策略王報價函式庫使用說明

版本 2.2.5

修改日期	版本編號	說明
2008/8/7	1.00.01	版本初次建立
2009/9/10	1.00.02	新增功能 1. 取得全部選擇權資料，此兩函式必須搭配使用 SKQuoteLib_AttachStrikePricesCallBack SKQuoteLib_GetStrikePrices 2. 技術分析資料，此兩函式必須搭配使用 SKQuoteLib_AttachKLineDataCallBack SKQuoteLib_GetKLine
		新增一個回傳值定義 #define SK_KLINE_DATA_TYPE_NOT_FOUND 8
2009/9/28	1.00.03	新增功能 1. 取得大盤成交張筆資料 SKQuoteLib_AttachMarketTotCallBack 2. 依鍵值(index)，取得大盤成交張筆資料 SKQuoteLib_GetMarketTot 3. 取得大盤成交買賣張筆數資料 SKQuoteLib_AttachMarketBuySellCallBack 4. 依鍵值(index)，取得大盤成交買賣張筆數資料 SKQuoteLib_GetMarketBuySell 5. 取得大盤成交上漲下跌家數資料 SKQuoteLib_AttachMarketHighLowCallBack 6. 依鍵值(index)，取得大盤成交上漲下跌家數資料 SKQuoteLib_GetMarketHighLow
		新增五種風險參數計算 1. SKQuoteLib_Delta 2. SKQuoteLib_Gamma 3. SKQuoteLib_Theta 4. SKQuoteLib_Vega 5. SKQuoteLib_Rho
2010/2/24	2.2.3	新增功能： 1. 直接回傳個股成交明細的 Call Back 函式 SKQuoteLib_AttachTicksGetCallBack 2. 直接回傳個股五檔價格的 Call Back 函式 SKQuoteLib_AttachBest5GetCallBack

		3. 要求傳送主機目前的時間 SKQuoteLib_RequestServerTime 4. 取得查詢的時間，此兩函式必須搭配使用。 SKQuoteLib_AttchServerTimeCallBack SKQuoteLib_GetServerTime 5. 選擇權詢價 SKQuoteLib_QuoteRequest 6. 取得選擇權詢價結果，此兩函式必須搭配使用。 SKQuoteLib_AttachQuoteRequestCallBack SKQuoteLib_GetQuoteRequest
		新增兩個回傳值定義 #define SK_SUBJECT_QUOTEREQUEST_NOT_FOUND 214 #define SK_SUBJECT_SERVER_TIME_NOT_FOUND 215
2010/7/7	2.2.4	SKQuoteLib_GetKLine 技術分析資料新增取得現貨的資料
		修正 SKQuoteLib_Initialize 問題，新增 #define SK_ERROR_PERMISSION_TIMEOUT 9
2010/12/3	2.2.5	新增 CallBack 函式，取得期貨、選擇權商品， 1. 總委託買進筆數 2. 總委託賣出筆數 3. 總委託買進口數 4. 總委託賣出口數 5. 總成交買進筆數 6. 總成交賣出筆數 SKQuoteLib_AttachFutureTradeInfoCallBack 當送出 Request 期貨、選擇權報價後，主動會傳送此項資料，其中選擇權僅有成交筆數之資料
		修正選擇商品年份問題，呼叫 GetStrikePrice 回傳的年份錯誤。

1、說明：

報價函式庫主要提供市場即時報價功能，商品包括證券、期貨以及選擇權。欲使用此項元件功能，必須先填寫申請表單，經過權限開放後方可使用，否則將會得到錯誤代碼為 SK_ERROR_PERMISSION_DENIED 的錯誤。

2、環境設定

所需要的檔案包括：

$Install\Data\Stock.dat	商品名稱定義
$Install\SKQuoteLib.dll	報價函式庫核心檔案
$Install\Config.ini	環境設定檔

2.1 Stock.dat

當每次報價連線後，會針對 Stock.dat 內遺漏的商品做下載名稱的動作，因此每次程式結束會自動更新檔案內容，如果此檔案遺漏，將會在初次連線下載全部市場的商品名稱，重新建立，因此將會導致初始的時間有所延遲的現象。

2.2 Config.ini

檔案主要設定函式庫程式的重要設定，設定內容初始如下：

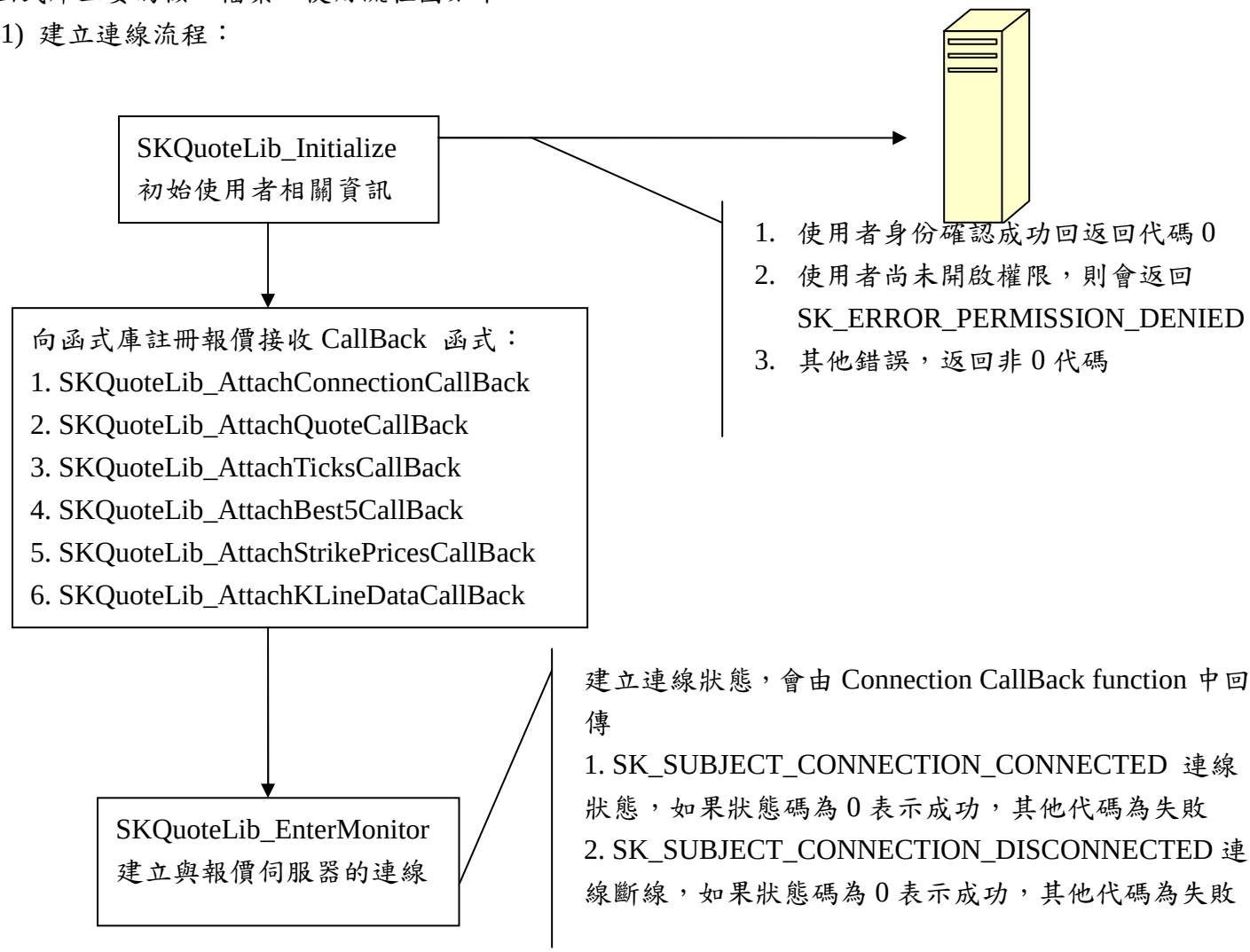
```
[version]
id=2.08.10

[Servers]
0=order2.capital.com.tw
#1=order3.capital.com.tw

[debug]
enable=1
```

file=C:\Source\SKOrderLib\Test\ObjectTester\debug.log

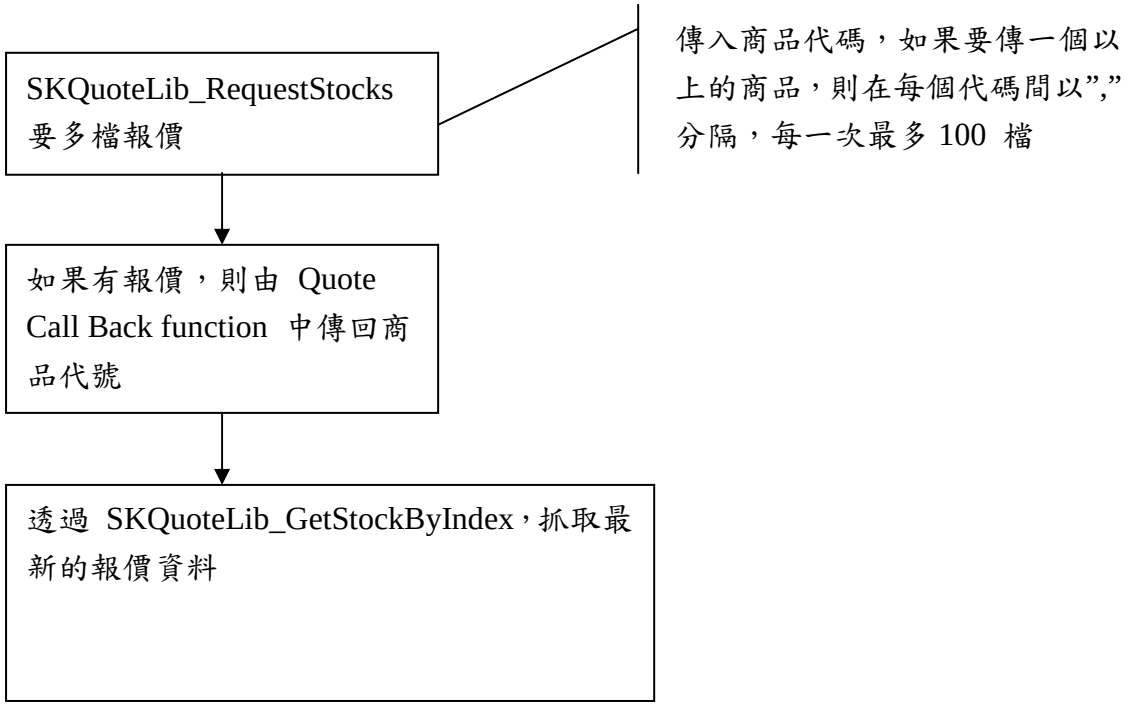
[Version] 區段主要為告知「使用者認證伺服器」目前使用軟體的版本編號，隨著程式的不斷修正或擴增功能，使用者的函式庫版本可能因異動而導致無法使用，因此「使用者認證伺服器」將會依照此序號做檢查的依據，如果版本過舊，伺服器將拒絕函式庫的使用。
因此發現函式庫無法使用，可以試著調整此序號繼續使用，但是建議先和群益相關人員洽詢，取得最新的版本，避免因為版本的不一致，導致產生問題。
[Servers] 區段主要放置「使用者認證伺服器」的位址。
[debug] 區段只要設定是否開啟除錯模式，以及除錯記錄的檔案位置與檔名。

2.3 SKQuoteLib.dll

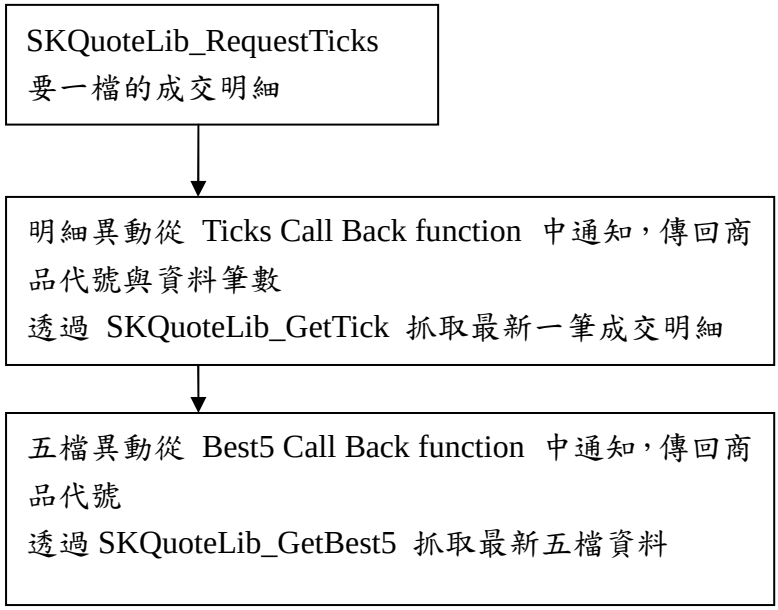
函式庫主要的核心檔案，使用流程圖如下：
(1) 建立連線流程：



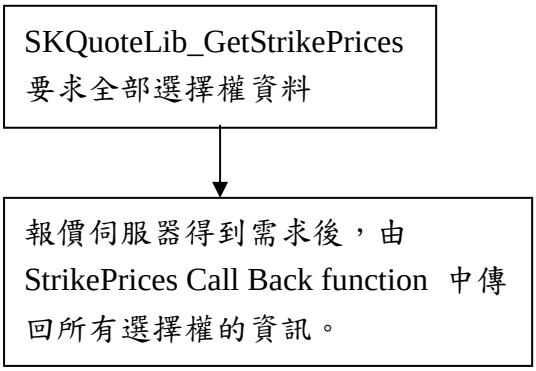
(2) 要求報價流程：



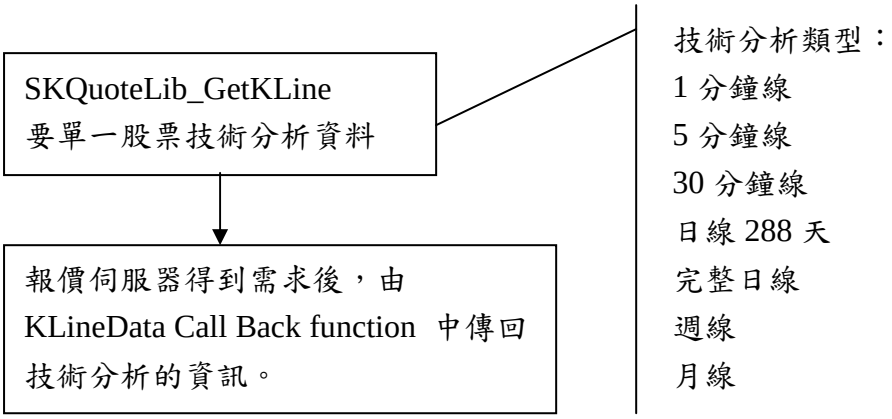
(3)要求成交明細與五檔流程：



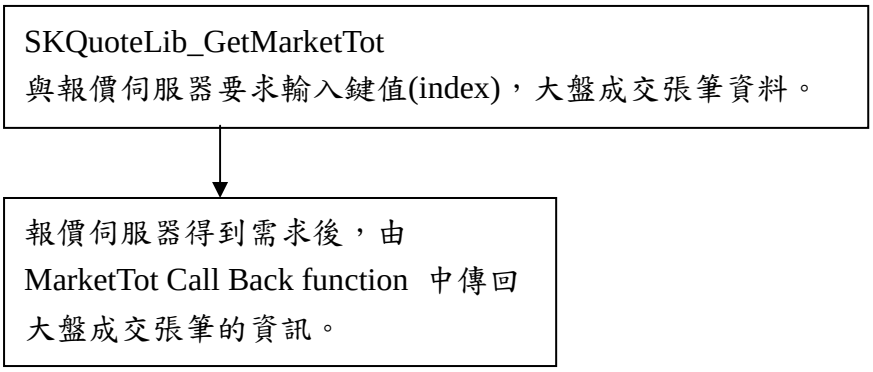
(4)要求選擇權資料流程：



(5)要求技術分析資料流程：



(6)要求大盤成交張筆資料流程：



(7)要求大盤成交買賣張筆數資料流程：

(8)要求大盤成交上漲下跌家數資料流程：

3.函式說明

注意，函式中所取回的價格都未經過小數點處理，例如 1101 價格為 「34.5」，則函式所傳回的價格為「3450」，因此使用者所開發的程式則必須接手處理。目前證券報價固定為 2 位小數，期貨、選擇權部份商品報價為 3 位小數，因此，可以從 TStock 物件的內容找到 m_sDecimal 的欄位，此欄位即定義該檔報價所必需呈現的小數位數。詳細 TStock 物件說明，請參考後續說明。

int __stdcall SKQuoteLib_GetVersion([out] char* lpszVersion, [in,out] int* pnSize)

說明：

取得函式庫版本編號。

參數：

lpszVersion 填入目前的版本值。

pnSize 帶入 lpszVersion 可填入的大小，回傳時內容為 lpszVersion 的字串長度。

回傳值：

SK_SUCCESS 表示成功。

int __stdcall SKQuoteLib_Initialize([in]char* lpszLoginID, [in]char* lpszPassword)

說明：

初始函式庫物件，同時向「使用者驗證伺服器」確認使用者資訊，確認是否有權限執行此函式庫。

參數：

lpszLoginID 使用者登入帳號。

lpszPassword 密碼。

回傳值：

SK_SUCCESS (0)，表示初始化成功，其餘非 0 數值都表示初始失敗。

SK_ERROR_PERMISSION_DENIED 表示使用者權限不足，尚未開啟使用權限。

SK_ERROR_INITIALIZE_FAIL 請檢查 config.ini 驗證伺服器網址設定是否正確。

SK_ERROR_PERMISSION_TIMEOUT 表示使用者太久沒有進行交易，因此使用權限暫時關閉，請重新洽詢營業員申請開放。

int __stdcall SKQuoteLib_EnterMonitor()

說明：

與報價伺服器建立連線。

參數：

無

回傳值：

SK_SUCCESS 表示成功，等待 Connection Call Back function 回傳處理結果。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

SK_FAIL 表示進行連線發生不明錯誤。

int __stdcall SKQuoteLib_LeaveMonitor()

說明：

中斷報價伺服器連線。

參數：

無。

回傳值：

SK_SUCCESS 表示成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

SK_ERROR_SERVER_NOT_CONNECTED 表示尚未與伺服器做連線動作。

其餘非 0 代碼表示其他未知錯誤原因。

int __stdcall SKQuoteLib_RequestStocks([in,out] short* psPageNo, [in]TCHAR* pStockNos)

說明：

要求伺服器針對 pStockNos 內的商品代號做報價通知動作。

參數：

psPageNo 每一次送出要求報價的動作都需要向報價伺服器指定一個特定的 Page 編號，伺服器會以編號當作 Key 值維護每一次送出不同的商品代號。當要變更某一 Page 索取報價的內容，即重複使用相同的 Page 編號，即可取消之前索取的內容，以新的內容取代。當 psPageNo=-1 時帶入，函式庫會指定一個新的編號，並回傳給呼叫端。Page 的範圍為 1~49，每一 Page 最大 100 檔。當代入 Page 為 50 則會取消該檔的報價。

pStockNos 為索取的商品股票代號，一筆以上的資料時，每檔股票代號以”,”做區隔。

舉例說明：

psPageNo = -1 時，pStockNos = “1101,1102,1103,1104,1108,1109,1110”，表示向伺服器索取這七檔股票的報價。如果是第一次執行，則 psPageNo 將會是回傳 1，表示這次的索取是用 Page 編號為 1 的 Page。

當想要取消這七檔的股票，改換其他股票時，則：

psPageNo = 1， pStockNos = “ bla bla…What you want” 帶入即可。

如果目前用的 Page 僅有 3 頁，但是帶入的 Page>3，則會新增一頁並取代，即回傳 Page=4。

如果帶入的股票數超過 100 檔，則僅以 100 檔處理，並不會回傳錯誤。

如果帶入的股票代號不存在，則會直接略過不處理，也不會回傳錯誤。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

SK_SUBJECT_QUOTE_PAGE_EXCEED 表示帶入的 Page 超過極限範圍。

SK_SUBJECT_QUOTE_PAGE_INCORRECT 表示帶入的 Page 不正確，可能為負值。

SK_SUBJECT_CONNECTION_DISCONNECT 表示報價連線已經斷線了。

int __stdcall SKQuoteLib_RequestTicks([in,out]short* psPageNo, [in]TCHAR* pStockNo)

說明：

要求傳送成交明細以及五檔。

參數：

psPageNo 每一次送出要求成交明細的動作都需要向報價伺服器指定一個特定的 Page 編號(請參考 RequestStocks 觀念說明)。

pStockNo 指定的股票代號，一次僅能索取一檔。

所以成交明細、五檔最多僅能同時索取 49 檔。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

SK_SUBJECT_TICK_PAGE_EXCEED 表示帶入的 Page 超過極限範圍。

SK_SUBJECT_TICK_PAGE_INCORRECT 表示帶入的 Page 不正確，可能為負值。

SK_SUBJECT_TICK_STOCK_NOT_FOUND 表示無法找到帶入的股票代號資訊。

SK_SUBJECT_CONNECTION_DISCONNECT 表示報價連線已經斷線了

int __stdcall SKQuoteLib_GetStockByIndex([in]short sMarketNo, [in]short sIndex,[in,out] TStock* pStock)

說明：

根據市場別編號與系統所編的特殊代號，取回股票報價的相關資訊。

市場編號分別為：

上市	0x00
上櫃	0x01
期貨	0x02
選擇權	0x03
興櫃	0x04

每一檔股票的編碼代號每天都有可能不同，因此不能以相同的代號在不同的交易日使用，以避免報價上產生錯誤。

參數：

cMarketNo 市場別代碼。

sIndex 系統編碼後的特定股票代碼。

pStock 回傳的個股報價內容。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

SK_FAIL 表示找不到股票代碼所對應的股票資訊。

int __stdcall SKQuoteLib_GetStockByNo([in] const TCHAR* lpszStockNo,[in,out] TStock* pStock)

說明：

根據股票代號取得報價資訊。

參數：

lpszStockNo　　股票代號，例如 1101。

pStock　　回傳股票代號所對應的報價資訊。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

SK_FAIL 表示找不到股票代碼所對應的股票資訊。

int __stdcall SKQuoteLib_GetTick([in]short sMarketNo, [in]short sStockidx, [in]int nPtr, [in,out]TTick* pTick)

說明：

取得成交明細資訊。

參數：

sMarketNo 市場別代號。

sStockidx 系統所定義的股票代碼，詳細說明請參考 SKQuoteLib_GetStockByIndex 中說明。

nPtr 目前成交明細的儲存位置。可以根據此 Index 取得該筆成交明細資訊。

pTick 成交明細內容。詳細物件內容說明，請參考後續說明。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

SK_SUBJECT_TICK_STOCK_NOT_FOUND 表示找不到股票代碼所對應的成交明細資訊。

int __stdcall SKQuoteLib_GetBest5([in]short sMarketNo, [in]short sStockidx, [in,out]TBest5* pBest5)

說明：

取得最佳五檔價格資訊。

參數：

sMarketNo 市場別代號。

sStockidx 系統定義的股票代碼。

pBest5 最佳五檔資訊。詳細參考後續說明。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

SK_SUBJECT_BEST5_DATA_NOT_FOUND 表示找不到股票代碼所對應的最佳五檔資訊。

int __stdcall SKQuoteLib_GetStrikePrices()

說明：

向報價伺服器提出，取得全部選擇權資訊需求。報價伺服器會透過向 AttachStrikePricesCallBack(long lCallBack)註冊的 Call Back 函式，回傳全部選擇權資訊。

參數：

無

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_GetKLine([in,out] char* caStockNo, [in] int KLineType)

說明：

向報價伺服器提出，取得單一上市股票技術分析資訊需求。報價伺服器會透過向 AttachKLineCallBack(long lCallBack)註冊的 Call Back 函式，回傳單一上市股票技術分析資訊。

參數：

caStockNo 上市股票代號。Ex. 6005。

KLineType 技術分析類型。

0 = 1 分鐘線。

1 = 5 分鐘線。

2 = 30 分鐘線。

3 =日線 288 天。

4 =完整日線。

5 =週線。

6 =月線。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

SK_KLINE_DATA_TYPE_NOT_FOUND 表示找不到要使用的技術分析類型。

int __stdcall SKQuoteLib_GetMarketTot([in] int index, [in] char cMarketNo , [out] short sPrt ,
[out] long* lTime, [out] long* lTotv,[out] long* lTots,[out] long* lTotc)

說明：

向報價伺服器提出，依輸入鍵值(index)，取得大盤成交張筆資料需求。報價伺服器會透過向 AttachMarketTotCallBack (long lCallBack)註冊的 Call Back 函式，回傳大盤成交張筆資料。

參數：

index	輸入要求第 index 筆成交明細資訊。 鍵值(0~n)，0 為當天第一筆成交資料。
cMarketNo	市場別代號 (0x00 上市, 0x01 上櫃)
sPrt	回傳目前成交明細的儲存位置。(同 index)
lTime	大盤成交資料時間
lTotv	大盤成交值
lTots	大盤成交筆數
lTotc	大盤成交張數

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_GetMarketBuySell([in] int index, [in] char cMarketNo , [out] short sPrt , [out] long* lTime, [out] long* lBc, [out] long* lSc, [out] long* lBs, [out] long* lSs)

說明：

向報價伺服器提出，依輸入鍵值(index)，取得大盤成交買賣張筆數資料需求。報價伺服器會透過向 AttachMarketBuySellCallBack(long lCallBack)註冊的 Call Back 函式，回傳大盤成交買賣張筆數資料。

參數：

index	輸入要求第 index 筆成交明細資訊。 鍵值(0~n)，0 為當天第一筆成交資料。
cMarketNo	市場別代號 (0x00 上市, 0x01 上櫃)
sPrt	回傳目前成交明細的儲存位置。(同 index)
lTime	大盤成交資料時間
lBc	大盤成交買進筆數
lSc	大盤成交賣出筆數
lBs	大盤成交買進張數
lSs	大盤成交賣出張數

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_GetMarketHighLow([in] int index, [in] char cMarketNo , [out] short sPrt ,
[out] long* lTime, [out] short* sUp, [out] short* sDown, [out] short* sHigh,
[out] short* sLow,
[out]short* sNoChange)

說明：

向報價伺服器提出，依輸入鍵值(index)，取得大盤成交上漲下跌家數資料需求。報價伺服器會透過向 AttachMarketHighLowCallBack(long lCallBack)註冊的 Call Back 函式，回傳大盤成交上漲下跌家數資料。

參數：

index	輸入要求第 index 筆成交明細資訊。 鍵值(0~n)，0 為當天第一筆成交資料。
cMarketNo	市場別代號 (0x00 上市, 0x01 上櫃)
sPrt	回傳目前成交明細的儲存位置。(同 index)
lTime	大盤成交資料時間
sUp	大盤成交上漲家數
sDown	大盤成交下跌家數
sHigh	大盤成交漲停家數
sLow	大盤成交跌停家數
sNoChange	大盤平盤家數

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_Delta([in] int nCallPut, [in] double S, [in] double K, [in] double R,
[in]double T, [in] double sigma, [out] double* dDelta)

說明：

輸入，

1. nCallPut　　標的別(Call: 0 , Put: 1)
2. S　　指數
3. K　　StrikePrice(履約價)
4. R　　RisklessRate(無風險利率)
5. T　　RemainMaturity(剩餘天數)
6. sigma

得到 Delta 值。

參數：

nCallPut	標的別(Call: 0 , Put: 1)
S	指數
K	StrikePrice(履約價)
R	RisklessRate(無風險利率)
T	RemainMaturity(剩餘天數)
sigma	sigma
dDelta	儲存回傳 Delta 值 參數

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_Gamma([in] double S, [in] double K, [in] double R, [in] double T,
[in]double sigma , [out] double* dGamma)

說明：

輸入，

1. S 指數
2. K StrikePrice(履約價)
3. R RisklessRate(無風險利率)
4. T RemainMaturity(剩餘天數)
5. sigma

得到 Gamma 值。

參數：

S	指數
K	StrikePrice(履約價)
R	RisklessRate(無風險利率)
T	RemainMaturity(剩餘天數)
sigma	sigma
dGamma	儲存回傳 Gamma 值 參數

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_Theta([in] int nCallPut, [in] double S, [in] double K, [in] double R,
[in]double T, [in] double sigma , [out] double* dTheta)

說明：

輸入，

1. nCallPut 標的別(Call: 0 , Put: 1)
2. S 台指期成交價
3. K StrikePrice(履約價)
4. R RisklessRate(無風險利率)
5. T RemainMaturity(剩餘天數)
6. v 波動率 = sigma

得到 Theta 值。

參數：

nCallPut	標的別(Call: 0 , Put: 1)
S	台指期成交價
K	StrikePrice(履約價)
R	RisklessRate(無風險利率)
T	RemainMaturity(剩餘天數)
sigma	V 波動率 = sigma
dTheta	儲存回傳 Theta 值 參數

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_Vega([in] double S, [in] double K, [in] double R, [in] double T,
[in]double sigma , [out] double* dVega)

說明：

輸入，

1. S 台指期成交價
2. K StrikePrice(履約價)
3. R RisklessRate(無風險利率)
4. T RemainMaturity(剩餘天數)
5. v 波動率 = sigma

得到 Vega 值。

參數：

S	台指期成交價
K	StrikePrice(履約價)
R	RisklessRate(無風險利率)
T	RemainMaturity(剩餘天數)
sigma	V 波動率 = sigma
dVega	儲存回傳 Vega 值 參數

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_Rho([in] int nCallPut, [in] double S, [in] double K, [in] double R, [in]double T, [in]double sigma , [out] double* dRho)

說明：

輸入，

1. nCallPut　　標的別(Call: 0 , Put: 1)
2. S　　台指期成交價
3. K　　StrikePrice(履約價)
4. R　　RisklessRate(無風險利率)
5. T　　RemainMaturity(剩餘天數)
6. v　　波動率 = sigma

得到 Rho 值。

參數：

nCallPut	標的別(Call: 0 , Put: 1)
S	台指期成交價
K	StrikePrice(履約價)
R	RisklessRate(無風險利率)
T	RemainMaturity(剩餘天數)
sigma	V 波動率 = sigma
dRho	儲存回傳 Rho 值 參數

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

TStock 物件說明：

```
typedef struct STOCK
{
    short m_sStockidx;          // 系統自行定義的股票代碼
    short m_sDecimal;           // 報價小數位數
    short m_sTypeNo;            // 類股分類
    char m_cMarketNo;           // 市場代號 0x00 上市;0x01 上櫃;0x02 期貨;0x03 選擇權;
                                   0x04 興櫃
    char m_caStockNo[20];       // 股票代號
    char m_caName[10];          // 股票名稱
    int m_nOpen;                // 開盤價
    int m_nHigh;                // 最高價
    int m_nLow;                 // 最低價
    int m_nClose;               // 成交價
    int m_nTickQty;             // 單量
    int m_nRef;                 // 昨收、參考價
    int m_nBid;                 // 買價
    int m_nBc;                  // 買量
    int m_nAsk;                 // 賣價
    int m_nAc;                  // 賣量
    int m_nTBc;                 // 買盤量
    int m_nTAc;                 // 賣盤量
    int m_nFutureOI;            // 期貨未平倉 OI
    int m_nTQty;                // 總量
    int m_nYQty;                // 昨量
    int m_nUp;                  // 漲停價
    int m_nDown;                // 跌停價
} TStock;
```

TTick 物件說明：

```
typedef struct TICK
{
    int m_nPtr;             // 資料所在的位置(Key)
    int m_nTime;            // 時間
    int m_nBid;             // 買價
    int m_nAsk;             // 賣價
    int m_nClose;           // 成交價
    int m_nQty;             // 成交量
}TTick;
```

TBest5 物件說明：

```
typedef struct BEST5
{
    int m_nBid1;                    // 第一檔買價
    int m_nBidQty1;                 // 第一檔買量
    int m_nBid2;                    // 第二檔買價
    int m_nBidQty2;                 // 第二檔買量
    int m_nBid3;                    // 第三檔買價
    int m_nBidQty3;                 // 第三檔買量
    int m_nBid4;                    // 第四檔買價
    int m_nBidQty4;                 // 第四檔買量
    int m_nBid5;                    // 第五檔買價
    int m_nBidQty5;                 // 第五檔買量
    int m_nExtendBid;               // 衍生一檔買價
    int m_nExtendBidQty;            // 衍生一檔買量
    int m_nAsk1;                    // 第一檔賣價
    int m_nAskQty1;                 // 第一檔賣量
    int m_nAsk2;                    // 第二檔賣價
    int m_nAskQty2;                 // 第二檔賣量
    int m_nAsk3;                    // 第三檔賣價
    int m_nAskQty3;                 // 第三檔賣量
    int m_nAsk4;                    // 第四檔賣價
    int m_nAskQty4;                 // 第四檔買量
    int m_nAsk5;                    // 第五檔賣價
    int m_nAskQty5;                 // 第五檔賣量
    int m_nExtendAsk;               // 衍生一檔賣價
    int m_nExtendAskQty;            // 衍生一檔賣量
} TBest5;
```

Call back 函式說明：

int __stdcall SKQuoteLib_AttachConnectionCallBack([in]long lCallBack)

說明：

向報價函式庫註冊接收連線狀態的 Call back 函式位址。函式的宣告必須符合以下格式：

typedef void (__stdcall* FOnNotifyConnection)(int nKind, int nCode)

會有兩個參數回傳：

nKind 表示訊息種類，可能得值有：

SK_SUBJECT_CONNECTION_CONNECTED 表示為連線事件。

SK_SUBJECT_CONNECTION_DISCONNECT 表示為斷線事件。

nCode 表示事件所獲得的代碼，代碼為 0 值即表示執行正確，非 0 表示例外事件。

參數：

lCallBack 帶入 Call back 的函式位址。當狀況發生時即可呼叫此函式進行處理。

請注意，如果傳入的函式定義與上面描述的不同，將可能導致系統發生嚴重錯誤，甚至導致電腦發生當機情況，因此請務必謹慎使用。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_AttachQuoteCallBack([in]long lCallBack)

說明：

當有索取的個股報價異動時，將透過此註冊的函式通知應用程式處理。

向報價函式庫註冊接收報價更新時處理的 Callback 函式，函式的宣告必須符合以下格式：

typedef void (__stdcall* FOnNotifyQuote)(short sMarketNo, short sStockidx)

兩個參數回傳：

sMarketNo 報價有異動的商品市場別。

sStockidx 系統自行定義的股票代碼。

透過以上兩個參數，可以再使用 SKQuoteLib_GetStockByIndex 取出報價的內容。

參數：

lCallBack 帶入 Call back 的函式位址。當有向報價伺服器索取的個股報價有所異動，即會透過呼叫此 Call back 函式，處理價格異動時使用者自行定義的程序。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_AttachTicksCallBack([in] long lCallBack)

說明：

當有索取的個股成交明細有所異動，即透過向此註冊的 Call back 函式進行處理。

所註冊的函式宣告必須符合以下格式：

typedef void (__stdcall* FOnNotifyTicks)(short sMarketNo, short sStockidx, int nPtr)

三個參數回傳：

sMarketNo 成交明細有異動的商品市場別。

sStockidx 系統自行定義的股票代碼。

nPtr 表示資料的位址(Key)。所以如果要取得今天該檔股票的全部成交明細，可以迴圈的方式逐筆取得。Index 的範圍從 0 ~ Ptr 所指定的位置。

參數：

lCallBack 帶入 Call back 的函式位址。當有向報價伺服器索取的個股報價有所異動，即會透過呼叫此 Call back 函式，處理成交明細異動時使用者自行定義的程序。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_AttachTicksGetCallBack([in] long lCallBack)

說明：

當有索取的個股成交明細有所異動,即透過向此註冊的 Call back 函式進行處理,此 Call Back 函式會回傳所異動的個股成交明細。

所註冊的函式宣告必須符合以下格式：

typedef void (__stdcall* FOnNotifyTicksGet)(short sMarketNo, short sStockidx, int nPtr, int nTime,int nBid, int nAsk, int nClose, int nQty)

八個參數回傳：

sMarketNo	成交明細有異動的商品市場別。
sStockidx	系統自行定義的股票代碼。
nPtr	表示資料的位址(Key)。所以如果要取得今天該檔股票的全部成交明細，可以迴圈的方式逐筆取得。Index 的範圍從 0 ~ Ptr 所指定的位置。
nTime	時間。
nBid	買價。
nAsk	賣價。
nClose	成交價。
nQty	成交量。

參數：

lCallBack 帶入 Call back 的函式位址。當有向報價伺服器索取的個股報價有所異動，即會透過呼叫此 Call back 函式，處理成交明細異動時使用者自行定義的程序。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_AttachBest5CallBack([in] long lCallBack)

說明：

當有索取的個股五檔價格有所異動，即透過向此註冊的 Call back 函式進行處理。

所註冊的函式宣告必須符合以下格式：

typedef void (__stdcall* FOnNotifyBest5)(short sMarketNo, short sStockidx)

兩個參數回傳：

sMarketNo 五檔有異動的商品市場別。

sStockidx 系統自行定義的股票代碼。

參數：

lCallBack 帶入 Call back 的函式位址。當有向報價伺服器索取的個股報價有所異動，即會透過呼叫此 Call back 函式，處理五檔價格異動時使用者自行定義的程序。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_AttachBest5GetCallBack([in] long lCallBack)

說明：

當有索取的個股五檔價格有所異動,即透過向此註冊的 Call back 函式進行處理。此 Call Back 函式會回傳所異動的個股五檔價格。

所註冊的函式宣告必須符合以下格式：

typedef void (__stdcall* FOnNotifyBest5Get)(short sMarketNo, short sStockidx, int nBestBid1, int nBestBidQty1, int nBestBid2, int nBestBidQty2, int nBestBid3, int nBestBidQty3, int nBestBid4, int nBestBidQty4,int nBestBid5, int nBestBidQty5,int nExtendBid, int nExtendBidQty, int nBestAsk1, int nBestAskQty1, int nBestAsk2, int nBestAskQty2, int nBestAsk3, int nBestAskQty3, int nBestAsk4, int nBestAskQty4, int nBestAsk5, int nBestAskQty5, int nExtendAsk,int nExtendAskQty)

二十六個參數回傳：

sMarketNo	五檔有異動的商品市場別。
sStockidx	系統自行定義的股票代碼。
nBestBid1	第一檔買價。
nBestBidQty1	第一檔買量。
nBestBid2	第二檔買價。
nBestBidQty2	第二檔買量。
nBestBid3	第三檔買價。
nBestBidQty3	第三檔買量。
nBestBid4	第四檔買價。
nBestBidQty4	第四檔買量。
nBestBid5	第五檔買價。
nBestBidQty5	第五檔買量。
nExtendBid	延伸一檔買價。
nExtendBidQty	延伸一檔買量。
nBestAsk1	第一檔賣價。
nBestAskQty1	第一檔賣量。
nBestAsk2	第二檔賣價。
nBestAskQty2	第二檔賣量。
nBestAsk3	第三檔賣價。
nBestAskQty3	第三檔賣量。
nBestAsk4	第四檔賣價。
nBestAskQty4	第四檔賣量。
nBestAsk5	第五檔賣價。
nBestAskQty5	第五檔賣量。

nExtendAsk　　延伸一檔賣價。
nExtendAskQty 延伸一檔賣量。

參數：

lCallBack　帶入 Call back 的函式位址。當有向報價伺服器索取的個股報價有所異動，即會透過呼叫此 Call back 函式，處理五檔價格異動時使用者自行定義的程序。

回傳值：

SK_SUCCESS 表示執行成功。
SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall AttachStrikePricesCallBack([in] long lCallBack)

說明：

報價伺服器會透過向此註冊的 Call back 函式，回傳全部選擇權資訊。

所註冊的函式宣告必須符合以下格式：

typedef void (__stdcall* FOnNotifyStrikePrices)(BSTR BProduct, BSTR BName, BSTR BCall BSTR BPut, int nStrikePrice, int nYearMonth)

六個參數回傳：

BProduct	商品代碼。例如，台指選商品代碼為 TXO，也就是選擇權代號前三碼。
BName	中文名稱。例如，台選。
BCall	Call 標的代碼。例如，TXO03800I9
BPut	Put 標的代碼。例如，TXO03800U9
nStrikePrice	履約價。例如，C102.5，nStrikePrice 資料為 10250，使用時要除一百，即 nStrikePrice/100。
nYearMonth	年＋月。例如，2009 年 9 月，nYearMonth 資料為 200909

參數：

lCallBack 帶入 Call back 的函式位址。當向報價伺服器索取全部選擇權資訊時，報價伺服器會透過呼叫此 Call back 函式，回傳選擇權資訊。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_AttachKLineDataCallBack([in] long lCallBack)

說明：

當報價伺服器會透過向此註冊的 Call back 函式，回傳技術分析資訊。

所註冊的函式宣告必須符合以下格式：

typedef void (__stdcall* FOnNotifyKLineData)(char * caStockNo, char * caData)

二個參數回傳：

caStockNo 上市股票代號。

caData 回傳字串，技術分析資料。傳回字串格式分成兩種:

(1.) 1 分鐘線，5 分鐘線，30 分鐘線。以逗號分開所有資料。

(月/日/年, 時:分, 開盤價, 最高價, 最低價, 收盤價, 成交量)

例如，06/18/2009, 09:05, 1365, 1385, 1365, 1380, 656

(2.) 日線 288 天，完整日線，週線，月線。以逗號分開所有資料。

(月/日/年, 開盤價, 最高價, 最低價, 收盤價, 成交量)

例如，06/18/2009, 1365, 1385, 1365, 1380, 656

函式中所取回的價格都未經過小數點處理，例如 6005 價格為 「13.65」，則函式所傳回的價格為「1365」，因此使用者所開發的程式則必須接手處理。

參數：

lCallBack 帶入 Call back 的函式位址。當向報價伺服器索取單一上市股票技術分析資料時，報價伺服器會透過呼叫此 Call back 函式，回傳技術分析資訊。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_AttachMarketTotCallBack([in] long lCallBack)

說明：

當報價伺服器會透過向此註冊的 Call back 函式，回傳大盤成交張筆資料。

所註冊的函式宣告必須符合以下格式：

typedef void (__stdcall* FOnNotifyMarketTot)(char cMarketNo , short sPrt, long lTime, long lTotv, long lTots, long lTotc)

六個參數回傳：

cMarketNo	市場別代號 (0x00 上市, 0x01 上櫃)
sPrt	目前成交明細的儲存位置。可以根據此 index 取得該筆成交明細資訊。
lTime	大盤成交時間。Ex. lTime = 92000 ，為早上九點 20 分(09:20)
lTotv	大盤成交值(億)。Ex. lTotv = 88542，為 885.42 (億)
lTots	大盤成交筆數
lTotc	大盤成交張數

函式中所取回的價格都未經過小數點處理，例如 6005 價格為 「13.65」，則函式所傳回的價格為「1365」，因此使用者所開發的程式則必須接手處理。

參數：

lCallBack 帶入 Call back 的函式位址。當向報價伺服器索取大盤成交張筆資料時，報價伺服器會透過呼叫此 Call back 函式，回傳大盤成交張筆資料。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_AttachMarketBuySellCallBack([in] long lCallBack)

說明：

當報價伺服器會透過向此註冊的 Call back 函式，回傳大盤成交買賣張筆數資料。

所註冊的函式宣告必須符合以下格式：

typedef void (__stdcall* FOnNotifyMarketBuySell)(char cMarketNo , short sPrt, long lTime, long lBc, long lSc, long lBs, long lSs)

七個參數回傳：\

cMarketNo	市場別代號 (0x00 上市, 0x01 上櫃)
sPrt	目前成交明細的儲存位置。可以根據此 index 取得該筆成交明細資訊。
lTime	大盤成交時間。Ex. lTime = 92000 ，為早上九點 20 分(09:20)
lBc	大盤成交買進筆數
lSc	大盤成交賣出筆數
lBs	大盤成交買進張數
lSs	大盤成交賣出張數

函式中所取回的價格都未經過小數點處理，例如 6005 價格為 「13.65」，則函式所傳回的價格為「1365」，因此使用者所開發的程式則必須接手處理。

參數：

lCallBack 帶入 Call back 的函式位址。當向報價伺服器索取大盤成交買賣張筆數資料時，報價伺服器會透過呼叫此 Call back 函式，回傳大盤成交買賣張筆數資料。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_AttachMarketHighLowCallBack([in] long lCallBack)

說明：

當報價伺服器會透過向此註冊的 Call back 函式，回傳大盤成交上漲下跌家數資料。

所註冊的函式宣告必須符合以下格式：

typedef void (__stdcall* FOnNotifyMarketHighLow)(char cMarketNo , short sPrt, long lTime, short sUp, short sDown, short sHigh, short sLow, short sNoChange)

八個參數回傳：

cMarketNo	市場別代號 (0x00 上市, 0x01 上櫃)
sPrt	目前成交明細的儲存位置。可以根據此 index 取得該筆成交明細資訊。
lTime	大盤成交時間。Ex. lTime = 92000 ，為早上九點 20 分(09:20)
sUp	大盤成交上漲家數
sDown	大盤成交下跌家數
sHigh,	大盤成交漲停家數
sLow	大盤成交跌停家數
sNoChange	大盤平盤家數

函式中所取回的價格都未經過小數點處理，例如 6005 價格為 「13.65」，則函式所傳回的價格為「1365」，因此使用者所開發的程式則必須接手處理。

參數：

lCallBack 帶入 Call back 的函式位址。當向報價伺服器索取大盤成交上漲下跌家數資料時，報價伺服器會透過呼叫此 Call back 函式，回傳大盤成交上漲下跌家數資料。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_RequestServerTime( )

說明：

要求報價主機傳送目前時間。

注意：為避免收盤後無報價資料傳送，導致連線被防火牆切斷，請固定五分鐘呼叫此函式，確保連線正常。

參數：

無。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_AttchServerTimeCallBack ([in] long lCallBack)

說明：

報價伺服器會透過向此註冊的 Call back 函式，回傳查詢主機時間的結果。

所註冊的函式宣告必須符合以下格式：

typedef void (__stdcall* FOnNotifyServerTime)(short sHour, short sMinute, short sSecond, int nTotal)

四個參數回傳：

sHour	時
sMinute	分
sSecond	秒
nTotal	總秒數

參數：

lCallBack 帶入 Call back 的函式位址。當向報價伺服器索取時間時，報價伺服器會透過呼叫此 Call back 函式，處理使用者自行定義的程序

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_GetServerTime ([in,out]CFormat05* pFormat05)

說明：

取得查詢的主機時間。

參數：

pFormat05　　時間資訊。詳細物件內容說明，請參考後續說明。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

SK_SUBJECT_SERVER_TIME_NOT_FOUND 表示取得時間時失敗。

CFormat05 物件說明：

```
typedef struct FORMAT05
{
    short m_sHour;          // 時
    short m_sMinute;        // 分
    short m_sSecond;        // 秒
    long m_lTotal;          // 總秒數
}CFormat05;
```

int __stdcall SKQuoteLib_QuoteRequest ( )

說明：

要求傳送選擇詢價的結果。

參數：

無。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_AttachQuoteRequestCallBack([in] long lCallBack)

說明：

報價伺服器會透過向此註冊的 Call back 函式，回傳選擇權的詢價結果。

所註冊的函式宣告必須符合以下格式：

typedef void (__stdcall* FOnNotifyQuoteRequest)(BSTR bstrStockNo, short sPtr, int nStockidx, short sHour, short sMinute, short sSecond, short sMSecond, int nDuration)

八個參數回傳：

bstrStockNo	股票代號。
sPrt	目前詢價資料的儲存位置。可以根據此 index 取得該詢價結果的資訊。
nStockidx	系統定義的股票代碼。
sHour	時。
sMinute	分。
sSecond	秒。
sMSecond	毫秒。
nDuration	持續時間(秒)。

參數：

lCallBack 帶入 Call back 的函式位址。當向報價伺服器索取選擇權詢價資料時，報價伺服器會透過呼叫此 Call back 函式，處理使用者自行定義的程序

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

int __stdcall SKQuoteLib_GetQuoteRequest([in] int nIndex, [in,out]CQuoteItem* pQuoteItem)

說明：

取得選擇權詢價資訊。

參數：

nIndex	欲取得詢價資訊的儲存位置。
pQuoteItem	詢價資訊。詳細物件內容說明，請參考後續說明。

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

SK_SUBJECT_QUOTEREQUEST_NOT_FOUND 表示找不到該位置所對應的詢價資訊

int __stdcall SKQuoteLib_AttachFutureTradeInfoCallBack ([in] long CallBack)

說明：

當向報價伺服器送出索取報價 (SKQuoteLib_RequestStocks) 或索取成交明細 (SKQuoteLib_RequestTicks) 即會透過註冊的 CallBack 函式，回傳相關資料。

向報價函式庫註冊接收期貨、選擇權交易商品的交易資訊的 Call back 函式位址，函式的宣告必須符合以下格式：

```
typedef void ( __stdcall* FOnNotifyFutureTradeInfo)( BSTR StockNo, int MarketNo, int
    nStockidx, int nBuyTotalCount, int nSellTotalCount, int nBuyTotalQty, int
    nSellTotalQty, int nBuyDealTotalCount, int nSellDealTotalCount)
```

參數：

CallBack 程式定義的回傳函式位址

回傳欄位內容如下:

StockNo 股票代號

nMarketNo 交易所代號

nStockidx 標的代號,透過 MarketNo 與 Stockidx ,可以使用 SKQuoteLib_GetStockByIndex 取得標的相關資訊

nBuyTotalCount 總委託買進筆數

nSellTotalCount 總委託賣出筆數

nBuyTotalQty 總委託買進口數

nSellTotalQty 總委託賣出口數

nBuyDealTotalCount 總成交買進筆數

nSellDealTotalCount 總成交賣出筆數

回傳值：

SK_SUCCESS 表示執行成功。

SK_ERROR_INITIALIZE_FAIL 表示 SKQuoteLib_Initialize 尚未執行完成。

CQuoteItem 物件說明：

```
typedef struct QUOTEITEM
{
    short m_sIndex;          // 資料所在位置(Key)
    int m_nStockidx;         // 系統定義的股票代碼
    int m_nHour;             // 詢價揭示時間 - 時
    int m_nMin;              // 詢價揭示時間 - 分
    int m_nSec;              // 詢價揭示時間 - 秒
    int m_nMSec;             // 詢價揭示時間 - 毫秒
    int m_nDuration;         // 詢價持續時間(秒)

}CQuoteItem;
```

4.附件

4.1 錯誤代碼定義：

#define SK_SUCCESS	0
#define SK_FAIL	-1
#define SK_ERROR_STRING_LENGTH_NOT_ENOUGH	-2
#define SK_ERROR_SERVER_NOT_CONNECTED	-3
#define SK_ERROR_INITIALIZE_FAIL	-4
#define SK_ERROR_ACCOUNT_NOT_EXIST	1
#define SK_ERROR_ACCOUNT_MARKET_NOT_MATCH	2
#define SK_ERROR_PERIOD_OUT_OF_RANGE	3
#define SK_ERROR_FLAG_OUT_OF_RANGE	4
#define SK_ERROR_BUYSELL_OUT_OF_RANGE	5
#define SK_ERROR_ORDER_SERVER_INVALID	6
#define SK_ERROR_PERMISSION_DENIED	7
#define SK_KLINE_DATA_TYPE_NOT_FOUND	8
#define SK_ERROR_PERMISSION_TIMEOUT	9
#define SK_SUBJECT_CONNECTION_CONNECTED	100
#define SK_SUBJECT_CONNECTION_DISCONNECT	101
#define SK_SUBJECT_QUOTE_PAGE_EXCEED	200
#define SK_SUBJECT_QUOTE_PAGE_INCORRECT	201
#define SK_SUBJECT_TICK_PAGE_EXCEED	210
#define SK_SUBJECT_TICK_PAGE_INCORRECT	211
#define SK_SUBJECT_TICK_STOCK_NOT_FOUND	212
#define SK_SUBJECT_BEST5_DATA_NOT_FOUND	213
#define SK_SUBJECT_QUOTEREQUEST_NOT_FOUND	214
#define SK_SUBJECT_SERVER_TIME_NOT_FOUND	215