
net_incr_repo	float	回购业务资金净增加额	元	
net_decr_repo	float	回购业务资金净减少额	元	证券
tax_rbt_rcv	float	收到的税费返还	元	
net_cash_rcv_trd	float	收到交易性金融资产现金净额	元	保险
cash_rcv_oth_oper	float	收到其他与经营活动有关的现金	元	
net_cash_agy_secu_trd	float	代理买卖证券收到的现金净额	元	证券
cash_rcv_pur_resell	float	买入返售金融资产收到的现金	元	证券、保险
net_cash_agy_secu_uw	float	代理承销证券收到的现金净额	元	证券
cash_rcv_dspl_debt	float	处置抵债资产收到的现金	元	银行
canc_loan_rcv	float	收回的已于以前年度核销的贷款	元	银行
cf_in_oper	float	经营活动现金流入小计	元	
cash_pur_gds_svc	float	购买商品、接受劳务支付的现金	元	
net_incr_ln_adv_cust	float	客户贷款及垫款净增加额	元	
net_decr_brw_cb	float	向中央银行借款净减少额	元	银行
net_incr_dpst_cb_ob	float	存放中央银行和同业款项净增加额	元	
net_incr_cb	float	存放中央银行款项净增加额	元	银行

字段名	类型	中文名称	量纲	说明
net_incr_ob_fin_inst	float	存放同业及其他金融机构款项净增加额	元	银行
net_decr_dpst_ob	float	同业及其他机构存放款减少净额	元	银行
net_decr_issu_cert_dpst	float	已发行存款证净减少额	元	银行
net_incr_lnto_pur_resell	float	拆出资金及买入返售金融资产净增加额	元	银行
net_incr_ln_to	float	拆出资金净增加额	元	银行、保险
net_incr_pur_resell	float	买入返售金融资产净增加额	元	银行、保险
net_decr_lnfm_sell_repo	float	拆入资金及卖出回购金融资产款净减少额	元	银行
net_decr_ln_fm	float	拆入资金净减少额	元	银行、证券
net_decr_sell_repo	float	卖出回购金融资产净减少额	元	银行、保险
net_incr_trd_fin	float	交易性金融资产净增加额	元	银行
net_decr_trd_liab	float	交易性金融负债净减少额	元	银行
cash_pay_indem_orig	float	支付原保险合同赔付款项的现金	元	
net_cash_pay_rin_biz	float	支付再保险业务现金净额	元	保险
cash_pay_int_fee	float	支付利息、手续费及佣金的现金	元	
cash_pay_int	float	其中：支付利息的现金	元	银行
cash_pay_fee	float	支付手续费及佣金的现金	元	银行
ph_dvd_pay	float	支付保单红利的现金	元	
net_decr_ph_dpst_inv	float	保户储金及投资款净减少额	元	保险
cash_pay_emp	float	支付给职工以及为职工支付的现金		

字段名	类型	中文名称	量纲	说明
cash_pay_tax	float	支付的各项税费	元	
net_cash_pay_trd	float	支付交易性金融资产现金净额	元	保险
cash_pay_oth_oper	float	支付其他与经营活动有关的现金	元	
net_incr_dspl_trd_fin	float	处置交易性金融资产净增加额	元	
cash_pay_fin_leas	float	购买融资租赁资产支付的现金	元	银行
net_decr_agy_secu_pay	float	代理买卖证券支付的现金净额（净减少额）	元	证券
net_decr_dspl_trd_fin	float	处置交易性金融资产的净减少额	元	证券
cf_out_oper	float	经营活动现金流出小计	元	
net_cf_oper	float	经营活动产生的现金流量净额	元	
二、投资活动产生的现金流量：				
cash_rcv_sale_inv	float	收回投资收到的现金	元	
inv_inc_rcv	float	取得投资收益收到的现金	元	
cash_rcv_dvd_prof	float	分得股利或利润所收到的现金	元	银行
cash_rcv_dspl_ast	float	处置固定资产、无形资产和其他长期资产收回的现金净额	元	
cash_rcv_dspl_sub_oth	float	处置子公司及其他营业单位收到的现金净额	元	
cash_rcv_oth_inv	float	收到其他与投资活动有关的现金	元	
cf_in_inv	float	投资活动现金流入小计	元	
pur_fix_intg_ast	float	购建固定资产、无形资产和其他长期资产支付的现金	元	
cash_out_dspl_sub_oth	float	处置子公司及其他营业单位流出的现金净额	元	保险
cash_pay_inv	float	投资支付的现金	元	

字段名	类型	中文名称	量纲	说明
net_incr_ph_plge_ln	float	保户质押贷款净增加额	元	保险
add_cash_pled_dpst	float	增加质押和定期存款所支付的现金	元	
net_incr_plge_ln	float	质押贷款净增加额	元	
net_cash_get_sub	float	取得子公司及其他营业单位支付的现金净额	元	
net_pay_pur_resell	float	支付买入返售金融资产现金净额	元	证券、保险
cash_pay_oth_inv	float	支付其他与投资活动有关的现金	元	
cf_out_inv	float	投资活动现金流出小计		
net_cf_inv	float	投资活动产生的现金流量净额	元	
三、筹资活动产生的现金流量：				
cash_rcv_cptl	float	吸收投资收到的现金	元	
sub_rcv_ms_inv	float	其中：子公司吸收少数股东投资收到的现金	元	
brw_rcv	float	取得借款收到的现金	元	
cash_rcv_bnd_iss	float	发行债券收到的现金	元	
net_cash_rcv_sell_repo	float	收到卖出回购金融资产款现金净额	元	保险
cash_rcv_oth_fin	float	收到其他与筹资活动有关的现金	元	
issu_cert_dpst	float	发行存款证	元	银行
cf_in_fin_oth	float	筹资活动现金流入其他项目	元	
cf_in_fin	float	筹资活动现金流入小计	元	
cash_rpay_brw	float	偿还债务支付的现金	元	
cash_pay_bnd_int	float	偿付债券利息支付的现金	元	银行
cash_pay_dvd_int	float	分配股利、利润或偿付利息支付的现金	元	

字段名	类型	中文名称	量纲	说明
sub_pay_dvd_prof	float	其中：子公司支付给少数股东的股利、利润	元	
cash_pay_oth_fin	float	支付其他与筹资活动有关的现金	元	
net_cash_pay_sell_repo	float	支付卖出回购金融资产款现金净额	元	保险
cf_out_fin	float	筹资活动现金流出小计	元	
net_cf_fin	float	筹资活动产生的现金流量净额	元	
efct_er_chg_cash	float	四、汇率变动对现金及现金等价物的影响	元	
net_incr_cash_eq	float	五、现金及现金等价物净增加额	元	
cash_cash_eq_bgn	float	加：期初现金及现金等价物余额	元	
cash_cash_eq_end	float	六、期末现金及现金等价物余额	元	
*补充资料**1**．将净利润调节为经营活动现金流量：*				
net_prof	float	净利润	元	
ast_impr	float	资产减值准备	元	
accr_prvs_ln_impa	float	计提贷款减值准备	元	银行
accr_prvs_oth_impa	float	计提其他资产减值准备	元	银行
accr_prem_rsv	float	提取的保险责任准备金	元	保险
accr_unearn_prem_rsv	float	提取的未到期的责任准备金	元	保险
defr_fix_prop	float	固定资产和投资性房地产折旧	元	
depr_oga_cba	float	其中:固定资产折旧、油气资产折耗、生产性生物资产折旧	元	
amor_intg_ast_lt_exp	float	无形资产及长期待摊费用等摊销	元	银行、证券、保险
amort_intg_ast	float	无形资产摊销	元	

字段名	类型	中文名称	量纲	说明
amort_lt_exp_ppay	float	长期待摊费用摊销	元	
dspl_ast_loss	float	处置固定资产、无形资产和其他长期资产的损失	元	
fair_val_chg_loss	float	固定资产报废损失	元	
fv_chg_loss	float	公允价值变动损失	元	
dfa	float	固定资产折旧	元	银行
fin_exp	float	财务费用	元	
inv_loss	float	投资损失	元	
exchg_loss	float	汇兑损失	元	银行、证券、保险
dest_incr	float	存款的增加	元	银行
loan_decr	float	贷款的减少	元	银行
cash_pay_bnd_int_iss	float	发行债券利息支出	元	银行
dfr_tax	float	递延所得税	元	
dfr_tax_ast_decr	float	其中:递延所得税资产减少	元	
dfr_tax_liab_incr	float	递延所得税负债增加	元	
invt_decr	float	存货的减少	元	
decr_rcv_oper	float	经营性应收项目的减少	元	
incr_pay_oper	float	经营性应付项目的增加	元	
oth	float	其他	元	
cash_end	float	现金的期末余额	元	
cash_bgn	float	减：现金的期初余额	元	
cash_eq_end	float	加:现金等价物的期末余额	元	
cash_eq_bgn	float	减:现金等价物的期初余额	元	
cred_impr_loss	float	信用减值损失	元	
est_liab_add	float	预计负债的增加	元	
dr_cnv_cptl	float	债务转为资本	元	

字段名	类型	中文名称	量纲	说明
cptl_bnd_expr_one_y	float	一年内到期的可转换公司债券	元	
fin_ls_fix_ast	float	融资租入固定资产	元	
amort_dfr_inc	float	递延收益摊销	元	
depr_inv_prop	float	投资性房地产折旧	元	
trd_fin_decr	float	交易性金融资产的减少	元	证券、保险
im_net_cf_oper	float	间接法-经营活动产生的现金流量净额	元	
im_net_incr_cash_eq	float	间接法-现金及现金等价物净增加额	元	

StkGetFundamentalsIncome - 查询利润表数据

查询指定时间段某一股票所属上市公司的利润表数据

函数原型：

```
public static GMDataList<FundamentalsIncome> StkGetFundamentalsIncome(string symbol, s
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	股票代码	Y	无	必填，只能填一个股票标的
fields	string	返回字段	Y	无	指定需要返回的财务字段，如有多个字段，中间用英文逗号分隔
rptType	int	报表类型	N	0	按报告期查询可指定以下报表类型： 1-一季度报 6-中报 9-前三季报 12-年报 默认 `0` 为不限
dataType	int	数据类型	N	0	在发布原始财务报告以后，上市公司可能会对数据进行修正。 101-合并原始 102-合并调整 201-母公司原始 202-母公司调整 默认 `0` 返回当期合并调整，如果没有调整返回合并原始
startDate	string	开始时间	N	null	开始时间，时间类型为报告日期，%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间，时间类型为报告日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
pubDate	string	发布日期	若数据类型选择合并原始(`dataType=101`)，则返回原始发布的发布日期 若数据类型选择合并调整(`dataType=102`)，则返回调整后最新发布日期 若数据类型选择母公司原始(`dataType=201`)，则返回母公司原始发布的发布日期 若数据类型选择母公司调整(`dataType=202`)，则返回母公司调整后最新发布日期
rptDate	string	报告日期	报告截止日期，财报统计的最后一天 在指定时间段[开始时间,结束时间]内的报告截止日期
rptType	int	报表类型	返回数据的报表类型：1-一季度报, 6-中报, 9-前三季报, 12-年报
dataType	int	数据类型	返回数据的数据类型：101-合并原始, 102-合并调整, 201-母公司原始, 202-母公司调整
fields	float	财务字段数据	指定返回 `fields` 字段的数值. 支持的字段名请参考 利润表

示例：

```
StkGetFundamentalsIncome("SHSE.600000", "inc_oper")
```

注意：

1. 当 `startDate` == `endDate` 时，取离 `endDate` 最近报告日期的一条数据，

当 `startDat` < `endDate` 时，取指定时间段的数据，

当 `startDate` > `endDate` 时，返回报错。

1. 若在指定历史时间段内，有多个同一类型报表（如不同年份的一季度报表），将按照报告日期顺序返回。
2. 如果 `fields` 参数的财务字段填写不正确，或填写空字段，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

利润表

字段名	类型	中文名称	量纲	说明
ttl_inc_oper	float	营业总收入	元	
inc_oper	float	营业收入	元	
net_inc_int	float	利息净收入	元	证券、银行、保险
exp_int	float	利息支出	元	
net_inc_fee_comm	float	手续费及佣金净收入	元	证券、银行
inc_rin_prem	float	其中：分保费收入	元	保险
net_inc_secu_agy	float	其中:代理买卖证券业务净收入	元	证券
inc_fee_comm	float	手续费及佣金收入	元	
in_prem_earn	float	已赚保费	元	保险
inc_in_biz	float	其中:保险业务收入	元	保险
rin_prem_cede	float	分出保费	元	保险
unear_prem_rsv	float	提取未到期责任准备金	元	保险
net_inc_uw	float	证券承销业务净收入	元	证券
net_inc_cust_ast_mgmt	float	受托客户资产管理业务净收入	元	证券
inc_fx	float	汇兑收益	元	
inc_other_oper	float	其他业务收入	元	
inc_oper_balance	float	营业收入平衡项目	元	
ttl_inc_oper_other	float	营业总收入其他项目	元	
ttl_cost_oper	float	营业总成本	元	
cost_oper	float	营业成本	元	
exp_oper	float	营业支出	元	证券、银行、保险
biz_tax_sur	float	营业税金及附加	元	
exp_sell	float	销售费用	元	
exp_adm	float	管理费用	元	
exp_rd	float	研发费用	元	

字段名	类型	中文名称	量纲	说明
exp_fin	float	财务费用	元	
int_fee	float	其中:利息费用	元	
inc_int	float	利息收入	元	
exp_oper_adm	float	业务及管理费	元	证券、银行、保险
exp_rin	float	减:摊回分保费用	元	保险
rfd_prem	float	退保金	元	保险
comp_pay	float	赔付支出	元	保险
rin_clm_pay	float	减:摊回赔付支出	元	保险
draw_insur_liab	float	提取保险责任准备金	元	保险
amor_insur_liab	float	减:摊回保险责任准备金	元	保险
exp_ph_dvd	float	保单红利支出	元	保险
exp_fee_comm	float	手续费及佣金支出	元	
other_oper_cost	float	其他业务成本	元	
oper_exp_balance	float	营业支出平衡项目	元	证券、银行、保险
exp_oper_other	float	营业支出其他项目	元	证券、银行、保险
ttl_cost_oper_other	float	营业总成本其他项目	元	
其他经营收益			元	
inc_inv	float	投资收益	元	
inv_inv_jv_p	float	对联营企业和合营企业的投资收益	元	
inc_ast_dspl	float	资产处置收益	元	
ast_impr_loss	float	资产减值损失(新)	元	
cred_impr_loss	float	信用减值损失(新)	元	
inc_fv_chg	float	公允价值变动收益	元	
inc_other	float	其他收益	元	

字段名	类型	中文名称	量纲	说明
oper_prof_balance	float	营业利润平衡项目	元	
oper_prof	float	营业利润	元	
inc_noper	float	营业外收入	元	
exp_noper	float	营业外支出	元	
ttl_prof_balance	float	利润总额平衡项目	元	
oper_prof_other	float	营业利润其他项目	元	
ttl_prof	float	利润总额	元	
inc_tax	float	所得税费用	元	
net_prof	float	净利润	元	
oper_net_prof	float	持续经营净利润	元	
net_prof_pcom	float	归属于母公司股东的净利润	元	
min_int_inc	float	少数股东损益	元	
end_net_prof	float	终止经营净利润	元	
net_prof_other	float	净利润其他项目	元	
eps_base	float	基本每股收益	元	
eps_dil	float	稀释每股收益	元	
other_comp_inc	float	其他综合收益	元	
other_comp_inc_pcom	float	归属于母公司股东的其他综合收益	元	
other_comp_inc_min	float	归属于少数股东的其他综合收益	元	
ttl_comp_inc	float	综合收益总额	元	
ttl_comp_inc_pcom	float	归属于母公司所有者的综合收益总额	元	
ttl_comp_inc_min	float	归属于少数股东的综合收益总额	元	
prof_pre_merge	float	被合并方在合并前实现利润	元	
net_rsv_in_contr	float	提取保险合同准备金净额	元	
net_pay_comp	float	赔付支出净额	元	
net_loss_ncur_ast	float	非流动资产处置净损失	元	
amod_fin_asst_end	float	以摊余成本计量的金融资产终止确认收益	元	

字段名	类型	中文名称	量纲	说明
cash_flow_hedging_pl	float	现金流量套期损益的有效部分	元	
cur_trans_diff	float	外币财务报表折算差额	元	
gain_ncur_ast	float	非流动资产处置利得	元	
afs_fv_chg_pl	float	可供出售金融资产公允价值变动损益	元	
oth_eqy_inv_fv_chg	float	其他权益工具投资公允价值变动	元	
oth_debt_inv_fv_chg	float	其他债权投资公允价值变动	元	
oth_debt_inv_cred_impr	float	其他债权投资信用减值准备	元	

StkGetFundamentalsBalancePt - 查询资产负债表截面数据（多标的）

查询指定日期截面的股票所属上市公司的资产负债表数据（point-in-time）

函数原型：

```
public static GMDataList<FundamentalsBalance> StkGetFundamentalsBalancePt(string symbo
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbols	string	股票代码	Y	无	必填，可输入多个，多个标的代码必须用英文逗号分割， 如：`SHSE.600008,SZSE.000002`
fields	string	返回字段	Y	无	指定需要返回的财务字段，如有多个字段，中间用英文逗号分隔
rptType	int	报表类型	N	0	按报告期查询可指定以下报表类型： 1-一季度报 6-中报 9-前三季报 12-年报 默认`0`为不限
dataType	int	数据类型	N	0	在发布原始财务报告以后，上市公司可能会对数据进行修正。 101-合并原始 102-合并调整 201-母公司原始 202-母公司调整 默认`0`返回当期合并调整，如果没有调整返回合并原始
date	string	查询日期	N	null	查询时间，时间类型为发布日期，%Y-%m-%d 格式，默认`null`表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
pubDate	string	发布日期	距查询日期最近的发布日期 若数据类型选择合并原始（`dataType=101`），则返回原始发布的发布日期 若数据类型选择合并调整（`dataType=102`），则返回调整后最新发布日期 若数据类型选择母公司原始（`dataType=201`），则返回母公司原始发布的发布日期 若数据类型选择母公司调整（`dataType=202`），则返回母公司调整后最新发布日期
rptDate	string	报告日期	报告截止日期，财报统计的最后一天
rptType	int	报表类型	返回数据的报表类型：1-一季度报, 6-中报, 9-前三季报, 12-年报
dataType	int	数据类型	返回数据的数据类型：101-合并原始, 102-合并调整, 201-母公司原始, 202-母公司调整
fields	list[float]	财务字段数据	指定查询 `fields` 字段的数值. 支持的字段名请参考 资产负债表

示例：

```
StkGetFundamentalsBalancePt("SHSE.600000,SZSE.000001", "fix_ast")
```

注意：

1. 为避免未来数据问题，指定查询日期 `date` 后，返回距离此日期最近发布的一条数据。
2. 如果 `fields` 参数的财务字段填写不正确，或填写空字段 `null`，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

StkGetFundamentalsCashflowPt - 查询现金流量表截面数据

查询指定日期截面的股票所属上市公司的现金流量表数据（point-in-time）

函数原型：

```
public static GMDataList<FundamentalsCashflow> StkGetFundamentalsCashflowPt(string sym
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbols	string	股票代码	Y	无	必填，可输入多个, 多个标的代码必须用英文逗号分割， 如： `SHSE.600008,SZSE.000002`
fields	string	返回字段	Y	无	指定需要返回的财务字段，如有多个字段，中间用英文逗号分隔
rptType	int	报表类型	N	0	按报告期查询可指定以下报表类型： 1-一季度报 6-中报 9-前三季报 12-年报 默认 `0` 为不限
dataType	int	数据类型	N	0	在发布原始财务报告以后，上市公司可能会对数据进行修正。 101-合并原始 102-合并调整 201-母公司原始 202-母公司调整 默认 `0` 返回当期合并调整，如果没有调整返回合并原始
date	string	查询日期	N	null	查询时间，时间类型为发布日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
pubDate	string	发布日期	距查询日期最近的发布日期 若数据类型选择合并原始(`dataType=101`)，则返回原始发布的发布日期 若数据类型选择合并调整(`dataType=102`)，则返回调整后最新发布日期 若数据类型选择母公司原始(`dataType=201`)，则返回母公司原始发布的发布日期 若数据类型选择母公司调整(`dataType=202`)，则返回母公司调整后最新发布日期
rptDate	string	报告日期	报告截止日期，财报统计的最后一天
rptType	int	报表类型	返回数据的报表类型：1-一季度报, 6-中报, 9-前三季报, 12-年报
dataType	int	数据类型	返回数据的数据类型：101-合并原始, 102-合并调整, 201-母公司原始, 202-母公司调整
fields	float	财务字段数据	指定查询 `fields` 字段的数值. 支持的字段名请参考 现金流量表

示例：

```
StkGetFundamentalsCashflowPt("SHSE.600000, SZSE.000001", "cash_pay_fee")
```

注意：

1. 为避免未来数据问题，指定查询日期 `date` 后，返回距离此日期最近发布的一条数据。
2. 如果 `fields` 参数的财务字段填写不正确，或填写空字段 `null` ，会报错提示“填写的 fields 无效”。

StkGetFundamentalsIncomePt - 查询利润表截面数据（多标的）

查询指定日期截面的股票所属上市公司的利润表数据（point-in-time）

函数原型：

```
public static GMDataList<FundamentalsIncome> StkGetFundamentalsIncomePt(string symbols
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbols	string	股票代码	Y	无	必填，可输入多个，多个标的代码必须用英文逗号分割， 如：`SHSE.600008,SZSE.000002`
fields	string	返回字段	Y	无	指定需要返回的财务字段，如有多个字段，中间用英文逗号分隔
rptType	int	报表类型	N	0	按报告期查询可指定以下报表类型： 1-一季度报 6-中报 9-前三季报 12-年报 默认`0`为不限
dataType	int	数据类型	N	0	在发布原始财务报告以后，上市公司可能会对数据进行修正。 101-合并原始 102-合并调整 201-母公司原始 202-母公司调整 默认`0`返回当期合并调整，如果没有调整返回合并原始
date	string	查询日期	N	null	查询时间，时间类型为发布日期，%Y-%m-%d 格式，默认`null`表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
pubDate	string	发布日期	距查询日期最近的发布日期 若数据类型选择合并原始(`dataType=101`)，则返回原始发布的发布日期 若数据类型选择合并调整(`dataType=102`)，则返回调整后最新发布日期 若数据类型选择母公司原始(`dataType=201`)，则返回母公司原始发布的发布日期 若数据类型选择母公司调整(`dataType=202`)，则返回母公司调整后最新发布日期
rptDate	string	报告日期	报告截止日期，财报统计的最后一天
rptType	int	报表类型	返回数据的报表类型：1-一季度报, 6-中报, 9-前三季报, 12-年报
dataType	int	数据类型	返回数据的数据类型：101-合并原始, 102-合并调整, 201-母公司原始, 202-母公司调整
fields	float	财务字段数据	指定查询 `fields` 字段的数值. 支持的字段名请参考 利润表

示例：

```
StkGetFundamentalsIncomePt("SHSE.600000, SZSE.000001", "inc_oper")
```

注意：

1. 为避免未来数据问题，指定查询日期 `date` 后，返回距离此日期最近发布的一条数据。
2. 如果 `fields` 参数的财务字段填写不正确，或填写空字段 `null` ，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

StkGetFinancePrime - 查询财务主要指标数据

查询指定时间段股票所属上市公司的财务主要指标

函数原型：

```
public static GMDataList<FinancePrime> StkGetFinancePrime(string symbol, string fields
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	股票代码	Y	无	必填，只能填一个股票标的
fields	string	返回字段	Y	无	指定需要返回的财务主要指标， 如有多个字段，中间用英文逗号分隔
rptType	int	报表类型	N	0	按报告期查询可指定以下报表类型： 1-一季度报 6-中报 9-前三季报 12-年报 默认 `0` 为不限
dataType	int	数据类型	N	0	在发布原始财务报告以后，上市公司可能会对数据进行修正。 101-合并原始 102-合并调整 201-母公司原始 202-母公司调整 默认 `0` 返回当期合并调整，如果没有调整返回合并原始
startDate	string	开始时间	N	null	开始时间，时间类型为报告日期，%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间，时间类型为报告日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
pubDate	string	发布日期	若数据类型选择合并原始(`dataType=101`)，则返回原始发布的发布日期 若数据类型选择合并调整(`dataType=102`)，则返回调整后最新发布日期 若数据类型选择母公司原始(`dataType=201`)，则返回母公司原始发布的发布日期 若数据类型选择母公司调整(`dataType=202`)，则返回母公司调整后最新发布日期
rptDate	string	报告日期	报告截止日期，财报统计的最后一天，在指定时间段[开始时间,结束时间]内的报告截止日期
rptType	int	报表类型	返回数据的报表类型：1-一季度报, 6-中报, 9-前三季报, 12-年报
dataType	int	数据类型	返回数据的数据类型：101-合并原始, 102-合并调整, 201-母公司原始, 202-母公司调整
fields	float	财务字段数据	指定返回 `fields` 字段的数值. 支持的字段名请参考 财务主要指标

示例：

```
StkGetFinancePrime("SHSE.600000", "eps_basic,eps_dil")
```

注意：

1. 当 `startDate` == `endDate` 时，取离 `endDate` 最近报告日期的一条数据，

当 `startDat` < `endDate` 时，取指定时间段的数据，

当 `startDate` > `endDate` 时，返回报错。

1. 若在指定历史时间段内，有多个同一类型报表（如不同年份的一季度报表），将按照报告日期顺序返回。
2. 如果 `fields` 参数的财务字段填写不正确，或填写空字段，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

财务主要指标

字段名	类型	中文名称	量纲	说明
eps_basic	float	基本每股收益	元	
eps_dil	float	稀释每股收益	元	
eps_basic_cut	float	扣除非经常性损益后的基本每股收益	元	
eps_dil_cut	float	扣除非经常性损益后的稀释每股收益	元	
net_cf_oper_ps	float	每股经营活动产生的现金流量净额	元	
bps_pcom_ps	float	归属于母公司股东的每股净资产	元	
ttl_ast	float	总资产	元	
ttl_liab	float	总负债	元	
share_cptl	float	股本	股	
ttl_inc_oper	float	营业总收入	元	
inc_oper	float	营业收入	元	
oper_prof	float	营业利润	元	
ttl_prof	float	利润总额	元	
ttl_eqy_pcom	float	归属于母公司股东的所有者权益	元	
net_prof_pcom	float	归属于母公司股东的净利润	元	
net_prof_pcom_cut	float	扣除非经常性损益后归属于母公司股东的净利润	元	
roe	float	全面摊薄净资产收益率	%	
roe_weight_avg	float	加权平均净资产收益率	%	
roe_cut	float	扣除非经常性损益后的全面摊薄净资产收益率	%	
roe_weight_avg_cut	float	扣除非经常性损益后的加权平均净资产收益率	%	
net_cf_oper	float	经营活动产生的现金流量净额	元	
eps_yoy	float	每股收益同比比例	%	
inc_oper_yoy	float	营业收入同比比例	%	
ttl_inc_oper_yoy	float	营业总收入同比比例	%	
net_prof_pcom_yoy	float	归母净利润同比比例	%	
bps_sh	float	归属于普通股东的每股净资产	元	
net_asset	float	归属于普通股东的净资产	元	

字段名	类型	中文名称	量纲	说明
net_prof	float	归属于普通股东的净利润	元	
net_prof_cut	float	扣除非经常性损益后归属于普通股股东的净利润	元	

StkGetFinanceDeriv - 查询财务衍生指标数据

查询指定时间段股票所属上市公司的财务衍生指标

函数原型：

```
public static GMDataList<FinanceDeriv> StkGetFinanceDeriv(string symbol, string fields
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	股票代码	Y	无	必填，只能填一个股票标的
fields	string	返回字段	Y	无	指定需要返回的财务衍生指标， 如有多个字段，中间用英文逗号分隔
rptType	int	报表类型	N	0	按报告期查询可指定以下报表类型： 1-一季度报 6-中报 9-前三季报 12-年报 默认 `0` 为不限
dataType	int	数据类型	N	0	在发布原始财务报告以后，上市公司可能会对数据进行修正。 101-合并原始 102-合并调整 201-母公司原始 202-母公司调整 默认 `0` 返回当期合并调整，如果没有调整返回合并原始
startDate	string	开始时间	N	null	开始时间，时间类型为报告日期，%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间，时间类型为报告日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
pubDate	string	发布日期	若数据类型选择合并原始(`dataType=101`)，则返回原始发布的发布日期 若数据类型选择合并调整(`dataType=102`)，则返回调整后最新发布日期 若数据类型选择母公司原始(`dataType=201`)，则返回母公司原始发布的发布日期 若数据类型选择母公司调整(`dataType=202`)，则返回母公司调整后最新发布日期
rptDate	string	报告日期	报告截止日期，财报统计的最后一天，在指定时间段[开始时间,结束时间]内的报告截止日期
rptType	int	报表类型	返回数据的报表类型：1-一季度报, 6-中报, 9-前三季报, 12-年报
dataType	int	数据类型	返回数据的数据类型：101-合并原始, 102-合并调整, 201-母公司原始, 202-母公司调整
fields	float	财务字段数据	指定返回 `fields` 字段的数值. 支持的字段名请参考 财务衍生指标指标

示例：

```
StkGetFinanceDeriv("SHSE.600000", "eps_basic,eps_dil2,eps_dil,eps_basic_cut", 9)
```

注意：

1. 当 `startDate` == `endDate` 时，取离 `endDate` 最近报告日期的一条数据，

当 `startDat` < `endDate` 时，取指定时间段的数据，

当 `startDate` > `endDate` 时，返回报错。

1. 若在指定历史时间段内，有多个同一类型报表（如不同年份的一季度报表），将按照报告日期顺序返回。
2. 如果 `fields` 参数的财务字段填写不正确，或填写空字段，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

财务衍生指标指标

字段名	类型	中文名称	量纲	说明
eps_basic	float	每股收益EPS(基本)	元	
eps_dil2	float	每股收益EPS(稀释)	元	
eps_dil	float	每股收益EPS(期末股本摊薄)	元	
eps_basic_cut	float	每股收益EPS(扣除/基本)	元	
eps_dil2_cut	float	每股收益EPS(扣除/稀释)	元	
eps_dil_cut	float	每股收益EPS(扣除/期末股本摊薄)	元	
bps	float	每股净资产BPS	元	
net_cf_oper_ps	float	每股经营活动产生的现金流量净额	元	
ttl_inc_oper_ps	float	每股营业总收入	元	
inc_oper_ps	float	每股营业收入	元	
ebit_ps	float	每股息税前利润	元	
cptl_rsv_ps	float	每股资本公积	元	
sur_rsv_ps	float	每股盈余公积	元	
retain_prof_ps	float	每股未分配利润	元	
retain_inc_ps	float	每股留存收益	元	
net_cf_ps	float	每股现金流量净额	元	
fcff_ps	float	每股企业自由现金流量	元	
fcfe_ps	float	每股股东自由现金流量	元	
ebitda_ps	float	每股EBITDA	元	
roe	float	净资产收益率ROE(摊薄)	%	
roe_weight	float	净资产收益率ROE(加权)	%	
roe_avg	float	净资产收益率ROE(平均)	%	
roe_cut	float	净资产收益率ROE(扣除/摊薄)	%	
roe_weight_cut	float	净资产收益率ROE(扣除/加权)	%	

字段名	类型	中文名称	量纲	说明
ocf_toi	float	经营性现金净流量/营业总收入		
eps_dil_yoy	float	稀释每股收益同比增长率	%	
net_cf_oper_ps_yoy	float	每股经营活动中产生的现金流量净额同比增长率	%	
ttl_inc_oper_yoy	float	营业总收入同比增长率	%	
inc_oper_yoy	float	营业收入同比增长率	%	
oper_prof_yoy	float	营业利润同比增长率	%	
ttl_prof_yoy	float	利润总额同比增长率	%	
net_prof_pcom_yoy	float	归属母公司股东的净利润同比增长率	%	
net_prof_pcom_cut_yoy	float	归属母公司股东的净利润同比增长率(扣除非经常性损益)	%	
net_cf_oper_yoy	float	经营活动产生的现金流量净额同比增长率	%	
roe_yoy	float	净资产收益率同比增长率(摊薄)	%	
net_asset_yoy	float	净资产同比增长率	%	
ttl_liab_yoy	float	总负债同比增长率	%	
ttl_asset_yoy	float	总资产同比增长率	%	
net_cash_flow_yoy	float	现金净流量同比增长率	%	
bps_gr_begin_year	float	每股净资产相对年初增长率	%	
ttl_asset_gr_begin_year	float	资产总计相对年初增长率	%	
ttl_eqy_pcom_gr_begin_year	float	归属母公司的股东权益相对年初增长率	%	
net_debt_eqy_ev	float	净债务/股权价值	%	

字段名	类型	中文名称	量纲	说明
int_debt_eqy_ev	float	带息债务/股权价值		
eps_bas_yoy	float	基本每股收益同比增长率	%	
ebit	float	EBIT(正推法)	元	
ebitda	float	EBITDA(正推法)	元	
ebit_inverse	float	EBIT(反推法)	元	
ebitda_inverse	float	EBITDA(反推法)	元	
nr_prof_loss	float	非经常性损益	元	
net_prof_cut	float	扣除非经常性损益后的净利润	元	
gross_prof	float	毛利润	元	
oper_net_inc	float	经营活动净收益	元	
val_chg_net_inc	float	价值变动净收益	元	
exp_rd	float	研发费用	元	
ttl_inv_cptl	float	全部投入资本	元	
work_cptl	float	营运资本	元	
net_work_cptl	float	净营运资本	元	
tg_asset	float	有形资产	元	
retain_inc	float	留存收益	元	
int_debt	float	带息债务	元	
net_debt	float	净债务	元	
curr_liab_non_int	float	无息流动负债	元	
ncur_liab_non_int	float	无息非流动负债	元	
fcff	float	企业自由现金流量FCFF	元	
fcfe	float	股权自由现金流量FCFE	元	
cur_depr_amort	float	当期计提折旧与摊销	元	
eqy_mult_dupont	float	权益乘数(杜邦分析)		
net_prof_pcom_np	float	归属母公司股东的净利润/净利润	%	
net_prof_tp	float	净利润/利润总额	%	
ttl_prof_ebit	float	利润总额/息税前利润	%	

字段名	类型	中文名称	量纲	说明
roe_cut_avg	float	净资产收益率ROE(扣除/平均)	%	
roe_add	float	净资产收益率ROE(增发条件)	%	
roe_ann	float	净资产收益率ROE(年化)	%	
roa	float	总资产报酬率ROA	%	
roa_ann	float	总资产报酬率ROA(年化)	%	
jroa	float	总资产净利率	%	
jroa_ann	float	总资产净利率(年化)	%	
roic	float	投入资本回报率ROIC	%	
sale_npm	float	销售净利率	%	
sale_gpm	float	销售毛利率	%	
sale_cost_rate	float	销售成本率	%	
sale_exp_rate	float	销售期间费用率	%	
net_prof_toi	float	净利润/营业总收入	%	
oper_prof_toi	float	营业利润/营业总收入	%	
ebit_toi	float	息税前利润/营业总收入	%	
ttl_cost_oper_toi	float	营业总成本/营业总收入	%	
exp_oper_toi	float	营业费用/营业总收入	%	
exp_admin_toi	float	管理费用/营业总收入	%	
exp_fin_toi	float	财务费用/营业总收入	%	
ast_impr_loss_toi	float	资产减值损失/营业总收入	%	
ebitda_toi	float	EBITDA/营业总收入	%	
oper_net_inc_tp	float	经营活动净收益/利润总额	%	
val_chg_net_inc_tp	float	价值变动净收益/利润总额	%	

字段名	类型	中文名称	量纲	说明
net_exp_noper_tp	float	营业外支出净额/利润总额		
inc_tax_tp	float	所得税/利润总额	%	
net_prof_cut_np	float	扣除非经常性损益的净利润/净利润	%	
eqy_mult	float	权益乘数		
curr_ast_ta	float	流动资产/总资产	%	
ncurr_ast_ta	float	非流动资产/总资产	%	
tg_ast_ta	float	有形资产/总资产	%	
ttl_eqy_pcom_tic	float	归属母公司股东的权益/全部投入资本	%	
int_debt_tic	float	带息负债/全部投入资本	%	
curr_liab_tl	float	流动负债/负债合计	%	
ncurr_liab_tl	float	非流动负债/负债合计	%	
ast_liab_rate	float	资产负债率	%	
quick_rate	float	速动比率		
curr_rate	float	流动比率		
cons_quick_rate	float	保守速动比率		
liab_eqy_rate	float	产权比率		
ttl_eqy_pcom_tl	float	归属母公司股东的权益/负债合计		
ttl_eqy_pcom_debt	float	归属母公司股东的权益/带息债务		
tg_ast_tl	float	有形资产/负债合计		
tg_ast_int_debt	float	有形资产/带息债务		
tg_ast_net_debt	float	有形资产/净债务		
ebitda_tl	float	息税折旧摊销前利润/负债合计		
net_cf_oper_tl	float	经营活动产生的现金流量净额/负债合计		

字段名	类型	中文名称	量纲	说明
net_cf_oper_int_debt	float	经营活动产生的现金流量净额/带息债务		
net_cf_oper_curr_liab	float	经营活动产生的现金流量净额/流动负债		
net_cf_oper_net_liab	float	经营活动产生的现金流量净额/净债务		
ebit_int_cover	float	已获利息倍数		
long_liab_work_cptl	float	长期债务与营运资金比率		
ebitda_int_debt	float	EBITDA/带息债务	%	
oper_cycle	float	营业周期	天	
inv_turnover_days	float	存货周转天数	天	
acct_rcv_turnover_days	float	应收账款周转天数(含应收票据)	天	
inv_turnover_rate	float	存货周转率	次	
acct_rcv_turnover_rate	float	应收账款周转率(含应收票据)	次	
curr_ast_turnover_rate	float	流动资产周转率	次	
fix_ast_turnover_rate	float	固定资产周转率	次	
ttl_ast_turnover_rate	float	总资产周转率	次	
cash_rcv_sale_oi	float	销售商品提供劳务收到的现金/营业收入	%	
net_cf_oper_oi	float	经营活动产生的现金流量净额/营业收入	%	
net_cf_oper_oni	float	经营活动产生的现金流量净额/经营活动净收益		
cptl_exp_da	float	资本支出/折旧摊销	%	
cash_rate	float	现金比率		
acct_pay_turnover_days	float	应付账款周转天数(含应付票据)	天	
acct_pay_turnover_rate	float	应付账款周转率(含应付票据)	次	
net_oper_cycle	float	净营业周期	天	
ttl_cost_oper_yoy	float	营业总成本同比增长率	%	

字段名	类型	中文名称	量纲	说明
net_prof_yoy	float	净利润同比增长率	%	
net_cf_oper_np	float	经营活动产生的现金流量净额/净利润	%	

StkGetDailyValuation - 查询估值指标每日数据

查询指定时间段股票的每日估值指标

函数原型：

```
public static GMDataList<DailyValuation> StkGetDailyValuation(string symbol, string fi
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	股票代码	Y	无	必填，只能填一个股票标的，使用时参考
fields	string	返回字段	Y	无	指定需要返回的财务字段，如有多个字段，中间用英文逗号分隔
startDate	string	开始时间	N	null	开始时间，时间类型为交易日期，%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间，时间类型为交易日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
tradeDate	string	交易日期	
fields	list[float]	指标字段数据	指定返回 `fields` 字段的数值. 支持的字段名请参考 估值指标

示例：

```
StkGetDailyValuation("SHSE.600000", "pe_ttm,pe_lyr,pe_mrq")
```

注意：

1. 当 `startDate` == `endDate` 时，取离 `endDate` 最近交易日期的一条数据，

当 `startDat` < `endDate` 时，取指定时间段的数据，

当 `startDate` > `endDate` 时，返回报错。

1. 如果 `fields` 参数的指标字段填写不正确，或填写空字段，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

估值指标

字段名	类型	中文名称	量纲	说明
pe_ttm	float	市盈率(TTM)	倍	
pe_lyr	float	市盈率(最新年报LYR)	倍	
pe_mrq	float	市盈率(最新报告期MRQ)	倍	
pe_1q	float	市盈率(当年一季×4)	倍	
pe_2q	float	市盈率(当年中报×2)	倍	
pe_3q	float	市盈率(当年三季×4/3)	倍	
pe_ttm_cut	float	市盈率(TTM) 扣除非经常性损益	倍	
pe_lyr_cut	float	市盈率(最新年报LYR) 扣除非经常性损益	倍	
pe_mrq_cut	float	市盈率(最新报告期MRQ) 扣除非经常性损益	倍	
pe_1q_cut	float	市盈率(当年一季×4) 扣除非经常性损益	倍	
pe_2q_cut	float	市盈率(当年中报×2) 扣除非经常性损益	倍	
pe_3q_cut	float	市盈率(当年三季×4/3) 扣除非经常性损益	倍	
pb_lyr	float	市净率(最新年报LYR)	倍	
pb_lf	float	市净率(最新公告)	倍	
pb_mrq	float	市净率(最新报告期MRQ)	倍	
pcf_ttm_oper	float	市现率(经营现金流,TTM)	倍	
pcf_ttm_ncf	float	市现率(现金净流量,TTM)	倍	
pcf_lyr_oper	float	市现率(经营现金流,最新年报LYR)	倍	
pcf_lyr_ncf	float	市现率(现金净流量,最新年报LYR)	倍	
ps_ttm	float	市销率(TTM)	倍	
ps_lyr	float	市销率(最新年报LYR)	倍	
ps_mrq	float	市销率(最新报告期MRQ)	倍	
ps_1q	float	市销率(当年一季×4)	倍	
ps_2q	float	市销率(当年中报×2)	倍	
ps_3q	float	市销率(当年三季×4/3)	倍	
peg_lyr	float	历史PEG值(当年年报增长率)	倍	
peg_1q	float	历史PEG值(当年1季*4较上年年报增长率)	倍	
peg_2q	float	历史PEG值(当年中报*2较上年年报增长率)	倍	

字段名	类型	中文名称	量纲	说明
peg_3q	float	历史PEG值(当年3季*4/3较上年年报增长率)	倍	
dy_ttm	float	股息率(滚动 12 月TTM)	%	
dy_lfy	float	股息率(上一财年LFY)	%	

StkGetDailyMktvalue - 查询市值指标每日数据

查询指定时间段股票的每日市值指标

函数原型：

```
public static GMDataList<DailyMktvalue> StkGetDailyMktvalue(string symbol, string fiel
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	股票代码	Y	无	必填，只能填一个股票标的
fields	string	返回字段	Y	无	指定需要返回的财务字段，如有多个字段，中间用英文逗号分隔
startDate	string	开始时间	N	null	开始时间，时间类型为交易日期，%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间，时间类型为交易日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
tradeDate	string	交易日期	
fields	float	指标字段数据	指定返回 `fields` 字段的数值. 支持的字段名请参考 市值指标

示例：

```
StkGetDailyMktvalue("SHSE.600000", "tot_mv,tot_mv_csrc,a_mv")
```

注意：

1. 当 `startDate` == `endDate` 时，取离 `endDate` 最近交易日期的一条数据，

当 `startDat` < `endDate` 时，取指定时间段的数据，

当 `startDate` > `endDate` 时，返回报错。

1. 如果 `fields` 参数的指标字段填写不正确，或填写空字段，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

市值指标

字段名	类型	中文名称	量纲	说明
tot_mv	float	总市值	元	
tot_mv_csrc	float	总市值(证监会算法)	元	
a_mv	float	A股流通市值(含限售股)	元	
a_mv_ex_ltd	float	A股流通市值(不含限售股)	元	
b_mv	float	B股流通市值(含限售股，折人民币)	元	
b_mv_ex_ltd	float	B股流通市值(不含限售股，折人民币)	元	
ev	float	企业价值(含货币资金)(EV1)	元	
ev_ex_curr	float	企业价值(剔除货币资金)(EV2)	元	
ev_ebitda	float	企业倍数	倍	
equity_value	float	股权价值	元	

StkGetDailyBasic - 查询基础指标每日数据

查询指定时间段股票的每日基础指标

函数原型：

```
public static GMDataList<DailyBasic> StkGetDailyBasic(string symbol, string fields, st
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	股票代码	Y	无	必填，只能填一个股票标的
fields	string	返回字段	Y	无	指定需要返回的财务字段，如有多个字段，中间用英文逗号分隔
startDate	string	开始时间	N	null	开始时间，时间类型为交易日期，%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间，时间类型为交易日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
tradeDate	string	交易日期	
fields	float	指标字段数据	指定返回 `fields` 字段的数值. 支持的字段名请参考 基础指标

示例：

```
StkGetDailyBasic("SHSE.600000", "tclose,turnrate,ttl_shr,circ_shr")
```

注意：

1. 当 `startDate` == `endDate` 时，取离 `endDate` 最近交易日期的一条数据，

当 `startDat` < `endDate` 时，取指定时间段的数据，

当 `startDate` > `endDate` 时，返回报错。

1. 如果 `fields` 参数的财务字段填写不正确，或填写空字段，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

基础指标

字段名	类型	中文名称	量纲	说明
tclose	float	收盘价	元	
turnrate	float	当日换手率	%	
ttl_shr	float	总股本	股	
circ_shr	float	流通股本（流通股本=无限售条件流通股本+有限售条件流通股本）	股	
ttl_shr_unl	float	无限售条件流通股本(行情软件定义的流通股)	股	
ttl_shr_ltd	float	有限售条件股本	股	
a_shr_unl	float	无限售条件流通股本（A股）	股	
h_shr_unl	float	无限售条件流通股本（H股）	股	

StkGetFinancePrimePt - 查询财务主要指标截面数据（多标的）

查询指定日期截面上，股票所属上市公司的财务主要指标数据（point-in-time）

函数原型：

```
public static GMDataList<FinancePrime> StkGetFinancePrimePt(string symbols, string fie
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbols	string or list	股票代码	Y	无	必填，可输入多个, 多个标的代码必须用英文逗号分割， 如： `SHSE.600008,SZSE.000002`
fields	string	返回字段	Y	无	指定需要返回的财务主要指标， 如有多个字段，中间用英文逗号分隔
rptType	int	报表类型	N	0	按报告期查询可指定以下报表类型： 1-一季度报 6-中报 9-前三季报 12-年报 默认 `0` 为不限
dataType	int	数据类型	N	0	在发布原始财务报告以后，上市公司可能会对数据进行修正。 101-合并原始 102-合并调整 201-母公司原始 202-母公司调整 默认 `0` 返回当期合并调整，如果没有调整返回合并原始
date	string	查询日期	N	null	查询时间，时间类型为发布日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
pubDate	string	发布日期	距查询日期最近的发布日期 若数据类型选择合并原始(`dataType=101`)，则返回原始发布的发布日期 若数据类型选择合并调整(`dataType=102`)，则返回调整后最新发布日期 若数据类型选择母公司原始(`dataType=201`)，则返回母公司原始发布的发布日期 若数据类型选择母公司调整(`dataType=202`)，则返回母公司调整后最新发布日期
rptDate	string	报告日期	报告截止日期，财报统计的最后一天，在指定时间段[开始时间,结束时间]内的报告截止日期
rptType	int	报表类型	返回数据的报表类型：1-一季度报, 6-中报, 9-前三季报, 12-年报
dataType	int	数据类型	返回数据的数据类型：101-合并原始, 102-合并调整, 201-母公司原始, 202-母公司调整
fields	float	财务字段数据	指定查询 `fields` 字段的数值. 支持的字段名请参考 财务主要指标

示例：

```
StkGetFinancePrimePt("SZSE.000001,SZSE.300002", "eps_basic,eps_dil")
```

注意：

1. 为避免未来数据问题，指定查询日期 `date` 后，返回距离此日期最近发布的一条数据。 若多个报告期报表的最新发布日期相同，返回报告日期rptDate距离查询日期date最近的一个报告期的报表数据。
2. 如果 `fields` 参数的财务字段填写不正确，或填写空字段 `null` ，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

StkGetFinanceDerivPt - 查询财务衍生指标截面数据（多标的）

查询指定日期截面上，股票所属上市公司的财务衍生指标数据（point-in-time）

函数原型：

```
public static GMDataList<FinanceDeriv> StkGetFinanceDerivPt(string symbols, string fie
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbols	string	股票代码	Y	无	必填，可输入多个，多个标的代码必须用英文逗号分割， 如： `SHSE.600008,SZSE.000002`
fields	string	返回字段	Y	无	指定需要返回的财务衍生指标， 如有多个字段，中间用英文逗号分隔
rptType	int	报表类型	N	0	按报告期查询可指定以下报表类型： 1-一季度报 6-中报 9-前三季报 12-年报 默认 `0` 为不限
dataType	int	数据类型	N	0	在发布原始财务报告以后，上市公司可能会对数据进行修正。 101-合并原始 102-合并调整 201-母公司原始 202-母公司调整 默认 `0` 返回当期合并调整，如果没有调整返回合并原始
date	string	查询日期	N	null	查询时间，时间类型为发布日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
pubDate	string	发布日期	距查询日期最近的发布日期 若数据类型选择合并原始(`dataType=101`)，则返回原始发布的发布日期 若数据类型选择合并调整(`dataType=102`)，则返回调整后最新发布日期 若数据类型选择母公司原始(`dataType=201`)，则返回母公司原始发布的发布日期 若数据类型选择母公司调整(`dataType=202`)，则返回母公司调整后最新发布日期
rptDate	string	报告日期	报告截止日期，财报统计的最后一天，在指定时间段[开始时间,结束时间]内的报告截止日期
rptType	int	报表类型	返回数据的报表类型：1-一季度报, 6-中报, 9-前三季报, 12-年报
dataType	int	数据类型	返回数据的数据类型：101-合并原始, 102-合并调整, 201-母公司原始, 202-母公司调整
fields	float	财务字段数据	指定查询 `fields` 字段的数值. 支持的字段名请参考 财务衍生指标指标

示例：

```
StkGetFinanceDerivPt("SZSE.000001,SZSE.300002", "eps_basic,eps_dil2")
```

注意：

1. 为避免未来数据问题，指定查询日期 `date` 后，返回距离此日期最近发布的一条数据。 若多个报告期报表的最新发布日期相同，返回报告日期rptDate距离查询日期date最近的一个报告期的报表数据。
2. 如果 `fields` 参数的财务字段填写不正确，或填写空字段 `null` ，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

StkGetDailyValuationPt - 查询估值指标单日截面数据（多标的）

查询指定日期截面上，股票的单日估值指标（point-in-time）

函数原型：

```
public static GMDataList<DailyValuation> StkGetDailyValuationPt(string symbols, strir
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbols	string	股票代码	Y	无	必填，可输入多个，多个标的代码必须用英文逗号分割， 如：`SHSE.600008,SZSE.000002`
fields	string	返回字段	Y	无	指定需要返回的交易衍生指标， 如有多个字段，中间用英文逗号分隔
tradeDate	string	查询日期	N	null	查询时间，时间类型为交易日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
tradeDate	string	交易日期	
fields	float	指标字段数据	指定查询 `fields` 字段的数值. 支持的字段名请参考 估值指标

示例：

```
StkGetDailyValuationPt("SZSE.000001,SZSE.300002", "pe_ttm,pe_lyr,pe_mrq")
```

注意：

1. 如果 `fields` 参数的财务字段填写不正确，或填写空字段 `null` ，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

StkGetDailyMktvaluePt - 查询市值指标单日截面数据（多标的）

查询指定日期截面上，股票的单日市值指标（point-in-time）

函数原型：

```
public static GMDataList<DailyMktvalue> StkGetDailyMktvaluePt(string symbols, string f
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbols	string	股票代码	Y	无	必填，可输入多个，多个标的代码必须用英文逗号分割， 如：`SHSE.600008,SZSE.000002`
fields	string	返回字段	Y	无	指定需要返回的交易衍生指标， 如有多个字段，中间用英文逗号分隔
tradeDate	string	查询日期	N	null	查询时间，时间类型为交易日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
tradeDate	string	交易日期	
fields	float	指标字段数据	指定查询 `fields` 字段的数值. 支持的字段名请参考 市值指标

示例：

```
StkGetDailyMktvaluePt("SZSE.000001,SZSE.300002", "tot_mv,tot_mv_csrc,a_mv")
```

注意：

1. 如果 `fields` 参数的财务字段填写不正确，或填写空字段 `null` ，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

StkGetDailyBasicPt - 查询基础指标单日截面数据（多标的）

查询指定日期截面上，股票的单日市值指标（point-in-time）

函数原型：

```
public static GMDataList<DailyBasic> StkGetDailyBasicPt(string symbols, string fields,
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbols	string	股票代码	Y	无	必填，可输入多个，多个标的代码必须用英文逗号分割， 如：`SHSE.600008,SZSE.000002`
fields	string	返回字段	Y	无	指定需要返回的交易衍生指标， 如有多个字段，中间用英文逗号分隔
tradeDate	string	查询日期	N	null	查询时间，时间类型为交易日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

一个结果集，结果集中返回字段如下：

字段名	类型	中文名称	说明
symbol	string	股票代码	
tradeDate	string	交易日期	
fields	list[float]	指标字段数据	指定查询 `fields` 字段的数值. 支持的字段名请参考 基础指标

示例：

```
StkGetDailyBasicPt("SZSE.000001,SZSE.300002", "tclose,turnrate,ttl_shr")
```

注意：

1. 如果 `fields` 参数的财务字段填写不正确，或填写空字段 `null` ，会报错提示“填写的 fields 不正确”。fields不能超过20个字段

股票增值数据函数（付费）

注意：vip特色数据权益，股票实盘客户已享有，非股票实盘客户可前往 权益中心 开通

StkGetIndustryCategory - 查询行业分类

查询指定行业来源的行业列表

函数原型：

```
public static GMDataList<StkIndustryCategory> StkGetIndustryCategory(string source = r
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
source	string	行业来源	N	'zjh2012'	'zjh2012'- 证监会行业分类 2012（默认）， 'sw2021'- 申万行业分类 2021
level	int	行业分级	N	1	1 - 一级行业（默认），2 - 二级行业，3 - 三级行业

返回值：

`StkIndustryCategory` 结构列表，参见 `StkIndustryCategory` 定义与 `GMDataList` 类的用法。

示例：

```
StkGetIndustryCategory("sw2021", 2)
```

注意：

1. 证监会行业分类 2012 没有三级行业，若输入 `source='zjh2012', level=3` 则参数无效，返回空

StkGetIndustryConstituents - 查询行业成分股

查询指定某个行业的成分股

函数原型：

```
public static GMDataList<StkIndustryConstituent> StkGetIndustryConstituents(string inc
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
industryCode	string	行业代码	Y	无	需要查询成分股的行业代码，可通过 `StkGetIndustryCategory` 获取
date	string	查询日期	N	null	查询行业成分股的指定日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

`StkIndustryConstituent` 结构列表，参见 `StkIndustryConstituent` 定义与 `GMDataList` 类的用法。

示例：

```
StkGetIndustryConstituents("A", "2022-09-05")
```

注意：

1. 只能指定一个行业代码查询成分股。

StkGetSymbolIndustry - 查询股票的所属行业

查询指定股票所属的行业

函数原型：

```
public static GMDataList<StkSymbolIndustry> StkGetSymbolIndustry(string symbols, strir
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbols	string	股票代码	Y	无	多个代码可用，多个标的代码必须用英文逗号分割 如：`"SHSE.600008,SZSE.000002"`
source	string	行业来源	N	null	'zjh2012'- 证监会行业分类 2012（默认），'sw2021'- 申万行业分类 2021)
level	int	行业分级	N	0	1 - 一级行业（默认），2 - 二级行业，3 - 三级行业
date	string	查询日期	N	null	查询行业分类的指定日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

`StkSymbolIndustry` 结构列表，参见 `StkSymbolIndustry` 定义与 `GMDataList` 类的用法。

示例：

```
StkGetSymbolIndustry("SHSE.600000, SZSE.000002", "zjh2012", 1)
```

注意：

1. 证监会行业分类 2012 没有三级行业，若输入 `source='zjh2012', level=3` 则参数无效，返回空

StkGetSectorCategory - 查询板块分类

查询指定类型的板块列表

函数原型：

```
public static GMDataList<StkSectorCategory> StkGetSectorCategory(string sectorType);
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
sectorType	string	板块类型	Y	无	只能选择一种类型，可选择 1001:市场类 1002:地域类 1003:概念类

返回值：

`StkSectorCategory` 结构列表，参见 `StkSectorCategory` 定义与 `GMDataList` 类的用法。

示例：

```
StkGetSectorCategory("1003")
```

StkGetSectorConstituents - 查询板块成分股

查询指定某个板块的成分股

函数原型：

```
public static GMDataList<StkSectorConstituent> StkGetSectorConstituents(string sectorC
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
sectorCode	string	板块代码	Y	无	需要查询成分股的板块代码，可通过 `StkGetSectorCategory` 获取

返回值：

`StkSectorConstituent` 结构列表，参见 `StkSectorConstituent` 定义与 `GMDataList` 类的用法。

示例：

```
StkGetSectorConstituents("007089")
```

注意：

1. 只能指定一个板块代码查询成分股。

StkGetSymbolSector - 查询股票的所属板块

查询指定股票所属的板块

函数原型：

```
public static GMDataList<StkSymbolSector> StkGetSymbolSector(string symbols, string se
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbols	string	股票代码	Y	无	多个代码可用, 多个标的代码必须用英文逗号分割 如： `"SHSE.600008,SZSE.000002"`
sectorType	string	板块类型	Y	无	只能选择一种类型，可选择 1001:市场类 1002:地域类 1003:概念类

返回值：

`StkSectorConstituent` 结构列表，参见 `StkSectorConstituent` 定义与 `GMDataList` 类的用法。

示例：

```
StkGetSymbolSector("SHSE.600008,SZSE.000002", "1002")
```

StkGetDividend - 查询股票分红送股信息

查询指定股票在一段时间内的分红送股信息

函数原型：

```
public static GMDataList<StockDividend> StkGetDividend(string symbol, string startDate
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	标的代码	Y	无	必填，只能填一个股票标的
startDate	string	开始时间	Y	无	必填，开始时间日期（除权除息日），%Y-%m-%d 格式
endDate	string	结束时间	Y	无	必填，结束时间日期（除权除息日），%Y-%m-%d 格式

返回值：

`StockDividend` 结构列表，参见 `StockDividend` 定义与 `GMDataList` 类的用法。

示例：

```
StkGetDividend("SHSE.600000", "2022-07-01", "2022-07-31")
```

注意：

1. 当 `startDate` 小于或等于 `endDate` 时取指定时间段的数据，
   当 `startDate` > `endDate` 时返回报错.

StkGetRation - 查询股票配股信息

查询指定股票在一段时间内的配股信息

函数原型：

```
public static GMDataList<StkRation> StkGetRation(string symbol, string startDate = nul
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	标的代码	Y	无	必填，只能填一个股票标的
startDate	string	开始时间	Y	无	必填, 开始时间日期（除权除息日），%Y-%m-%d 格式
endDate	string	结束时间	Y	无	必填, 结束时间日期（除权除息日），%Y-%m-%d 格式

返回值：

`StkRation` 结构列表，参见 `StkRation` 定义与 `GMDataList` 类的用法。

示例：

```
StkGetRation("SZSE.000728", "2005-01-01", "2022-09-30")
```

注意：

1. 当 `startDate` 小于或等于 `endDate` 时取指定时间段的数据,当 `startDate` > `endDate` 时返回报错.

StkGetAdjFactor - 查询股票的复权因子

查询某只股票在一段时间内的复权因子

函数原型：

```
public static GMDataList<StkAdjFactor> StkGetAdjFactor(string symbol, string startDate
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	标的代码	Y	无	必填，只能填一个股票标的
startDate	string	开始时间	N	null	开始时间交易日期，%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间交易日期，%Y-%m-%d 格式，默认 `null` 表示最新时间
baseDate	string	复权基准日	N	null	前复权的基准日，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

`StkAdjFactor` 结构列表，参见 `StkAdjFactor` 定义与 `GMDataList` 类的用法。

示例：

```
StkGetAdjFactor("SZSE.000651", "2015-01-01", "2022-09-01")
```

注意：

1. T+1 日复权因子会二次更新，分别约在 T 日 19:00 和 T+1 日 19:00 更新
2. 复权价格计算： `T日后复权价格 = T日不复权价格 * T日累计后复权因子` `T日前复权价格 = T日不复权价格 * T日前复权因子`
3. 上市首日后复权因子和累计后复权因子为 1，最近一次除权除息日后的前复权因子为 1
4. 前复权基准日 `baseDate` 应不早于设定的结束日期 `endDate` ，不晚于最新交易日。若设定的基准日早于 `endDate` 则等同于 `endDate` ，若设定的基准日晚于最新交易日则等同于最新交易日。
5. 当 `startDate` 小于或等于 `endDate` 时取指定时间段的数据,当 `startDate` > `endDate` 时返回报错.

StkGetShareholderNum - 查询股东户数

查询上市公司股东总数，A 股股东、B 股股东、H 股股东总数

函数原型：

```
public static GMDataList<StkShareholderNum> StkGetShareholderNum(string symbol, string
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	股票代码	Y	无	必填，只能填一个股票标的
startDate	string	开始时间	N	null	开始时间日期（公告日期），%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间日期（公告日期），%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

`StkShareholderNum` 结构列表，参见 `StkShareholderNum` 定义与 `GMDataList` 类的用法。

示例：

```
StkGetShareholderNum("SZSE.002594", "2022-01-01", "2022-08-01")
```

注意：

当 `startDate == endDate` 时，取离 `endDate` 最近公告日期的一条数据， 当 `startDat < endDate` 时，取指定时间段的数据， 当 `startDate > endDate` 时，返回报错。

StkGetTopShareholder - 查询十大股东

查询上市公司前十大股东的持股情况，包括持股数量，所持股份性质等

函数原型：

```
public static GMDataList<StkShareholder> StkGetTopShareholder(string symbol, string st
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	股票代码	Y	无	必填，只能填一个股票标的
startDate	string	开始时间	N	null	开始时间日期（公告日期），%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间日期（公告日期），%Y-%m-%d 格式，默认 `null` 表示最新时间
tradableHolder	bool	是否流通股东	N	false	false-十大股东（默认）、true-十大流通股东 默认 `false` 表示十大股东

返回值：

`StkShareholder` 结构列表，参见 `StkShareholder` 定义与 `GMDataList` 类的用法。

示例：

```
StkGetTopShareholder("SHSE.603906", "2022-06-01", "2022-08-01")
```

注意：

当 `startDate == endDate` 时，取离 `endDate` 最近公告日期的一条数据， 当 `startDat < endDate` 时，取指定时间段的数据， 当 `startDate > endDate` 时，返回报错。

StkGetShareChange - 查询股本变动

查询上市公司的一段时间内公告的股本变动情况

函数原型：

```
public static GMDataList<StkShareChange> StkGetShareChange(string symbol, string start
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	股票代码	Y	无	必填，只能填一个股票标的
startDate	string	开始时间	N	null	开始时间日期（发布日期），%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间日期（发布日期），%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

`StkShareChange` 结构列表，参见 `StkShareChange` 定义与 `GMDataList` 类的用法。

示例：

```
StkGetShareChange("SHSE.605090", "2020-01-01", "2022-10-01")
```

注意：

当 `startDate == endDate` 时，取离 `endDate` 最近发布日期的一条数据， 当 `startDat < endDate` 时，取指定时间段的数据， 当 `startDate > endDate` 时，返回报错。

期货基础数据函数（免费）

FutGetContinuousContracts - 查询连续合约对应的真实合约

查询指定时间段连续合约在每个交易日上对应的真实合约

函数原型：

```
public static GMDataList<FutContinuousContractsInfo> FutGetContinuousContracts(string
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
csymbol	string	连续合约代码	Y	无	必填，只能输入一个 支持主力合约、次主力、前 5 个月份连续和加权指数合约代码，如： 1000 股指期货主力连续合约：CFFEX.IM，1000 股指期货次主力连续合约：CFFEX.IM22， 1000 股指期货当月连续合约：CFFEX.IM00， 1000 股指期货下月连续合约：CFFEX.IM01， 1000 股指期货下季连续合约：CFFEX.IM02， 1000 股指期货隔季连续合约：CFFEX.IM03， 1000 股指期货加权指数合约：CFFEX.IM99
startDate	string	开始时间	N	null	开始时间日期，%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

`FutContinuousContractsInfo` 结构列表，参见 `FutContinuousContractsInfo` 定义与 `GMDataList` 类的用法。

示例：

```
FutGetContinuousContracts("SHFE.NI", "2022-09-01", "2022-09-15")
```

注意：

1. 具体合约（真实合约）：`交易所.品种名到期月份`对应期货具体合约 symbol，如 CFFEX.IF2206
2. 主力连续合约（期货连续合约，由真实合约拼接）：`交易所.品种名` 对应主力连续合约 symbol，如 CFFEX.IF，CFFEX.IC
3. 主力连续合约切换规则
    i. 每个品种只选出唯一一个主力合约。
    ii. 日成交量和持仓量都为最大的合约，确定为新的主力合约，每日收盘结算后判断，于下一交易日进行指向切换，日内不会进行主力合约的切换。
    iii. 按照第二条规定产生新的主力合约之前，维持原来的主力合约不变。
    iv. 若出现当前主力合约的成交量和持仓量都不是最大的情况，当前指向合约在下一个交易日必须让出主力合约身份，金融期货新主力指向成交量最大的合约（中金所），商品期货新主力指向持仓量最大的合约（上期所、大商所、郑商所、上期能源）。
4. 次主力连续合约（期货连续合约，由真实合约拼接）：`交易所.品种名 22` 对应次主力连续合约 symbol，如 CFFEX.IF22，CFFEX.IC22
5. 次主力连续合约切换规则
    i. 每个品种只选出唯一一个次主力合约。
    ii. 金融期货日成交量第二大、或商品期货日持仓量第二大的合约，确定为新的次主力合约，每日收盘结算后判断，于下一交易日进行指向切换，日内不会进行次主力合约的切换。
    iii. 按照第二条规定产生新的次主力合约之前，维持原来的次主力合约不变。
    iv. 若金融期货出现当前次主力合约的成交量、或商品期货出现当前次主力合约持仓量不是第二大的情况，当前指向合约在下一个交易日必须让出次主力合约身份，金融期货新主力指向成交量第二大的合约（中金所），商品期货新主力指向持仓量第二大的合约（上期所、大商所、郑商所、上期能源）。
6. 月份连续合约（期货连续合约，由真实合约拼接）：`交易所.品种名 月份排序` 对应月份连续合约 symbol，如 SHFE.RB00，SHFE.RB01，...，SHFE.RB04（同一品种最多有最近 5 个月的月份连续合约）
7. 月份连续合约的切换规则
    i. 该品种上市合约按交割月份排序
    ii. 00 对应最近月份合约，01 对应其后一个合约，02 对应再后一个合约，依次类推
    iii. 合约最后交易日盘后切换。
8. 当 `startDate 小于或等于 endDate` 时取指定时间段的数据,当 `startDate > endDate` 时返回报错。

期货增值数据函数（付费）

注意：vip特色数据权益，可前往 权益中心 开通

FutGetContractInfo - 查询期货标准品种信息

查询交易所披露的期货标准品种的合约规格/合约文本

函数原型：

```
public static GMDataList<FutContractInfo> FutGetContractInfo(string productCodes);
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
productCodes	string	品种代码	Y	无	必填，交易品种代码，如：IF，AL 多个代码可用，多个标的代码必须用英文逗号分割，如：`IF, AL`

返回值：

`FutContractInfo` 结构列表，参见 `FutContractInfo` 定义与 `GMDataList` 类的用法。

示例：

```
FutGetContractInfo("IF")
```

FutGetTransactionRankings - 查询期货每日成交持仓排名

查询期货合约每日成交量/持买单量/持卖单量排名

函数原型：

```
public static GMDataList<FutTransactionRanking> FutGetTransactionRankings(string symbo
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbols	string	期货合约代码	Y	无	必填，期货真实合约代码
tradeDate	string	交易日期	N	null	交易日期，%Y-%m-%d 格式，默认 null 表示最新交易日
indicators	string	排名指标	N	null	排名指标，即用于排名的依据，可选：'volume'-成交量排名（默认）, 'long'-持买单量排名, 'short'-持卖单量排名, 支持一次查询多个排名指标，如有多个指标，中间用英文逗号分隔, 默认None表示成交量排名

返回值：

`FutTransactionRanking` 结构列表，参见 `FutTransactionRanking` 定义与 `GMDataList` 类的用法。

示例：

```
FutGetTransactionRankings("SHFE.ag2212", "2022-10-10", "volume")
```

注意：

1. 当上一交易日没有进入前 20 名， `rankingChange` 返回 0.
2. 数据日频更新，当日更新前返回前一交易日的排名数据，约在交易日 20 点左右更新当日数据。

FutGetWarehouseReceipt - 查询期货仓单数据

查询交易所在交易日期货品种的注册仓单数量、有效预报

- 期货仓单是指由期货交易所指定交割仓库，按照期货交易所指定的程序，签发的符合合约规定质量的实物提货凭证。记录了交易所所有期货实物的库存情况以及变更情况。

函数原型：

```
public static GMDataList<FutWarehouseReceiptInfo> FutGetWarehouseReceipt(string produc
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
productCode	string	品种代码	Y	无	必填，只能填写一个交易品种代码，如：AL
startDate	string	开始时间	N	null	开始时间日期，%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

`FutWarehouseReceiptInfo` 结构列表，参见 `FutWarehouseReceiptInfo` 定义与 `GMDataList` 类的用法。

示例：

```
FutGetWarehouseReceipt("AL", "2023-06-20", "2023-06-29")
```

注意：

1. 支持郑商所、大商所、上期所和上海国际能源交易中心的期货品种。
2. 注册仓单数量每日更新，其余数据上期所一周一披露，郑商所一天一披露。
3. 当 `startDate` 小于或等于 `endDate` 时, 取指定时间段的数据,当 `startDate` > `endDate` 时, 返回报错。

基金增值数据函数（付费）

FndGetEtfConstituents - 查询ETF最新成分股

查询某只 ETF 在最新交易日的成分股持有情况和现金替代信息

注意：vip特色数据权益，股票实盘客户已享有，非股票实盘客户可前往 权益中心 开通

函数原型：

```
public static GMDataList<FndEtfConstituents> FndGetEtfConstituents(string symbol);
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	ETF代码	Y	无	必填，只能输入一个symbol, 如： `SZSE.159919`

返回值：

`FndEtfConstituents` 结构列表，参见 `FndEtfConstituents` 定义与 `GMDataList` 类的用法。

示例：

```
FndGetEtfConstituents("SHSE.510050")
```

注意：

1. 只返回上交所、深交所的成分股，不提供其余交易所的成分股票。

FndGetStockPortfolio - 查询基金资产组合（股票投资组合）

查询某只基金在指定日期的基金资产组合（股票投资组合）

注意：vip特色数据权益，可前往 权益中心 开通

函数原型：

```
public static GMDataList<FndPortfolioStockInfo> FndGetStockPortfolio(string symbol, ir
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	基金代码	Y	无	必填，只能输入一个基金的[symbol]， 如： `SZSE.161133`
reportType	int	报表类别	Y	无	公布持仓所在的报表类别，必填，可选： 1:第一季度 2:第二季度 3:第三季报 4:第四季度 6:中报 12:年报
startDate	string	开始时间	N	null	开始时间日期（公告日），%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间日期（公告日），%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

`FndPortfolioStockInfo` 结构列表，参见 `FndPortfolioStockInfo` 定义与 `GMDataList` 类的用法。

示例：

```
FndGetStockPortfolio("SHSE.510300", 1)
```

注意：

1. 仅提供场内基金（ETF、LOF、FOF-LOF）的资产组合数据。
2. 当 `startDate == endDate` 时，取离 `endDate` 最近公告日期的一条数据，
   当 `startDat < endDate` 时，取指定时间段的数据， 当 `startDate > endDate` 时，返回报错。

FndGetBondPortfolio - 查询基金资产组合（债券投资组合）

查询某只基金在指定日期的基金资产组合（债券投资组合）

注意：vip特色数据权益，可前往 权益中心 开通

函数原型：

```
public static GMDataList<FndPortfolioBondInfo> FndGetBondPortfolio(string symbol, int
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	基金代码	Y	无	必填，只能输入一个基金的[symbol]， 如： `SZSE.161133`
reportType	int	报表类别	Y	无	公布持仓所在的报表类别，必填，可选： 1:第一季度 2:第二季度 3:第三季报 4:第四季度 6:中报 9:前三季报 12:年报
startDate	string	开始时间	N	null	开始时间日期（公告日），%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间日期（公告日），%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

`FndPortfolioBondInfo` 结构列表，参见 `FndPortfolioBondInfo` 定义与 `GMDataList` 类的用法。

示例：

```
FndGetBondPortfolio("SHSE.510300", 1)
```

注意：

1. 仅提供场内基金（ETF、LOF、FOF-LOF）的资产组合数据。
2. 当 `startDate == endDate` 时，取离 `endDate` 最近公告日期的一条数据，
   当 `startDat < endDate` 时，取指定时间段的数据， 当 `startDate > endDate` 时，返回报错。

FndGetFundPortfolio - 查询基金资产组合（基金投资组合）

查询某只基金在指定日期的基金资产组合（基金投资组合）

注意：vip特色数据权益，可前往 权益中心 开通

函数原型：

```
public static GMDataList<FndPortfolioFundInfo> FndGetFundPortfolio(string symbol, int
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	基金代码	Y	无	必填，只能输入一个基金的[symbol]， 如： `SZSE.161133`
reportType	int	报表类别	Y	无	公布持仓所在的报表类别，必填，可选： 1:第一季度 2:第二季度 3:第三季报 4:第四季度 6:中报 9:前三季报 12:年报
startDate	string	开始时间	N	null	开始时间日期（公告日），%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间日期（公告日），%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

`FndPortfolioFundInfo` 结构列表，参见 `FndPortfolioFundInfo` 定义与 `GMDataList` 类的用法。

注意：

1. 仅提供场内基金（ETF、LOF、FOF-LOF）的资产组合数据。
2. 当 `startDate == endDate` 时，取离 `endDate` 最近公告日期的一条数据，
   当 `startDat < endDate` 时，取指定时间段的数据， 当 `startDate > endDate` 时，返回报错。

FndGetNetValue - 查询基金净值数据

查询某只基金在指定时间段的基金净值数据

注意：vip特色数据权益，可前往 权益中心 开通

函数原型：

```
public static GMDataList<FndNetValueInfo> FndGetNetValue(string symbol, string startDa
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	基金代码	Y	无	必填，只能输入一个基金的[symbol]如：`SZSE.159919`
startDate	string	开始时间	N	null	开始时间日期，%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间日期，%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

`FndNetValueInfo` 结构列表，参见 `FndNetValueInfo` 定义与 `GMDataList` 类的用法。

示例：

```
FndGetNetValue("SHSE.510300")
```

注意：

1. 仅提供场内基金（ETF、LOF、FOF-LOF）的净值数据。
2. 当 `startDate == endDate` 时，取离 `endDate` 最近日期的一条数据， 当 `startDate < endDate` 时，取指定时间段的数据， 当 `startDate > endDate` 时，返回报错。

FndGetAdjFactor - 查询基金复权因子

查询某只基金在一段时间内的复权因子

注意：vip特色数据权益，可前往 权益中心 开通

函数原型：

```
public static GMDataList<FndAdjFactorInfo> FndGetAdjFactor(string symbol, string start
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	基金代码	Y	无	必填，只能输入一个基金的[symbol]， 如： `SZSE.159919`
startDate	string	开始时间	N	null	开始时间交易日期，%Y-%m-%d 格式， 默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间交易日期，%Y-%m-%d 格式， 默认 `null` 表示最新时间
baseDate	string	复权基准日	N	null	前复权的基准日，%Y-%m-%d 格式， 默认 `null` 表示最新时间

返回值：

`FndAdjFactorInfo` 结构列表，参见 `FndAdjFactorInfo` 定义与 `GMDataList` 类的用法。

示例：

```
FndGetAdjFactor("SHSE.510300")
```

注意：

1. T+1 日复权因子会二次更新，分别在 T 日 19:00 和 T+1 日 19:00 更新。仅提供场内基金（ETF、LOF、FOF-LOF）的复权因子数据。
2. 复权价格计算： `T日后复权价格 = T日不复权价格 T日累计后复权因子` `T日前复权价格 = T日不复权价格 T日前复权因子`
3. 上市首日后复权因子合累计后复权因子为 1，最近一次除权除息日后的交易日前复权因子为 1
4. 前复权基准日 `baseDate` 应不早于设定的结束日期 `endDate` ，不晚于最新交易日。若设定的基准日早于 `endDate` 则等同于 `endDate` ，若设定的基准日晚于最新交易日则等同于最新交易日。
5. 当 `startDate` 小于或等于 `endDate` 时取指定时间段的数据,当 `startDate` > `endDate` 时返回报错.

FndGetDividend - 查询基金分红信息

查询指定基金在一段时间内的分红信息

注意：vip特色数据权益，可前往 权益中心 开通

函数原型：

```
public static GMDataList<FndDividendInfo> FndGetDividend(string symbol, string startDa
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	基金代码	Y	无	必填，只能输入一个基金的[symbol], 如： `SZSE.159919`
startDate	string	开始时间	Y	无	必填，开始时间日期（场内除息日），%Y-%m-%d 格式
endDate	string	结束时间	Y	无	必填，结束时间日期（场内除息日），%Y-%m-%d 格式

返回值：

`FndDividendInfo` 结构列表，参见 `FndDividendInfo` 定义与 `GMDataList` 类的用法。

示例：

```
FndGetDividend("SHSE.510300", "2021-1-1", "2023-1-1")
```

FndGetSplit - 查询基金拆分折算信息

查询指定基金在一段时间内的拆分折算信息

注意：vip特色数据权益，可前往 权益中心 开通

函数原型：

```
public static GMDataList<FndSplitInfo> FndGetSplit(string symbol, string startDate, st
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	基金代码	Y	无	必填，只能输入一个基金的[symbol]，如：`SZSE.159919`
startDate	string	开始时间	Y	无	必填，开始时间日期（场内除权日），%Y-%m-%d 格式
endDate	string	结束时间	Y	无	必填，结束时间日期（场内除权日），%Y-%m-%d 格式

返回值：

`FndSplitInfo` 结构列表，参见 `FndSplitInfo` 定义与 `GMDataList` 类的用法。

示例：

```
FndGetSplit("SZSE.161725", "2000-01-01", "2022-09-07")
```

可转债增值数据函数（付费）

注意：vip特色数据权益，可前往 权益中心 开通

BndGetConversionPrice - 查询可转债转股价变动信息

查询可转债一段时间的转股价变动和转股结果

函数原型：

```
public static GMDataList<BndConversionPrice> BndGetConversionPrice(string symbol, stri
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	可转债代码	Y	无	必填，只能输入一个可转债的[symbol]
startDate	string	开始时间	N	null	开始时间日期（转股价格生效日），%Y-%m-%d 格式， 默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间日期（转股价格生效日），%Y-%m-%d 格式， 默认 `null` 表示最新时间

返回值：

`BndConversionPrice` 结构列表，参见 `BndConversionPrice` 定义与 `GMDataList` 类的用法。

示例：

```
BndGetConversionPrice("SZSE.123015")
```

注意：

1. 本期转股数、累计转股金额、债券流通余额在执行日期收盘后才有数据。
2. 当 `startDate == endDate` 时，取离 `endDate` 最近转股价格生效日期的一条数据，当 `startDate < endDate` 时，取指定时间段的数据， 当 `startDate > endDate` 时，返回报错。

BndGetCallInfo - 查询可转债赎回信息

查询可转债一段时间内的赎回情况

函数原型：

```
public static GMDataList<BndCallInfo> BndGetCallInfo(string symbol, string startDate =
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	可转债代码	Y	无	必填，只能输入一个可转债的[symbol]
startDate	string	开始时间	N	null	开始时间日期（公告日），%Y-%m-%d 格式， 默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间日期（公告日），%Y-%m-%d 格式， 默认 `null` 表示最新时间

返回值：

`BndCallInfo` 结构列表，参见 `BndCallInfo` 定义与 `GMDataList` 类的用法。

示例：

```
BndGetCallInfo("SHSE.110041")
```

注意：

当 `startDate == endDate` 时，取离 `endDate` 最近公告日的一条数据， 当 `startDate < endDate` 时，取指定时间段的数据， 当 `startDate > endDate` 时，返回报错。

BndGetPutInfo - 查询可转债回售信息

查询可转债一段时间内的回售情况

函数原型：

```
public static GMDataList<BndPutInfo> BndGetPutInfo(string symbol, string startDate = r
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	可转债代码	Y	无	必填，只能输入一个可转债的[symbol]
startDate	string	开始时间	N	null	开始时间日期（公告日），%Y-%m-%d 格式， 默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间日期（公告日），%Y-%m-%d 格式， 默认 `null` 表示最新时间

返回值：

`BndPutInfo` 结构列表，参见 `BndPutInfo` 定义与 `GMDataList` 类的用法。

示例：

```
BndGetPutInfo("SZSE.128015")
```

注意：

当 `startDate == endDate` 时，取离 `endDate` 最近公告日的一条数据， 当 `startDate < endDate` 时，取指定时间段的数据， 当 `startDate > endDate` 时，返回报错。

BndGetAmountChange - 查询可转债剩余规模变动

查询可转债转股、回售、赎回等事件导致的剩余规模变动的情况

函数原型：

```
public static GMDataList<BndAmountChange> BndGetAmountChange(string symbol, string sta
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
symbol	string	可转债代码	Y	无	必填，只能输入一个可转债的[symbol]
startDate	string	开始时间	N	null	开始时间日期（变动日期），%Y-%m-%d 格式，默认 `null` 表示最新时间
endDate	string	结束时间	N	null	结束时间日期（变动日期），%Y-%m-%d 格式，默认 `null` 表示最新时间

返回值：

`BndAmountChange` 结构列表，参见 `BndAmountChange` 定义与 `GMDataList` 类的用法。

示例：

```
BndGetAmountChange("SZSE.123015")
```

注意：

1. 变动类型指定为首发时，返回的剩余金额为发行金额。
2. 当 `startDate == endDate` 时，取离 `endDate` 最近变动日期的一条数据，当 `startDate < endDate` 时，取指定时间段的数据， 当 `startDate > endDate` 时，返回报错。

结果集合类

GMDataList

GMDataList 类定义

GMDataList 模板类是所有返回列表型数据函数的标准返回， 表示一个列表数据存储。类声明如下:

```
public class GMDataList<T>
{
    public List<T> data;            //数据
    public int status;              //状态码， 0表示成功， 其它表示错误码
}
```

使用举例

```
//查询一段tick行情
GMDataList<Tick> dl = GMApi.HistoryTicks("SHSE.600000", "2018-07-16 09:30:00", "2018-0

if (dl.status == 0) //判断查询是否成功
{
    //遍历行情数组
    foreach(var v in dl.data)
    {
        Console.WriteLine("{0} {1}", v.symbol, v.price);
    }
}
```

GMData

GMData 类定义

GMData 模板类是所有返回类数据函数的标准返回， 表示一个对象数据。类声明如下:

```
public class GMData<T>
{
    public T data;                  //数据
    public int status;              //状态码， 0表示成功， 其它表示错误码
}
```

使用举例

```
//获取深交所最新的代码信息
GMData<DataTable> ds = GMApi.GetInstruments(null, "SZSE");
if(ds.status == 0)
{
    var dt = ds.data;
    for (int i = 0; i < dt.Rows.Count; i++)
    {
        Console.WriteLine("{0},{1},{2} ", dt.Rows[i]["symbol"], dt.Rows[i]["sec_level"
    }
}
```

数据结构

数据类

Tick - Tick结构

逐笔行情数据

```
public class Tick
{
    public string symbol;                   //symbol

    public DateTime createdAt;              //时间
    public float price;                     //最新价
    public float open;                      //开盘价
    public float high;                      //最高价
    public float low;                       //最低价
    public double cumVolume;                //成交总量
    public double cumAmount;                //成交总金额/最新成交额，累计值
    public System.Int64 cumPosition;        //合约持仓量（期），累计值
    public double lastAmount;               //瞬时成交额
    public int lastVolume;                  //瞬时成交量
    public int tradeType;                   //交易类型，对应多开，多平等类型
    public Quote[] quotes;                  //报价，下标从0开始，0-表示第一档，1-表示第二

};
```

报价 `Quote`

```
public class Quote
{
    public float bidPrice;            //本档委买价
    public System.Int64 bidVolume;    //本档委买量
    public float askPrice;            //本档委卖价
    public System.Int64 askVolume;    //本档委卖量
};
```

Bar - Bar结构

bar数据是指各种频率的行情数据

```
public class Bar
{
    public string symbol;
    public DateTime bob;            //bar的开始时间
    public DateTime eob;            //bar的结束时间
    public float open;              //开盘价
    public float close;             //收盘价
    public float high;              //最高价
    public float low;               //<最低价
    public double volume;           //成交量
    public double amount;           //成交金额
    public float preClose;          //昨收盘价，只有日频数据赋值

    public System.Int64 position;   //持仓量
    public string frequency;        //bar频度
};
```

SymbolInfo - 标的交易静态信息

```
public struct SymbolInfo
{
    //标的代码
    public string symbol;
    //证券品种大类
    public int secType1;
    //证券品种细类
    public int secType2;
    //板块
    public int board;
    //交易所代码
    public string exchange;
    //交易所标的代码
    public string secId;
    //交易所标的名称
    public string secName;
    //交易所标的简称
    public string secAbbr;
    //最小变动单位
    public double priceTick;
    //交易制度
    public int tradeN;
    //上市日期
    public DateTime listedDate;
    //退市日期
    public DateTime delistedDate;
    //标的资产
    public string underlyingSymbol;
    //行权方式
    public string optionType;
    //期权保证金计算参数1
    public double optionMarginRatio1;
    //期权保证金计算参数2
    public double optionMarginRatio2;
    //合约类型
    public string callOrPut;
    //可转债开始转股日期
    public DateTime conversionStartDate;
}
```

Symbol - 标的交易信息

```
public struct Symbol
{
    public DateTime trade_date;
    // 合约调整
    public bool is_adjusted;
    // 是否停牌
    public bool is_suspended;
    // 持仓量
    public long position;
    // 结算价
    public double settle_price;
    // 昨结算价
    public double pre_settle;
    // 昨收盘价
    public double pre_close;
    // 涨停价
    public double upper_limit;
    // 跌停价
    public double lower_limit;
    // 换手率
    public double turn_rate;
    // 复权因子
    public double adj_factor;
    // 保证金比例
    public double margin_ratio;
    // 转股价
    public double conversion_price;
    // 行权价
    public double exercise_price;
    // 合约乘数
    public long multiplier;
    // 包含 info 中的字段
    public SymbolInfo info;
    // 是否ST
    public bool is_st;
    // 做市 市场分层 '0'-基础层，'1'-创新层，'2'-北交所
    public int market_level;
    // 做市 做市商数量 有做市商提供做市服务的证券，返回其做市商数量；没有做市商提供做市服务的
    public int total_market_makers;
    // 做市 交易方式 'T'-协议交易方式，'M'-做市交易方式，'B'-集合竞价+连续竞价交易方式，'C'
    public string trade_type;
    // 做市 买数量单位 单笔做市买入申报数量是unit_volume_buy的整数倍
    public int unit_volume_buy;
    // 做市 卖数量单位 单笔做市卖出申报数量是unit_volume_sell的整数倍
    public int unit_volume_sell;
}
```

TradeDate - 年度交易日历

```
public struct TradeDate
{
    //自然日期，查询年份的自然日日期
    public DateTime date;
    //交易日期
    public DateTime tradeDate;
    //下一交易日
    public DateTime nextTradeDate;
    //上一交易日
    public DateTime preTradeDate;
}
```

TradingSession - 交易时段

```
public struct TradingSession
{
    //交易时间
    public struct Session
    {
        public string start;
        public string end;
    }

    //标的代码
    public string symbol;
    //交易所代码
    public string exchange;
    //连续竞价时段
    public List<Session> timeTrading;
    //集合竞价时段
    public List<Session> timeAuctions;
}
```

ContractExpireRestDays - 合约到期剩余天数

```
public struct ContractExpireRestDays
{
    //日期
    public string date;
    //标的代码
    public string symbol;
    //
    public string daysToExpire;
}
```

IndustryCategory - 行业分类

```
public struct IndustryCategory
{
    // 行业代码
    public string industryCode;
    // 行业名称
    public string industryName;
}
```

IndustryConstituent - 行业成分股

```
public struct IndustryConstituent
{
    // 行业代码
    public string industryCode;
    // 行业名称
    public string industryName;
    // 成分股票代码, 格式: exchange.secId
    public string symbol;
    // 成分股票名称
    public string secName;
    // 纳入日期, 本地时间
    public DateTime dateIn;
    // 剔除日期, 本地时间
    public DateTime dateOut;
}
```

SymbolIndustry - 股票所属行业

```
public struct SymbolIndustry
{
    // 股票代码, 格式: exchange.secId
    public string symbol;
    // 股票名称
    public string secName;
    // 行业代码
    public string industryCode;
    // 行业名称
    public string industryName;
}
```

SectorCategory - 板块分类

```
public struct  SectorCategory
{
    //版块代码
    public string sectorCode;
    //版块名称
    public string sectorName;
}
```

SectorConstituent - 板块成分股

```
public struct SectorConstituent
{
    // 板块代码
    public string sectorCode;
    // 板块名称
   public string sectorName;
    // 股票代码, 格式: exchange.sec_id
  public  string symbol;
    // 成分股票名称
   public  string secName;
}
```

SymbolSector - 股票所属板块

```
public struct SymbolSector
{
    // 股票代码，格式：exchange.sec_id
    public string symbol;
    // 股票名称
    public string secName;
    // 板块代码
    public string sectorCode;
    // 板块名称
    public string sectorName;
}
```

IndexConstituent - 指数成分股

```
public struct IndexConstituent
{
    // 指数代码
    public string index;
    // 成分股代码
    public string symbol;
    // 成分股权重
    public double weight;
    // 交易日期，本地时间，格式为：YYYY-MM-DD
    public DateTime date;
    //总市值
    public double marketValueTotal;
    //流通市值
    public double marketValueCirc;
}
```

StockDividend - 股票分红送股信息

```
public struct StockDividend
{
    // 股票代码
    public string symbol;
    // 分配方案，如现金分红，送股，配股，转增
    public string schemeType;
    // 公告日，本地时间
    public DateTime pubDate;
    // 除权除息日，本地时间
    public DateTime exDate;
    // 股权登记日，本地时间
    public DateTime equityRegDate;
    // 现金红利发放日(派息日)，本地时间，格式为：YYYY-MM-DD
    public DateTime cashPayDate;
    // 送（转增）股份到账日，本地时间，格式为：YYYY-MM-DD
    public DateTime shareAcctDate;
    // 红股上市日，送（转增）股份上市交易日，本地时间，格式为：YYYY-MM-DD
    public DateTime shareLstDate;
    // 税后红利（元/10股）
    public double cashAfTax;
    // 税前红利（元/10股）
    public double cashBfTax;
    // 送股比例，10:X
    public double bonusRatio;
    // 转增比例，10:X
    public double convertRatio;
    // 盈余公积金转增比例，10:X
    public double surRsvRatio;
    // 资本公积金转增比例，10:X
    public double capRsvRatio;
    // 股本基准日
    public string baseDate;
    // 股本基数(基准股本)
    public double baseShare;
    // 配股比例
    public double rationRatio;
    // 配股价格
    public double rationPrice;
}
```

StockRation - 股票配股信息

```
public struct StockRation
{
    // 标的代码
    public string symbol;
    // 公告日
    public DateTime pubDate;
    //股权登记日
    public DateTime equityRegDate;
    //除权除息日
    public DateTime ex_date;
    //配股比例
    public double rationRatio;
    //配股价格
    public double rationPrice;
}
```

AdjFactor - 股票复权因子

```
public struct AdjFactor
{
    // 交易日期
    public DateTime tradeDate;
    // 当日后复权因子，T日后复权因子=T-1日的收盘价/T日前收价
    public double adjFactorBwd;
    // 当日累计后复权因子，T日累计后复权因子=T日后复权因子*T-1日累计后复权因子
    //（第一个累计后复权因子=第一个后复权因子）
    public double adjFactorBwdAcc;
    // 当日前复权因子，T日前复权因子=T日后复权因子/复权基准日后复权因子
    public double adjFactorFwd;
    // 当日累计前复权因子，T日累计前复权因子=T日后复权因子
    // T-1日累计前复权因子=T日后复权因子*T-1日后复权因子
    //（第一个累计前复权因子=最新累计后复权因子）
    public double adjFactorFwdAcc;
}
```

ShareholderNum - 股东户数

```
public struct ShareholderNum
{
    // 股票代码
    public string symbol ;
    // 股票名称
    public string secName;
    // 公告日期
    public DateTime pubDate ;
    // 截止日期
    public DateTime expiryDate ;
    // 股东总数
    public long totalShare ;
    // A股股东总数
    public long totalShareA ;
    // 流通B股股东总数
    public long totalShareB ;
    // 流通H股股东总数
    public long totalShareH ;
    // 其他股东户数
    public long otherShare ;
    // 优先股股东总数（表决权恢复）
    public long totalSharePfd ;
    // 股东户数（含融资融券）
    public long totalShareMgn ;
    // 股东户数（不含融资融券）
    public long totalShareNoMgn ;
}
```

Shareholder - 十大股东

```
public struct Shareholder
{
    // 股票代码
    public string symbol;
    // 股票名称
    public string secName;
    // 公告日期
    public DateTime pubDate;
    // 截止日期
    public DateTime expiryDate;
    // 股东名称
    public string holderName;
    // 股东序号（名次）
    public int holderRank;
    // 股东类型
    public string holderType;
    // 股东性质
    public string holderAttr;
    // 股份类型(股份性质)
    public string shareType;
    // 持有数量（股）
    public double shareNum;
    // 持股比例1，持股占总股本比例（%）
    public double shareRatio1;
    // 持股比例2，持股占已上市流通股比例（%）
    public double shareRatio2;
    // 质押股份数量，股权质押涉及股数（股）
    public double sharePledge;
    // 冻结股份数量，股权冻结涉及股数（股）
    public double shareFreeze;
}
```

ShareChange - 股本变动

```
public struct ShareChange
{
    // 股票代码
    public string symbol;
    // 公司名称
    public string companyName;
    // 发布日期
    public DateTime pubDate;
    // 股本变动日期
    public DateTime chgDate;
    // 股本变动原因
    public string chgReason;
    // 股本变动事件
    public string chgEvent;
    // 总股本，未流通股份+已流通股份，单位：股
    public double shareTotal;
    // 未流通股份
    public double shareTotalNlf;
    // 发起人股份：国有发起人股 + 发起社会法人股 + 其他发起人股份，单位：股
    public double shareProm;
    // 国有发起人股：国家持股+国有法人股，单位：股
    public double sharePromState;
    // 国家股
    public double shareState;
    // 国有法人股
    public double shareStateLp;
    // 发起社会法人股：境内社会法人股+境外法人股，单位：股
    public double sharePromSoc;
    // 境内社会法人股
    public double shareDcLp;
    // 境外法人股
    public double shareOsLp;
    // 其他发起人股份
    public double sharePromOther;
    // 募集人股份：募集国家股+募集境内法人股+募集境外法人股，单位：股
    public double shareRs;
    // 募集国家股
    public double shareRsState;
    // 募集境内法人股：募集境内国有法人股+募集境内社会法人股，单位：股
    public double shareRsDcLp;
    // 募集境内国有法人股
    public double shareRsStateLp;
    // 募集境内社会法人股
    public double shareRsSocLp;
    // 募集境外法人股
    public double shareRsOsLp;
    // 内部职工股
    public double shareEmpNlf;
    // 优先股
    public double sharePfdNlf;
    // 其他未流通股份
    public double shareOthNlf;
    // 流通股份
    public double shareCirc;
    // 无限售条件股份
    public double shareTtlUnl;
    // 人民币普通股（A股）
    public double shareAUnl;
    // 境内上市外资股（B股）
    public double shareBUnl;
    // 境外上市外资股（H股）
    public double shareHUnl;
    // 其他已流通股份
    public double shareOthUnl;
    // 有限售条件股份
```

```
    public double shareTtlLtd;
    // 一般有限售条件股份: 限售国家持股+ 限售国有法人持股+ 限售其他内资持股+ 限售外资持股+ 锁
    public double shareGenLtd;
    // 限售国家持股
    public double shareStateLtd;
    // 限售国有法人持股
    public double shareStateLpLtd;
    // 限售其他内资持股: 限售境内非国有法人持股+限售境内自然人持股, 单位: 股
    public double shareOthDcLtd;
    // 限售境内非国有法人持股
    public double shareNstDcLpLtd;
    // 限售境内自然人持股
    public double shareDcNpLtd;
    // 限售外资持股: 限售境外法人持股+限售境外自然人持股, 单位: 股
    public double shareFornLtd;
    // 限售境外法人持股
    public double shareOsLpLtd;
    // 限售境外自然人持股
    public double shareOsNpLtd;
    // 锁定股份
    public double shareLkLtd;
    // 高管持股(原始披露)
    public double shareGmLtd;
    // 配售法人持股: 战略投资者配售股份+一般法人投资者配售+ 证券投资基金配售股份, 单位: 股
    public double sharePlcLpLtd;
    // 战略投资者配售股份
    public double sharePlcSiLtd;
    // 一般法人投资者配售股份
    public double sharePlcLpGenLtd;
    // 证券投资基金配售股份
    public double sharePlcFndLtd;
    // 限售流通A股
    public double shareALtd;
    // 限售流通B股
    public double shareBLtd;
    // 限售流通H股
    public double shareHLtd;
    // 其他限售股份
    public double shareOthLtd;
    // 变动股份上市日
    public DateTime shareListDate;
}
```

FundamentalsBalance - 资产负债表

```
public struct FundamentalsBalance
{
    // 股票代码
    public string symbol;
    // 发布日期
    // 在指定时间段[开始时间,结束时间]内的最新发布日期,
    // 若数据类型选择合并原始(data_type=101), 则返回原始发布的发布日期
    // 若数据类型选择合并调整(data_type=102), 则返回调整后最新发布日期
    public DateTime pubDate;
    // 报告截止日期, 财报统计的最后一天
    public DateTime rptDate;
    // 相应指定查询 fields字段的值. 字典key值请参考 资产负债表
    public Dictionary<string, string> data;
}
```

FundamentalsCashflow - 利润表

```
public struct FundamentalsCashflow
{
    // 股票代码
    public string symbol;
    // 发布日期
    // 在指定时间段[开始时间,结束时间]内的最新发布日期,
    // 若数据类型选择合并原始(data_type=101)，则返回原始发布的发布日期
    // 若数据类型选择合并调整(data_type=102)，则返回调整后最新发布日期
    public DateTime pubDate;
    // 报告截止日期，财报统计的最后一天
    public DateTime rptDate;
    // 相应指定查询 fields字段的值. 字典key值请参考 利润表
    public Dictionary<string, string> data;
}
```

FundamentalsIncome - 现金流量表

```
public struct FundamentalsIncome
{
    // 股票代码
    public string symbol;
    // 发布日期
    // 在指定时间段[开始时间,结束时间]内的最新发布日期,
    // 若数据类型选择合并原始(data_type=101)，则返回原始发布的发布日期
    // 若数据类型选择合并调整(data_type=102)，则返回调整后最新发布日期
    public DateTime pubDate;
    // 报告截止日期，财报统计的最后一天
    public DateTime rptDate;
    // 相应指定查询 fields字段的值. 字典key值请参考 现金流量表
    public Dictionary<string, string> data;
}
```

FinancePrime - 财务主要指标

```
public struct FinancePrime
{
    // 股票代码
    public string symbol;
    // 发布日期
    // 在指定时间段[开始时间,结束时间]内的最新发布日期,
    // 若数据类型选择合并原始(data_type=101)，则返回原始发布的发布日期
    // 若数据类型选择合并调整(data_type=102)，则返回调整后最新发布日期
    public DateTime pubDate;
    // 在指定时间段[开始时间,结束时间]内的报告截止日期,
    // 报告截止日期，财报统计的最后一天
    public DateTime rptDate;
    // 报表类型
    public int rptType;
    // 数据类型
    public int dataType;
    // 相应指定查询 fields字段的值. 字典key值请参考 财务主要指标
    public  Dictionary<string, string> data;
}
```

FinanceDeriv - 财务衍生指标

```
public struct FinanceDeriv
{
    // 股票代码
    public string symbol;
    // 发布日期
    // 在指定时间段[开始时间,结束时间]内的最新发布日期,
    // 若数据类型选择合并原始(data_type=101)，则返回原始发布的发布日期
    // 若数据类型选择合并调整(data_type=102)，则返回调整后最新发布日期
    public DateTime pubDate;
    // 在指定时间段[开始时间,结束时间]内的报告截止日期,
    // 报告截止日期，财报统计的最后一天
    public DateTime rptDate;
    // 报表类型
    public int rptType;
    // 数据类型
    public int dataType;
    // 相应指定查询 fields字段的值. 字典key值请参考 财务衍生指标
    public Dictionary<string, string> data;
}
```

DailyValuation - 交易衍生指标-估值类

```
public struct DailyValuation
{
    // 股票代码
    public string symbol;
    // 交易日期
    public DateTime tradeDate;
    // 相应指定查询 fields字段的值. 字典key值请参考 交易衍生指标-估值类
    public Dictionary<string, string> data;
}
```

DailyMktvalue - 交易衍生指标-市值类

```
public struct DailyMktvalue
{
    // 股票代码
    public string symbol;
    // 交易日期
    public DateTime tradeDate;
    // 相应指定查询 fields字段的值. 字典key值请参考 交易衍生指标-市值类
    public Dictionary<string, string> data;
}
```

DailyBasic - 交易衍生指标-基础类

```
public struct DailyBasic
{
    // 股票代码
    public string symbol;
    // 交易日期
    public DateTime tradeDate;
    // 相应指定查询 fields字段的值. 字典key值请参考 交易衍生指标-基础类
    public Dictionary<string, string> data;
}
```

ContinuousContractsInfo - 期货连续合约映射

```
public struct ContinuousContractsInfo
{
    // 标的代码
    public string symbol;
    // 交易日期
    public DateTime tradeDate;
}
```

FutContractInfo - 期货标准品种信息

```
public struct FutContractInfo
{
    // 交易品种  --交易品种名称，如：沪深300指数，铝
    public string productName;
    // 交易代码  --交易品种代码，如：IF，AL
    public string productCode;
    // 合约标的 --如：SHSE.000300， AL
    public string underlyingSymbol;
    // 合约乘数  --如：200，5
    public int multiplier;
    // 交易单位  --如：每点人民币200元，5吨/手
    public string tradeUnit;
    // 报价单位   --如：指数点，元（人民币）/吨
    public string priceUnit;
    // 价格最小变动单位  --如：0.2点，5元/吨
    public string priceTick;
    // 合约月份  --如：当月、下月及随后两个季月，1～12月
    public string deliveryMonth;
    // 交易时间  --如：“9:30-11:30，13:00-15:00”，“上午9:00－11:30 ， 下午1:30－3:00和交易
    public string tradeTime;
    // 涨跌停板幅度  --每日价格最大波动限制，如：“上一个交易日结算价的±10%”，“上一交易日结算
    public string priceRange;
    // 最低交易保证金  --交易所公布的最低保证金比例，如：“合约价值的8%”，“合约价值的5%”
    public string minimumMargin;
    // 最后交易日   -- 如：“合约到期月份的第三个星期五，遇国家法定假日顺延”，“合约月份的15日
    public string lastTradeDate;
    // 交割日期  --如：“同最后交易日”，“最后交易日后连续三个工作日”
    public string deliveryDate;
    // 交割方式  --如：现金交割，实物交割
    public string deliveryMethod;
    // 交易所名称 --上市交易所名称，如：中国金融期货交易所，上海期货交易所
    public string exchangeName;
    // 交易所代码  --上市交易所代码，如：CFFEX，SHFE
    public string exchange;

}
```

FutTransactionRanking - 期货每日成交持仓排名

```
public struct FutTransactionRanking
{
// 期货合约代码  --必填，使用时参考symbol
public string symbol;
    // 交易日期  --
    public DateTime tradeDate;
    // 期货公司会员简称
    public string memberName;
    // 排名指标
    public string indicator;
    // 排名指标数值  --单位：手。视乎所选的排名指标indicator，分别为：成交量（indicator为'v
    public int indicatorNumber;
    // 排名指标比上交易日增减  --单位：手
    public int indicatorChange;
    // 排名名次
    public int ranking;
    // 排名名次比上交易日增减
    public int rankingChange;
    // 判断 ranking_change 的值是否为空
    public bool rankingChangeIsNull;
}
```

WarehouseReceiptInfo - 期货仓单数据

```
public struct WarehouseReceiptInfo
{
    // 交易日期 --
    public DateTime tradeDate;
    // 期货交易所代码 --期货品种对应交易所代码，如：CFFEX，SHFE
    public string exchange;
    // 期货交易所名称 --上市交易所名称，如：中国金融期货交易所，上海期货交易所
    public string exchangeName;
    // 交易代码 --交易品种代码，如：IF，AL
    public string productCode;
    // 交易品种 --交易品种名称，如：沪深300指数，铝
    public string productName;
    // 注册仓单数量 --
    public int onWarrant;
    // 仓单单位 -- 仅支持郑商所品种
    public string warrantUnit;
    // 仓库名称 --
    public string warehouse;
    // 期货库存 --
    public int futureInventory;
    // 期货库存增减 --
    public int futureInventoryChange;
    // 可用库容量 --
    public int warehouseCapacity;
    // 可用库容量增减 --
    public int warehouseCapacityChange;
    // 库存小计 --
    public int inventorySubtotal;
    // 库存小计增减 --
    public int inventorySubtotalChange;
    // 有效预报 --仅支持郑商所品种
    public int effectiveForecast;
    // 升贴水 --
    public int premium;
}
```

TransactionRankingInfo - 期货每日成交持仓排名

```
public struct TransactionRankingInfo
{
    // 沽购类型
    public string callOrPut;
    // 交易日期
    public string tradeDate;
    // 会员公司简称
    public string memberName;
    // indicator_number :排名指标数值 --单位：手。视乎所选的排名指标indicator，分别为：
    // 成交量（indicator为'volume'时）
    // 持买单量（indicator为'long'时）
    // 持卖单量（indicator为‘short’时）
    public int indicatorNumber;
    // 排名指标比上交易日增减 --单位：手
    public int indicatorChange;
    // 排名名次 --指标具体排名
    public int ranking;
    // 排名名次比上交易日增减
    public int rankingChange;
}
```

EtfConstituents - ETF基金成分股

```
public struct EtfConstituents
{
    // ETF代码
    public string etf;
    // ETF名称
    public string etfName;
    // 交易日期
    public DateTime tradeDate;
    // 成分股代码
    public string symbol;
    // 股票数量
    public double amount;
    // 现金替代标志
    public string cashSubsType;
    // 固定替代金额
    public double cashSubsSum;
    // 现金替代溢价比例 --单位：%
    public double cashPremiumRate;
}
```

PortfolioStockInfo - 基金资产组合（股票投资组合）

```
public struct PortfolioStockInfo
{
    // 基金代码  --查询资产组合的基金代码
    public string fund;
    // 基金名称
    public string fundName;
    // 公告日期  --在指定时间段[开始时间,结束时间]内的公告日期
    public DateTime pubDate;
    // 报告期 -- 持仓截止日期
    public DateTime periodEnd;
    // 股票代码
    public string symbol;
    // 股票名称
    public string secName;
    // 持仓股数
    public double holdShare;
    // 持仓市值
    public double holdValue;
    // 占净值比例  --单位：%
    public double nvRate;
    // 占总股本比例 --单位：%
    public double ttlShareRate;
    // 占流通股比例 --单位：%s
    public double circShareRate;
}
```

PortfolioBondInfo - 基金资产组合（债券投资组合）

```
public struct PortfolioBondInfo
{
    // 基金代码  --查询资产组合的基金代码
   public  string fund;
    // 基金名称
    public string fundName;
    // 公告日期  --在指定时间段[开始时间,结束时间]内的公告日期
    public DateTime pubDate;
    // 报告期 -- 持仓截止日期
    public DateTime periodEnd;
    // 债券代码
    public string symbol;
    // 债券名称
    public string secName;
    // 持仓数量
    public double holdShare;
    // 持仓市值
    public double holdValue;
    // 占净值比例 --单位：%
    public double nvRate;
}
```

PortfolioFundInfo - 基金资产组合（基金投资组合）

```
public struct PortfolioFundInfo
{
    // 基金代码  --查询资产组合的基金代码
   public string fund;
    // 基金名称
    public string fundName;
    // 公告日期  --在指定时间段[开始时间,结束时间]内的公告日期
    public DateTime pubDate;
    // 报告期 -- 持仓截止日期
    public DateTime periodEnd;
    // 基金代码
    public string symbol;
    // 基金名称
    public string secName;
    // 持有份额
    public double holdShare;
    // 期末市值
    public double holdValue;
    // 占净值比例 --单位：%
    public double nvRate;
}
```

NetValueInfo - 基金净值

```
public struct NetValueInfo
{
    // 基金代码  --查询净值的基金代码
    public string fund;
    // 交易日期
    public DateTime tradeDate;
    // 单位净值  --T日单位净值是每个基金份额截至T日的净值（也是申赎的价格）
    public double unitNv;
    // 累计单位净值  --T日累计净值是指，在基金成立之初投资该基金1元钱，在现金分红方式下，截至
    public double unitNvAccu;
    // 复权单位净值  --T日复权净值是指，在基金成立之初投资该基金1元钱，在分红再投资方式下，截
    public double unitNvAdj;
}
```

FndAdjFactorInfo - 基金复权因子

```
public struct FndAdjFactorInfo
{
    // 交易日期    --最新交易日的日期
    public DateTime tradeDate;
    // 当日后复权因子 --T日后复权因子=T-1日的收盘价/T日前收价
    //public double adjFactorBwd;
    // 当日累计后复权因子  --T日累计后复权因子=T日后复权因子*T-1日累计后复权因子（第一个累计
    public double adjFactorBwdAcc;
    // 当日前复权因子   --T日前复权因子=T日后复权因子/复权基准日后复权因子
    public double adjFactorFwd;
    // 当日累计前复权因子  --T日累计前复权因子=T日后复权因子 T-1日累计前复权因子=T日后复权因
    //public double adjFactorFwdAcc;
}
```

FndDividendInfo - 基金分红信息

```
public struct FndDividendInfo
{
    // 基金代码   --查询分红信息的基金代码
    public string fund;
    // 公告日
    public DateTime pubDate;
    // 方案进度
    public string eventProgress;
    // 派息比例 --10:X，每10份税前分红
    public double dvdRatio;
    // 分配收益基准日
    public DateTime dvdBaseDate;
    // 权益登记日
    public DateTime rtRegDate;
    // 实际除息日
    public DateTime exActDate;
    // 场内除息日
    public DateTime exDvdDate;
    // 场内红利发放日
    public DateTime payDvdDate;
    // 场内红利款账户划出日
    public DateTime transDvdDate;
    // 红利再投资确定日
    public DateTime reinvestCfmDate;
    // 红利再投资份额到账日
    public DateTime riShrArrDate;
    // 红利再投资赎回起始日
    public DateTime riShrRdmDate;
    // 可分配收益 --单位：元
    public double earnDistr;
    // 本期实际红利发放 --单位：元
    public double cashPay;
    // 基准日基金份额净值
    public double baseUnitNv;
}
```

SplitInfo - 基金拆分折算信息

```
public struct SplitInfo
{
    // 基金代码
    public string fund;
    // 公告日
    public DateTime pubDate;
    // 拆分折算类型
    public string splitType;
    // 拆分折算比例
    public double splitRatio;
    // 拆分折算基准日
    public DateTime baseDate;
    // 拆分折算场内除权日
    public DateTime exDate;
    // 基金份额变更登记日
    public DateTime shareChangeRegDate;
    // 基金披露净值拆分折算日
    public DateTime nvSplitPubDate;
    // 权益登记日
    public DateTime rtRegDate;
    // 场内除权日(收盘价)
    public DateTime exDateClose;
}
```

ConversionPrice - 可转债转股价变动信息

```
public struct ConversionPrice
{
    // 公告日期
    public DateTime pubDate;
    // 转股价格生效日期
    public DateTime effectiveDate;
    // 执行日期
    public DateTime executionDate;
    // 转股价格 --单位：元
    public double conversionPrice;
    // 转股比例 --单位：%
    public double conversionRate;
    // 单位：股
    public double conversionVolume;
    // 累计转股金额 --单位：万元，累计转债已经转为股票的金额，累计每次转股金额
    public double conversionAmountTotal;
    // 债券流通余额 --单位：万元
    public double bondFloatAmountRemain;
    // 事件类型  --初始转股价，调整转股价，修正转股价
    public string eventType;
    // 转股价变动原因
    public string changeReason;
}
```

CallInfo - 可转债赎回信息

```
public struct CallInfo
{
    // 公告日 --赎回公告日
    public DateTime pubDate;
    // 赎回日 --发行人行权日（实际），公布的赎回日如遇节假日会顺延为非节假日
    public DateTime callDate;
    // 赎回登记日 --理论登记日，非节假日
    public DateTime recordDate;
    // 赎回资金到账日 --投资者赎回款到账日
    public DateTime cashDate;
    // 赎回类型 --部分赎回，全部赎回
    public string callType;
    // 赎回原因 --1:满足赎回条件，2:强制赎回，3:到期赎回
    public string callReason;
    // 赎回价格 --单位：元/张，每百元面值赎回价格（元），债券面值 加当期应计利息（含税）
    public double callPrice;
    // 赎回金额
    public double callAmount;
    // 是否包含利息  -- 0:不包含，1:包含
    public bool interestIncluded;
}
```

PutInfo - 可转债回售信息

```
public struct PutInfo
{
    // 公告日 --赎回公告日
    public DateTime pubDate;
    // 回售起始日 --投资者行权起始日
    public DateTime putStartDate;
    // 回售截止日 --投资者行权截止日
    public DateTime putEndDate;
    // 回售资金到账日 --投资者回售款到账日
    public DateTime cashDate;
    // 回售原因 --1:满足回售条款，2:满足附加回售条款
    public string putReason;
    // 回售价格 --单位：元/张，每百元面值回售价格（元），债券面值 加当期应计利息（含税）
    public double putPrice;
    // 回售金额
    public double putAmount;
    // 是否包含利息  -- 0:不包含，1:包含
    public bool interestIncluded;
}
```

AmountChange - 可转债剩余规模变动

```
public struct AmountChange
{
    // 公告日
    public DateTime pubDate;
    // 变动类型 --首发、增发、转股 、赎回、回售(注销)、到期
    public string changeType;
    // 变动日期
    public DateTime changeDate;
    // 本次变动金额 --单位：万元
    public double changeAmount;
    // 剩余金额 --变动后金额，单位：万元
    public double remainAmount;
}
```

OptContractInfo - 期权标的基础信息

```
public struct OptContractInfo
{
    // 交易代码
    public string productCode;
    // 合约标的物名称
    public string underlying;
    // 合约乘数
    public int multiplier;
    // 交易单位
    public string tradeUnit;
    // 报价单位
    public string priceUnit;
    // 价格最小变动单位
    public double priceTick;
    // 合约月份
    public string deliveryMonth;
    // 交易时间
    public string tradeTime;
    // 涨跌幅限制
    public string priceRange;
    // 行权方式
    public string optionType;
    // 行权价格
    public string exercisePriceRule;
    // 最后交易日
    public string lastTradeDate;
    // 交割方式
    public string deliveryMethod;
    // 交易所名称
    public string exchangeName;
    // 交易所代码
    public string exchange;
}
```

RiskValueInfo - 期权波动率

```
public struct RiskValueInfo
{
    // 交易日期
    public string tradeDate;
    // Delta
    public double delta;
    // Theta
    public double theta;
    // Gamma
    public double gamma;
    // Vega
    public double vega;
    // Rho
    public double rho;
    // 隐含波动率
    public double iv;
}
```

GetSymbolsByInAtOutReq - 期权档位合约信息

```
public struct GetSymbolsByInAtOutReq
{
    // 合约标的物
    // 参数用法说明:
    // 必填，标的物symbol，全部大写，不指定具体到期月份，
    // 标的物为商品期货的，可参考主力合约代码，如：'CZCE.CF'
    // 标的物为股指的，可填：'CFFEX.IO'，'CFFEX.MO'
    // 标的物为ETF的，可填ETF的symbol，如：'SHSE.510050'
    public string underlyingSymbol;
    // 沽购类型
    // 参数用法说明:
    // 沽购类型，
    // 认购期权（看涨期权，买权）：'C'
    // 认沽期权（看跌期权，卖权）：'P'
    // 默认None表示不区分沽购类型，同时包括认购和认沽
    public string callOrPut;
    // 合约月份
    // 参数用法说明:
    // 合约月份，按到期月份从近至远从小到大排序，支持最近交割的4个月份
    // 可选1，2，3，4
    // 默认None为全部月份
    public int executeMonth;
    // 档位计数
    // 参数用法说明:
    // 档位计数，实值档位为正，虚值档位为负，平值为0，
    // 默认None时为所有档位（含平值）
    public int inAtOut;
    // 交易日期
    // 参数用法说明:
    // 查询时间，本地时间，格式为：YYYY-MM-DD
    // 为空时，表示当前日期
    public string tradeDate;
    // 标的物价格
    // 参数用法说明:
    // 标的物价格，只能输入自定义具体价格
    public double s;
    // 标的物价格类型
    // 参数用法说明:
    // 标的物价格s为None时，此参数才生效。'pre_close'为昨收价（默认），'open'为今开价，'las
    // 收盘价，trade_date包含分钟就取该1分钟bar的close）默认None为昨收价。
    public string sType;
    // 调整合约
    // 参数用法说明:
    // 表示是否过滤除权后的调整合约，'M'表示不返回调整合约，只返回标准合约（默认）'A'表示只返
    // 默认None为只返回标准合约。
    public string adjustFlag;
}
```

交易类

Account - 账户结构

```
public class Account
{
    public string accountId;              //账号ID
    public string accountName;            //账户登录名
    public string title;                  //账号名称
    public string intro;                  //账号描述
    public string comment;                //账号备注
};
```

AccountStatus - 账户状态结构

```
public class AccountStatus
{
    public string accountId;              //账号ID
    public string accountName;            //账户登录名
    public int state;                     //账户状态
    public int errorCode;                 //错误码
    public string errorMsg;               //错误信息
};
```

Order - 委托结构

```
public class Order
{
    public string strategyId;                   //策略ID
    public string accountId;                    //账号ID
    public string accountName;                  //账户登录名
    public string clOrdId;                      //委托客户端ID
    public string orderId;                      //委托柜台ID
    public string exOrdId;                      //委托交易所ID
    public string symbol;                       //symbol
    public int side;                            //买卖方向，取值参考enum OrderSide
    public int positionEffect;                  //开平标志，取值参考enum PositionEffe
    public int positionSide;                    //持仓方向，取值参考enum PositionSide

    public int orderType;                       //委托类型，取值参考enum OrderType
    public int orderDuration;                   //委托时间属性，取值参考enum OrderDura
    public int orderQualifier;                  //委托成交属性，取值参考enum OrderQual
    public int orderSrc;                        //委托来源，取值参考enum OrderSrc

    public int status;                          //委托状态，取值参考enum OrderStatus
    public int ordRejReason;                    //委托拒绝原因，取值参考enum OrderReje
    public string ordRejReasonDetail;           //委托拒绝原因描述

    public double price;                        //委托价格
    public double stopPrice;                    //委托止损/止盈触发价格

    public int orderStyle;                      //委托风格，取值参考 enum OrderStyle
    public Int64 volume;                        //委托量
    public double value;                        //委托额
    public double percent;                      //委托百分比
    public Int64 targetVolume;                  //委托目标量
    public double targetValue;                  //委托目标额
    public double targetPercent;                //委托目标百分比

    public Int64 filledVolume;                  //已成量
    public double filledVwap;                   //已成均价
    public double filledAmount;                 //已成金额
    public double filledCommission;             //已成手续费

    public DateTime createdAt;                  //委托创建时间
    public DateTime updatedAt;                  //委托更新时间
};
```

ExecRpt - 回报结构

```
public class ExecRpt
{
    public string strategyId;                   //策略ID
    public string accountId;                    //账号ID
    public string accountName;                  //账户登录名
    public string clOrdId;                      //委托客户端ID
    public string orderId;                      //委托柜台ID
    public string execId;                       //委托回报ID
    public string symbol;                       //symbol

    public int positionEffect;                  //开平标志，取值参考enum PositionEffect
    public int side;                            //买卖方向，取值参考enum OrderSide
    public int ordRejReason;                    //委托拒绝原因，取值参考enum OrderRejectR
    public string ordRejReasonDetail;           //委托拒绝原因描述
    public int execType;                        //执行回报类型，取值参考enum ExecType

    public double price;                        //委托成交价格
    public Int64 volume;                        //委托成交量
    public double amount;                       //委托成交金额
    public double commission;                   //委托成交手续费
    public double cost;                         //委托成交成本金额
    public DateTime createdAt;                  //回报创建时间

};
```

Cash - 资金结构

```
public class Cash
{
    public string accountId;                    //账号ID
    public string accountName;                  //账户登录名


    public int currency;                        //币种

    public double nav;                          //净值(cum_inout + cum_pnl + fpnl - cum
    public double pnl;                          //净收益(nav-cum_inout)
    public double fpnl;                         //浮动盈亏(sum(each position fpnl))
    public double frozen;                       //持仓占用资金
    public double orderFrozen;                  //挂单冻结资金
    public double available;                    //可用资金

    public double balance;                      //资金余额

    public double cumInout;                     //累计出入金
    public double cumTrade;                     //累计交易额
    public double cumPnl;                       //累计平仓收益(没扣除手续费)
    public double cumCommission;                //累计手续费

    public double lastTrade;                    //上一次交易额
    public double lastPnl;                      //上一次收益
    public double lastCommission;               //上一次手续费
    public double lastInout;                    //上一次出入金
    public int changeReason;                    //资金变更原因，取值参考enum CashPositior
    public string changeEventId;                //触发资金变更事件的ID

    public DateTime createdAt;                  //资金初始时间
    public DateTime updatedAt;                  //资金变更时间

};
```

Position - 持仓结构

```
public class Position
{
    public string accountId;                //账号ID
    public string accountName;              //账户登录名

    public string symbol;                   //symbol
    public int side;                        //持仓方向，取值参考enum PositionSide
    public Int64 volume;                    //总持仓量; 昨持仓量(volume-volume_today)
    public Int64 volumeToday;               //今日持仓量
    public double vwap;                     //持仓均价
    public double amount;                   //持仓额(volume*vwap*multiplier)

    public double price;                    //当前行情价格
    public double fpnl;                     //持仓浮动盈亏((price-vwap)*volume*multipli
    public double cost;                     //持仓成本(vwap*volume*multiplier*margin_ra
    public Int64 orderFrozen;               //挂单冻结仓位
    public Int64 orderFrozenToday;          //挂单冻结今仓仓位
    public Int64 available;                 //非挂单冻结仓位(volume-order_frozen); 可平
    public Int64 availableToday;           //非挂单冻结今仓位(volume_today-order_frozen
    public Int64 availableNow;             //当前可用仓位

    public double lastPrice;                //上一次成交价
    public Int64 lastVolume;                //上一次成交量
    public Int64 lastInout;                 //上一次出入持仓量
    public int changeReason;                //仓位变更原因，取值参考enum CashPositionCha
    public string changeEventId;            //触发资金变更事件的ID

    public int hasDividend;                 //持仓区间有分红配送
    public DateTime createdAt;              //建仓时间（实盘不支持）
    public DateTime updatedAt;              //仓位变更时间（实盘不支持）

};
```

Indicator - 绩效指标结构

```
public class Indicator
{
    public string accountId;               //账号ID
    public double pnlRatio;                //累计收益率(pnl/cum_inout)
    public double pnlRatioAnnual;          //年化收益率
    public double sharpRatio;              //夏普比率
    public double maxDrawdown;             //最大回撤
    public double riskRatio;               //风险比率
    public int openCount;                  //开仓次数
    public int closeCount;                 //平仓次数
    public int winCount;                   //盈利次数
    public int loseCount;                  //亏损次数
    public double winRatio;                //胜率

    public DateTime createdAt;             //指标创建时间
    public DateTime updatedAt;             //指标变更时间
};
```

Parameter - 动态参数结构

```
public class Parameter
{
    public string key;                  //参数键
    public double value;                //参数值
    public double min;                  //可设置的最小值
    public double max;                  //可设置的最大值
    public string name;                 //参数名
    public string intro;                //参数说明
    public string group;                //组名
    public bool readonlyFlag;           //是否只读
};
```

枚举常量

OrderStatus - 委托状态

```
enum OrderStatus
{
    OrderStatus_Unknown = 0,
    OrderStatus_New = 1,                    //已报
    OrderStatus_PartiallyFilled = 2,        //部成
    OrderStatus_Filled = 3,                 //已成
    OrderStatus_Canceled = 5,               //已撤
    OrderStatus_Rejected = 8,               //已拒绝
    OrderStatus_PendingNew = 10,            //待报
    OrderStatus_Expired = 12,               //已过期

};
```

OrderSide - 委托方向

```
enum OrderSide
{
    OrderSide_Unknown = 0,
    OrderSide_Buy = 1,          //买入
    OrderSide_Sell = 2,         //卖出
};
```

sectype---标的类别

```
enum SecType
{
SEC_TYPE_STOCK = 1,          //股票
SEC_TYPE_FUND = 2,           //基金
SEC_TYPE_INDEX = 3,          //指数
SEC_TYPE_FUTURE = 4,         //期货
SEC_TYPE_OPTION = 5,         //期权
SEC_TYPE_CONFUTURE = 10      //虚拟合约
}
```

OrderType - 委托类型

```
enum OrderType
{
    OrderType_Unknown = 0,
    OrderType_Limit = 1,          //限价委托
    OrderType_Market = 2,         //市价委托
};
```

ExecType - 执行回报类型

```
enum ExecType
{
    ExecType_Unknown = 0,
    ExecType_Trade = 15,                    //成交
    ExecType_CancelRejected = 19,           //撤单被拒绝
};
```

PositionEffect - 开平仓类型

```
enum PositionEffect
{
    PositionEffect_Unknown = 0,
    PositionEffect_Open = 1,                //开仓
    PositionEffect_Close = 2,               //平仓,具体语义取决于对应的交易所
    PositionEffect_CloseToday = 3,          //平今仓
    PositionEffect_CloseYesterday = 4,      //平昨仓
};
```

PositionSide - 持仓方向

```
enum PositionSide
{
    PositionSide_Unknown = 0,
    PositionSide_Long = 1,     //多方向
    PositionSide_Short = 2,    //空方向
};
```

OrderRejectReason - 订单拒绝原因

```
enum OrderRejectReason
{
    OrderRejectReason_Unknown = 0,                          //未知原因
    OrderRejectReason_RiskRuleCheckFailed = 1,              //不符合风控规则
    OrderRejectReason_NoEnoughCash = 2,                     //资金不足
    OrderRejectReason_NoEnoughPosition = 3,                 //仓位不足
    OrderRejectReason_IllegalAccountId = 4,                 //非法账户ID
    OrderRejectReason_IllegalStrategyId = 5,                //非法策略ID
    OrderRejectReason_IllegalSymbol = 6,                    //非法交易代码
    OrderRejectReason_IllegalVolume = 7,                    //非法委托量
    OrderRejectReason_IllegalPrice = 8,                     //非法委托价
    OrderRejectReason_AccountDisabled = 10,                 //交易账号被禁止交易
    OrderRejectReason_AccountDisconnected = 11,             //交易账号未连接
    OrderRejectReason_AccountLoggedout = 12,                //交易账号未登录
    OrderRejectReason_NotInTradingSession = 13,             //非交易时段
    OrderRejectReason_OrderTypeNotSupported = 14,           //委托类型不支持
    OrderRejectReason_Throttle = 15,                        //流控限制
    OrderRejectReason_SymbolSusppended = 16,                //交易代码停牌
    OrderRejectReason_Internal = 999,                       //内部错误

    CancelOrderRejectReason_OrderFinalized = 101,           //委托已完成
    CancelOrderRejectReason_UnknownOrder = 102,             //未知委托
    CancelOrderRejectReason_BrokerOption = 103,             //柜台设置
    CancelOrderRejectReason_AlreadyInPendingCancel = 104,   //委托撤销中
};
```

CashPositionChangeReason - 仓位变更原因

```
enum CashPositionChangeReason
{
    CashPositionChangeReason_Unknown = 0,
    CashPositionChangeReason_Trade = 1,       //交易
    CashPositionChangeReason_Inout = 2,       //出入金/出入持仓
    CashPositionChangeReason_Dividend = 3,    //分红送股
};
```

AccountState - 交易账户状态

```
enum AccountState
{
    State_UNKNOWN = 0,        //未知
    State_CONNECTING = 1,     //连接中
    State_CONNECTED = 2,      //已连接
    State_LOGGEDIN = 3,       //已登录
    State_DISCONNECTING = 4, //断开中
    State_DISCONNECTED = 5,  //已断开
    State_ERROR = 6           //错误
};
```

StrategyMode - 策略模式

```
public enum StrategyMode
{
    MODE_UNDEF = 0,          //未定义， 策略不会运行
    MODE_LIVE = 1,           //实盘与仿真模式
    MODE_BACKTEST = 2        //回测模式
};
```

Adjust - 复权方式

```
public enum Adjust
{
    ADJUST_NONE = 0,        //(不复权)
    ADJUST_PREV = 1,        //(前复权)
    ADJUST_POST = 2         //(后复权)
}
```

错误码

错误码	描述
0	成功
1000	错误或无效的token
1001	无法连接到终端服务
1010	无法获取东方财富量化服务器地址列表
1011	消息包解析错误
1012	网络消息包解析错误
1013	交易服务调用错误
1014	历史行情服务调用错误
1015	策略服务调用错误
1016	动态参数调用错误
1017	基本面数据服务调用错误
1018	回测服务调用错误
1019	交易网关服务调用错误
1020	无效的ACCOUNT_ID
1021	非法日期格式
1100	交易消息服务连接失败
1101	交易消息服务断开
1200	实时行情服务连接失败
1201	实时行情服务连接断开
1300	初始化回测失败，可能是终端未启动或无法连接到终端
1301	回测时间区间错误
1302	回测读取缓存数据错误
1303	回测写入缓存数据错误