Tac FinTech

TacFeed 行情服务
API 使用手册

V4.0

上海广策信息技术有限公司

2022 年 11 月 08 日

声明

商标声明

注意

本文档内容仅作指导使用，本文档内容会根据产品版本升级或其他原因进行不定期更新，除另有约定外，本文档所载内容不构成任何担保。

用户购买的产品受与广策签订的商业合同条款约束，本文档内容所陈述的产品性能，服务也许全部或部分与用户所购产品不符，除合同另有约定，广策对本文档所载内容不做任何明示或暗示的保证。

上海广策信息技术有限公司
地址：上海市浦东新区达尔文路88号半岛科技园10号楼2层
电话：021-68547307
传真：021-68547307
邮箱：support@tacfintech.cn

文档修订历史

版本	日期	描述	修改人
4.0.15	2022.11.08	增加支持能源行情	潘鹰
4.0.0	2021.07.01	文档重新整理、完善	周鹏辉

目 录

第一章 系统介绍

1.1.系统简介

1.1.1. 系统概述

本系统是基于硬件加速技术，自主研发的纳秒级行情加速系统，为极速交易提供高速、准确、连续的行情数据，实现第一时间获取市场信息、提升交易策略报单成交率的作用。

1.1.2. 运行平台

目前发布了以下操作系统平台的版本：

- Linux x64：CentOS 7

(.so 文件在 CentOS 7.2/gcc 4.8.5 下完成编译。)

注意:必须使用操作系统中gcc编译器对应tacfeed API版本才可正常使用， 默认给的为gcc4.8.5版本的API，如若编译器为高版本，请找客服人员索取即可，目前支持的gcc编译器版本如下(如有其他特需版本可找客服人员协商定制)：
gcc4, gcc6, gcc8, gcc9

1.2.支持范围

Tacfeed API 目前支持接收上期所（SHFE）Level 1 一档行情、Level 2 五档行情，MBL 10 档行情和大商所（DCE）Level 1 一档行情数据。API 需要配合相应的 TacFeed 硬件行情源使用。

第二章 API 使用说明

2.1.行情接收方法

TacFeed API 通过回调函数的方式将行情数据传递给用户，用户在使用 API 时需要自定义回调函数，并通过 TacFeedSubscribe 函数将回调函数传递给 API，当 API 收到行情更新后会调用用户传递进来的回调函数

2.2.行情数据结构说明

2.2.1. Level 1 行情数据结构体

```
typedef struct Level1QuoteDataT{
    Char        InstrumentID[31];     行情的合约代码
    Char        ActionDay[9];         行情更新日期
    Char        UpdateTime[12];       行情更新时间，格式为 hh:mm:ss
    int         UpdateMillisec;       行情更新时间的毫秒值
    int         Volume;               成交量
    double      LastPrice;            成交价
    double      Turnover;             成交金额
    double      OpenInterest;         开仓量
    double      BidPrice1;            申买价一
    int         BidVolume1;           申买量一
    double      AskPrice1;            申卖价一
    int         AskVolume1;           申卖量一
} Level1QuoteDataT;
```

2.2.2. Level 2 行情数据结构体

```
typedef struct Level2QuoteDataT{
    Char        InstrumentID[31];   行情的合约代码
    Char        ActionDay[9];       行情更新日期
    Char        UpdateTime[12];     行情更新时间，格式为 hh:mm:ss
    int         UpdateMillisec;     行情更新时间的毫秒值
    int         Volume;             成交量
    double      LastPrice;          成交价
    double      Turnover;           成交金额
    double      OpenInterest;       开仓量
    double      BidPrice1;          申买价一
    int         BidVolume1;         申买量一
    double      AskPrice1;          申卖价一
    int         AskVolume1;         申卖量一
    double      BidPrice2;          申买价二
    int         BidVolume2;         申买量二
    double      AskPrice2;          申卖价二
    int         AskVolume2;         申卖量二
    double      BidPrice3;          申买价三
    int         BidVolume3;         申买量三
    double      AskPrice3;          申卖价三
    int         AskVolume3;         申卖量三
    double      BidPrice4;          申买价四
    int         BidVolume4;         申买量四
    double      AskPrice4;          申卖价四
    int         AskVolume4;         申卖量四
    double      BidPrice5;          申买价五
    int         BidVolume5;         申买量五
    double      AskPrice5;          申卖价五
    int         AskVolume5;         申卖量五
} Level2QuoteDataT;
```

2.2.3. MBL 行情数据结构体

```
typedef struct MBLQuoteDataT{
    Char        InstrumentID[8];    行情的合约代码
    double      AskPrice1;          卖一挂单价
    int         AskVolume1;         卖一挂单量
    double      AskPrice2;          卖二挂单价
    int         AskVolume2;         卖二挂单量
    double      AskPrice3;          卖三挂单价
    int         AskVolume3;         卖三挂单量
    double      AskPrice4;          卖四挂单价
    int         AskVolume4;         卖四挂单量
    double      AskPrice5;          卖五挂单价
    int         AskVolume5;         卖五挂单量
    double      AskPrice6;          卖六挂单价
    int         AskVolume6;         卖六挂单量
    double      AskPrice7;          卖七挂单价
    int         AskVolume7;         卖七挂单量
    double      AskPrice8;          卖八挂单价
    int         AskVolume8;         卖八挂单量
    double      AskPrice9;          卖九挂单价
    int         AskVolume9;         卖九挂单量
    double      AskPrice10;         卖十挂单价
    int         AskVolume10;        卖十挂单量
    double      BidPrice1;          买一挂单价
    int         BidVolume1;         买一挂单量
    double      BidPrice2;          买二挂单价
    int         BidVolume2;         买二挂单量
    double      BidPrice3;          买三挂单价
    int         BidVolume3;         买三挂单量
    double      BidPrice4;          买四挂单价
    int         BidVolume4;         买四挂单量
    double      BidPrice5;          买五挂单价
    int         BidVolume5;         买五挂单量
    double      BidPrice6;          买六挂单价
    int         BidVolume6;         买六挂单量
    double      BidPrice7;          买七挂单价
    int         BidVolume7;         买七挂单量
    double      BidPrice8;          买八挂单价
    int         BidVolume8;         买八挂单量
    double      BidPrice9;          买九挂单价
    int         BidVolume9;         买九挂单量
    double      BidPrice10;         买十挂单价
    int         BidVolume10;        买十挂单量
} MBLQuoteDataT;
```

2.3.函数与参数说明

2.3.1. 回调函数原型

- Level 1 行情数据回调函数原型
  typedef std::function<void (Level1QuoteDataT *pData)> Level1Callback;

- Level 2 行情数据回调函数原型
  typedef std::function<void (Level2QuoteDataT *pData)> Level2Callback;

- MBL 行情数据回调函数原型
  typedef std::function<void (MBLQuoteDataT *pData)> MBLCallback;

2.3.2. 初始化参数

- 初始化参数结构体

typedef struct TacFeedInitParam{

const char* LocalIP;	(必填)Tacfeed 行情接收本地ip 地址 注意：填写本地接收行情的端口的 IP 地址，不能填写 127.0.0.1
const char* LoginServerIP;	(必填)Tacfeed 行情登录服务ip 地址

```
    uint16_t LoginServerPort;       (必填)Tacfeed 行情登录服务端口
    const char* UserName;           (必填)行情用户登录用户名
    const char* Password;           (必填)行情用户登录密码
    Level1Callback L1Callback; (选填)用户自定义 Level 1 行情处理的回调函数
    Level2Callback L2Callback; (选填)用户自定义 Level 2 行情处理的回调函数
    MBLCallback MblCallback; (选填)用户自定义 MBL 行情处理的回调函数
    TacFeedApiSelectMode            (选填)行情接收模式。默认为自动，根据用户网
ApiSelectMode;                      卡自动选择接收模式。
    TacFeedFlag Flag                (选填)API 配置标志
} TacFeedInitParam;
```

■ 行情接收模式

```
enum TacFeedApiSelectMode {
    ApiSelectAuto = 0,              自动选择接收方式
    ApiSelectLinux = 1,             使用 Linux socket 接收行情
    ApiSelectTcpDirect = 2,         使用 TCP Direct 接收行情
    ApiSelectTacnic = 3,            使用 Tacnic 接收行情
};
```

■ 配置标志

```
typedef struct TacFeedFlag {
    Unsigned int    UseHwFilter : 1;    使用硬件过滤器（仅适用于 tacnic 网卡）
    Unsigned int    UsePcieBus : 1;     使用 PCIE 总线接收（仅适用于 tacfeed 本
                                        地版）
    Unsigned int    Reserve : 30;       保留字段
} TacFeedFlag;
```

备注：配置标志目前只适用于 Tacnic 网卡

2.3.3. TacFeedSubscribe 函数

```
// 订阅 TacFeed 行情
int TacFeedSubscribe( const TacFeedInitParam* init_param);
```

参数：

init_param　初始化参数

返回值：

错误码，成功则返回 0

2.3.4. TacFeedGetApiVersion 函数

```
// 获取API 版本号
char* TacFeedGetApiVersion();
```

返回值：

TacFeed API 版本号

2.3.5. TacFeedSetCpuAffinity 函数

```
// 设定TacFeed 工作线程CPU 亲和性
int TacFeedSetCpuAffinity(int cpu);
```

参数：

cpu　绑定的cpu 编号

返回值：

错误码，成功则返回 0

备注: TacFeedSetCpuAffinity 函数必须在 TacFeedSubscribe 之前调用

2.3.6. TacFeedRelease 函数

```
// 释放工作线程和资源
int TacFeedRelease();
```

返回值：

错误码，成功则返回 0