OMS Application Programming Interface User Manual Version 1.0.0.19

Last Updated: JANUARY 2nd, 2019

File version	Update time	Remark
V1.0.0.0	2016-04-11	Initialize version
V1.0.0.1	2016-04-27	1. Support multi-currencies trading 2. Support all ITS mode: Account, User, Server, SSO
V1.0.0.2	2016-05-10	Update API function, remove SetSynchronizeMode function
V1.0.0.3	2016-05-18	Change API function return value.
V1.0.0.4	2016-06-02	Add API Connection status response.
V1.0.0.5	2016-07-05	Support single currency mode
V1.0.0.6	2016-08-03	Add API Order operator response function, Add RegisterAppId function, RegisterOperatorFlag function
V1.0.0.7	2016-09-26	1. Add ReqVerifySession function, use by client send verify session session to keep session 2. Add ReqOrderCalculate function 3. Add ReqOrderHistoryOrder function 4. Add SubscribeCalculate function 5. Change ReqLogin function Paramter 6. Change ReqPasswordUpdate functipn parameter 7. Change ReqOrderInsert function parameter 8. Change ReqOrderAction function parameter 9. Add OnRspGenericError function 10. Add OnSessionTerminate function 11. Remove OnRspOrderInsertOrAction function 12. Remove OnLogoutTrigger function 13. Add OnRspOrderCalculateUpdate function
V1.0.0.8	2016-10-06	1. remove ReqLoginField's Devicename and MachineID, add AppID Field. 2. remove RegisterAppID function, instead set AppID in ReqLoginField 3. add WebTradingLimit, BodWebLimit in AccountBalanceField 4. Update ReqQryHistoryOrder function parameter 5. Add ReqOrderCalculate description
V1.0.0.9	2016-10-24	1. Add TradeDate field in OrderField for ReqQryHistoryOrder function 2. Change LoginID in ReqVerifySession to optional
V1.0.0.10	2016-11-22	Add Error Code for Order
V1.0.0.11	2016-12-27	Add OrderVersion(tag#1109) in OrderField Add OrderCommand(tag#187) in OrderField Add CounterParty(tag#45) in TradeField

V1.0.0.12	2017-01-10	Add GTC&GTD order support 1. add GTDDate(tag#495) in OrderInputField 2. add GTDDate(tag#495) in OrderField 3. add new struct GTCInfo(tag#1304) in OrderField 4. add GTDDate(tag#495) in OrderChargesField 5. add new struct GTCInfo(tag#1304) in OrderChargesField 6. add OrderType(tag#40) in OrderChargesField 7. add GTDDate(tag#495) in OrderActionField
V1.0.0.13	2017-04-18	Add order status remark of StatusType in Appendix
V1.0.0.14	2017-09-21	Support 2FA 1. Add LocalIP, DeviceOS, Location, Browser in ReqLoginField. 2. Add DeviceOS, Location, ClientIP, AppVersion, AppID, LocalIP, Browser in ReqVerifyTokenField
V1.0.0.15	2018-05-11	2018. New api support: EnableEncryption SetQueryTimeout ReqQryTradeByUser ReqSubOrderByUser ReqBodPositionByUser 2. function ReqOrderInsert support param BCAN.
V1.0.0.16	2018-05-12	1. New api support ReqAttribute
V1.0.0.17	2018-06-07	1. New api support ReqAccountBCAN
V1.0.0.18	2018-11-13	1. New api support RegisterDesk
V1.0.0.19	2019-1-2	1. Add TradeType(tag#43) in TradeField

Contents

1. Introduction

This document specifies the details of communication interface between Order Management System (OMS) and remote applications. It defines the API functions and procedure on different operation a client application can perform on OMS. It also provides client application developer a general concept on the system architecture.

The intended reader of this document is the technical personnel of the application development party. It is assumed that the reader has sufficient business knowledge on stock market and trading system.

OMS API is one open library basing on C++; third party can implement all trading function VIA using or extending the library. OMS API allows customers' applications to commutate with EBROKER OMS system and provides customers with trading services.

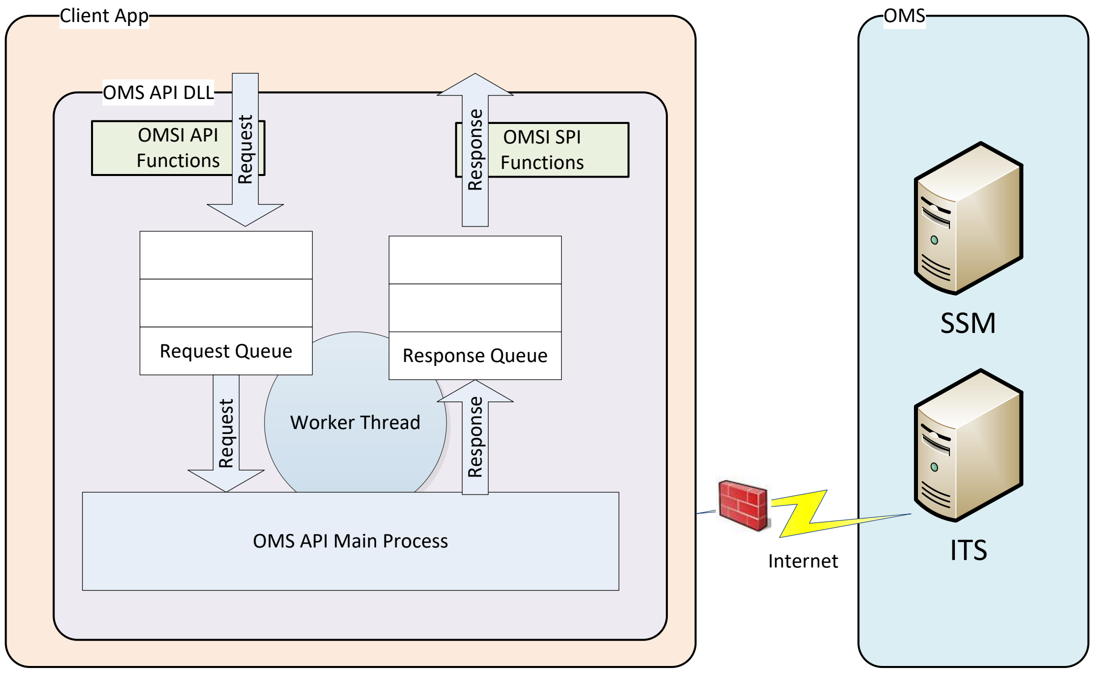
The whole package include below 5 files:

File Name	Version	File Description
OMSAPI.h	V1.0.0.22	Header file of API
OMSAPIDataType.h	V1.0.0.22	Header file for common data type definition
OMSAPIStruct.h	V1.0.0.22	Header file for business data type definition
OMSAPI.dll	V1.0.0.22	The main DLL
OMSAPI.lib	V1.0.0.22	Link library

*Currently only support Windows platform (Window 7 or above)
*Complied with MS VS 2008.

2. Overview of OMS API

2.1 System Overview



- There are two main public class in OMS API DLL, one is for request actions, and the other is for handling the response callback, Request Class(OMSAPI) can be used once create an instance, while Response Class(OMSSPI) should be inherited and implemented the callback functions
- All the request function calls are **thread-safe**; and the call back functions will be called from one same sub-thread that called Worker Thread.
- Worker Thread will process items in Request Queue, and send the request to corresponding OMS Servers via **TCP** socket.
- Worker Thread will process items in Response Queue and response to client program with OMSSPI function call directly.
- All the TCP connections with OMS have auto-reconnect mechanism; client program doesn’t need to handle it.

2.2 Working Mechanism

- Session Layer

  Client program should follow below steps to establish session with OMS.

Step	Client Program Action	API Function

1	Create OMSAPI instance by CreateOMSAPI	**OMSAPI :: CreateOMSAPI**
2	Inherit a class from OMSSPI and implement the response functions, and then create OMSSPI instance	**OMSSPI**
3	Register OMSSPI, Operator Flag and Desk	**RegisterSpi**, **RegisterOperatorFlag, RegisterDesk**
4(Optional)	Set log path by OpenLogFile and OpenDumpLog	**OpenLogFile**, **OpenDumpLog**
5	Connect to OMS System with connection string	**Connect**
6	Subscribe Order, Error-Order, Sub-Order, Trade	**SubscribeOrder**, **SubscribeErrorOrder**, **SubscribeSubOrder**, **SubscribeTrade**
7	Login to OMS System	**ReqLogin**

In normal mode, it only supports ONE login session (either account or AE) with ONE OMSAPI instance. If client program requires multiple login session, it should create multiple OMSAPI instance.
In server mode, OMSAPI can support multiple account login session for within ONE OMSAPI instance.
Multiple AE login with ONE OMSAPI instance is not supported in normal mode or server mode.
Developer should contact eBroker to clarify which mode is deployed in OMSAPI.

- Business Layer
  After establish session, client program can use different API calls according to different business needs.

2.3 Session Maintenance

A login session will expire if there is no verification operation on certain period of time (default is 15 minutes). It is an important duty for a client application to keep the login session active. OMSAPI provide a **ReqVerifySession** API to allow client application keep login session active. The timer will be renewed if a verification message of that particular session is received before it is expired.

3. OMS API Specification

3.1 OMS API Function

This section describes the request functions, including synchronous functions and asynchronous functions, if function name begins with "**Subscribe**", it indicates the function is asynchronous, else the function is synchronous.

3.1.1 CreateOMSAPI

Purpose:	create OMSAPI instance.
Function:	static OMSAPI* CreateOMSAPI ()
Output:	return OMSAPI's instance.

3.1.2 ReleaseOMSAPI

Purpose:	release OMSAPI instance. Client should be call this function when client detach from API
Function:	public void ReleaseOMSAPI ()
Remark:	release OMSAPI's instance.

3.1.3 RegisterSpi

Purpose:	register an OMSSPI's instance.
Function:	void RegisterSpi(OMSSPI *pSPI)
Parameter:	set OMSSPI's instance.

3.1.4 RegisterOperatorFlag

Purpose:	register client's operator flag.
Function:	void RegisterOperatorFlag (const char* pszOperatorFlag)
Parameter:	set client's operator flag.

3.1.5 RegisterDesk

Purpose:	register client’s desk.
Function:	void RegisterDesk (const char* pszDesk)
Parameter:	set client’s desk.

3.1.6 OpenLogFile

Purpose:	Open log file.
Function:	void OpenLogFile(const char* pszFileName)
Parameter:	log’s filename.
Remark:	Log File will reset by daily.

3.1.7 OpenDumpLog

Purpose:	Open dump log.
Function:	void OpenDumpLog(const char* pszFileName)
Parameter:	dump log’s filename.
Remark:	Log File will reset by daily.

3.1.8 EnableEncryption

Purpose:	support security socket.
Function:	public void EnableEncryption(ValueType Encryption)
Parameter:	Encryption :true or false.
Remark:	the API will make the socket encryption or not, default not encryption.it should be call before call connect API.SetQueryTimeout

3.1.9 SetQueryTimeout

Purpose:	set request timeout ,default is 5s.
Function:	public void SetQueryTimeout(int timeout);
Parameter:	timeout, values means milliseconds

3.1.10 Connect

Purpose: set API's IP and Port, and then connect API.

Function: int Connect (char* pszAddress);

Parameter: pszAddress: indicate API's IP and Port.

Remark: pszAddress' format: "ipaddress:port[:ipaddress:port]"

*eBroker will provide the address string.

Output: return value: **0**: means connect success. **-1**: Not connect API

3.1.11 ReqLogin

Purpose: user send login request.

Function: int ReqLogin(const ReqLoginField& reqLoginInfo,
RspLoginField& rspLoginInfo)

Parameter: reqLoginInfo: request login info

rspLoginInfo: response login info

Field summary of ReqLoginField:

Field	**Format**	**Req's**	**Remark**
LoginID	LoginIdType	Yes	Login System ID
Password	PasswordType	Yes	Login Password
AppID	AppNameType	Yes	Client Application ID
AppVersion	AppVersionType	No	Client Application Version
ClientIP	ClientIPType	Yes	Public IP address of the client. Need to provide Internet IP instead of LAN IP since SFC required, unless Gateway keeps record on their own.
LocalIP	LocalIPTYpe	NO	Local IP address of the client. Need to provide Internet IP instead of LAN IP since SFC required, unless Gateway keeps

			record on their own.
DeviceOS	DeviceOSType	NO	E.g.Windows7
Location	LocationType	NO	Location resolved by IP address
Browser	BrowserType	NO	Web Brower name, version info, depends on B/S mode

Field summary of RspLoginField:

Field	Format	Remark
SessionID	SessionType	Unique Session ID
LoginID	LoginIdType	user identity
LoginType	UserType	Login type
PinPos	PinPosType	Random PIN number positions (optional),separated by comma
LastLoginStatus	LoginStatusType	last logon status -1=false, 0=success
LastLoginAppName	ClientNameType	last logon application name
LastLoginTime	DateTimeType	last logon time, in format"yyyy-MM-dd HH:mm:ss";
PwdExpireDate	DateTimeType	password expiry date time, in format "yyyy/MM/dd HH:mm:ss"
SvrCurrentTime	DateTimeType	Server side date time, in format “yyyy/MM/dd HH:mm:ss”
SessionKey	SessionType	The session key of the logon session, used to identify the session of the order
ErrorCode	ErrorCodeType	Error Code
ErrorMessage	ErrorMsgType	Error Message
NeedTradePassword	bool	Whether the trade password is required for order actions.
SubAccountList	AccountList	Sub-account, use by SSO mode
NotifyAddress	NotifyAddressType	OTP notify email address or

		phone number
OTPRsndTimeout	TimeIntervalType	Use by ResendOTP interval time (s)
TokenType	TokenTypeType	Token verify type: 0: single password 1: PIN 2: OTP for Email 3: OTP for SMS 4: Software token 5: eBrokerKey

Output: return value: **0**: means success, **-1**: Not connect API,,**-3**:query timeout,else error code please refer **Appendix 4.1**

3.1.12 ReqVerifySession

Purpose: when client register auto verify is false, client should be call this function to keep heartbeat.

Function: int ReqVerifySession (const ReqVerifyField& reqVerifyInfo)

Parameter: reqVerifyInfo: indicate verify record.

Field summary of ReqVerifyField:

Field	Format	Req's	Remark
LoginID	LoginIdType	No	user identity
SessionID	SessionType	Yes	Unique Session ID

Output: return value: **0**: means sent message to system success. **-1**: Not connect API

3.1.13 ReqLogout

Purpose: user send logout request.

Function: int ReqLogout(const char* loginId)

Parameter: loginId: indicate Customer's Id.

Output: return value: **0**: means success. **-1**: Not connect API, else error code please Refer **Appendix 4.1**

3.1.14 ReqPasswordUpdate

Purpose: user change password request.

Function: int ReqPasswordUpdate(const ReqPasswordUpdateField& reqPasswordInfo, RspStatusField& info)

Parameter: reqPasswordInfo: request updates password info

Info: response updates password info

Field summary of ReqPasswordUpdateField:

Field	Format	Req's	Remark
LoginID	LoginIdType	Yes	Login System ID
OldPassword	PasswordType	Yes	Old password
LoginType	UserType	No	loginId Type: 0=AE, 1=Account , optinal
NewPassword	PasswordType	Yes	New password

Field summary of RspStatusField:

Field	Format	Remark
ErrorID	ErrorCodeType	Error Code
ErrorMsg	ErrorMsgType	Error Message

Output: return value: **0**: means success. **-1**: Not connect API, else error code please Refer **Appendix 4.1**

3.1.15 ReqOrderInsert

Purpose: user send new order request.

Function: int ReqOrderInsert(const OrderInputField& reqOrderInfo);

Parameter: reqOrderInfo: indicate input order info.

* client can define "ClientLocalID" for reference query

Field summary of OrderInputField:

Field	Format	Req's	Remark
Symbol	SymbolType	Yes	Stock symbol
Price	PriceType	Yes	Limited price
Quantity	VolumeType	Yes	Quantity
AccountID	AccountType	Yes	Trade account ID

BuySell	BuySellType	Yes	Order buy sell direction: 0=Buy, 1=Sell
OrderType	OrderTypeType	Yes	Order type: 0 = limit order 3 = odd lot order 128 = Fill And Kill 256 = Fill Or Kill 512=Auction Order 2048=Stop Limit Order
UserID	UserNoType	No	The user ID placing this order. (not mandatory when login type is 1, and mandatory when login type is 0)
ClientReference	ClientLocalIDType	No	Free text used for client's reference (optional)
TradingPassword	PasswordType	Yes	Trading password. (mandatory when ReqLogin return value's NeedTradePassword is True)
ShortSellingFlag	ShortSellType	No	short selling flag: 1=Yes, 0=No
BasketID	OrderNoType	No	Basket or bulk parent order ID
StopPrice	PriceType	No	stop price
GTDDate	DateTimeType	No	Required for GTD order, used to specified the expiry date, in format "yyyy/MM/dd"
Currency	CurrencyType	NO	Symbol currency
OpenClose	OpenCloseType	NO	1=Open,2=Close, 3=ForceClose

BCAN	BCANType	No	Required if placing A-Share order, numeric data with range 1 ~ 9999999999

Output: return value: **0**: means sent message to system success. **-1**: Not connect API

Remark: If order accepted by OMS, order will update through **OMSSPI**::**OnRspOrderUpdate.** If order rejected by OMS, error order will update through **OMSSPI**::**OnRspErrorOrderUpdate.**

Depending on account setting, order actions might require a trading password for verification. Fail to provide a required trading password would result in operation denied. Client program should check the value of NeedTradePassword in RspLoginField to detect whether the password is required

3.1.16 ReqOrderAction

Purpose: user send order-operation request, include change order, cancel order.

Function: int ReqOrderAction(const OrderActionField& order);

Parameter: order: indicate order action info.

Field summary of OrderActionField:

Field	**Format**	**Req's**	**Remark**
Price	PriceType	No	Limited price (optional)
StopPrice	PriceType	No	Stop Price
Quantity	VolumeType	No	Quantity (optional)
OrderID	OrderNoType	Yes	Order number identifying order to change
AccountID	AccountType	Yes	Trade account ID
UserID	UserNoType	No	The user ID who change the order. (not mandatory when login type is 1, and mandatory when login type is 0)
ClientReference	ClientLocalIDType	No	Free text used for client's reference (optional)

OrderAction	OrderOperation	Yes	OrderAction: ordChange, ordCancel
Password	PasswordType	Yes	Trading password, (mandatory when ReqLogin return value's NeedTradePassword is True)
AcitonReason	ErrorMsgType	No	Reason
GTDDate	DateTimeType	No	Required for GTD order, used to specified the expiry date, in format "yyyy/MM/dd"

Output: return value: **0**: means sent message to system success. **-1**: Not connect API

Remark: If order action accepted by OMS, order will update through **OMSSPI**::**OnRspOrderUpdate.** If order rejected by OMS, error order will update through **OMSSPI**::**OnRspErrorOrderUpdate.**

Depending on account setting, order actions might require a trading password for verification. Fail to provide a required trading password would result in operation denied. Client program should check the value of NeedTradePassword in RspLoginField to detect whether the password is required.

3.1.17 ReqOrderCalculate

Purpose: pre-calculate the charges if placing an order with given price and quantity.

Function: public int ReqOrderCalculate (const OrderInputField & reqChargesInfo)

Parameter: reqChargesInfo: indicate input order info.

Field summary of OrderInputField:

Field	**Format**	**Req's**	**Remark**
Symbol	SymbolType	Yes	Stock symbol
Price	PriceType	Yes	Limited price
Quantity	VolumeType	Yes	Quantity
AccountID	AccountType	Yes	Trade account ID
BuySell	BuySellType	Yes	Order buy sell direction: 0=Buy,

			1=Sell
OrderType	OrderTypeType	Yes	Order type: 0 = limit order 3 = odd lot order 128 = Fill And Kill 256 = Fill Or Kill 512=Auction Order 2048=Stop Limit Order
UserID	UserNoType	No	The user ID placing this order. (not mandatory when login type is 1, and mandatory when login type is 0)
Currency	CurrencyType	No	Symbol currency
ClientReference	ClientLocalIDType	No	Free text used for client's reference (optional)
TradingPassword	PasswordType	Yes	Trading password. (mandatory when ReqLogin return value's NeedTradePassword is True)
GTDDate	DateTimeType	No	Required for GTD order, used to specified the expiry date, in format "yyyy/MM/dd"
Currency	CurrencyType	NO	Symbol currency
OpenClose	OpenCloseType	NO	1=Open,2=Close, 3=ForceClose

Output: return value: <u>**0**</u>: means sent message to system success, <u>**-1**</u>: Not connect API.

Remark: Please note that the calculation won't hold fund for the trading account, but just shows the charges. Order calculate result will update through **OMSSPI**:: **OnRspOrderCalculateUpdate.**

Depending on account setting, calculate actions might require a trading password for verification. Fail to provide a required trading password would result in operation denied. Client program should check the value of NeedTradePassword in RspLoginField to detect whether the password is required.

3.1.18 ReqQryHistoryOrder

Purpose: query history order request

Function: int ReqQryHistoryOrder(const ReqOrderHistoryField& reqOrderHistory, OrderList& orders);

Parameter: reqOrderHistory: request history field.

orders: query's result.

Field summary of ReqOrderHistoryField:

Field	**Format**	**Remark**
AccountID	AccountType	Trade Account ID
StartDate	DateTimeType	Query Start Date, format: yyyy/MM/dd
EndDate	DateTimeType	Query End Date, format: yyyy/MM/dd
ClientReference	ClientLocalIDType	Free text used for client's reference (optional)

Field summary of OrderField returned by ReqQryHistoryOrder:

Field	**Format**	**Remark**
Symbol	SymbolType	stock symbol
Price	PriceType	limited price
Quantity	VolumeType	Quantity
OrderStatus	StatusType	refer Order status Type
OrderID	OrderNoType	Order number (identity of a particular order)
AccountID	AccountType	Trade account ID
BuySell	BuySellType	order buy sell direction:0=buy, 1=sell
AEUserID	UserNoType	corresponding AE account ID

OrderingTime	DateTimeType	ordering time
TradeDate	DateTimeType	Order date, format, yyyy/MM/dd, only used by this function
FillQuantity	VolumeType	Filled quantity
OperatorFlag	OperatorFlagType	Operator flag indicating where the order entry from, must assigned with eBroker agree
OrderType	OrderTypeType	limit order or stop order
ExchangeID	ExchangeIDType	name of exchange
ErrorMessage	ErrorMsgType	error message
ErrorCode	ErrorCodeType	error code
ClientReference	ClientLocalIDType	free text comments on this order entry
AvgPrice	PriceType	verage price
AccountCredit	PriceType	account credit
QueueQuantity	VolumeType	Queue quantity
WorkingQuantity	VolumeType	Working quantity
MinorCode	ErrorCodeType	Minor code
OrderCreateTime	DateTimeType	Order creation time
TradeDate	DateTimeType	Order date
ResourceID	ResourceNoType	Resource ID
OpenClose	OpenCloseType	1=Open,2=Close, 3=ForceClose
OrderCommand	CommandType	Order Command
OrderVersion	VersionType	Order Version
GTDDate	DateTimeType	Required for GTD order, used to specified the expiry date, in format "yyyy/MM/dd"
GTCInfo	GTCInfoField	GTC Info field in format <Order-Create-Date>,<Original Price>,<Order-Quantity>,<Pre Filled Qty >,<Pre Filled Amt >

Output: return value: **0**: means success. **-1**: Not connect API, else error code please

Refer **Appendix 4.1**

3.1.19 ReqQrySubOrder

Purpose: query Sub-Order request

Function: int ReqQrySubOrder (const char* AccountID, SubOrderList& subOrders);

Parameter: AccountID: account id.

subOrders: query's result.

Field summary of SubOrderList:

Field	Format	Remark
SubOrderCount	Int	Sub-Order count
SubOrders	SubOrderField*	Sub-Order of array

Field summary of SubOrderField:

Field	Format	Remark
Symbol	SymbolType	stock symbol;
Price	PriceType	limit price
Quantity	VolumeType	Quantity
SubOrderStatus	StatusType	refer satus type
OrderID	OrderNoType	Order number (identity of a particular order)
SubOrderID	OrderNoType	Sub-Order number
AccountID	AccountType	trade account ID
BuySell	BuySellType	order buy sell direction:0=buy, 1=sell
UserID	UserNoType	AE-UserID
ExchangeID	ExchangeIDType	name of exchange
DestExchangeID	ExchangeIDType	Destination exchange ID
OrderingTime	DateTimeType	orderint time
OrderCreateTime	DateTimeType	Order creation time
ClientReference	ClientLocalIDType	Client Use Reference
OpenClose	OpenCloseType	1=Open, 2=Close, 3=ForceClose

Output: return value: **0**: means success. **-1**: Not connect API, else error code please Refer **Appendix 4.1**

3.1.20 ReqQryTrade

Purpose: query trade request.

Function: int ReqQryTrade(const char* account, TradeList& trades);

Parameter: account: account id

trades: query's result.

Field summary of TradeList:

Field	Format	Remark
TradeCount	Int	Trade's count
Trade	TradeField*	Trade of array

Field summary of TradeField:

Field	Format	Remark
Symbol	SymbolType	stock symbol;
Price	PriceType	limit price
Quantity	VolumeType	Quantity
OrderStatus	StatusType	refer satus type
SubOrderID	OrderNoType	Sub-Order number
TradeID	TradeNoType	Trade number
AccountID	AccountType	trade account ID
BuySell	BuySellType	order buy sell direction:0=buy, 1=sell
UserID	UserNoType	AE-UserID
ExchangeID	ExchangeIDType	name of exchange
TradingTime	DateTimeType	trade time
OpenClose	OpenCloseType	1=Open, 2=Close, 3=ForceClose
CounterParty	BrokerType	Counter Party
ClientReference	ClientLocalIDType	Client Use reference
TradeType	TradeTypeType	Trade Type OddLot=1,Manual=2, Preopen=3, Oversea=4

Output: return value: 0: means success. -1: Not connect API, else error code please Refer **Appendix 4.1**

3.1.21 ReqQryPosition

Purpose: query position request.

Function: int ReqQryPosition(const char* account, PositionList& positions);

Parameter: account: indicate query's account id.

positions: query's result.

Field summary of PositionList:

Field	Format	Remark
SymbolCount	Int	Position's count
Position	PositionField*	Position of array

Field summary of PositionField:

Field	Format	Remark
Account	AccountType	Trade account ID
Symbol	SymbolType	Stock symbol
Amount	AmountType	Amount (not to be used)
LongQty	VolumeType	Long Quantity
ShortQty	VolumeType	Short Quantity
AvgPrice	PriceType	Average Price

Output: return value: **0**: means success. **-1**: Not connect API, Else error code please Refer **Appendix 4.1**

3.1.22 ReqQryBodPosition

Purpose: query Bod position request.

Function: int ReqQryPosition(const char* account, PositionList& positions);

Parameter: account: indicate query's account id.

positions: query's result.

Field summary of PositionList:

Field	Format	Remark
SymbolCount	Int	Position's count
Position	PositionField*	Position of array

Field summary of PositionField:

Field	Format	Remark
Account	AccountType	Trade account ID
Symbol	SymbolType	Stock symbol
Amount	AmountType	Amount (not to be used)
LongQty	VolumeType	Long Quantity
ShortQty	VolumeType	Short Quantity
AvgPrice	PriceType	Average Price

Output: return value: **0**: means success. **-1**: Not connect API, else error code please Refer **Appendix 4.1**

3.1.23 ReqQryAccountInfo

Purpose: query account info request.

Function: int ReqQryAccountInfo(const char* account, AccountField& accountInfo);

Parameter: account: indicate query's account id.

accountInfo: query's result.

Field summary of AccountField:

Field	Format	Remark
Account	AccountType	Trade account ID
AccountName	AcctNameType	Trade account Name
Margin_Type	AcctMarginType	Account Margin Type
Client_Class	AcctClientType	Account Client Type

Output: return value: **0**: means success. **-1**: Not connect API, else error code please Refer **Appendix 4.1**

3.1.24 ReqQryAccountBalanceInfo

Purpose: query account balance info request.

Function: int ReqQryAccountBalanceInfo(const char* AccountID, AccountBalanceField& rspBalanceInfo);

Parameter: AccountID: indicate query's account id.

rspBalanceInfo: query's result.

Field summary of AccountBalanceField:

Field	Format	Remark
AccountID	AccountType	Account ID
Currency	CurrencyType	Currency ID, use by multi-Currency mode
TradingLimit	AmountType	Account Trading Limit
CashBalance	AmountType	Account Balance
BodTradingLimit	AmountType	Account Begin of day trading limit
BodCashBalance	AmountType	Account begin of day cash balance
InitialMargin	AmountType	Initial Margin
MainMargin	AmountType	Maintenance margin
PNL	AmountType	P&L
MarginCall	AmountType	MarginCall / Margin call
WebTradingLimit	AmountType	Web Tradeing Limit
BodWebLimit	AmountType	Account begin of day web tradeing limit

Output: return value: **0**: means success. **-1**: Not connect API, else error code please Refer **Appendix 4.1**

Remark: If this request accepted by OMS, account balance will update through **OMSSPI**:: **OnRspAccountBalanceInfoUpdate.**

3.1.25 ReqQryCurrencyBalance

Purpose: query account balance info request, by currency.

This function only working in multi-currency mode

Function: int ReqQryCurrencyBalance (const char* account, CurrencyBalanceList & currencyBalance);

Parameter: account: indicate query's account id.

currencyBalance: query's result.

Field summary of CurrencyBalanceList:

Field	Format	Remark
CurrencyCount	Int	Account-balance's count
AccountBalance	AccountBalanceField*	Account-Balance of array

Field summary if AccountBalanceField:

Field	Format	Remark
AccountID	AccountType	Account ID
Currency	CurrencyType	Currency ID, use by multi-Currency mode
TradingLimit	AmountType	Account Trading Limit
CashBalance	AmountType	Account Balance
BodTradingLimit	AmountType	Account Begin of day trading limit
BodCashBalance	AmountType	Account begin of day cash balance
InitialMargin	AmountType	Initial Margin
MainMargin	AmountType	Maintenance margin
PNL	AmountType	P&L
MarginCall	AmountType	MarginCall / Margin call
WebTradingLimit	AmountType	Web Tradeing Limit
BodWebLimit	AmountType	Account begin of day web tradeing limit

Output: return value: **0**: means success.**-1**: Not connect API, else error code please Refer **Appendix 4.1**

Remark: If this request accepted by OMS, account balance will update through **OMSSPI**:: **OnRspCurrencyBalanceInfoUpdate.**

3.1.26 ReqQryUserAccounstInfo

Purpose: query AE user's account info request.
This function only work AE-mode

Function: int ReqQryUserAccountsInfo(const char* user, AccountList & accountInfo);

Parameter: user: indicate query's AE user id.
accountInfo: query's result.

Field summary of AccountList:

Field	Format	Remark
AccountCount	Int	Account's count
Account	AccountType*	Account of array

Output: return value: **0**: means success. **-1**: Not connect API, else error code please Refer **Appendix 4.1**

3.1.27 ReqQryMarginRatioInfo

Purpose: query account symbol Margin info request.

Function: int ReqQryMarginRatioInfo(const char* account, const char* symbol MarginRatioField& marginInfo);

Parameter: account: indicate query's account id.

symbol: indicate query's symbol id.

marginInfo: query's result.

Field summary of MarginRatioField:

Field	Format	Remark
Symbol	SymbolType	Stock symbol
MarginRatio	float	N/A

Output: return value: **0**: means success. **-1**: Not connect API, else error code please Refer **Appendix 4.1**

3.1.28 ReqQryCurrencyListInfo

Purpose: query currency info request.

Function: int ReqQryCurrencyListInfo(const char* account, CurrencyList& currenys);

Parameter: currencys: query's result.

Field summary of CurrencyList:

Field	Format	Remark
CurrencyCount	Int	Currency's count
Currency	CurrencyType*	Currency of array

Output: return value: **0**: means success. **-1**: Not connect API, else error code please Refer **Appendix 4.1**

3.1.29 ReqQryExchangeRate

Purpose: query Currency Ratio request.

Function: int ReqQryExchangeRate (const char* account, const char* currency, CurrencyRatioField & currencyRatios);

Parameter: account: account id

currency: indicate query's currency code, can query all currencies if it's empty

currencyRatios: query's result.

Field summary of CurrencyRatioField:

Field	**Format**	**Remark**
Currency	CurrencyType	Currency
Ratio	Float	N/A

Output: return value: **0**: means success. **-1**: Not connect API, else error code please Refer **Appendix 4.1**

3.1.30 ReqSubOrderByUser

Purpose: query suborder by user.

Function: int ReqSubOrderByUser(const char* userID, SubOrderList& subOrders);

Parameter: userID: user id

subOrders: query result.

Field summary of SubOrderList:

Field	**Format**	**Remark**
SubOrderCount	Int	SubOrderCount's count
SubOrders	SubOrderField *	SubOrder of array

Field summary of SubOrderField:

Field	**Format**	**Remark**
Symbol	SymbolType	stock symbol;
Price	PriceType	limit price
Quantity	VolumeType	Quantity
SubOrderStatus	StatusType	refer satus type
OrderID	OrderNoType	Order number (identity of a particular order)

SubOrderID	OrderNoType	Sub-Order number
AccountID	AccountType	trade account ID
BuySell	BuySellType	order buy sell direction:0=buy, 1=sell
UserID	UserNoType	AE-UserID
ExchangeID	ExchangeIDType	name of exchange
DestExchangeID	ExchangeIDType	Destination exchange ID
OrderingTime	DateTimeType	orderint time
OrderCreateTime	DateTimeType	Order creation time
ClientReference	ClientLocalIDType	Client Use Reference
OpenClose	OpenCloseType	1=Open, 2=Close, 3=ForceClose

Output: return value: **0**: means success, **-1**: Not connect API,**,-3**:query timeout,else error code please refer **Appendix 4.1**

3.1.31 ReqQryTradeByUser

Purpose: query trade by user.

Function: int ReqQryTradeByUser(const char* userID, TradeList& trades);

Parameter: userID: user id

trades: query result.

Field summary of TradeList:

Field	Format	Remark
TradeCount	Int	Trade's count
Trade	TradeField*	Trade of array

Field summary of TradeField:

Field	Format	Remark
Symbol	SymbolType	stock symbol;
Price	PriceType	limit price
Quantity	VolumeType	Quantity
OrderStatus	StatusType	refer satus type
SubOrderID	OrderNoType	Sub-Order number
TradeID	TradeNoType	Trade number

AccountID	AccountType	trade account ID
BuySell	BuySellType	order buy sell direction:0=buy, 1=sell
UserID	UserNoType	AE-UserID
ExchangeID	ExchangeIDType	name of exchange
TradingTime	DateTimeType	trade time
OpenClose	OpenCloseType	1=Open, 2=Close, 3=ForceClose
CounterParty	BrokerType	Counter Party
ClientReference	ClientLocalIDType	Client Use reference
TradeType	TradeTypeType	Trade Type OddLot=1,Manual=2, Preopen=3, Oversea=4

Output: return value: **0**: means success, **-1**: Not connect API,,**-3**:query timeout,else error code please refer **Appendix 4.1**

3.1.32 ReqBodPositionByUser

Purpose: query bod position by user.

Function: int ReqSubOrderByUser(const char* userID, SubOrderList& subOrders);

Parameter: userID: user id

subOrders: query result.

Field summary of PositionList:

Field	Format	Remark
SymbolCount	Int	Position's count
Position	PositionField*	Position of array

Field summary of PositionField:

Field	Format	Remark
Account	AccountType	Trade account ID
Symbol	SymbolType	Stock symbol
Amount	AmountType	Amount (not to be used)
LongQty	VolumeType	Long Quantity
ShortQty	VolumeType	Short Quantity
AvgPrice	PriceType	Average Price

Output: return value: **0**: means success, **-1**: Not connect API,,**-3**:query timeout,else error code please refer **Appendix 4.1**

3.1.33 ReqAttribute

Purpose: update attribute status

Function: int ReqAttribute(const AttributeField& reqAttributeInfo, AttributeField& rspAttributeInfo);

Parameter: reqAttributeInfo: indicate request attribute info

rspAttributeInfo: indicate respond attribute info

ield summary of AttributeField

Field	Format	Remark
AccountID	AccountType	Trade account ID
Symbol	SymbolType	Stock symbol Must be: AccountAttribute
AttributeType	AttributeTypeType	Attribute Type Must be: NT
ObjectType	ObjectTypeType	Object Type Must be: ACCT
AttributeID	AttributeIDType	Attribute ID Must be: OptOutTradeNotify
AttributeValue	AttributeValueType	Attribute value Must be: NONE
TimeStamp	DateTimeType	Request time
ActionType	ActionTypeType	0 for delete/1 for add or update

The request message should be:

AttributeType	AttributeID	CategoryID	ObjectType	ValueType	Symbol
NT	OptOutTradeNotify	ACCTENTITLE	ACCT	NONE	AccountAttribute

Output: return value: **0**: means success, **-1**: Not connect API,,**-3**:query timeout,else error code please refer **Appendix 4.1**

3.1.34 ReqAccountBCAN

Purpose: query account BCAN

Function: int ReqQryAccountBCAN (const char* account,AccountBCANList& BCANS);

Parameter: AccountID: account id

BCANS:BCAN list

Output: return value: **0**: means success, **-1**: Not connect API,**,-3**:query timeout,else error code please refer **Appendix 4.1**

3.1.35 SubscribeOrder

Purpose: subscribe order info request.

Function: int SubscribeOrder();

Output: return value: **0**: means sent message to system success. **-1**: Not connect API

Remark: client should be calling this function after connect success, else the order update will not response. If called this function, order will update through **OMSSPI**:: **OnRspOrderUpdate.**

3.1.36 SubscribeSubOrder

Purpose: subscribe Sub-Order info request.

Function: int SubscribeSubOrder ();

Output: return value: **0**: means sent message to system success.**-1**: Not connect API

Remark: client should be calling this function after connect success, else the sub-order update will not response. If called this function, sub-order will update through **OMSSPI**:: **OnRspSubOrderUpdate.**

3.1.37 SubscribeTrade

Purpose: subscribe trade info request.

Function: int SubscribeTrade();

Output: return value: **0**: means sent message to system success.**-1**: Not connect API

Remark: client should be calling this function after connect success, else the trade update will not response. If called this function, trade will update through **OMSSPI**:: **OnRspTradeUpdate**

3.1.38 SubscribeErrorOrder

Purpose: subscribe ErrorOrder info request.

Function: int SubscribeErrorOrder ();

Output: return value: **0**: means sent message to system success.**-1**: Not connect API

Remark: client should be calling this function after connect success, else the error order update will not response. If called this function, error order will update through **OMSSPI**:: **OnRspErrorOrderUpdate.**

3.1.39 SubscribeCalculate

Purpose: subscribe calculate order info request.

Function: int SubscribeCalculate ();

Output: return value: **0**: means sent message to system success. **-1**: Not connect API

Remark: client should be calling this function after connect success, else the calculate order update will not response. If called this function, calculate order will update through **OMSSPI**:: **OnRspOrderCalculateUpdate**

3.2 OMS SPI Callback Function

This section describes the response call back basically raised by **Subscribe** functions in API functions. Except OnSessionTerminate and OnRspConnectionStatus, other SPI functions are required subscribe manually.

3.2.1 OnSessionTerminate

Purpose: response for client login session had been terminated.

Function: OnSessionTerminate(const char* account, RspStatusField& info);

Parameter: loginId: return logout' user or account id.

info: response status info

Remark: login session will be terminated in one of the following cases:

Cases	Error Code
User attempt to create another login session	50
Login session expired	51

Being kicked by system operator	52
Logout by client application	53

Client application should logout the session when it is no longer needed.

Filed summary RspStatusField:

Field	Format	Remark
ErrorID	ErrorCodeType	Error Code
ErrorMsg	ErrorMsgType	Error message

3.2.2 OnRspOrderUpdate

Purpose: response for client subscribe order request.

Function: void OnRspOrderUpdate(OrderField& order, RspStatusField & info);

Parameter: order: return user order info.

info: response status info.

Remark: after client call **SubscribeOrder** function; once there is order update, this function will be called.

* Only Order is GTC or GTD order, the GTCInfo field will be filled.

Field summary of OrderField:

Field	Format	Remark
Symbol	SymbolType	stock symbol
Price	PriceType	limited price
Quantity	VolumeType	Quantity
OrderStatus	StatusType	refer Order status Type
OrderID	OrderNoType	Order number (identity of a particular order)
AccountID	AccountType	Trade account ID
BuySell	BuySellType	Order buy sell direction:0=buy, 1=sell
OrderType	OrderTypeType	Order type: 0 = limit order 3 = odd lot order 128 = Fill And Kill 256 = Fill Or Kill

		512=Auction Order 2048=Stop Limit Order
AEUserID	UserNoType	corresponding AE account ID
ExchangeID	ExchangeIDType	name of exchange
ErrorMessage	ErrorMsgType	error message
ClientReference	ClientLocalIDType	free text comments on this order entry
OrderingTime	DateTimeType	ordering time
FillQuantity	VolumeType	Filled quantity
ErrorCode	ErrorCodeType	error code;
AvgPrice	PriceType	average price
AccountCredit	PriceType	account credit
QueueQuantity	VolumeType	Queue quantity;
OperatorFlag	OperatorFlagType	Operator flag indicating where the order entry from, must assigned with eBroker agree
WorkingQuantity	VolumeType	Working quantity
MinorCode	ErrorCodeType	Minor code (refer Appendix 4.2)
OrderCreateTime	DateTimeType	Order creation time
ResourceID	ResourceNoType	Resource ID
OpenClose	OpenCloseType	1 --> Open, 2 --> Close, 3 --> ForceClose
OrderCommand	CommandType	Last command
OrderVersion	VersionType	Order version
GTDDate	DateTimeType	Required for GTD order, used to specified the expiry date, in format “yyyy/MM/dd”
GTCInfo	GTCInfoField	GTC Info field in format <Order-Create-Date>,<Original Price>,<Order-Quantity>,<Pre Filled Qty >,<Pre Filled Amt >
TradeDate	DateTimeType	Order date, format, yyyy/MM/dd, only used by this function

BCAN	BCANType	Required if placing A-Share order, numeric data with range 1 ~ 9999999999

3.2.3 OnRspTradeUpdate

Purpose: response for client subscribe trade request.

Function: void OnRspTradeUpdate(TradeField& trade, RspStatusField & info);

Parameter: trade: return user trade info.

info: response status info.

Remark: after client call **SubscribeTrade** function; once there is trade update, this function will be called.

Field summary of TradeField:

Field	**Format**	**Remark**
Symbol	SymbolType	stock symbol;
Price	PriceType	limit price
Quantity	VolumeType	Quantity
OrderStatus	StatusType	refer satus type
SubOrderID	OrderNoType	Sub-Order number
TradeID	TradeNoType	Trade number
AccountID	AccountType	trade account ID
BuySell	BuySellType	Order buy sell direction:0=buy, 1=sell
UserID	UserNoType	AE-UserID
ExchangeID	ExchangeIDType	name of exchange
TradingTime	DateTimeType	trade time
OpenClose	OpenCloseType	1=Open, 2=Close, 3=ForceClose
CounterParty	BrokerType	Counter Party
ClientReference	ClientLocalIDType	Client Use reference
TradeType	TradeTypeType	Trade Type OddLot=1,Manual=2, Preopen=3, Oversea=4

3.2.4 OnRspSubOrderUpdate

Purpose: response for client subscribe Sub-Order request.

Function: void OnRspSubOrderUpdate (SubOrderField& subOrder,
RspStatusField & info);

Parameter: subOrder: return user sub order info.
info: response status info.

Remark: after client call **SubscribeSubOrder** function; once there is Sub-Order update, this function will be called.

Field summary of SubOrderField:

Field	Format	Remark
Symbol	SymbolType	stock symbol;
Price	PriceType	limit price
Quantity	VolumeType	Quantity
SubOrderStatus	StatusType	refer status type
OrderID	OrderNoType	Order number (identity of a particular order)
SubOrderID	OrderNoType	Sub-Order number
AccountID	AccountType	trade account ID
BuySell	BuySellType	Order buy sell direction:0=buy, 1=sell
UserID	UserNoType	AE-UserID
ExchangeID	ExchangeIDType	name of exchange
DestExchangeID	ExchangeIDType	Destination exchange ID
OrderingTime	DateTimeType	orderint time
OrderCreateTime	DateTimeType	Order creation time
ClientReference	ClientLocalIDType	Client Use Reference
OpenClose	OpenCloseType	1=Open, 2=Close, 3=ForceClose

3.2.5 OnRspErrorOrderUpdate

Purpose: response for client subscribe Error-Order request.

Function: void OnRspErrorOrderUpdate (OrderField& errOrder,
RspStatusField & info);

Parameter: errOrder: return user error orderinfo.

info: response status info.

Remark: after client call **SubscribeErrorOrder** function; once there is Error-Order update, this function will be called.

Field summary of OrderField:

Field	Format	Remark
Symbol	SymbolType	stock symbol
Price	PriceType	limited price
Quantity	VolumeType	Quantity
OrderStatus	StatusType	refer Order status Type
OrderID	OrderNoType	Order number (identity of a particular order)
AccountID	AccountType	Trade account ID
BuySell	BuySellType	Order buy sell direction:0=buy, 1=sell
AEUserID	UserNoType	corresponding AE account ID
ExchangeID	ExchangeIDType	name of exchange
ErrorMessage	ErrorMsgType	error message
ClientReference	ClientLocalIDType	free text comments on this order entry
OrderingTime	DateTimeType	ordering time
FillQuantity	VolumeType	Filled quantity
ErrorCode	ErrorCodeType	error code;
AvgPrice	PriceType	average price
AccountCredit	PriceType	account credit
QueueQuantity	VolumeType	Queue quantity;
OperatorFlag	OperatorFlagType	Operator flag indicating where the order entry from, must assigned with eBroker agree
WorkingQuantity	VolumeType	Working quantity
MinorCode	ErrorCodeType	Minor code (refer Appendix 4.2)

OrderCreateTime	DateTimeType	Order creation time
ResourceID	ResourceNoType	Resource ID
OpenClose	OpenCloseType	1 --> Open, 2 --> Close, 3 --> ForceClose
GTDDate	DateTimeType	Required for GTD order, used to specified the expiry date, in format “yyyy/MM/dd”
GTCInfo	GTCInfoField	GTC Info field in format <Order-Create-Date>,<Original Price>,<Order-Quantity>,<Pre Filled Qty >,<Pre Filled Amt >
TradeDate	DateTimeType	Order date, format, yyyy/MM/dd, only used by this function
OrderCommand	CommandType	Order Command
OrderVersion	VersionType	Order Version
OrderType	OrderTypeType	Order’s type, ref appendix 4.5, order type summary

3.2.6 OnRspAccountBalanceInfoUpdate

Purpose: response for client account balance real time update.

Function: void OnRspAccountBalanceInfoUpdate (AccountBalanceField& acctbalance, RspStatusField & info);

Parameter: acctbalance: return account balance info.

info: response status info.

Remark: after client call ReqQryAccountBalance function; once there is account balance update, this function will be called.

Field summary AccountBalanceField:

Field	Format	Remark
AccountID	AccountType	Account ID
Currency	CurrencyType	Currency ID, use by multi-Currency mode
TradingLimit	AmountType	Account Trading Limit
CashBalance	AmountType	Account Balance
BodTradingLimit	AmountType	Account Begin of day trading limit

BodCashBalance	AmountType	Account begin of day cash balance
InitialMargin	AmountType	Initial Margin
MainMargin	AmountType	Maintenance margin
PNL	AmountType	P&L
MarginCall	AmountType	MarginCall / Margin call
WebTradingLimit	AmountType	Web Tradeing Limit
BodWebLimit	AmountType	Account begin of day web tradeing limit

3.2.7 OnRspCurrencyBalanceInfoUpdate

Purpose: response for client account balance(by currency) real time update.

Function: void OnRspCurrencyBalanceInfoUpdate (AccountBalanceField& balance, RspStatusField & info);

Parameter: balance: return account balance(by currency) info.

info: response status info.

Remark: after client call ReqQryCurrencyBalancefunction; once there is account balance(by currency) update, this function will be called.

Field summary of AccountBalanceField:

Field	**Format**	**Remark**
AccountID	AccountType	Account ID
Currency	CurrencyType	Currency ID, use by multi-Currency mode
TradingLimit	AmountType	Account Trading Limit
CashBalance	AmountType	Account Balance
BodTradingLimit	AmountType	Account Begin of day trading limit
BodCashBalance	AmountType	Account begin of day cash balance
InitialMargin	AmountType	Initial Margin
MainMargin	AmountType	Maintenance margin
PNL	AmountType	P&L
MarginCall	AmountType	MarginCall / Margin call
WebTradingLimit	AmountType	Web Tradeing Limit
BodWebLimit	AmountType	Account begin of day web

		tradeing limit

3.2.8 OnRspConnectionStatus

Purpose: response for client connect with API' status.

Function: void OnRspConnectionStatus(ConnectionStatusField& connectionStatus);

Parameter: connectionStatus: return connects status info.

Remark: After client connect API function; once there is connect status is update, this function will be called.

Field summary of ConnectionStatusField:

Field	Format	Remark
TargetServer	TargetServerType	Server Type
Status	Bool	Server connect status

3.2.9 OnRspOrderCalculateUpdate

Purpose: response for client calculate order request.

Function: void OnRspOrderCalculateUpdate(OrderChargesField& rspChargesInfo, RspStatusField& info)

Parameter: rspChargesInfo : charges calculate fields.

info: response status info.

Remark: after client call **SubscribeCalculate** function.

* Only Order is GTC or GTD order, the GTCInfo field will be filled.

Field summary of OrderChargesField:

Field	Format	Remark
Symbol	SymbolType	Stock symbol
Price	PriceType	Limited price
Quantity	VolumeType	Quantity
AccountID	AccountType	Trade account ID
BuySell	BuySellType	Order type: 0=Buy, 1=Sell
UserID	UserNoType	Order user ID
Currency	CurrencyType	Symbol currency
OperatorFlag	OperatorFlagType	Operator Flag indicating where the

		Order entry from.
ClientReference	ClientLocalIDType	Client use reference
CcassFees	AmountType	CCASS fees
Commission	AmountType	Commission
StampDuty	AmountType	Stamp Duty
Levy	AmountType	Levy
TotalFees	AmountType	Total Fees = CCASS Fees + Stamp Duty + Lev
TradeValue	AmountType	Order Amount excluding fees and commission
OrderType	OrderTypeType	Order's type, ref appendix 4.5, order type summary
GTDDate	DateTimeType	Required for GTD order, used to specified the expiry date, in format "yyyy/MM/dd"
GTCInfo	GTCInfoField	GTC Info field in format <Order-Create-Date>,<Original Price>,<Order-Quantity>,<Pre Filled Qty >,<Pre Filled Amt >

3.2.10 OnRspGenericError

Purpose: response for client general error.

Function: void OnRspGenericError (GeneralErrorField& generalError)

Parameter: generalError: general error field.

Remark: when client call ReqOrderInsert, ReqOrderAction, ReqOrderCalculate, and Occur error, this function will be triggure.

Field summary of GeneralErrorField:

Field	Format	Remark
TargetServer	TargetServerType	Server Type
ClientLocalID	ClientLocalIDType	Client use reference
ReferenceID	ReferenceIDType	Reference ID(function name, which function trigger)
ErrorCode	ErrorCodeType	Error Code
ErrorMsg	ErrorMsgType	Error message

4. Appendix

4.1. Login and Session Related Error Code List

Error Code	Error Message
0	No Error
10	Login failed
11	Password change failed
12	Password expired
13	User suspended/disable
14	Password checking failed
15	Acct suspended/disable
16	User inactive
17	Conflict with existing old password
18	Conflict with existing old PIN
19	PIN expired
20	verify failed
21	PIN change failed
30	remote verify timeout
40	logout failed
41	operation denied
50	Session Terminated due to new session
51	OMS_SSMERR_NEWSESSION
52	Session is kicked
53	User logout normally
54	Session Terminated due to verify retry failed
60	The License file does not contain this certificate
61	The application excesses license limit
62	No application ID in login command
65	The counterpart ssm is not avaliable for any reason
66	The counterpart ssm exist
67	The ssm is waiting counterpart's response
68	The backup ssm is expired
70	PIN number verify failed
71	Challenge session ID not found
72	Login session denied by end user
73	Challenge session timeout
80	Verify token failed
81	Update DB failed

82	Session ID error
83	Token verification already done
84	Token verification required.
85	Service not available
86	Send notify failed
100	Include invalid symbol in password
101	Password not meet length limit condition
102	Password not meet number char condition
103	Password not meet low- alphabet condition
104	Password not meet Upper- alphabet condition
105	Password include continue char
106	Password include continue char by Keyboard
107	Password include repeat char
108	Password not begin with alphabet
110	command unknown
111	ITS command invalid
112	verification required
113	Cipher key not set
114	Verification duplicated
115	Service not available
116	Address access denied
117	Another Login in progress
118	OMS_ITSERR_DUPLOGIN
120	Account ID invalid
121	Invalid number of while querying history
122	Cipher key invalid
123	Trading password verification failed
130	Required record not found
131	Settle instruction failed
132	Market closed. Operation rejected.
133	Requested Custom Query is not defined
134	Query executing error
135	Order Validation Error
136	Date Invalid

4.2. List of Error Code for Order

Error Code	Minor Code	Error Message
1	10000	Order size is odd lot
-1	-10001	Account not specified
-2	-20000	Please specify Buy or Sell

Error Code	Minor Code	Error Message
-3	-30000	Invalid Symbol
-4	-40000	Please specify Quantity
-5	-50001	Insufficient position
	-50002	Insufficient close position
	-50003	Insufficient sell limit
	-50004	Insufficient stock position %.0n
-6	-60001	Cannot calculate credit amount for AO market order
	-60002	Insufficient trading balance/limit: need additional credit of %.0n
	-60003	Cannot %s from bank account
	-60004	Order over internet trading limit
-7	-70000	Price is not at spread level
-8	-80001	Bid is over market by %.0n%%
	-80002	Bid is below market by %.0n%%
	-80003	Ask is below market by %.0n%%
	-80004	Ask is over market by %.0n%%
	-80005	Bid/Ask is not available
	-80006	Daily up limit check failed
	-80007	Decimal length check failed
-9	-90001	Invalid Quantity
	-90002	Work quantity larger than order quantity.
	-90003	Order results in too many orders to the exchange. Please enter orders separately.
-10	-100001	Cannot save order:
	-100002	Cannot book trade:
-11	-110001	Account SELL Only for securities
	-110002	Account Close Only for futures/options
	-110003	Client account is suspended.
	-110004	User is suspended
	-110005	Security can be sold only.
	-110006	Order over upfront ratio.
	-110007	Account cannot trade in the exchange XXX
	-110008	Account cannot trade in the market YYY
	-110009	Insufficient approval limit: need additional amount of
	-110010	Insufficient per order approval limit: difference is
	-110011	Previous Change/Cancel still pending. Please RESET order
	-110012	Cannot change INAC order
	-110013	Exchange Order not replied
	-110015	Order quantity over max lots
	-110016	Internet user cannot change AE order
	-110017	Internet user cannot cancel AE order
	-110018	Client account is suspended via channel

Error Code	Minor Code	Error Message
	-110019	User is suspended via channel
	-110020	Exceed the maximum buying limit
	-110021	Account cannot trade this product
	-110022	Account is suspended for [EXCH] via channel [operator flag].
	-110023	User is suspended for [EXCH] via channel [operator flag].
	-110024	Account is suspended for product(%s: %s) via channel %s.
	-110025	User is suspended for product(%s: %s) via channel %s.
	-110026	Account is suspended for product(%s) via channel %s.
	-110027	User is suspended for product(%s) via channel %s.'
	-110028	User cannot trade this product(%s).
	-110029	Exceed the maximum sell volume limit.
	-110030	Total net-buy-limit exceeded. Required: %s
	-110031	Account cannot trade this symbol via channel %s
	-110032	Account need approval to trade this symbol via channel %s
	-110033	User is set inactive.
	-110034	Cross trade order is not allow to update via this channel.
-12	-120000	ORD-2:Order not outstanding
-14	-140000	Invalid exchange
-16	-160000	ORD-1:Change order failed. Trying to cancel order. Please inactivate order if it fails
-17	-170001	Stop price is over limit price
	-170002	Stop price is already at nominal
	-170003	Limit price is over stop price
-18	-180001	Order Amount over Transaction Limit
-19	-190000	Invalid Symbol
	-190001	Order passed last trading date.
-20	-200000	Lotsize of is zero
-21	-210002	Account Sell Only (ICL < 0).
-22	-220001	Market is closed
	-220002	Symbol status changed to day closed
-23	-230001	Cannot change AO order during market open
-24	-240001	Cannot approve SI instruction Order <order_no> not found Cannot approve non-inactive order Cannot approve Buy/Sell Transaction
-25	-250001	Cannot reduce work order quantity below <n>
-26	-260001	Only <n> order can wait for approval
-27	-270001	Cannot hold fund. Insufficient fund.
	-270002	Cannot hold fund. System failed.
	-270003	Cannot release fund
	-270004	Cannot release fund in change

Error Code	Minor Code	Error Message
	-270005	Cannot change hold fund
-28	-280001	cannot amend up
-29	-290001	Order amendment is blocked in current market status
-30	-300001	Order amount is over the warning limit.
	-300002	Add order met delisted symbol.
	-300003	Change order when symbol become delisted.

4.3. Data type list.

Data type	Base data type	Data type explain
AccountType	Char[20]	Account
SymbolType	Char[20]	symbol
symbolNameType	Char[50]	Symbol's name
imagekeyType	Char[64]	N/A
imageValueType	Char[1024]	N/A
UserNoType	Char[10]	User number
AcctNameType	Char[150]	Account's name
AddressType	Char[100]	N/A
PhoneNoType	Char[20]	N/A
EmailType	Char[100]	N/A
ClientLocalIDType	Char[128]	N/A
OrderNoType	Char[64]	Order number
TradeNoType	Char[64]	Trade number
ErrorMsgType	Char[256]	N/A
ExchangeIDType	Char[16]	N/A
ProductSeriesType	Char[32]	N/A
CurrencyType	Char[16]	N/A
ReferenceIDType	Char[25]	N/A
LoginIdType	Char[20]	N/A
DateTimeType	Char[20]	N/A
BrokerType	Char[20]	N/A
ClientNameType	Char[20]	N/A
ResourceNoType	Char[16]	N/A
CommandType	Char[16]	N/A
DeviceType	Char[16]	N/A
MachineType	Char[16]	N/A
AppIDType	Char[32]	N/A
AppVersionType	Char[32]	N/A
ClientIPType	Char[32]	N/A
SessionType	Char[128]	N/A

PinPosType	Char[32]	N/A
AppNameType	Char[32]	N/A
TokenCodeType	Char[16]	N/A
NotifyAddressType	Char[128]	N/A
LocalIPType	Char[32]	N/A
DeviceOSType	Char[50]	N/A
LocationType	Char[100]	N/A
BrowserType	Char[50]	N/A
BCANType	Char[16]	N/A
PasswordType	Char[25]	N/A
AcctClientType		N/A
OperatorFlagType		N/A
AttributeTypeType	Char[8]	N/A
ObjectTypeType	Char[8]	N/A
AttributeIDType	Char[24]	N/A
AttributeValueType	Char[8]	N/A
AmountType	Double	N/A
PriceType	Double	N/A
VolumeType	int	N/A
ErrorCodeType	int	N/A
LotSizeType	int	N/A
CallPutType	Int	N/A
BuySellType	Int	N/A
OpenCloseType	Int	N/A
StatusType	Int	Reject=-1, Pend=1, Partial=2, Fill=3, Cancel=4, Inactive=5 Order Status “Pending” implies either the order is stored in the system or already sent out to the exchange for queuing. But if order queue quantity > 0, the order is queuing in the exchange. Otherwise it is stored in the system.
AcctMarginType	Int	N/A
AcctStatusType	Int	N/A
OrderTypeType	Int	N/A
ShortSellType	Int	N/A
UserType	Int	N/A
APIModeType	Int	N/A

LoginStatusType	Int	N/A
DateType	Int	N/A
TokenTypeType	Int	N/A
TimeIntervalType	Int	N/A
VersionType	Int	N/A
Productype	Int	N/A
ExchangeStatusType	Int	N/A

4.4. Struct List

```
struct ReqLoginField
{
        LoginIdType         LoginID;                // Account ID of the user
        PasswordType Password;                      // Account Password
        AppVersionType      AppVersion;                     // Application version
        ClientIPType    ClientIP;               // IP address of the client, Need to provide internet
IP instead of LAN IP since SFC required, unless Gateway keeps record on their own
        AppNameType         AppID;                          // APP ID
};
```

```
struct RspLoginField {
        SessionType             SessionID;              // Unique Session ID
        LoginIdType             LoginID;            // user identity
        UserType            LoginType;              // account type
        PinPosType              PinPos;                 //   Random   PIN   number   positions
(optional),separated by comma
        LoginStatusType         LastLoginStatus;    // last logon status -1=false, 0=success
        ClientNameType          LastLoginAppName;       // last logon application name
        DateTimeType        LastLoginTime;          //  last  logon  time,  in  format"yyyy-MM-dd
HH:mm:ss";
        DateTimeType        PwdExpireDate;          //  password  expiry  date  time,  in  format
"yyyy/MM/dd HH:mm:ss"
        DateTimeType        SvrCurrentTime;         //   Server   side   date   time,   in   format
“yyyy/MM/dd HH:mm:ss”
        SessionType             SessionKey;             // The session key of the logon session,
used to identify the session of the order
        ErrorCodeType           ErrorCode;              // Error Code
        ErrorMsgType        ErrorMessage;       // Error Message
        bool                NeedTradePassword;  // y or n trade password when place order
        AccountList             SubAccountList;         // Sub-account, use by SSO mode
};
```

```
struct ReqPasswordUpdateField {
        LoginIdType             LoginID;        // Account ID of the user
        PasswordType        OldPassword;  // Old password
        UserType            LoginType;          // Account Type: 0=AE, 1=Account , optinal
        PasswordType        NewPassword;// New password
        DeviceType              DeviceName;         // name of device
};
```

```
struct PositionField
{
        AccountType Account;
        SymbolType Symbol;
        AmountType Amount;
        VolumeType LongQty;
        VolumeType ShortQty;
        PriceType AvgPrice; // no use
};
```

```
struct PositionList
{
        Int SymbolCount;
        PositionField* Position;
};
```

```
struct AccountField
{
        AccountType Account;
        AcctNameType AccountName;
        AcctMarginType Margin_Type;
        AcctClientType Client_Class;
};
```

```
struct AccountBalanceField {
        AccountType             AccountID;                      // Account ID
        CurrencyType        Currency;               // Currency: HKD
        AmountType              TradingLimit;           // Account Trading Limit
        AmountType              CashBalance;            // Account Balance
        AmountType              BodTradingLimit;        // Account Begin of day trading limit
        AmountType              BodCashBalance;             // Account begin of day cash balance
        AmountType              InitialMargin;          // Initial Margin
        AmountType              MainMargin;                 // Maintenance margin
        AmountType              PNL;                    // P&L
        AmountType          MarginCall;         // MarginCall / Margin call
        AmountType          WebTradingLimit;
        AmountType          BodWebLimit;
};
```

```
struct CurrencyBalanceList {
        int CurrencyCount;
        AccountBalanceField* AccountBalance;
};
```

```
struct OrderChargesField {
        SymbolType          Symbol;
        PriceType       LimitPrice;
        VolumeType          Quantity;
        AccountType         AccountID;
        BuySellType         BuySell;
        UserNoType          UserID;
        CurrencyType    Currency;       // optional
        OperatorFlagType OperatorFlag;      // optional
        ClientLocalIDType ClientReference;  // optional
        PriceType       CcassFees;
        PriceType       Commission;
        PriceType       StampDuty;
        PriceType       Levy;
        PriceType       TotalFees;
        PriceType       TradeValue;
};
```

```
struct OrderInputField {
        SymbolType          Symbol;             // stock symbol
        PriceType       LimitPrice;                 // limit price
        VolumeType          Quantity;               // Quantity
        AccountType         AccountID;              // trade account ID
        BuySellType         BuySell;            // order type: 0=buy, 1=sell
        UserNoType          UserID;             // the user ID placing this order
        OrderTypeType       OrderType;                  // limit order or stop order
        PriceType       AvgPrice;               // average price
        ClientLocalIDType ClientReference;      // free text used for client's reference
        ShortSellType   ShortSellingFlag;       // short selling flag: 1=Yes, 0=No
        OrderNoType         BasketID;               // Basket or bulk parent order ID
        PasswordType    TradingPassword;        // Trading password
        PriceType       StopPrice;                  // stop price
        OpenCloseType       OpenClose;
```

```
	CurrencyType		Currency;			// Symbol currency
	DateTimeType		GTDDate;
};
```

```
struct OrderField {
	SymbolType			Symbol;				// stock symbol
	PriceType		Price;				// limited price
	VolumeType			Quantity;			// quantity
	StatusType			OrderStatus;			// refer Order status Type
	OrderNoType			OrderID;			// Order number (identity of a particular
order)
	AccountType			AccountID;				// Trade account ID
	BuySellType			BuySell;			// order type:0=buy, 1=sell
	UserNoType			AEUserID;				// corresponding AE account ID
	OrderTypeType			OrderType;				// limit order or stop order
	ExchangeIDType			ExchangeID;				// name of exchange
	ErrorMsgType		ErrorMessage;			// error message
	ClientLocalIDType	ClientReference;		// free text comments on this order entry
	DateTimeType		OrderingTime;			// ordering time
	VolumeType			FillQuantity;			// Filled quantity
	ErrorCodeType			ErrorCode;				// error code;
	PriceType		AccountCredit;			// account credit
	VolumeType			QueueQuantity;				// Queue quantity;
	OperatorFlagType	OperatorFlag;			// Operator flag indicating where the order
entry from, must assigned with eBroker agree
	VolumeType			WorkingQuantity;		// Working quantity
	ErrorCodeType			MinorCode;				// Minor code
	DateTimeType		OrderCreateTime;		// Order creation time
	ResourceNoType			ResourceID;				// Resource ID
	OpenCloseType			OpenClose;
	DateTimeType		GTDDate;
	GTCInfoField		GTCInfo;
};
```

```
struct SubOrderField {
	SymbolType			Symbol;				// stock symbol;
	PriceType		Price;				// limit price
	VolumeType			Quantity;			// quantity
	StatusType			SubOrderStatus;				// refer satus type
	OrderNoType			OrderID;			// Order number (identity of a particular
order)
	OrderNoType			SubOrderID;				// Sub-Order number
	AccountType			AccountID;				// trade account ID
```

```
        BuySellType             BuySell;                    // order type:0=buy, 1=sell
        UserNoType              UserID;                         // AE-UserID
        ExchangeIDType          ExchangeID;                     // name of exchange
        ExchangeIDType          DestExchangeID;                 // Destination exchange ID
        DateTimeType        OrderingTime;           // ordering time
        DateTimeType        OrderCreateTime;        // Order creation time
        ClientLocalIDType   ClientReference;
        OpenCloseType           OpenClose;
};
```

```
struct OrderActionField {
        PriceType           Price;                  // limit price
        PriceType           StopPrice;                  // stop price
        VolumeType              Quantity;               // quantity
        AccountType             AccountID;                      // trade account ID
        OrderNoType             OrderID;                // order number identifying order to image
        UserNoType              UserID;                         // The user ID who change the order
        ClientLocalIDType   ClientReference;        // free text used for client's reference
        PasswordType        Password;                   // Thrading password
        ErrorMsgType        ActionReason;           //
        OrderOperation          OrderAction;
};
```

```
struct OrderList
{
        Int OrderCount;
        OrderField* Orders;
};
```

```
struct SubOrderList
{
        int SubOrderCount;
        SubOrderField* SubOrders;
};
```

```
struct TradeField {
        SymbolType              Symbol;
        PriceType           LimitPrice;
        VolumeType              Quantity;
        StatusType              OrderStatus;
        OrderNoType             SubOrderID;
        TradeNoType             TradeID;
        AccountType             AccountID;
        BuySellType                  BuySell;
        UserNoType                   UserID;
        ExchangeIDType               ExchangeID;
        DateTimeType         TradingTime;
        OpenCloseType                OpenClose;
        ClientLocalIDType    ClientReference;
        TradeTypeType                TradeType;
};
```

```
struct TradeList
{
        int TradeCount;
        TradeField* Trade;
};
```

```
struct ConnectionStatusField
{
        TargetServerType TargetServer;
        bool Status;
};
```

```
struct CurrencyRatioField
{
        CurrencyType Currency;
        Float Ratio;
};
```

```
struct CurrencyRatioList
{
        Int CurrencyCount;
        CurrencyRatioField* CurrencyRatio;
};
```

```
struct RspStatusField
{
        ErrorCodeType ErrorId;
        ErrorMsgType ErrorMsg;
};
```

```
struct MarginRatioField
{
        SymbolType symbol;
```

```
	Float MarginRatio;
};
```

```
struct CurrencyList
{
	Int CurrencyCount;
	CurrencyType* Currenct;
};
```

```
struct AccountList
{
	Int AccountCount;
	AccountType* Account;
};
```

```
struct SubOrderList
{
	int SubOrderCount;
	OrderField* SubOrder;
};
```

```
enum TargetServerType
{
	tgtSSM,
	tgtDDS,
	tgtITS
};
```

```
/***********                          Order                          Type
*********************************************************/
#define ORD_TYPY_LIMIT                    0
#define ORD_TYPE_FAK                  128
#define ORD_TYPE_FOK                  256
#define ORD_TYPE_AUCTION              512
#define ORD_TYPE_STOPLIMIT                2048
/*****************************************************************************
***/

/*********** Stop Limit Order Trigger Condition *********************************/
#define omsOrderTriggerUp             8388608
#define omsOrderTriggerDown               16777216
/*****************************************************************************
***/
```

```
struct GTCInfoField {
        DateTimeType        CreatedDate;
        VolumeType          Quantity;
        PriceType           Price;
        VolumeType          FilledQuantity;
        PriceType           FilledPrice;
};
```

4.5. API support order type summary

Order Type	Value to be used
Limit Order	0 (default value)
Odd Lot Order	3
Fill And Kill	128
Fill Or Kill	256
Auction Order	512
Stop Limit Order	2048
GTC Order	1048576