SPSYSTEM

SP Native API R8.6839

交易接口说明书

2015/09/29

内容目錄

1.介绍

SPTrader交易系统API接口是一个基于C++的类库,通过扩展类库提供的接口来实现相关交易功能,其中包括订单处理,持仓查询,成交记录查询,价格订阅,市场成交记录查询,户口资料查询等.该类库下面包含以下文件:

文件名	版本	日期	备注
spapidll32.dll	V1.0, R8.6839	20150929	动态连接
spapidll.h	V1.0, R8.6839	20150929	接口头文件

注：登录已经更改为两种模式 AE 与普通账户

1. AE 模式下:登录端口为 8081,需要 AE 开通 AE 模式,没开通登录时会自动登出.此模式下用户拿到的与回调的都是 AE 账户下的订单与成交数据.

2.普通账户模式下:登录端口为 8080,此模式下用户拿到的与回调的都是此账户的订单与成交数据.

取订单数据方法：

SPAPI_GetOrderCount,SPAPI_GetOrder,SPAPI_GetOrderByOrderNo, SPAPI_LoadOrderReport

订单回调方法：SPAPI_RegisterOrderReport

取成交数据方法：

SPAPI_GetTradeCount,SPAPI_GetTrade,SPAPI_GetTradeByTradeNo,SPAPI_LoadTradeReport

成交回调方法：SPAPI_RegisterTradeReport

2. 开发接口

1.1 SPAPI_Initialize 方法

该方法用于API初始化。

函数原型:

```
int  SPAPI_Initialize()
```

返回值:

0, 代表成功.

1.2 SPAPI_Uninitialize 方法

该方法用于释放API.

函数原型:

```
int  SPAPI_Uninitialize()
```

返回值:

0, 代表成功.

1.3 SPAPI_Poll 方法

该方法用来建立一个轮询.
(建议用于单线程编程.)

函数原型:

```
void  SPAPI_Poll()
```

1.4 SPAPI_SetBackgroundPoll 方法

该方法用来建立一个后台轮询.
(建议用于多线程编程.)

函数原型:

```
void  SPAPI_SetBackgroundpoll(bool  onoff)
```

参数:

onoff: 当它为true或1时建立通信.
flase或0时关闭通信.

1.5 SPAPI_SetLoginInfo 方法

设定登录信息.

函数原型:

```
void  SPAPI_SetLoginInfo(char *host, int port, char *license,
char *app_id, char *user_id, char *password)
```

参数:

host: 服务器地址.
port: 连接端口.
license: 许可加密字符串.
app_id: 应用编号.
user_id: 用户名.
password: 用户密码.

1.6 SPAPI_Login 方法

发送登录请求.

函数原型:

```
int  SPAPI_Login();
```

返回值:

0, 代表成功发送登录请求.
-9,表示已经有一个程序在访问DLL.

1.7 SPAPI_Logout 方法

发送登出请求.

函数原型:

```
int  SPAPI_Logout()
```

返回值:

0, 代表成功发送登出请求.

1.8 SPAPI_ChangePassword 方法

修改用户密码.

函数原型:

```
int  SPAPI_ChangePassword(char *old_psw, char *new_psw)
```

参数:

old_psw: 原始密码.
new_psw: 新的密码.

返回值:

0, 代表成功发送登出请求.

1.9 SPAPI_GetLoginStatus 方法

查询交易连接状态与行情连接状态.

函数原型:

```
int  SPAPI_GetLoginStatus(short  host_id)
```

参数:

host_id:

80,81, 表示交易连接.(用户登录状态)
83, 表示一般价格连接.
87, 表示详细价格深度连接.
88, 表示一般资讯连接.

返回值:

0,断开连接.	1,连接中.	2,已连接.	3,连接错误.
4,登录中.	5,已登录.	6,登出中.	7,已登出.
8,登录失败	9,连接丢失	10,断开中	11,主机请求

注意:

80,81参考:SPAPI_RegisterLoginStatusUpdate
83,87,88参考:SPAPI_RegisterPServerLinkStatusUpdate

1.10 SPAPI_AddOrder 方法

该方法用来添加订单.

函数原型:

```
int  SPAPI_AddOrder(SPApiOrder *order)
```

参数:

***order,** 表示一个指针订单结构.

订单结构:

```
typedef  struct
{
    double Price;               //价格
    double StopLevel;           //止损价格
    double UpLevel;             //上限水平
    double UpPrice;             //上限价格
    double DownLevel;           //下限水平
    double DownPrice;           //下限价格
    bigint ExtOrderNo;          //外部指示
    long IntOrderNo;            //用户订单编号
    long Qty;                   //剩下数量
    long TradedQty;             //已成交数量
    long TotalQty;              //订单全部数量
    long ValidTime;             //有效时间
```

```
    long SchedTime;             //预订发送时间
    long TimeStamp;             //服务器接收订单时间
    u_long OrderOptions;        //0=默认,1=T+1
    STR16 AccNo;                //用户帐号
    STR16 ProdCode;             //合约代号
    STR16 Initiator;            //下单用户
    STR16 Ref;                  //参考
    STR16 Ref2;                 //参考
    STR16 GatewayCode;          //网关
    STR40 ClOrderId;            //用户自定义参考
    char BuySell;               //买卖方向
    char StopType;              //止损类型
    char OpenClose;             //开平仓
    tinyint CondType;           //订单条件类型
    tinyint OrderType;          //订单类型
    tinyint ValidType;          //订单有效类型
    tinyint Status;             //状态
    tinyint DecInPrice;         //合约小数位
} SPApiOrder;
```

返回值:

0， 表示成功.

1.11 SPAPI_ChangeOrder 方法

该方法用来修改工作中的订单.

函数原型:

```
int  SPAPI_ChangeOrder (SPApiOrder *order, double org_price,
long org_qty);
```

参数:

***order,** 表示一个指针订单结构.

org_price, 之前订单的价格,(注意:修改订单价格时,要将修改后的价格给结构体中Price,而之前的价格要给参数org_price.)

org_qty, 之前订单中的数量.

返回值:

0，表示成功.

1.12 SPAPI_GetOrderByOrderNo 方法

根据订单编号查询该编号工作中的订单信息.

函数原型:

```
int  SPAPI_GetOrderByOrderNo(char *acc_no, long
```

int_order_no, SPApiOrder *order)

参数:

*acc_no, 账户.(client Mode:返回，AE Mode:输入)

int_order_no, 输入要查询的订单编号.(输入)

*order, 表示返回的订单信息.(返回)

返回值:

0，表示请求成功.

1.13 SPAPI_GetOrderCount 方法

查询已加载的工作中订单数量.

函数原型:

int SPAPI_GetOrderCount()

返回值:

工作中订单数量.

1.14 SPAPI_GetOrder 方法

该方法用来获取工作中的订单信息.

函数原型:

int SPAPI_GetOrder(int idx, SPApiOrder *order)

参数:

idx, 表示订单列表中的序号.(序号是从0开始.)

*order, 表示返回的订单信息.

返回值:

0, 表示请求成功.

1.15 SPAPI_DeleteOrder 方法

该方法用来删除订单.

函数原型:

int SPAPI_DeleteOrder(SPApiOrder *order)

参数:

*order,表示一个指针订单结构.

删除订单结构,删除订单时只需在结构中输入用户订单编号与合约代号.

返回值:

0, 表示成功.

1.16 SPAPI_ActivateOrder 方法

该方法用来设置有效订单.

函数原型:

```
int SPAPI_SetOrderInactive(SPApiOrder *order);
```

返回值:

0，表示成功.

1.17 SPAPI_InactivateOrder 方法

该方法用来设置无效订单.

函数原型:

```
int  SPAPI_InactivateOrder()
```

返回值:

0，表示成功.

1.18 SPAPI_GetPosCount 方法

该方法用来获取持仓数量.

函数原型:

```
int SPAPI_GetPosCount()
```

返回值:

持仓数量.

1.19 SPAPI_GetPos 方法

该方法用来通过序号获取持仓信息.

函数原型:

```
int  SPAPI_GetPos(int idx, SPApiPos *pos)
```

参数:

idx, 表示持仓列表中的序号.(序号是从0开始.)

***Pos,** 表示返回的持仓信息.

持仓结构体:

```
typedef  struct
{
    long Qty;                   //上日仓位
    long DepQty;                //存储仓位
    long LongQty;               //今日长仓
    long ShortQty;              //今日短仓
    double TotalAmt;            //上日成交
    double DepTotalAmt;         //上日持仓总数(数量*价格)
```

```
    double LongTotalAmt;      //今日长仓总数(数量*价格)
    double ShortTotalAmt;     //今日短仓总数(数量*价格)
    double PLBaseCcy;         //盈亏(基本货币)
    double PL;                //盈亏
    double ExchangeRate;      //汇率
    STR16 AccNo;              //用户帐号
    STR16 ProdCode;           //合约代码
    char LongShort;           //上日持仓买卖方向
    tinyint DecInPrice;       //小数点
    } SPApiPos;
```

返回值:

0,表示请求成功.

1.20 SPAPI_GetPosByProduct 方法

该方法通过合约代号查询该合约的持仓信息.

函数原型:

```
int  SPAPI_GetPosByProduct(char *prod_code, SPApiPos *pos)
```

参数:

prod_code, 输入要查询的合约代号.

***pos**, 返回的持仓信息.

返回值:

0，表示请求成功.

1.21 SPAPI_GetTradeCount 方法

查询已成交数量.

函数原型:

```
int  SPAPI_GetTradeCount()
```

返回值:

成交数量.

1.22 SPAPI_GetTrade 方法

该方法用来通过序号获取已成交信息.

函数原型:

```
int  SPAPI_GetTrade(int idx, SPApiTrade *trade);
```

参数:

idx, 表示成交列表中的序号.(序号是从0开始.)
*trade, 返回的成交信息.
成交结构:

```
typedef  struct
{
        double RecNo;                   //成交记录
        double Price;                   //成交价格
        double AvgPrice;                //成交价格
        bigint TradeNo;                 //成交编号
        bigint ExtOrderNo;              //外部指示
        long IntOrderNo;                //用户订单编号
        long Qty;                       //成交数量
        long TradeDate;                 //成交日期
        long TradeTime;                 //成交时间
        STR16 AccNo;                    //用户帐号
        STR16 ProdCode;                 //合约代码
        STR16 Initiator;                //下单用户
        STR16 Ref;                      //参考
        STR16 Ref2;                     //参考
        STR16 GatewayCode;              //网关
        STR40 ClOrderId;                //用户自定义参考
        char BuySell;                   //买卖方向
        char OpenClose;                 //开平仓
        tinyint Status;                 //状态
        tinyint DecInPrice;             //小数位
} SPApiTrade;
```

返回值:
0, 表示请求成功.

1.23 SPAPI_GetTradeByTradeNo 方法

该方法通过订单编号与成交编号查询指定成交记录信息.

函数原型:

```
int  SPAPI_GetTradeByTradeNo(long int_order_no, bigint trade_no, SPApiTrade *trade);
```

参数:
int_order_no, 订单编号.
trade_no, 成交编号.
*trade, 返回的成交记录信息结构体.

返回值:
0,表示请求成功.

1.24 SPAPI_SubscribePrice 方法

该方法用来订阅/取消市场数据.

函数原型:

```
int  SPAPI_SubscribePrice(char *prod_code, int mode);
```

参数:

prod_code, 要订阅的合约代码.

mode, 订阅的类型.

0:取消市场数据.

1:订阅市场数据.

返回值:

0, 表示请求成功.

1.25 SPAPI_GetPriceCount 方法

该方法用来获取已订阅的合约数量.

函数原型:

```
int  SPAPI_GetPriceCount()
```

返回值:

已订阅合约的数量.

1.26 SPAPI_GetPrice 方法

该方法用来通过序号获取价格资讯信息.

函数原型:

```
int  SPAPI_GetPrice(int idx, SPApiPrice *price);
```

参数:

idx, 表示价格资讯列表中的序号.(序号是从0开始.)

*price, 价格资讯信息.

价格资讯结构体:

```
typedef  struct
{
double Bid[SP_MAX_DEPTH];       //买入价
long BidQty[SP_MAX_DEPTH];      //买入量
long BidTicket[SP_MAX_DEPTH];   //买指令数量
double Ask[SP_MAX_DEPTH];       //卖出价
long AskQty[SP_MAX_DEPTH];      //卖出量
long AskTicket[SP_MAX_DEPTH];   //卖指令数量
double Last[SP_MAX_LAST];       //成交价
long LastQty[SP_MAX_LAST];      //成交数量
long LastTime[SP_MAX_LAST];     //成交时间
```

```
double Equil;                    //平衡价
double Open;                     //开盘价
double High;                     //最高价
double Low;                      //最低价
double Close;                    //收盘价
long CloseDate;                  //收市日期
double TurnoverVol;              //总成交量
double TurnoverAmt;              //总成交额
long OpenInt;                    //未平仓
STR16 ProdCode;                  //合约代码
STR40 ProdName;                  //合约名称
char DecInPrice;                 //合约小数位
} SPApiPrice;
```

返回值 ：

0, 表示请求成功.

1.27 SPAPI_GetPriceByCode 方法

该方法通过合约代码获取价格信息.

函数原型:

```
int  SPAPI_GetPriceByCode(char *prod_code, SPApiPrice *price);
```

参数:

prod_code, 合约代码.

***price,** 价格资讯信息.

返回值:

0, 代表请求成功.

1.28 SPAPI_LoadInstrumentList 方法

加载所有合约系列信息.

函数原型:

```
int  SPAPI_LoadInstrumentList()
```

返回值:

0:表示请求成功.

1.29 SPAPI_GetInstrumentCount 方法

该方法用来获取市场产品系列的数量.

函数原型:

```
int  SPAPI_GetInstrumentCount()
```

返回值:

市场产品系列的数量.

1.30 SPAPI_GetInstrument 方法

该方法通过序号获取产品系列的信息.

函数原型:

```
int  SPAPI_GetInstrument(int idx, SPApiInstrument
*inst)
```

参数:

idx, 表示产品系列列表中的序号.(序号是从0开始.)

***inst,** 产品系列信息.

产品系列结构:

```
typedef  struct
{
        double Margin;
        long ContractSize;
        STR16 MarketCode;     //市场代码
        STR16 InstCode;       //产品系列代码
        STR40 InstName;       //英文名称
        STR40 InstName1;      //繁体名称
        STR40 InstName2;      //简体名称
        STR4 Ccy;             //产品系列的交易币种
        char DecInPrice;      //产品系列的小数位
        char InstType;        //产品系列的类型
} SPApiInstrument;
```

返回值:

0, 表示请求成功.

1.31 SPAPI_GetInstrumentByCode 方法

该方法通过产品代码获取该产品的产品系列信息.

函数原型:

```
int  SPAPI_GetInstrumentByCode)(char *inst_code,
SPApiInstrument *inst);
```

参数:

inst_code, 要查询产品系列代码.

***inst,** 产品系列信息.

1.32 SPAPI_GetProductCount 方法

该方法用来获取全部合约的数量.

函数原型:

```
int  SPAPI_GetProductCount()
```

返回值:

市场合约的数量.

1.33 SPAPI_GetProduct 方法

该方法通过序号获取产品合约信息.

函数原型:

```
int SPAPI_GetProduct(int idx, SPApiProduct *prod)
```

参数:

idx, 表示产品列表中的序号.(序号是从0开始.)

prod, 产品信息.

产品信息结构:

```
typedef  struct
{
    STR16 ProdCode;       //产品代码
    char ProdType;        //产品类型
    STR40 ProdName;       //产品英文名称
    STR16 Underlying;     //关联的期货合约
    STR16 InstCode;       //产品系列名称
    long ExpiryDate;      //产品到期时间
    char CallPut;         //期权方向认购与认沽
    long Strike;          //期权行使价
    long LotSize;         //手数
    STR40 ProdName1;      //产品繁体名称
    STR40 ProdName2;      //产品简体名称
    char OptStyle;        //期权类型
    long TickSize;        //产品价格最小变动位数
}SPApiProduct;
```

返回值:

0, 表示请求成功.

1.34 SPAPI_GetProductByCode 方法

该方法通过合约代码获取该产品的合约信息

函数原型:

```
int SPAPI_GetProductByCode(char *prod_code,
 SPApiProduct *prod)
```

参数:

prod_code, 要查询的合约代码.

prod, 返回的合约产品信息.

返回值:

0, 代表请求成功.

1.35 SPAPI_GetAccBalCount 方法

该方法用来获取现金结余的数量.

函数原型:

```
int  SPAPI_GetAccBalCount()
```

返回值:

返回一个整型的帐户现金结余数量.

1.36 SPAPI_GetAccBal 方法

该方法通过序号获取账户现金结余信息.

函数原型:

```
int SPAPI_GetAccBal(int idx, SPApiAccBal *acc_bal)
```

参数:

idx, 表示现金结余列表中的序号.(序号是从0开始.)

***acc_bal,** 现金结余信息.

帐户现金结余结构:

```
typedef  struct
{
        double CashBf;          //上日结余
        double TodayCash;       //今日存取
        double NotYetValue;     //未交收
        double Unpresented;     //未兑现
        double TodayOut;        //提取要求
        STR4 Ccy;               //货币
} SPApiAccBal;
```

现金结余 = CashBf + TodayCash + NotYetValue

参考兑换率：请参考GetCcyRate。

现金(基本货币) 现金结余 * 兑换率

返回值:

0, 表示请求成功.

1.37 SPAPI_GetAccBalByCurrency 方法

该方法通过币种获取现金结余信息.

函数原型:

```
int  SPAPI_GetAccBalByCurrency(char *ccy, SPApiAccBal
```

*acc_bal)

参数:

ccy, 要查询的币种.

***acc_bal,** 现金结余信息.

返回值:

0, 表示请求成功.

1.38 SPAPI_SubscribeTicker 方法

该方法用来订阅/取消指定合约的市场成交信息.

函数原型:

```
int  SPAPI_SubscribeTicker(char *prod_code, int mode)
```

参数:

prod_code, 合约代码.

mode, 0:取消市场成交记录.
1:订阅市场成交记录.

返回值:

0, 表示请求成功.

1.39 SPAPI_GetAccInfo 方法

该方法用来获取帐户信息

函数原型:

```
int SPAPI_GetAccInfo(SPApiAccInfo *acc_info)
```

参数:

acc_info, 账户信息.

账户信息结构:

```
typedef  struct
{
double NAV;                    //资产净值
double BuyingPower;            //购买力
double CashBal;                //现金结余
double MarginCall;             //追收金额
double CommodityPL;            //商品盈亏
double LockupAmt;              //冻结金额
double CreditLimit;            //信贷限额
double MaxMargin;              //最高保证金
double MaxLoanLimit;           //最高借贷上限
double TradingLimit;           //信用交易额
double RawMargin;              //原始保证金
double IMargin;                //基本保证金
double MMargin;                //维持保证金
double TodayTrans;             //交易金额
```

```
    double LoanLimit;           //证券可按总值
    double TotalFee;            //费用总额
    double LoanToMR             //借贷/可按值%
    double LoanToMV             //借贷/市值%
    STR16 AccName;              //赢名称
    STR4 BaseCcy;               //基本币种
    STR16 MarginClass;          //保证金类别
    STR16 TradeClass;           //交易额别
    STR16 ClientId;             //客户
    STR16 AEId;                 // 经纪
    char AccType;               //户口类别
    char CtrlLevel;             //控制级数
    char Active;                // 生效
    char MarginPeriod;          //时段
} SPApiAccInfo;
```

返回值:

0, 表示请求成功.

1.40 SPAPI_LoadOrderReport 方法

调用该方法用于加载所有工作中订单.(该方法只会返回一次有效数据)

函数原型:

```
int SPAPI_LoadOrderReport(char *acc_no);
```

参数:

***acc_no,** 用户帐号

返回值:

0, 表示请求成功.

-1, 表示已经请求过该方法.

1.41 SPAPI_LoadTradeReport 方法

调用该方法用于加载所有已成交记录.(该方法只会返回一次有效数据)

函数原型:

```
int SPAPI_LoadTradeReport (char *acc_no);
```

参数:

***acc_no,** 用户帐号

返回值:

0, 表示请求成功.

-1, 表示已经请求过该方法.

1.42 SPAPI_GetDllVersion 方法

查询 DLL 版本信息.

函数原型:

```
int  SPAPI_GetDllVersion(char *dll_ver_no, char *dll_rel_no, char *dll_suffix)
```

参数:

dll_ver_no, DLL的版本信息.

dll_rel_no, 发布版本号

dll_suffix, 更新时间.

返回值:

0, 表示成功.

1.43 SPAPI_LoadProductInfoListByCode 方法

根据产品系列代码加载该系列下的合约信息.

函数原型:

```
int  SPAPI_LoadProductInfoListByCode(char *inst_code)
```

参数:

inst_code, 产品系列代码.

返回值:

0, 表示成功.

1.44 SPAPI_SendAccControl 方法

户口控制，该方法只针对 AE。

函数原型:

```
int SPAPI_SetAccControl
                (char *acc , char ctrl_mask,char ctrl_level)
```

参数:

acc, 户口.

ctrl_mask,

```
#define CTRLMASK_CTRL_LEVEL     1  //户口控制:级别
#define CTRLMASK_KICKOUT        2  //户口控制:踢走
```

当ctrl_mask为1设置户口级数，为2时踢出用户登录状态。

ctrl_level,

0:"级别0 - 正常客户使用",
1:"级别1 - 暂停客户交易",
2:"级别2 - 暂停客户登入及交易",
3:"级别3 - 冻结户口",
4:"级别4 - 只限客户交易",
当ctrl_mask为1，ctrl_level输入0-4的级别,
当ctrl_mask为2，ctrl_level输入0就可以了。

返回值:

0, 表示成功.

1.45 SPAPI_SetApiLogPath 方法

设置日志的存放位置.

函数原型:

```
int  SPAPI_SetApiLogPath(char *path)
```

参数:

path, 路径.

返回值:

0, 表示成功.

1.46 SPAPI_GetCcyRateCount 方法

该方法用来获取参考兑换率的数量。

函数原型:

```
int  SPAPI_GetCcyRateCount ()
```

返回值:

返回数量。

1.47 SPAPI_GetCcyRate 方法

该方法在 AE 模式情况下需先 AccounLogin 一个用户才能获取所有兑换率。

函数原型:

```
int  SPAPI_GetCcyRate(int idx, SPAPICcyRate *ccy_rate)
```

参数:

idx, 表示兑换率列表中的序号.(序号是从0开始.)

ccy_rate, 参考兑换率信息

兑换率结构

```
typedef struct
{
    STR4 Ccy;        //货币
    double Rate;     //兑换率
} SPAPICcyRate;
```

返回值:

0, 表示成功.

-1, 表示失败.

-2，没用 Access Login

1.48 SPAPI_GetCcyRateByCcy 方法

该方法在 AE 模式情况下需先 AccounLogin 一个用户才能获取所有兑换率。

函数原型:

```
int  SPAPI_GetCcyRateByCcy(char *ccy, double rate)
```

参数:

ccy, 想要获取兑换率的货币名

rate, 得到指定货币的兑换率

返回值:

0, 表示成功.

-1, 表示失败.

-2，没用 Access Login

1.49 SPAPI_AccountLogin 方法

该方法只针对 AE,当 AE 登录后可选择性登录账户,登录成功同样会触发 SPAPI_RegisterLoginAccInfo.

函数原型:

```
int  SPAPI_AccountLogin(char *acc_no)
```

参数:

acc_no, 账户名.

返回值:

0, 表示成功.

-1, 表示失败. 不是 AE 调用此方法返回-1.

1.50 SPAPI_AccountLogout 方法

该方法只针对 AE,当 AE 选择账户后可用它来释放账户.

函数原型:

```
int  SPAPI_AccountLogout(char *acc_no)
```

参数:

acc_no, 账户名.

返回值:

0, 表示成功.

-1, 表示失败. 不是 AE 调用此方法返回-1.

3 接口回调方法

1.1 SPAPI_RegisterLoginReply 方法

注册一个登录回调方法.

函数原型:

```
void SPAPI_RegisterLoginReply(LoginReplyAddr  addr)
```

参数:

LoginReplyAddr addr,

LoginReplyAddr 参数原型:

```
void (SPDLLCALL *LoginReplyAddr)(long ret_code, char *ret_msg);
```

LoginReplyAddr 参数:

ret_code: 返回一个长整型编号.0:表示登录成功.如果登录失败也会有相应的错误编号.请查看错误信息列表.

ret_msg: 返回的登录信息.如果登录成功返回的是一个空字符串.如果登录失败会返回相应的错误提示.

1.2 SPAPI_RegisterLogoutReply 方法

注册一个登出回调方法.

函数原型:

```
void SPAPI_RegisterLogoutReply(LogoutReplyAddr  addr)
```

参数:

LogoutReplyAddr addr,

LogoutReplyAddr 参数原型:

```
void (SPDLLCALL *LogoutReplyAddr)(long ret_code, char *ret_msg);
```

LogoutReplyAddr 参数:

ret_code: 返回一个长整型编号.0:表示登出成功.如果登出失败也会有相应的错误编号.请查看错误信息列表.

ret_msg: 返回的登出信息.如果登出成功返回的是一个空字符串.如果登出失败会返回相应的错误提示.

1.3 SPAPI_RegisterPswChangeReply 方法

注册一个修改密码的回调方法.

函数原型:

```
void SPAPI_RegisterPswChangeReply(PswChangeReplyAddr addr)
```

参数:

PswChangeReplyAddr addr

PswChangeReplyAddr 参数原型:

```
void (SPDLLCALL *PswChangeReplyAddr)
                        (long ret_code, char *ret_msg);
```

PswChangeReplyAddr 参数:

ret_code: 返回一个长整型编号.0:表示密码修改成功.如果修改失败也会有相应的错误编号.请查看错误信息列表.

ret_msg: 返回的密码修改信息.如果密码成功返回的是一个空字符串.如果修改失败会返回相应的错误提示.

1.4 SPAPI_RegisterLoginStatusUpdate 方法

注册一个登录状态更新的回调方法.

函数原型:

```
void SPAPI_RegisterLoginStatusUpdate(
                        LoginStatusUpdateAddr  addr)
```

参数:

LoginStatusUpdateAddr addr

LoginStatusUpdateAddr 参数原型:

```
void (SPDLLCALL *LoginStatusUpdateAddr)(long
login_status);
```

LoginStatusUpdateAddr 参数:

login_status, 返回一个长整型的状态编号.0-11.

0,断开连接.　1,连接中.　2,已连接.　3,连接错误.

4,登录中.　5,已登录.　6,登出中.　7,已登出.

8,登录失败　9,连接丢失.　10,断开中. 11, 主机请求

1.5 SPAPI_RegisterOrderRequestFailed 方法

注册一个订单请求失败回调方法.

函数原型:

```
void SPAPI_RegisterOrderRequestFailed (
                        ApiOrderRequestFailedAdd addr)
```

参数:

ApiOrderRequestFailedAdd addr

ApiOrderRequestFailedAdd 参数原型:

```
void (SPDLLCALL *ApiOrderRequestFailedAdd)(tinyint
action, SPApiOrder *order, long err_code, char
*err_msg);
```

ApiOrderRequestFailedAdd 参数:

action: 订单操作编号
***order:** 订单信息
err_code: 错误编码
***err_msg:** 错误信息

1.6 SPAPI_RegisterLoginAccInfo 方法

注册一个客户登入信息回调的方法.

函数原型:

```
void  SPAPI_RegisterLoginAccInfo(
                              LoginAccInfoAddr  addr);
```

参数:

LoginAccInfoAddr addr

LoginAccInfoAddr 参数原型:

```
void (SPDLLCALL *LoginAccInfoAddr)(char *acc_no, int
max_bal, int max_pos, int max_order);
```

LoginAccInfoAddr 参数:

acc_no, 返回客户登录成功的帐号.
max_bal, 返回客户最大的交易币种.
max_pos, 返回客户最大的持仓数量.
max_order, 返回客户最大的订单数量.

1.7 SPAPI_RegisterTradeReport 方法

注册一个成交记录更新回调的方法.

函数原型:

```
void  SPAPI_RegisterTradeReport(
                              ApiTradeReportAddr addr)
```

TradeUpdateAddr addr 参数原型:

```
void (SPDLLCALL *ApiTradeReportAddr)(long rec_no, SPApiTrade
     *trade);
```

ApiTradeReportAddr参数:

rec_no, 成交记录在服务器中的记录编号.
***trade,** 已成交的订单信息.

1.8 SPAPI_RegisterLoadTradeEnd 方法

注册一个普通客户请求成交记录后,成交记录全部返回完成后，执行此回调.

函数原型:

```
void SPAPI_RegisterLoadTradeEnd(LoadTradeEndAddr addr);
```

LoadTradeEndAddr addr 参数原型:

```
void (SPDLLCALL *LoadTradeEndAddr)(char *acc_no);
```

LoadTradeEndAddr 参数:

acc_no:用户账号

1.9 SPAPI_RegisterLoadAETradeEnd 方法

注册一个 AE 用户请求成交记录后,成交记录全部返回完成后，执行此回调.

函数原型:

```
void SPAPI_RegisterLoadAETradeEnd(LoadAETradeEndAddr addr);
```

LoadAETradeEndAddr 参数原型:

```
void (SPDLLCALL *LoadAETradeEndAddr)();
```

1.10 SPAPI_RegisterApiPriceUpdate 方法

注册行情更新回调的方法.

函数原型:

```
void SPAPI_RegisterApiPriceUpdate(
                                ApiPriceUpdateAddr addr)
```

ApiPriceUpdateAddr addr 参数原型:

```
void (SPDLLCALL *ApiPriceUpdateAddr)(
                                    SPApiPrice *price)
```

ApiPriceUpdateAddr参数:

price, 合约更新的信息.

SPApiPrice结构体:

```
typedef  struct
{
    double Bid[SP_MAX_DEPTH];       //买入价
    long BidQty[SP_MAX_DEPTH];      //买入量
    long BidTicket[SP_MAX_DEPTH];   //买指令数量
    double Ask[SP_MAX_DEPTH];       //卖出价
    long AskQty[SP_MAX_DEPTH];      //卖出量
    long AskTicket[SP_MAX_DEPTH];   //卖指令数量
    double Last[SP_MAX_LAST];       //成交价
    long LastQty[SP_MAX_LAST];      //成交数量
    long LastTime[SP_MAX_LAST];     //成交时间
    double Equil;                   //平衡价
```

```
        double Open;                    //开盘价
        double High;                    //最高价
        double Low;                     //最低价
        double Close;                   //收盘价
        long CloseDate;                 //收市日期
        double TurnoverVol;             //总成交量
        double TurnoverAmt;             //总成交额
        long OpenInt;                   //未平仓
        STR16 ProdCode;                 //合约代码
        STR40 ProdName;                 //合约名称
        char DecInPrice;                //合约小数位
    } SPApiPrice;
```

1.11 SPAPI_RegisterTickerUpdate 方法

注册一个市场成交记录的回调方法.

函数原型:

```
  void SPAPI_RegisterTickerUpdate(
                                ApiTickerUpdateAddr  addr)
```

ApiTickerUpdateAddr addr 参数原型:

```
   void (SPDLLCALL *ApiTickerUpdateAddr)(
                                    SPApiTicker *ticker);
```

ApiTickerUpdateAddr 参数:

ticker, 新的市场成交记录信息.

SPApiTicker结构:

```
typedef  struct
{
        double Price;           //价格
        long Qty;               //成交量
        long TickerTime;        //时间
        long DealSrc;           //来源
        STR16 ProdCode;         //合约代码
        char DecInPrice;        //小数位
} SPApiTicker;
```

1.12 SPAPI_RegisterPServerLinkStatusUpdate 方法

注册行情登入连接状态的回调方法.

函数原型:

```
     void  SPAPI_RegisterPServerLinkStatusUpdate)(
                        PServerLinkStatusUpdateAddr  addr);
```

PServerLinkStatusUpdateAddr addr参数原型:

```
void (SPDLLCALL *PServerLinkStatusUpdateAddr)(
                                short host_id, long
                                con_status);
```

PServerLinkStatusUpdateAddr 参数:

hsot_id, 返回发生改变的行情状态服务器的ID.

83, 表示一般价格连接.

87, 表示详细价格深度连接.

88, 表示一般资讯连接.

con_status, 连接状态.(参考:2.1.8)

1.13 SPAPI_RegisterConnectionErrorUpdate 方法

注册一个连接状态错误更新回调的方法.

函数原型:

```
void  SPAPI_RegisterConnectionErrorUpdate)(
                                ConnectionErrorAddr  addr);
```

ConnectionErrorAddr addr 参数原型:

```
void (SPDLLCALL *ConnectionErrorAddr)(
                                    short host_id, long
                                link_err);
```

ConnectionErrorAddr 参数:

host_id, 连接错误的服务器ID.

link_err, 连接错误的编号.

1.14 SPAPI_RegisterOrderReport 方法

注册一个订单报告回调方法.

函数原型:

```
void SPAPI_RegisterOrderReport)(
                                ApiOrderReportAddr addr);
```

ApiOrderReportAddr addr 参数原型:

```
void (SPDLLCALL *ApiOrderReportAddr)(long rec_no,
SPApiOrder *order);
```

ApiOrderReportAddr 参数:

rec_no, 订单在服务器中的记录编号

*order, 订单信息

1.15 SPAPI_RegisterInstrumentListReply 方法

注册一个产品系列信息回调方法.

函数原型:

```
void SPAPI_RegisterInstrumentListReply(
                              InstrumentListReplyAddr  addr);
```

ProductListReplyAddr addr 参数原型:

```
void (SPDLLCALL *InstrumentListReplyAddr)(bool is_ready
                                          char *ret_msg);
```

ProductListReplyAddr 参数:

is_ready, 是否加载成功:

true:产品系列信息加载成功

false:产品系列信息加载没有完成

ret_msg, 返回的提示信息.

1.16 SPAPI_RegisterBusinessDateReply 方法

登录成功后返回一个交易日期。

函数原型:

```
SPAPI_RegisterBusinessDateReply(BusinessDateReplyAddr addr)
```

BusinessDateReplyAddr addr 参数原型:

```
void (SPDLLCALL *BusinessDateReplyAddr)(long business_date);
```

BusinessDateReplyAddr 参数:

business_date:返回一个 Unix 时间戳

1.17 SPAPI_RegisterProductListByCodeReply 方法

注册一个根据产品系列名返回合约信息的回调方法.

函数原型:

```
void SPAPI_RegisterProductListByCodeReply(
                              ProductListByCodeReplyAddr  addr);
```

ProductListByCodeReplyAddr addr 参数原型:

```
void (SPDLLCALL *ProductListByCodeReplyAddr)(
          char *inst_code, bool is_ready, char *ret_msg);
```

ProductListByCodeReplyAddr 参数:

inst_code, 产品系列代码

is_ready, 是否加载成功:

true:产品合约信息加载成功

false:产品合约信息加载没有完成

ret_msg, 返回的提示信息.

4.字符对照表

1.1 买卖动作

买入	‘B’
卖出	‘S’

1.2 止损触发类型

限价止损	‘L’
升市触发	‘U”
跌市触发	‘D’

1.3 竞价指定价格

竞价标准价格	((long)0x7fffffff)

1.4 指令类型

限价指令类型	0
竞价指令类型	2
市场价指令类型	6

主要应用于 SPApiOrder 结构体中的<char OrderType>.
当 OrderType = 0 时, 指令为限价指令, 可再设置为多种条件指令类型.
当 OrderType = 2 时, 指令为竞价指令, 此时价格设置可参照 4.3.
当 OrderType = 6 时, 指令为市价指令, 此时价格可设置为 0, 实际价格将按市价设定.

1.5 指令条件类型

普通指令	0
止损指令	1
指定时间发送指令	3
双向限价指令	4
追踪止损指令	6
开仓平仓指令(开仓)	8
开仓平仓指令(平仓)	9
*止损指令(以价格计算)	11
*双向限价指令(以价格计算)	14
*追踪止损指令(以价格计算)	16

1.6 指令有效期

当天有效	0
成交并取消	1
成交或取消	2
直到有效期	3
直到自定时间	4

1.7 发送指令动作

新增指令	1
更改指令	2
删除指令	3

1.8 指令状态

发送中	0

工作中	1
无效	2
待定	3
新增中	4
更改中	5
删除中	6
无效中	7
部分成交且工作中	8
已成交	9
已删除	10
等待批准	18
成交已覆盘	20
删除已覆盘	21
同步异常中	24
部分成交已删除	28
部分成交并删除已覆盘	29
交易所無效	30

1.9 Ticker 来源指令

DS_NORMAL(auto matching)	1
DS_CROSS(crossing)	5
DS_STDC(standard combo)	7
DS_AUC (auction)	20
DS_COMBO(combo match with out-right)	43

1.10 开仓平仓指令

		Combo Open(Stop)	Combo Open (Stop)	Combo Close (Stop)	Combo Open(Trail)	Combo Open (Trail)
OrderType	no. (0~n)	0	0	0	0	0
CondType	no. (0~n)	8	8	9	8	8
StopType	Char L/U/D	0/StopType	0/StopType	L	0/StopType	0/StopType
BuySell	Char B/S	B/S	B/S	S/B	B/S	B/S
Price	long	OpenPrice	OpenPrice	StopPrice	OpenPrice	OpenPrice
StopPrice	long	0/StopLevel	0/StopLevel	StopLevel	0/StopLevel	0/StopLevel
UpLevel	long	SubCondType=1	11	1	6	16
UpPrice	long	0	0	0	CloseTrailingStep	CloseTrailingStep
DownLevel	long	CloseLossDelta	CloseLossLevel	0	CloseLossDelta	CloseLossDelta
DownPrice	long	CloseLossTol	CloseLossTol	0	CloseLossTol	CloseLossTol
SchedTime	long	OutTime	OutTime	SendTime	OutTime	OutTime
		StopLevel= TradePrice- CloseLossDelta StopPrice= TradePrice- CloseLossDelta- CloseLossTol	StopLevel= CloseLossLevel StopPrice= CloseLossLevel- CloseLossTol		StopLevel= TradePrice- CloseLossDelta StopPrice= TradePrice- CloseLossDelta- CloseLossTol	StopLevel= CloseLossLevel StopPrice= CloseLossLevel - CloseLossTol

	Combo Close (Trail)	Combo Open (OCO)	Combo Open (OCO)	Combo Close (OCO)	Combo Open (Time)	Combo Close (Time)
OrderType	0	0	0	0	0	0
CondType	9	8	8	9	8	9
StopType	L	0/StopType	0/StopType	0	0/StopType	0
BuySell	S/B	B/S	B/S	S/B	B/S	S/B
Price	StopPrice	OpenPrice	OpenPrice	ProfitPrice	OpenPrice	0(Market)
StopPrice	StopLevel	0/StopLevel	0/StopLevel	0	0/StopLevel	0
UpLevel	6	4	14	4	3	3
UpPrice	TrailingStep	CloseProftDelta	CloseProftDelta	0	0	0
DownLevel	StopLevel(Init)	CloseLossDelta	CloseLossDelta	LossLevel	0	0
DownPrice	MarketPrice	CloseLossTol	CloseLossTol	LossTol	0	0
SchedTime	SendTime	OutTime	OutTime	SendTime	OutTime	SendTime
		LossTol= CloseLossTol LossLevel= TradePrice- CloseLossDelta ProfitPrice= TradePrice+ CloseProfitDelta	LossTol= CloseLossTol LossLevel= CloseLossLevel ProfitPrice= CloseProftLevel			

1.11 错误指令代码表

指令	中文	English
-10260001		Invalid Socket
-10260002	无效的登入端口	Invalid Entry Port
-10260003	没有权限	No User Right
-10260004	密码错误	Wrong Password
-10260005		User is challenging
-11050005	已登入	User already login
-11050002	登入已满	User login full
-11150005	已登入	User already login
-11650002	登入已满	User login full
-11230001	用户或密码错误	No Such User or Wrong Password
-11460001	没有权限:在服务器地址上	No User Right: On Server Address
-11460002	没有权限:在服务器端口上	No User Right: On Server Port
-11460003	没有权限:在客户端地址上	No User Right: On Client Address
-11460004	没有权限:在客户端端口上	No User Right: On Client Port
-11460005	没有权限:在登入端口上	No User Right: On Entry Point
-11460006	密码已过期,请更新	Password Expired
-11460007	多次密码错误,请与经纪联络	Too Many Password Err., Call Broker
-11460008	用户或密码错误	No Such User or Wrong Password
-11460009	转移登入	Redirect Login
-11460010	只能在公司内部的网路登入	Login from in-house network only
-11460011	未知的许可证密钥	Unknown license key
-11460012	未知的应用编号	Unknown application id
-11460013	在其他装置上已登入	User already login in another device
-11860001	版本太旧,请升级	Version Too Old, Please Upgrade First
-13330001	客户不存在	No Such Client
-13530000	户口未启动	Account Is Inactive
-13530001	户口已冻结	Account is frozen
-13530002	此户口之客户已被暂停	The client of this account is suspended

-15260003	没有权限作交易指示	No User Right To Access Order
-15260004	用户已停用	User Suspended
-15260005	密码错误	Wrong Password
-15260006	系统已停止	System Closed
-15260007	系统已暂停	System Suspended
-15260008	买卖繁忙	Order Request Busy
-15260009	此产品暂时未能提供电子交易，请联络阁下经纪	Product currently not available for e-trading, please contact your AE
-15260010	只限客户作交易指示	Client Trade Only
-15260011	用户在公司以外的网路登入，只能查看户口	User can view only when outside of In-house network
-15260012	没有权限作批核指示	No User Right To Approve Order
-15260013	没有权限作手动交易	No User Right For Manual Dealing
-15260014	没有权限作拒绝指示	No User Right To Reject Order
-15260015	没有权限作放弃指示	No User Right To Abort Order
-15260016	没有权限作成交入账	No User Right For Trade Booking
-15260017	不准许更改为相同密码	Change to same password not allowed
-15260018	新密码不能与最近的旧密码相同	Password cannot same as old one
-15260019	系统价格连结中断，请联络阁下经纪	System price feed not linked, please contact your AE
-15270001	价格不正确	Invalid Price
-15270002	数量不正确	Invalid Quantity
-15270003	买卖不正确	Invalid Buy/Sell
-15270004	产品不正确	Invalid Product
-15270005	有效期不正确	Invalid Validity
-15270006	日期时间不正确	Invalid Specific Time
-15270007	指示种类不正确	Invalid Order Type
-15270008	止损价不正确	Invalid Stop Price
-15270009	指示不提供	Order Not Supported
-15270010	指示已满	Order Is Full
-15270011	价格不符	Price Not Matched
-15270012	价格过高	Price Over Limit
-15270013	价格过低	Price Under Limit

-15270014	持仓已满	Position Is Full
-15270015	此指示不能无效	Inactive Order Not Allowed For This Type
-15270016	指示已经有效	Order Is Already Active
-15270017	指示已经无效	Order Is Already Inactive
-15270018	小数点不符	Decimal Point Not Matched
-15270019	户口不能买卖此产品	Account cannot trade this product
-15270020	户口不正确	Invalid Account
-15270021	读取户口错误	Get Account Error
-15270022	户口类型不正确	Invalid Acc Code Type
-15270023	获取新的请求编号错误	Get New Request No. Error
-15270024	获取新的订单编号错误	Get New Order No. Error
-15270025	获取用户市场信息错误	Get Acc Market Error
-15270026	获取订单错误	Get Order Error
-15270027	更新指示错误	Update Order Error
-15270028	此时段不允许这动作	Action Not Allowed In This Status
-15270029	此指示不允许更改	Change Not Allowed For This Order
-15270030	这动作未能提供	Action Not Supported
-15270031	当条件被触发时,指示价格偏离太远	Order Outside Price Limit When Condition Triggered
-15270032	产品已停止买卖	Product Is Stopped Trading
-15270033	指示未能执行,请等待回覆	Order not executed, please wait for reply
-15270034	指示已成交，警告:持仓可能不正确,请即联络经纪	Order Already Traded, Warning: check position with broker
-15270035	产品不允许沽出	Product Not Allowed To Sell
-15270036	产品不允许买入	Product Not Allowed To Buy
-15270037	产品只允许平仓	Product Must Be Closed Only
-15270038	预定时间已过了，指示不接受	Schedule Time Is Under Current Time, Order Not Accepted
-15270039	止损水平会被即时触发，指示不接受	Stop Level Will Be Triggered Immediately, Order Not Accepted
-15270040	OCO 止损水平可能先被触发，指示不接受	OCO Stop Level Will Be Triggered First, Order Not Accepted
-15270041	超出批核金额上限	Over Approval Limit

-15270042	已过了最后交易日	Over Last Trading Date
-15270043	超出单项指示上限	Over Single Order Limit
-15270044		Get Position Error
-15270045	交易指示批核请求进行中,请等候或先删除现有之请求	Order approval requesting, please wait or delete the existing request first
-15270046	不允许期权交易	Option Trading Not Allowed
-15270047	不允许沽空期权	Option Short Sell Not Allowed
-15270048	不允许沽空证券	Securities Short Sell Not Allowed
-15270049	预定时间超出本交易日，指示不接受	Schedule Time over current trade day, order not accepted
-15270050	超出允许的最大数量	Over the allowed quantity
-15270051	成交并/或取消未能提供	Fill and/or Kill Not Supported
-15270052	指示不允许双边开仓	Order not allowed to open on both sides
-15270053	指示不允许超出平仓数量	Order not allowed to over close
-15270054	指示不允许增加数量	Add Quantity Not Allowed
-15270055	超出持仓上限	Over Position Limit
-15270056	超出用户每日指示上限	Over user daily order limit
-15270057	不允许股票以外之证券交易	Non-Stock Product Not Allowed
-15270058	差价不正确	Invalid Tick Size
-15270099	请求批核	Approval Request
-15280000	保证金不足	Insufficient Margin
-15280001	信用查核错误	Credit Check Error
-15288001	此产品未提供保证金	Margin is not set for this product
-15289000	购买力不足	Insufficient Buying Power
-15289001	不准沽空	Short Sell Not Allowed
-15289002	购买力不足	Insufficient Buying Power
-15289003	购买力不足	Insufficient Buying Power
-15289004	超出最高保证金	Over Maximum Margin
-15230000	指示正在处理中	Order Requesting
-15999996	过期指示已被删除	The Expired Order Is Deleted
-15999997	系统增加指示	Order Added By Gateway
-15999998	系统更改指示	Order Changed By Gateway

-15999999	系统删除指示	Order Deleted By Gateway
-90000012	GW 错误:指示错误，请联络阁下经纪	GW: Transaction Failed, please contact your AE
-90002008	GW 错误:未连线，请联络阁下经纪	GW: Not Connected, please contact your AE
-90002012	GW 错误:连线错误，请联络阁下经纪	GW: Connection Error, please contact your AE
-90002014	GW 错误:段落错误，请联络阁下经纪	GW: Session Aborted, please contact your AE
-90009010	GW 错误:登入错误，请联络阁下经纪	GW: Login Error, please contact your AE
-90009011	GW 错误:产品不正确	GW: Invalid Product
-90009012	GW 错误:有效期不正确	GW: Invalid Validity
-90009013	GW 错误:买卖不正确	GW: Invalid Buy/Sell
-90009014	GW 错误:指示种类不正确	GW: Invalid Order Type
-90009015	GW 错误:价格不正确	GW: Invalid Price
-90009016	GW 错误:数量不正确	GW: Invalid Quantity
-90009021	GW 错误:[成交并取消]没有成交	GW: No Matched Order For FaK
-90009022	GW 错误:[成交或取消]没有成交	GW: No Matched Order For FoK
-90009023	GW 错误:指示没有放出市场	GW: No Order Placed
-90009024	GW 错误:未开市	GW: Market Not Open
-90009025	GW 错误:价格不准许	GW: Price Not Allowed
-90009026	GW 错误:不准许竞价盘	GW: Auction Order Not Allowed
-90009027	GW 错误:更改数量错误	GW: Change Quantity Failed
-90009028	GW 错误:指示不存在	GW: Order Not Exists
-90009029	GW 错误:不能更改价格及数量	GW: Price and Quantity Cannot Be Changed
-90009030	GW 错误:价格超出范围	GW: Price Out Of Range
-90009031	GW 错误:更改错误导致指示已被删除	GW: Order Deleted Due To Unrecoverable Change Error
-90009032	GW 错误:不正确结果	GW: Incorrect Result
-90009033	GW 错误:此时段不准许限价盘	GW: Limit Order Not Allowed At This Trading State
-90009034	GW 错误:数量过高	GW: Quantity Too High

-90009035	GW 错误:此时段不准许更改价格	GW: Price Change Not Allowed At This Trading State
-90009036	GW 错误:市场繁忙	GW: Market Is Busy
-90009037	GW 错误:有效期不提供	GW: Validity Not Supported
-90009038	GW 错误:指示种类不提供	GW: Order Type Not Supported
-90009039	GW 错误:此要求不提供	GW: Request Not Supported
-90009040	GW 错误:已暂停接受指示	GW: Gateway Is Inactive
-90009041	GW 错误:不准许所选有效期	GW: Given Time Validity Not Allowed
-90009042	GW 错误:买卖差价不正确	GW: Invalid Tick Size
-90009043	GW: 指示状态不确定	GW: Order Status Uncertain
-90009044	GW: 此时段不接受非 T+1 指示	GW: Non T+1 Order Not Allowed At This Trading State
-92000201	GW 错误:指示被拒绝	GW: Order Rejected
-92000202	GW 错误:指示被取消	GW: Order Cancelled
-90009101	GW 错误:已收市	GW: Market Closed
-90009102	GW 错误:已暂停市	GW: Market Paused
-90009103	GW 错误:此时段不准许更改指示	GW: Change Order Not Allowed At This Trading State
-90009104	GW 错误:此时段不准许删除指示	GW: Delete Order Not Allowed At This Trading State
-90009105	GW 错误:此时段不准许交易	GW: Transaction Not Allowed At This Trading State
-90009106	GW 错误:此指示种类不准许更改指示	GW: Change Not Allowed For This OrderType
-16160001	新密码太短	New Password Too Short
-16160002	密码必须为字母或数字	New Password Must Be Alpha-Numeric
-17260003	没有权限:现金存取	No User Right: Cash Change
-17260004	密码错误	Wrong Password
-17260010	金额不正确	Invalid Amount
-17260011	买卖不正确	Invalid Buy Sell
-17260012	货币不正确	Invalid Currency
-17260013	现金不足	Not Enough Cash
-17260014	计算购买力错误	Get Buying Power Error
-17260015	购买力不足	Over Buying Power

-17260016	已超出现金存取的请求限额	Over Cash Request Limit
-17260017	已超出现金存取的批核限额	Over Cash Approval Limit
-17260018	现金存取请求进行中,请等候或先删除现有之请求	Cash Change Requesting, please wait or delete the existing request first
-18260003	没有权限:更改信贷限额	No User Right: On Credit Limit Change
-18260004	没有权限:更改保证金上限	No User Right: On Max Margin Change
-18260005	密码错误	Wrong Password
-18260006	没有权限:更改最高借贷上限	No User Right: On Max Loan Limit Change
-18260007	没有权限:更改信用交易额	No User Right: On Trading Limit Change
-18260010	信贷限额不正确	Invalid Credit Limit Amount
-18260011	保证金上限不正确	Invalid Max Margin Amount
-18260012	最高借贷上限不正确	Invalid Max Loan Limit Amount
-18260013	已超出准许可更改的上限	Over Allowed Changable Limit
-18260014	信用交易额不正确	Invalid Trading Limit Amount
-20260003	没有权限:户口控制	No User Right: On Account Control
-20260004	密码错误	Wrong Password
-20430004	客户已登出	Client Already Logged Out
-22150004	客户未登入	User Is Not Logged In
-23260001	不准许自我批核	Self Approval Not Allowed
-24260003	没有权限:股票/商品存取	No User Right: Stock/Commodity In/Out
-25660001	处理中有指示存在产品不正确	Invalid Product
-25660002	指示处理中导致不能平仓,请待处理完成后再试	Invalid Status
-25660003	处理中有指示删除失败	Delete Order Failed
-25660004	未能提供市价导致不能平仓,请待提供市价后再试	No Market Price To Close Position
-25660005	处理中有平仓指示失败	Close Position Failed
-25660006	平仓指示并没有进行	Close Position Not Requested

1.12 错误指令范围表

指令范围	中文	English
-1015xxxx	用户配置文件读取错误	Read User Profile Error
-1026xxxx	验证失败	Validation Failed
-1105xxxx	添加用户错误	Add User Error
-1115xxxx	检查用户配置文件错误	Check User Profile Error
-1123xxxx	获取用户配置文件错误	Get User Profile Error
-1133xxxx	获取用户权限错误	Get User Right Error
-1153xxxx	获取账户编号错误	Get Account No. Error
-1165xxxx	添加用户配置文件错误	Add User Profile Error
-1173xxxx	获取账户错误	Get Account Error
-1171xxxx	发送错误	Send Error
-1205xxxx	用户读取错误	Read User Error
-1215xxxx	读取用户配置文件错误	Read User Profile Error
-1225xxxx	删除用户配置文件错误	Delete User Profile Error
-1231xxxx	发送错误	Send Error
-1305xxxx	用户读取错误	Read User Error
-1315xxxx	读取用户配置文件错误	Read User Profile Error
-1326xxxx	账户登录验证失败	Account Login Validation Failed
-1333xxxx	获取客户端配置文件错误	Get Client Profile Error
-1343xxxx	获取客户端账户编号错误	Get Client Account No. Error
-1353xxxx		Get Client Access Error
-1365xxxx	修改用户配置文件错误	Change User Profile Error
-1373xxxx	获取账户错误	Get Account Error
-1371xxxx	发送错误	Send Error
-1383xxxx	获取账户信息错误	Get Account Information Error
-1405xxxx	用户读取错误	Read User Error
-1415xxxx	读取用户配置文件错误	Read User Profile Error
-1426xxxx	账户登出验证失败	Account Logout Validation Failed
-1435xxxx	修改用户配置文件错误	Change User Profile Error

-1441xxxx	发送错误	Send Error
-1505xxxx	用户读取错误	Read User Error
-1515xxxx	读取用户配置文件错误	Read User Profile Error
-1526xxxx	订单请求验证失败	Order Request Validation Failed
-1527xxxx	订单检查失败	Order Check Failed
-1528xxxx		Credit Check Error
-1523xxxx	订单请求	Order Requesting
-1533xxxx	获取账户错误	Get Account Error
-1543xxxx	添加请求错误	Add Request Error
-1553xxxx	获取账户日志错误	Get Account Log Error
-1561xxxx	发送错误	Send Error
-1570xxxx		Exchange Rate Error
-1583xxxx		Get Approval Data Error
-9000xxxx	OM:请求错误	OM: Request Error
-1616xxxx	密码验证失败	Password Validation Failed
-1715xxxx	用户配置文件读取错误	Read User Profile Error
-1726xxxx	现金修改验证失败	Cash Change Validation Failed
-1733xxxx	获取账户错误	Get Account Error
-1743xxxx	修改账户错误	Change Account Error
-1751xxxx	发送错误	Send Error
-1760xxxx		Exchange Rate Error
-1773xxxx		Get Approval Data Error
-1815xxxx	读取用户配置文件错误	Read User Profile Error
-1826xxxx	市场数据验证失败	Market Data Validation Failed
-1833xxxx	获取账户错误	Get Account Error
-1843xxxx	修改账户错误	Change Account Error
-1851xxxx	发送指令	Send Error
-1860xxxx		Exchange Rate Error
-2326xxxx		Approval Validation Failed
-2323xxxx		Get Approval Request Error
-2426xxxx		Stock/Commodity In/Out Validation Failed

-2515xxxx	加载账户错误	Load Account Error
-2525xxxx	添加请求错误	Add Request Error
-2535xxxx	释放账户错误	Release Account Error
-2545xxxx	删除请求错误	Delete Request Error
-2555xxxx	获取账户错误	Get Account Error
-2566xxxx	平仓验证错误	Close Pos Validation Error

5. 使用说明

1.首先要向证券或经纪开通 AE 账户(例如开通 AE 账户名为 AE_01),再向 AE_01 关联多个普通子账户(例如在 AE_01 下开通 1000,1001,1002.....)。

2.AE 登录与普通账户登录不同，port 要用 8081,而普通账户则用 8080.填写好登录信息后登录。

3.AE 登录成功后与 Client 登录成功后的区别:当 Client 登录成功可以直接对“下单,持仓,用户信息等”进行操作,而 AE 需要登录指定户口 SPAPI_AccountLogin，
当登录成功后,才能对用户下单与查询用户的信息(见下表)。

4.AE 登录成功后，当子账户下单时都会收到子账户的下单信息.如果想拿到 AE 登录前的所有订单信息，可以用 LoadOrderReport 加载登录前的订单信息.(Trader 相同)

5.AE 下使用子账户,SPAPI 提供子账户登录与登出。(User_Id 是指登录账户，Acc_No 则是指子账户,当普通用户登录时 User_ID 与 Acc_No 是相同的，用 AE 登录例如 AE_01 这时 User_Id 是 AE_01,如果你想操作 1000 你就 AE 下(Access)登录 1000 这时 Acc_No 就等于 1000)

6.AE 下订单操作。AE_01 登录后再指定登录 1000，这时 User_Id=AE_01,Acc_No=1000 就可以对 1000 子账户进行订单操作(添加删除修改订单)

7.AE 下取持仓 户口信息。也是指定登录子账户后可以查询该子账户信息.

8.成交信息. 只能取 AE 下的订单成交记录.(AE_01 用 1000 下了一个 HSIK4 成交了,1000 自己下了一个 HSIM4 成交了.AE 只能拿到自己下的 HSIK4 记录，拿不到 1000 自己下单的成交记录).

9.SPAPI_LoadOrderReport 是请求登入前的在工作中与部分成交工作中的订单，此请求只能请求一次。(请求成交相同)

10.AE Mode 如果想看 Client Account 的订单或成交。 需要先 Acc Access 一个客户才能去请求这个客户的订单

11.AE Mode 只能查看自己名下的 Clietnt 的订单与成交。

12.有关成交时间，只要交易所有提供的都会有。(例如香港股票有因交易所有提供；而香港期货沒有，因交易所沒有提供)

注：Client Mode 登录后可以对全部功能进行操作(AccountLogin 与 AccountLogin 除外)，AE Mode 下表打“✔”的功能需要 Access Login 后才能使用。(见下表)

SPAPI_AccountLogin 与 SPAPI_AccountLogout 只有是 AE Mode 登录后才能用，Client Mode 这两个方法是不可用的。

	功能	AE Mode	方法
订单(Order)	下单	✔	SPAPI_AddOrder
	删除订单	✔	SPAPI_DeleteOrder
	修改订单	✔	SPAPI_ChangeOrder
	查询订单	✘	SPAPI_LoadOrderReport(加载登录前订单) SPAPI_GetOrderCount SPAPI_GetOrder SPAPI_GetOrderByOrderNo
	设有效订单	✔	SPAPI_ActivateOrder
	设无效订单	✔	SPAPI_InactivateOrder
持仓(Pos)	查询	✔	SPAPI_GetPosCount SPAPI_GetPos SPAPI_GetPosByProduct
成交(Trader)	查询	✘	SPAPI_LoadTradeReport(加载登录前成交) SPAPI_GetTradeCount SPAPI_GetTrade SPAPI_GetTradeByTradeNo
行情(Price)	订阅与取消	✘	SPAPI_SubscribePrice
	查询	✘	SPAPI_GetPriceCount SPAPI_GetPrice SPAPI_GetPriceByCode
产品系列	加载	✘	SPAPI_LoadInstrumentList

(Instrument)			查询前需用它加载
	查询	✘	SPAPI_GetInstrumentCount SPAPI_GetInstrument SPAPI_GetInstrumentByCode
产品信息 (Product)	加载	✘	SPAPI_LoadProductInfoListByCode 查询前需用它加载
	查询	✘	SPAPI_GetProductCount SPAPI_GetProduct SPAPI_GetProductByCode
Ticker	订阅与取消	✘	SPAPI_SubscribeTicker
用户信息 (Acc Info)	查询	✔	SPAPI_GetAccInfo
户口资金 (Acc Bal)	查询	✔	SPAPI_GetAccBalCount SPAPI_GetAccBal SPAPI_GetAccBalByCurrency