Table of Contents

1. Introduction ........................................................................................................................... 5

2. API Reference ..................................................................................................................... 7

   1.1 SPAPI_Initialize Method ................................................................................................ 7
   1.2 SPAPI_Uninitialize Method ............................................................................................ 7
   1.3 SPAPI_Poll Method ......................................................................................................... 7
   1.4 SPAPI_SetBackgroundPoll Method ............................................................................... 7
   1.5 SPAPI_SetLoginInfo Method .......................................................................................... 8
   1.6 SPAPI_Login Method ..................................................................................................... 8
   1.7 SPAPI_Logout Method ..................................................................................................... 8
   1.8 SPAPI_ChangePassword Method ................................................................................... 8
   1.9 SPAPI_GetLoginStatus Method ...................................................................................... 9
   1.10 SPAPI_AddOrder Method ............................................................................................. 9
   1.11 SPAPI_ChangeOrder Method ....................................................................................... 10
   1.12 SPAPI_GetOrderByOrderNo Method .......................................................................... 10
   1.13 SPAPI_GetOrderCount Method ................................................................................... 11
   1.14 SPAPI_GetOrder Method ............................................................................................. 11
   1.15 SPAPI_DeleteOrder Method ....................................................................................... 11
   1.16 SPAPI_ActivateOrder Method ..................................................................................... 12
   1.17 SPAPI_InactivateOrder Method .................................................................................. 12
   1.18 SPAPI_GetPosCount Method ...................................................................................... 12
   1.19 SPAPI_GetPos Method ................................................................................................. 12
   1.20 SPAPI_GetPosByProduct Method .............................................................................. 13
   1.21 SPAPI_GetTradeCount Method ................................................................................... 13
   1.22 SPAPI_GetTrade Method ............................................................................................. 13
   1.23 SPAPI_GetTradeByTradeNo Method .......................................................................... 14
   1.24 SPAPI_SubscribePrice Method ................................................................................... 15
   1.25 SPAPI_GetPriceCount Method ................................................................................... 15
   1.26 SPAPI_GetPrice Method ............................................................................................. 15
   1.27 SPAPI_GetPriceByCode Method ................................................................................. 16
<table>
<thead>
<tr>
<th>Method Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.28 SPAPI_LoadInstrumentList Method</td>
<td>16</td>
</tr>
<tr>
<td>1.29 SPAPI_GetInstrumentCount Method</td>
<td>16</td>
</tr>
<tr>
<td>1.30 SPAPI_GetInstrument Method</td>
<td>17</td>
</tr>
<tr>
<td>1.31 SPAPI_GetInstrumentByCode Method</td>
<td>17</td>
</tr>
<tr>
<td>1.32 SPAPI_GetProductCount Method</td>
<td>18</td>
</tr>
<tr>
<td>1.33 SPAPI_GetProduct Method</td>
<td>18</td>
</tr>
<tr>
<td>1.34 SPAPI_GetProductByCode Method</td>
<td>18</td>
</tr>
<tr>
<td>1.35 SPAPI_GetAccBalCount Method</td>
<td>19</td>
</tr>
<tr>
<td>1.36 SPAPI_GetAccBal Method</td>
<td>19</td>
</tr>
<tr>
<td>1.37 SPAPI_GetAccBalByCurrency Method</td>
<td>20</td>
</tr>
<tr>
<td>1.38 SPAPI_SubscribeTicker Method</td>
<td>20</td>
</tr>
<tr>
<td>1.39 SPAPI_GetAccInfo Method</td>
<td>20</td>
</tr>
<tr>
<td>1.40 SPAPI_LoadOrderReport Method</td>
<td>21</td>
</tr>
<tr>
<td>1.41 SPAPI_LoadTradeReport Method</td>
<td>21</td>
</tr>
<tr>
<td>1.42 SPAPI_GetDllVersion Method</td>
<td>22</td>
</tr>
<tr>
<td>1.43 SPAPI_LoadProductInfoListByCode Method</td>
<td>22</td>
</tr>
<tr>
<td>1.44 SPAPI_SendAccControl Method</td>
<td>23</td>
</tr>
<tr>
<td>1.45 SPAPI_SetApiLogPath Method</td>
<td>23</td>
</tr>
<tr>
<td>1.46 SPAPI_GetCcyRateCount Method</td>
<td>23</td>
</tr>
<tr>
<td>1.47 SPAPI_GetCcyRate Method</td>
<td>24</td>
</tr>
<tr>
<td>1.48 SPAPI_GetCcyRateByCcy Method</td>
<td>24</td>
</tr>
<tr>
<td>1.49 SPAPI_AccountLogin Method</td>
<td>24</td>
</tr>
<tr>
<td>1.50 SPAPI_AccountLogout Method</td>
<td>25</td>
</tr>
<tr>
<td>3 Reply Method</td>
<td>26</td>
</tr>
<tr>
<td>1.1 SPAPI_RegisterLoginReply Method</td>
<td>26</td>
</tr>
<tr>
<td>1.2 SPAPI_RegisterLogoutReply Method</td>
<td>26</td>
</tr>
<tr>
<td>1.3 SPAPI_RegisterPswChangeReply Method</td>
<td>27</td>
</tr>
<tr>
<td>1.4 SPAPI_RegisterLoginStatusUpdate Method</td>
<td>27</td>
</tr>
<tr>
<td>1.5 SPAPI_RegisterOrderRequestFailed Method</td>
<td>27</td>
</tr>
<tr>
<td>1.6 SPAPI_RegisterLoginAccInfo Method</td>
<td>28</td>
</tr>
</tbody>
</table>
Developer’s Guide (SPNativeAPI 8.6839 20150929)

1.7 SPAPI_RegisterTradeReport Method..............................................................................28
1.8 SPAPI_RegisterLoadTradeEnd Method.............................................................................29
1.9 SPAPI_RegisterLoadAETradeEnd Method........................................................................29
1.10 SPAPI_RegisterApiPriceUpdate Method..........................................................................29
1.11 SPAPI_RegisterTickerUpdate Method................................................................................30
1.12 SPAPI_RegisterPServerLinkStatusUpdate Method...............................................................31
1.13 SPAPI_RegisterConnectionErrorUpdate Method.................................................................31
1.14 SPAPI_RegisterOrderReportMethod...................................................................................31
1.15 SPAPI_RegisterInstrumentListReply Method.....................................................................32
1.16 SPAPI_RegisterBusinessDateReply Method.......................................................................32
    business_date: Unix Business Date.....................................................................................32
1.17 SPAPI_RegisterProductListByCodeReply Method...............................................................32

4. Value Table..........................................................................................................................33
1.1 Buy/Sell Type.....................................................................................................................33
1.2 Stop Order Type................................................................................................................33
1.3 AO Price.............................................................................................................................33
1.4 Order Type........................................................................................................................33
1.5 Order Conditional Type.....................................................................................................34
1.6 Validity................................................................................................................................34
1.7 Order Action.......................................................................................................................34
1.8 Order Status........................................................................................................................35
1.9 Ticker DealSrc....................................................................................................................35
1.10 Combo Open/Closes Action..............................................................................................35
1. Introduction

SPsystem API is based on the C++ class library interface provided by the expansion of library related transactions, including order handling, prices subscription, account information inquiry, etc. All class library contains the following files:

<table>
<thead>
<tr>
<th>Filename</th>
<th>Version</th>
<th>Date</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>spapidll32.dll</td>
<td>V1.0, R8.6839</td>
<td>29 Sep 2015</td>
<td>Dynamic Linked Library</td>
</tr>
<tr>
<td>spapidll.h</td>
<td>V1.0, R8.6839</td>
<td>29 Sep 2015</td>
<td>Structure Definition</td>
</tr>
</tbody>
</table>

Note: Two Modes To Login (AE & Client)
1. AE Mode: Login Port is 8081. AE needs AE Mode Right to Login otherwise system will be forced to logout immediately. User can get AE Orders and AE Trades at AE Mode.

2. Client Mode: Login port is 8080. At this mode, User can get Orders and Trades belong to the Client account.

Functions to Get Order:

Order Callback: SPAPI_RegisterOrderReport

Functions to Get Trade:
- SPAPI_GetTradeCount, SPAPI_GetTrade, SPAPI_GetTradeByTradeNo, SPAPI_LoadTradeReport

Trade Callback: SPAPI_RegisterTradeReport
2. API Reference

1.1 SPAPI_Initialize Method

This method is used for API initialization.

Function:
   int SPAPI_Initialize()

Returns:
   0, means successful.

1.2 SPAPI_Uninitialize Method

This method is used to uninitialize.

Function:
   int SPAPI_Uninitialize()

Returns:
   0, means successful.

1.3 SPAPI_Poll Method

This method is used to establish a polling.
(for single thread programming)

Function:
   void SPAPI_Poll()

1.4 SPAPI_SetBackgroundPoll Method

This method is used to establish a background polling.
(for mult-thread programming)

Function:
   void SPAPI_SetBackgroundpoll(bool onoff)

Arguments:
   onoff: True OR 1, means establish background polling.
   False OR 0, means close background polling.
1.5  **SPAPI_SetLoginInfo Method**

Sets the login information.

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

**Arguments:**
- **host**, means Host IP Address
- **port**, means Port No.
- **license**, means License Key
- **app_id**, means Application ID
- **user_id**, means User ID
- **password**, means User Password

1.6  **SPAPI_Login Method**

*Issue login request*

**Function:**
int SPAPI_Login();

**Returns:**
- 0, means successfully sent the login request.
- -9, means the DLL has been accessed

1.7  **SPAPI_Logout Method**

*Issue logout request*

**Function:**
int SPAPI_Logout();

**Returns:**
- 0, means that request was successful.

1.8  **SPAPI_ChangePassword Method**

This method is used to change password.

**Function:**
int SPAPI_ChangePassword(char *old_psw, char *new_psw)

**Arguments:**
- **old_psw**, means old password.
- **new_psw**, means new password.

**Returns:**
0, means that request was successful.

1.9  SPAPI_GetLoginStatus Method

Query the status of trading connection and price connection.

Function:
int  SPAPI_GetLoginStatus(short  host_id)

Arguments:
host_id:
81, means Transaction Link (User Login Status)
83, means Price Link
87, means Long Price Depth Link
88, means Information Link

Returns:
0, Closed.  1, Connecting.  2, Connected.  3, Connect Error.
4, Logging In.  5, Logged In.  6, Logging Out.  7,Logged Out.
8, Login Fail  9, Connection Lost 10, Closing 11, Host Request

Note:
80,81 Ref: SPAPI_RegisterLoginStatusUpdate
83,87,88 Ref: SPAPI_RegisterPServerLinkStatusUpdate

1.10 SPAPI_AddOrder Method

This method is used to add order.

Function:
int  SPAPI_AddOrder(SPApiOrder *order)

Arguments:
*order, means the pointer of order structure

Order Structure:
typedef struct
{
    double Price;     //Price
    double StopLevel; //Stop Price
    double UpLevel;   //Up Trigger Level
    double UpPrice;   //Up Trigger Price
    double DownLevel; //Down Trigger Level
    double DownPrice; //Down Trigger Price
    bigint ExtOrderNo; //Ext. order no.
    long IntOrderNo;  //Int. order no.
    long Qty;        //Remain Qty
    long TradedQty;  //Traded Qty
    long TotalQty;   //Total Qty
    long ValidTime;  //Valid Time
}
Developed's Guide (SPNativeAPI 8.6839 20150929)

```c
long SchedTime;  //Schedule Time
long TimeStamp;  //Server Timestamp
u_long OrderOptions;  //0=default, 1=T+1
STR16 AccNo;  //Account No.
STR16 ProdCode;  //Product Code
STR16 Initiator;  //Initiator
STR16 Ref;  //Reference
STR16 Ref2;  //Reference #2
STR16 GatewayCode;  //Gateway Code
STR40 ClOrderId;  //User Define Order ID
char BuySell;  //Buy or Sell
char StopType;  //Stop Order Type
char OpenClose;  //Open or Close
tinyint CondType;  //Condition Type
tinyint OrderType;  //Order Type
tinyint ValidType;  //Validity Type
tinyint Status;  //Order Status
tinyint DecInPrice;  //Dec. Place of Price
}

SPApiOrder;
```

Returns:
0, means successful.

### 1.11 SPAPI_ChangeOrder Method

This method is used to change order.

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

Arguments:
- `*order`, means the pointer of order structure
- `org_price`, means original price (At the same time, please assign new price to “Price” in order structure.)
- `org_qty`, means original quantity.

Returns:
0, means successful.

### 1.12 SPAPI_GetOrderByOrderNo Method

This method is used to get order by int order no.

Function:
```c
int  SPAPI_GetOrderByOrderNo(char *acc_no, long
```
int_order_no, SPApiOrder *order)
Arguments:
    *acc_no,       account. (ClientMode: output , AE Mode: Input)
    int_order_no, means order no. (input)
    *order,      returns order information. (output)
Returns:
    0, means successful.

1.13 SPAPI_GetOrderCount Method

This method is used to get count of working order.
Function:
    int  SPAPI_GetOrderCount()
Returns:
    the count. of working order.

1.14 SPAPI_GetOrder Method

This method is used to get the order information by index.
Function:
    int  SPAPI_GetOrder(int idx, SPApiOrder *order)
Arguments:
    idx, means order index. (Index is started from 0)
    *order, returns order information.
Returns:
    0, means successful.

1.15 SPAPI_DeleteOrder Method

This method is used to delete order.
Function:
    int SPAPI_DeleteOrder(SPApiOrder *order)
Arguments:
    *order, means the pointer of order structure.
    (For delete order, simply assign int. order no. in the structure)
Returns:
    0, means successful.
1.16 SPAPI_ActivateOrder Method

This method is used to active an order.

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

Returns:
0, means successful.

1.17 SPAPI_InactivateOrder Method

This method is used to inactive an order.

Function:
```
int SPAPI_InactivateOrder()
```

Returns:
0, means successful.

1.18 SPAPI_GetPosCount Method

This method is used to get the count of position.

Function:
```
int SPAPI_GetPosCount()
```

Returns:
the count of position

1.19 SPAPI_GetPos Method

This method is used to get position information by index.

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

Arguments:
```
idx, means position index. (Index is started from 0)
*Pos, returns position information.
```

Position Structure:
```
typedef struct
{
    long Qty;       //Previous Qty
    long DepQty;    //Deposit Qty
    long LongQty;   //Day Long Qty
    long ShortQty;  //Day Short Qty
    double TotalAmt; //Previous Amount
    double DepTotalAmt; //Deposit Amount(Qty*Price)
} SPApiPos;
```

double LongTotalAmt; //Day Long Amount (Qty*Price)
double ShortTotalAmt; //Day Short Amount (Qty*Price)
double PLBaseCcy; //P/L (Base Ccy)
double PL; //PL
double ExchangeRate; //Exchange Rate
STR16 AccNo; //Account No.
STR16 ProdCode; //Product Code
char LongShort; //Previous Buy/Sell
tinyint DecInPrice; //Decimal Place
} SPApiPos;

Returns:
0, means that request was successful.

### 1.20 SPAPI_GetPosByProduct Method

This method is used to get position by product code.

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

**Arguments:**
- `prod_code`, means product code.
- `*pos`, means position information.

**Returns:**
0, means that request was successful.

### 1.21 SPAPI_GetTradeCount Method

This method is used to get count of trade.

**Function:**
```
int SPAPI_GetTradeCount()
```

**Returns:**
the count of trade.

### 1.22 SPAPI_GetTrade Method

This method is used to get trade information by index.

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

**Arguments:**
- `idx`, means trade index. (Index is started from 0)
*trade, returns trade information.

Trade Structure:
```c
typedef struct
{
    long RecNo;         //Trade Log
    double Price;       //Trade Price
    double AvgPrice;    //Trade Avg Price
    bigint TradeNo;     //Trade No.
    bigint ExtOrderNo;  //Ext. Order No.
    long IntOrderNo;    //Int. Order No.
    long Qty;           //Traded Qty
    long TradeDate;     //Trade Date
    long TradeTime;     //Trade Time
    STR16 AccNo;        //Account No.
    STR16 ProdCode;     //Product Code
    STR16 Initiator;    //Initiator
    STR16 Ref;          //Reference
    STR16 Ref2;         //Reference #2
    STR16 GatewayCode;  //Gateway Code
    STR40 C1OrderId;    //User Define Order ID
    char BuySell;       //Buy or Sell
    char OpenClose;     //Open or Close
    tinyint Status;     //Order Status
    tinyint DecInPrice; //Dec. Place
} SPApiTrade;
```

Returns:
0, means that request was successful.

### 1.23 SPAPI_GetTradeByTradeNo Method

This method is used to get trade by internal order no and trade no.

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

Arguments:
- `int_order_no`, means Int. Order no.
- `trade_no`, means Trade No.
- `*trade`, returns trade information.

Returns:
0, means that request was successful.
1.24 **SPAPI_SubscribePrice Method**

This method is used to subscribe/unsubscribe market data.

**Function:**

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

**Arguments:**

- `prod_code`, Subscribe Product Code.
- `mode`, Subscribe Type.
  - 0: Unsubscribe Market Data
  - 1: Subscribe Market Data

**Returns:**

0, means that request was successful.

1.25 **SPAPI_GetPriceCount Method**

This method is used to get count of product.

**Function:**

```c
int SPAPI_GetPriceCount();
```

**Returns:**

the count of product

1.26 **SPAPI_GetPrice Method**

This method is used to get price information by index.

**Function:**

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

**Arguments:**

- `idx`, means index (The index start from 0).
- `*price`, returns the price information.

**Price Structure:**

```c
typedef struct {
    double Bid[SP_MAX_DEPTH]; // Bid Price
    long BidQty[SP_MAX_DEPTH]; // Bid Qty
    long BidTicket[SP_MAX_DEPTH]; // No. of Bid Ticket
    double Ask[SP_MAX_DEPTH]; // Ask Price
    long AskQty[SP_MAX_DEPTH]; // Ask Qty
    long AskTicket[SP_MAX_DEPTH]; // No. Of Ask Ticket
    double Last[SP_MAX_LAST]; // Last
} SPApiPrice;
```
Returns:
0, means that request was successful.

1.27 SPAPI_GetPriceByCode Method

This method is used to get price by product code.

Function:
int SPAPI_GetPriceByCode(char *prod_code, SPApiPrice *price);

Arguments:
prod_code, means product code
*price, returns price information.

Returns:
0, means that request was successful.

1.28 SPAPI_LoadInstrumentList Method

This method is used to load all existing Instrument.

Function:
int SPAPI_LoadInstrumentList()

Returns:
0, means that request was successful.

1.29 SPAPI_GetInstrumentCount Method

This method is used to get the count of instrument.
Function:
int SPAPI_GetInstrumentCount()

Returns:
the count of instrument

1.30 SPAPI_GetInstrument Method

This method is used to get the instrument information by index.

Function:
int SPAPI_GetInstrument(int idx, SPApiInstrument *inst)

Arguments:
idx, means index (The index start from 0).
*inst, returns the instrument information.

Instrument Structure:
typedef struct
{
    double Margin;
    long ContractSize;
    STR16 MarketCode; //Market Code
    STR16 InstCode; //Instrument Code
    STR40 InstName; //Instrument Name (EN)
    STR40 InstName1; //Instrument Name (TC)
    STR40 InstName2; //Instrument Name (SC)
    STR4 Ccy; //Currency
    char DecInPrice; //Dec. place
    char InstType; //Instrument Type
} SPApiInstrument;

Returns:
0, means that request was successful.

1.31 SPAPI_GetInstrumentByCode Method

This method is used to get instrument information by instrument code.

Function:
int SPAPI_GetInstrumentByCode(char *inst_code, SPApiInstrument *inst);

Arguments:
inst_code, means instrument code.
*inst, returns instrument information.
1.32 SPAPI_GetProductCount Method

This method is used to get count of product.

**Function:**

```c
int SPAPI_GetProductCount()
```

**Returns:**

the count of product

1.33 SPAPI_GetProduct Method

This method is used to get product information by index.

**Function:**

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

**Arguments:**

- `idx`, means index (The index start from 0).
- `*prod`, returns product information.

**Product Structure:**

```c
typedef struct {
    STR16 ProdCode;  // Product Code
    char ProdType;    // Product Type
    STR40 ProdName;  // Product Name (EN)
    STR16 Underlying;  // Underlying
    STR16 InstCode;  // Instrument Code
    long ExpiryDate;  // Expiry Date
    char CallPut;    // Call or Put
    long Strike;     // Strike Price
    long LotSize;    // Lot Size
    STR40 ProdName1; // Product Name (TC)
    STR40 ProdName2; // Product Name (SC)
    char OptStyle;  // Option Style
    long TickSize;  // Tick Size
}
SPApiProduct;
```

**Returns:**

0, means that request was successful.

1.34 SPAPI_GetProductByCode Method

This method is used to get product information by product code.
Function:

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

Arguments:

- `prod_code`, means product code.
- `*prod`, returns product information.

Returns:

- `0`, means that request was successful.

### 1.35 SPAPI_GetAccBalCount Method

This method is used to get count of account balance.

Function:

```c
int SPAPI_GetAccBalCount()
```

Returns:

- the count of account balance

### 1.36 SPAPI_GetAccBal Method

This method is used to get account balance by index.

Function:

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

Arguments:

- `idx`, means index (The index start from 0).
- `*acc_bal`, returns account balance.

Account Balance Structure:

```c
typedef struct
{
    double CashBf; //Cash B/F
    double TodayCash; //Today Cash In/Out
    double NotYetValue; //Unsettle
    double Unpresented; //Unpresent
    double TodayOut; //Withdrawal Request
    STR4 Ccy; //Currency Code
} SPApiAccBal;
```

CashBalance = CashBf + TodayCash + NotYetValue

Ref.FxRate : Ref GetCcyRate.

Cash(Base Ccy)= CashBalance * Ref FxRate.

Returns:

- `0`, means that request was successful.
1.37 SPAPI_GetAccBalByCurrency Method

This method is used to get account balance by currency.

Function:  
int SPAPI_GetAccBalByCurrency(char *ccy, SAPIAccBal *acc_bal)

Arguments:  
ccy, means currency code.
*acc_bal, returns account balance.

Returns:  
0, means that request was successful.

1.38 SPAPI_SubscribeTicker Method

This method is used to subscribe/unsubscribe ticker.

Function:  
int SPAPI_SubscribeTicker(char *prod_code, int mode)

Arguments:  
prod_code, means product code
Mode, 0: Unsubscribe Ticker
1: Subscribe Ticker

Returns:  
0, means that request was successful.

1.39 SPAPI_GetAccInfo Method

This method is used to get account information.

Function:  
int SPAPI_GetAccInfo(SAPIAccInfo *acc_info)

Arguments:  
acc_info, return account information.

SAPIAccInfo Structure:

typedef struct
{
    double NAV; //nav
    double BuyingPower; //Buying Power
    double CashBal; //Cash Balance
    double MarginCall; //Margin Call
    double CommodityPL; //Commodity P/L
    double LockupAmt; //Lockup Amp
    double CreditLimit; //Credit Limit
double MaxMargin; //Max Margin
double MaxLoanLimit; //Max Loan Limit
double TradingLimit; //Trading Limit
double RawMargin; //Raw Margin
double IMargin; //Initial Margin
double MMargin; //Maintenance Margin
double TodayTrans; //Today Transaction
double LoanLimit; //Loan Limit
double TotalFee; //Total Fee
double LoanToMR //Loan/Marginable
double LoanToMV //Loan/Mkt Value
STR16 AccName; //Account Name
STR4 BaseCcy; //Base Currency
STR16 MarginClass; //Margin Class
STR16 TradeClass; //Trade Class
STR16 ClientId; //Client Login ID
STR16 AEId; //AE Code
char AccType; //Account Type
char CtrlLevel; //Control Level
char Active; //Account Status
char MarginPeriod; //Margin Period

} SPApiAccInfo;

Returns:
0, means that request was successful.

1.40 SPAPI_LoadOrderReport Method

This method is used to load all existing orders.

Function:
:int SPAPI_LoadOrderReport(char *acc_no);

Arguments:
*acc_no,  means account no.

Returns:
0, means that request was successful.
-1, means that this request has already sent.

1.41 SPAPI_LoadTradeReport Method

This method is used to load all existing trades.

Function:
:int SPAPI_LoadTradeReport (char *acc_no);
Developer’s Guide (SPNativeAPI 8.6839 20150929)

Arguments:
*acc_no, means account no.

Returns:
0, means that request was successful.
-1, means that this request has already sent.

1.42 SPAPI_GetDllVersion Method

This method is used to get DLL version.

Function:
int SPAPI_GetDllVersion(char *dll_ver_no, char
*dll_rel_no, char *dll_suffix)

Arguments:
dll_ver_no, means DLL version no.
dll_rel_no, means DLL release no.
dll_suffix, means DLL suffix

Returns:
0, means successful.

1.43 SPAPI_LoadProductInfoListByCode Method

This method is used to load By instrument code existing Product.

Function:
int SPAPI_LoadProductInfoListByCode(char *inst_code)

Arguments:
inst_code, instrument code.

Returns:
0, means successful.
1.44 SPAPI_SendAccControl Method

This method is user set Account control.

Function:
int SPAPI_SendAccControl(char *acc, char ctrl_mask, char ctrl_level)

Arguments:
acc: Account.
ctrl_mask:
#define CTRLMASK_CTRL_LEVEL 1 //AccControl:Level
#define CTRLMASK_KICKOUT 2 //AccControl:Kickout
When ctrl_mask 1 set Account Level.2 Kickout Account.
ctrl_level:
0: “Level 0 - Normal Client Access”,
1: “Level 1 - Disable Client Trading”,
2: “Level 2 - Suspend Client Login and Trading”,
3: “Level 3 - Freeze Account”,
4: “Level 4 - Client Trade Only”,
When ctrl_mask 1:ctrl_level input 0-4 Ctrl Level.
When ctrl_mask 2:ctrl_level input 0 is ok,

Returns:
0, means successful.

1.45 SPAPI_SetApiLogPath Method

This method is user set api log path.

Function:
int SPAPI_SetApiLogPath(char *path)

Arguments:
path: The specified location

Returns:
0, means successful.

1.46 SPAPI_GetCcyRateCount Method

This method is used to get the count of CcyRate.

Function:
int SPAPI_GetCcyRateCount()

Returns:
the count of CcyRate
1.47  SPAPI_GetCcyRate Method

This method is used to get CcyRate by index. AE mode AccountLogin.

Function:
int  SPAPI_GetCcyRate(int idx, SPApiCcyRate *ccy_rate)

Arguments:
idx, means CcyRate index. (Index is started from 0)
*Pos, returns CcyRate information.

CcyRate Structure:
typedef struct
{
    char* ccy;   //Ccy
    double rate; //Ccy rate
}

Returns:
0, means successful.
-1, means fail.
-2, AE: Not AccountLogin.

1.48  SPAPI_GetCcyRateByCcy Method

This method is used to get CcyRate by Ccy code. AE mode AccountLogin

Function:
int  SPAPI_GetCcyRateByCcy(char *ccy, SPApiCcyRate *ccy_rate)

Arguments:
ccy, means ccy code.
*ccy_rate, means ccy rate information.

Returns:
0, means that request was successful.

1.49  SPAPI_AccountLogin Method

This method only for AE. AE can select account by this method.

Function:
int  SPAPI_AccountLogin(char *acc_no)

Arguments:
acc_no, account.

Returns:
0, means successful.
-1, means fail. Not AE Login.
1.50 SPAPI_AccountLogout Method

This method only for AE. AE can release account by this method.

Function:
   int SPAPI_AccountLogout(char *acc_no)
Arguments:
   acc_no, account.
Returns:
   0, means successful.
   -1, means fail. Not AE Logout.
3 Reply Method

1.51

1.1 SPAPI_RegisterLoginReply Method

This method is used to register “Login Reply” callback.

Function:

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

Arguments:

- `LoginReplyAddr addr`,
- `LoginReplyAddr`:
  ```c
  void (SPDLLCALL *LoginReplyAddr)(long ret_code, char *ret_msg);
  ```

 `LoginReplyAddr` Arguments:

- `ret_code`, returns a long integer. (0=successful; otherwise=fail;)
- `ret_msg`, returns error message if login fail.

1.2 SPAPI_RegisterLogoutReply Method

This method is used to register “Logout Reply” callback.

Function:

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

Arguments:

- `LogoutReplyAddr addr`,
- `LogoutReplyAddr`:
  ```c
  void (SPDLLCALL *LogoutReplyAddr)(long ret_code, char *ret_msg);
  ```

 `LogoutReplyAddr` Arguments:

- `ret_code`, returns a long integer. (0=successful; otherwise=fail;)
- `ret_msg`, returns error message if login fail.
1.3  **SPAPI_RegisterPswChangeReply Method**

This method is used to register “password change” callback.

**Function:**

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

**Arguments:**

- `PswChangeReplyAddr`:

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

  **PswChangeReplyAddr Arguments:**

  - `ret_code`, returns a long integer. (0=successful; otherwise=fail;)
  - `ret_msg`, returns error message if login fail.

1.4  **SPAPI_RegisterLoginStatusUpdate Method**

This method is used to register “Login Status Update” callback.

**Function:**

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

**Arguments:**

- `LoginStatusUpdateAddr`:

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

  **LoginStatusUpdateAddr Arguments:**

  - `login_status`, returns status no. (0-11)

    0, Closed  1, Connecting  2, Connected  3, Connect Error
    4, Logging In  5, Logged In  6, Logging Out  7, Logged Out
    8, Login Fail  9, Connection Lost  10, Closing  11, Host Request

1.5  **SPAPI_RegisterOrderRequestFailed Method**

This method is used to register “order request failed” callback.

**Function:**

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

**Arguments:**

- `ApiOrderRequestFailedAdd addr`
ApiOrderRequestFailedAdd:

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

**ApiOrderRequestFailedAdd Arguments:**
- `action`: means order action
- `*order`: means order information
- `err_code`: means error code
- `*err_msg`: means error message

### 1.6 SPAPI_RegisterLoginAccInfo Method

This method is used to register “Login Account Information” callback.

**Function:**
```c
void SPAPI_RegisterLoginAccInfo(
    LoginAccInfoAddr addr);
```

**Arguments:**
- `LoginAccInfoAddr addr`

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

**LoginAccInfoAddr Arguments:**
- `acc_no`: returns account no.
- `max_bal`: returns max. no. of balance
- `max_pos`: returns max. no. of position
- `max_order`: returns max. no. of order

### 1.7 SPAPI_RegisterTradeReport Method

This method is used to register “Trade Report” callback.

**Function:**
```c
void SPAPI_RegisterTradeReport(
    ApiTradeReportAddr addr);
```

**Arguments:**
- `TradeUpdateAddr addr`

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

**ApiTradeReportAddr Arguments:**
- `rec_no`: means trade record no.
1.8 SPAPI_RegisterLoadTradeEnd Method

This method is used to register client “Load Trade End” callback
Function:
void SPAPI_RegisterLoadTradeEnd(LoadTradeEndAddr addr);
Arguments:
LoadTradeEndAddr addr
LoadTradeEndAddr Arguments:
acc_no: user id;

1.9 SPAPI_RegisterLoadAETradeEnd Method

This method is used to register AE “Load Trade End” callback
Function:
void SPAPI_RegisterLoadAETradeEnd(LoadAETradeEndAddr addr);
Arguments:
LoadAETradeEndAddr addr
LoadAETradeEndAddr Arguments:

1.10 SPAPI_RegisterApiPriceUpdate Method

This method is used to register “API Price Update” callback.
Function:
void SPAPI_RegisterApiPriceUpdate(
    ApiPriceUpdateAddr addr)
Arguments:
    ApiPriceUpdateAddr addr
    ApiPriceUpdateAddr:
    void (SPDLLCALL *ApiPriceUpdateAddr)(
        SPApiPrice *price)
    ApiPriceUpdateAddr Arguments:
    price, returns price update

    SPApiPrice Structure:
    typedef struct
    {
        double Bid[SP_MAX_DEPTH];   //Bid Price
1.11 SPAPI_RegisterTickerUpdate Method

This method is used to register “Ticker Update” callback.

Function:

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

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

Arguments:

`ticker`, returns the following structure if ticker update.

`SPApiTicker Structure`:

```c
typedef struct {
    double Price;  // Price
    long Qty;      // Quantity
    long TickerTime;  // Ticker Time
    long DealSrc;   // Deal Source
} SPApiPrice;
```
1.12 **SPAPI_RegisterPServerLinkStatusUpdate Method**

This method is used to register “Price Server Link Status Update” callback.

**Function:**

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

**PServerLinkStatusUpdateAddr** Arguments:
- `host_id`, returns Host ID.
  - 83, means Price Link
  - 87, means Long Price Depth Link
  - 88, means Information Link
- `con_status`, means connection status (Reference: 2.1.8)

1.13 **SPAPI_RegisterConnectionErrorUpdate Method**

This method is used to register “Connection Error Update” callback.

**Function:**

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

**ConnectionErrorAddr** Arguments:
- `host_id`, Means Host ID occurs connection error
- `link_err`, Means error no.

1.14 **SPAPI_RegisterOrderReportMethod**

This method is used to register “Order Report” callback.

**Function:**

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

**ApiOrderReportAddr Arguments:**
- rec_no, means record no. of order
- *order, means order information

1.15 **SPAPI_RegisterInstrumentListReply Method**

This method is used to register “Instrument List Reply” callback.

**Function:**
void SPAPI_RegisterInstrumentListReply()
  InstrumentListReplyAddr addr;

**InstrumentListReplyAddr addr:**
void (SPDLLCALL *InstrumentListReplyAddr)(
  bool is_ready, char *ret_msg);

**InstrumentListReplyAddr Arguments:**
- is_ready, true, means that load was successful.
  false, means that load was not ready yet.
- ret_msg, means prompt message.

1.16 **SPAPI_RegisterBusinessDateReply Method**

This method is used to register “Business Date” callback.

**Function:**
SPAPI_RegisterBusinessDateReply(BusinessDateReplyAddr addr)

**BusinessDateReplyAddr addr:**
void (SPDLLCALL *BusinessDateReplyAddr)(long business_date);

**BusinessDateReplyAddr:**
- business_date: Unix Business Date.

1.17 **SPAPI_RegisterProductListByCodeReply Method**

This method is used to register “Product List Reply(By instrument code)” callback.

**Function:**
void SPAPI_RegisterProductListByCodeReply(
  ProductListByCodeReplyAddr addr);

**ProductListByCodeReplyAddr addr:**
void (SPDLLCALL *ProductListByCodeReplyAddr)(
  char* inst_code, bool is_ready, char *ret_msg);
ProductListByCodeReplyAddr Arguments:
- inst_code, instrument code.
- is_ready, true, means that load was successful.
  false, means that load was not ready yet.
- ret_msg, means prompt message.

4. Value Table

1.1 Buy/Sell Type

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Buy</td>
<td>‘B’</td>
</tr>
<tr>
<td>Sell</td>
<td>‘S’</td>
</tr>
</tbody>
</table>

1.2 Stop Order Type

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Limit Stop</td>
<td>‘L’</td>
</tr>
<tr>
<td>Up Trigger</td>
<td>‘U’</td>
</tr>
<tr>
<td>Down Trigger</td>
<td>‘D’</td>
</tr>
</tbody>
</table>

1.3 AO Price

| AO Price | ((long)0x7fffffff) |

1.4 Order Type

<table>
<thead>
<tr>
<th>Order Type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Limit Order Type</td>
<td>0</td>
</tr>
<tr>
<td>AO Order Type</td>
<td>2</td>
</tr>
<tr>
<td>Market Order Type</td>
<td>6</td>
</tr>
</tbody>
</table>
If OrderType = 2, please reference 4.3 to set price value.
If OrderType = 6, please set price value to 0.

### 1.5 Order Conditional Type

<table>
<thead>
<tr>
<th>None</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop Order (Point)</td>
<td>1</td>
</tr>
<tr>
<td>Schedule Time</td>
<td>3</td>
</tr>
<tr>
<td>OCO Stop (Point)</td>
<td>4</td>
</tr>
<tr>
<td>Trailing Stop (Point)</td>
<td>6</td>
</tr>
<tr>
<td>Combo (Open)</td>
<td>8</td>
</tr>
<tr>
<td>Combo (Close)</td>
<td>9</td>
</tr>
<tr>
<td>*Stop Order(Price)</td>
<td>11</td>
</tr>
<tr>
<td>*OCO Stop (Price)</td>
<td>14</td>
</tr>
<tr>
<td>*Trailing Stop (Price)</td>
<td>16</td>
</tr>
</tbody>
</table>

### 1.6 Validity

<table>
<thead>
<tr>
<th>DAY</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAK (Fill and Kill)</td>
<td>1</td>
</tr>
<tr>
<td>FOK (Fill or Kill)</td>
<td>2</td>
</tr>
<tr>
<td>GTC (Good-till-cancel)</td>
<td>3</td>
</tr>
<tr>
<td>GTD (Good-till-date)</td>
<td>4</td>
</tr>
</tbody>
</table>

### 1.7 Order Action

<table>
<thead>
<tr>
<th>Add Order</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Order</td>
<td>2</td>
</tr>
<tr>
<td>Delete Order</td>
<td>3</td>
</tr>
</tbody>
</table>
1.8 Order Status

<table>
<thead>
<tr>
<th>Status</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sending</td>
<td>0</td>
</tr>
<tr>
<td>Working</td>
<td>1</td>
</tr>
<tr>
<td>Inactive</td>
<td>2</td>
</tr>
<tr>
<td>Pending</td>
<td>3</td>
</tr>
<tr>
<td>Adding</td>
<td>4</td>
</tr>
<tr>
<td>Changing</td>
<td>5</td>
</tr>
<tr>
<td>Deleting</td>
<td>6</td>
</tr>
<tr>
<td>Inactivating</td>
<td>7</td>
</tr>
<tr>
<td>Partial Traded and Working</td>
<td>8</td>
</tr>
<tr>
<td>Traded</td>
<td>9</td>
</tr>
<tr>
<td>Deleted</td>
<td>10</td>
</tr>
<tr>
<td>Wait Approval</td>
<td>18</td>
</tr>
<tr>
<td>Traded and Reported</td>
<td>20</td>
</tr>
<tr>
<td>Deleted and Reported</td>
<td>21</td>
</tr>
<tr>
<td>Resync. (Unknown)</td>
<td>24</td>
</tr>
<tr>
<td>Partial Traded and Deleted</td>
<td>28</td>
</tr>
<tr>
<td>Partial Traded and Deleted and Reported</td>
<td>29</td>
</tr>
<tr>
<td>Exchange inactive</td>
<td>30</td>
</tr>
</tbody>
</table>

1.9 Ticker DealSrc

<table>
<thead>
<tr>
<th>DealSrc</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS_NORMAL(auto matching)</td>
<td>1</td>
</tr>
<tr>
<td>DS_CROSS(crossing)</td>
<td>5</td>
</tr>
<tr>
<td>DS_STDC(standard combo)</td>
<td>7</td>
</tr>
<tr>
<td>DS_AUC (auction)</td>
<td>20</td>
</tr>
<tr>
<td>DS_COMBO(combo match with out-right)</td>
<td>43</td>
</tr>
</tbody>
</table>

1.10 Combo Open/Closes Action
<table>
<thead>
<tr>
<th>SchedTime</th>
<th>DownPrice</th>
<th>DownLevel</th>
<th>UpPrice</th>
<th>UpLevel</th>
<th>StopPrice</th>
<th>Price</th>
<th>BuySell</th>
<th>StopType</th>
<th>CondType</th>
<th>OrderType</th>
</tr>
</thead>
<tbody>
<tr>
<td>long</td>
<td>long</td>
<td>long</td>
<td>long</td>
<td>long</td>
<td>long</td>
<td>long</td>
<td>Char B/S</td>
<td>Char 0-1</td>
<td>no. (0-1)</td>
<td>no. (0-1)</td>
</tr>
</tbody>
</table>

**Stop Level** = TradePrice - CloseLossDelta

**Stop Price** = TradePrice - CloseLossDelta - CloseLossTol

**Out Time** = CloseLossDelta

**Close Loss Tolerance** = CloseLossDelta

**Close Trailing Step** = CloseLossDelta
<table>
<thead>
<tr>
<th>SendTime</th>
<th>OutTime</th>
<th>SendTime</th>
<th>OutTime</th>
<th>SendTime</th>
<th>OutTime</th>
<th>SendTime</th>
<th>OutTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>CloseLoss Tol</td>
<td>ClosePrice</td>
<td>CloseLoss Tol</td>
<td>ClosePrice</td>
<td>CloseLoss Tol</td>
<td>ClosePrice</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>CloseLoss Tol</td>
<td>ClosePrice</td>
<td>CloseLoss Tol</td>
<td>ClosePrice</td>
<td>CloseLoss Tol</td>
<td>ClosePrice</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>CloseLoss Tol</td>
<td>ClosePrice</td>
<td>CloseLoss Tol</td>
<td>ClosePrice</td>
<td>CloseLoss Tol</td>
<td>ClosePrice</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Combo Close (Trail)**

SendTime 0/OutTime 0

**Combo Open (OCO)**

SendTime 0/OutTime 0

**Combo Open (Time)**

SendTime 0/OutTime 0

**Combo Close (Time)**

SendTime 0/OutTime 0

---

*Sharp Point Limited*