策略王COM元件使用說明

-下單-國內證券-

## 函式範例

### SKOrderLib(國內證券)

*函式*

1. 查詢

|  |  |  |
| --- | --- | --- |
| **功能** | **函式名稱** | **備註** |
| 1. **證券即時庫存** | [GetRealBalanceReport](#_GetRealBalanceReport) |  |
| 1. **查詢集保庫存** | [GetBalanceQuery](#_GetBalanceQuery) | V2.13.54後，不再提供此函式 |
| 1. **查詢證券資券配額** | [GetMarginPurchaseAmountLimit](#_GetMarginPurchaseAmountLimit_1) |  |
| 1. **新損益試算查詢** | [GetProfitLossGWReport](#_GetProfitLossGWReport) |  |
| 1. **查詢昨日未沖銷明細** | [GetAvgCost](#_GetAvgCost) |  |
| 1. **一戶通、餘額、可用金額查詢** | [GetBalance](#_GetBalance) |  |
| 1. **近三日交割款查詢** | [GetT3DueAmt](#_GetT3DueAmt) |  |
| 1. **台股圈存查詢** | [GetIPAAmt](#_4-2-111_GetIPAAmt) |  |

1. 一般下單/刪改單

|  |  |  |
| --- | --- | --- |
| **功能** | **函式名稱** | **備註** |
| 1. **證券委託** | [SendStockOrder](#_SendStockOrder) | ＊適用逐筆交易 |
| 1. **證券盤中零股委託** | [SendStockOddLotOrder](#_SendStockOddLotOrder) |  |
| 1. **證期權改價** | [CorrectPriceBySeqNo](#_CorrectPriceBySeqNo_1) | 依序號改價 |
| 1. **證期權改價** | [CorrectPriceByBookNo](#_CorrectPriceByBookNo_1) | 依書號改價 |
| 1. **委託減量** | [DecreaseOrderBySeqNo](#_DecreaseOrderBySeqNo) |  |
| 1. **國內刪單-序號** | [CancelOrderBySeqNo](#_CancelOrderBySeqNo_1) | 13 碼序號 |
| 1. **國內刪單-書號** | [CancelOrderByBookNo](#_CancelOrderByBookNo_1) | 依5碼書號刪單 |
| 1. **國內刪單-商品代號** | [CancelOrderByStockNo](#_CancelOrderByStockNo_1) | 依帳號所屬**登入ID**+商品代號 |
| 1. **國內刪單-商品代號(By帳號所屬登入ID+商品代號+買賣別+委託價格)** | [CancelOrderByStockNoAdvance](#_CancelOrderByStockNoAdvance) |  |

1. **Proxy下單**/**刪改單**

|  |  |  |
| --- | --- | --- |
| **功能** | **函式名稱** | **備註** |
| 1. **證券下單** | [SendStockProxyOrder](#_SendStockProxyOrder) |  |
| 1. **證券刪改單** | [SendStockProxyAlter](#_SendStockProxyAlter_1) |  |
| 1. **證券特殊刪改單** | [SendStockProxyPreAlter](#_4-7-17_SendStockProxyPreAlter) |  |

*事件*

|  |  |  |
| --- | --- | --- |
| **功能** | **事件名稱** | **備註** |
| 1. **證券即時庫存** | [OnRealBalanceReport](#_OnRealBalanceReport) |  |
| 1. **查詢集保庫存** | [OnBalanceQuery](#_OnBalanceQuery_1) |  |
| 1. **查詢證券資券配額** | [OnMarginPurchaseAmountLimit](#_OnMarginPurchaseAmountLimit_1) |  |
| 1. **新損益試算資料** | [OnProfitLossGWReport](#_OnProfitLossGWReport_1) |  |
| 1. **非同步委託結果** | [OnAsyncOrder](#_OnAsyncOrder) |  |
| 1. **Proxy委託結果** | [OnProxyOrder](#_OnProxyOrder) |  |

#### 查詢

##### GetRealBalanceReport

|  |  |  |
| --- | --- | --- |
| 查詢證券即時庫存內容。 | | |
| **宣告** | Long GetRealBalanceReport([in] BSTR bstrLogInID, [in] BSTR bstrAccount); | |
| **參數** | bstrLogInID | 登入ID。 |
| bstrAccount | 證券帳號，分公司四碼＋帳號7碼。 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。 | |
| **備註** | 結果由[OnRealBalanceReport](#_OnRealBalanceReport)事件回傳。 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別ReadForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # buttonGetRealBalanceReport          self.buttonGetRealBalanceReport = tk.Button(self)          self.buttonGetRealBalanceReport["text"] = "即時庫存"          self.buttonGetRealBalanceReport["command"] = self.buttonGetRealBalanceReport\_Click          self.buttonGetRealBalanceReport.grid(row=0, column=1) |

|  |
| --- |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib) |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonGetRealBalanceReport\_Click(self):          nCode = m\_pSKOrder.GetRealBalanceReport(comboBoxUserID.get(), comboBoxAccount.get())          msg = "【GetRealBalanceReport】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode)          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### GetBalanceQuery

|  |  |  |
| --- | --- | --- |
| 集保庫存查詢。 | | |
| **宣告** | Long GetBalanceQuery ([in] BSTR bstrLogInID, [in] BSTR bstrAccount[in] BSTR bstrStockNo) | |
| **參數** | bstrLogInID | 登入ID。 |
| bstrAccount | 證券帳號，分公司四碼＋帳號7碼，帶空為所有證券帳號查詢。 |
| bstrStockNo | 商品代碼，代空為全部商品回傳 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。 | |
| **備註** | 結果由 [OnBalanceQuery](#_OnBalanceQuery) 事件回傳。  V2.13.54後，不再提供此函示，請改使用[GetRealBalanceReport](#_4-2-28_GetRealBalanceReport)，謝謝。 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別ReadForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # buttonGetBalanceQuery          self.buttonGetBalanceQuery = tk.Button(self)          self.buttonGetBalanceQuery["text"] = "集保庫存"          self.buttonGetBalanceQuery["command"] = self.buttonGetBalanceQuery\_Click          self.buttonGetBalanceQuery.grid(row=1, column=1)              # Entry          tk.Label(self, text="商品代碼，代空為全部商品回傳").grid(row=1, column=2)          self.bstrStockNo = tk.Entry(self)          self.bstrStockNo.grid(row=1, column=3) |

|  |
| --- |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib) |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonGetBalanceQuery\_Click(self):          nCode = m\_pSKOrder.GetBalanceQuery(comboBoxUserID.get(), comboBoxAccount.get(), self.bstrStockNo.get())          msg = "【GetBalanceQuery】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode)          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### GetMarginPurchaseAmountLimit

|  |  |  |
| --- | --- | --- |
| 資券配額查詢。 | | |
| **宣告** | Long GetMarginPurchaseAmountLimit ([in] BSTR bstrLogInID, [in] BSTR bstrAccount[in] BSTR bstrStockNo) | |
| **參數** | bstrLogInID | 登入ID。 |
| bstrAccount | 證券帳號，分公司四碼＋帳號7碼。 |
| bstrStockNo | 商品代碼，代空為全部商品回傳 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。 | |
| **備註** | 結果由 [OnMarginPurchaseAmountLimit](#_OnMarginPurchaseAmountLimit) 事件回傳。 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別ReadForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # buttonGetMarginPurchaseAmountLimit          self.buttonGetMarginPurchaseAmountLimit = tk.Button(self)          self.buttonGetMarginPurchaseAmountLimit["text"] = "資券配額"          self.buttonGetMarginPurchaseAmountLimit["command"] = self.buttonGetMarginPurchaseAmountLimit\_Click          self.buttonGetMarginPurchaseAmountLimit.grid(row=2, column=1)              # Entry          tk.Label(self, text="商品代碼，代空為全部商品回傳").grid(row=2, column=2)          self.bstrStockNo = tk.Entry(self)          self.bstrStockNo.grid(row=2, column=3) |

|  |
| --- |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib) |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonGetMarginPurchaseAmountLimit\_Click(self):          nCode = m\_pSKOrder.GetMarginPurchaseAmountLimit(comboBoxUserID.get(), comboBoxAccount.get(), self.bstrStockNo.get())          msg = "【GetMarginPurchaseAmountLimit】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode)          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### GetProfitLossGWReport

|  |  |  |
| --- | --- | --- |
| 未實現損益試算。 | | |
| **宣告** | Long GetProfitLossGWReport ([in] BSTR bstrLogInID, [in] struct **TSPROFITLOSSGWQUERY**\* pPLGWQuery); | |
| **參數** | bstrLogInID | 登入ID。 |
| pPLGWQuery | SKCOM元件中的[**TSPROFITLOSSGWQUERY**](#_未實現)物件，將未實現損益查詢條件填入該物件後，再帶入此欄位中。 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。錯誤代碼可參考對照表。 | |
| **備註** | 結果由[OnProfitLossGWReport](#_OnProfitLossGWReport)事件回傳。 | |

|  |  |  |
| --- | --- | --- |
| 已實現損益查詢。 | | |
| **宣告** | Long GetProfitLossGWReport ([in] BSTR bstrLogInID, [in] struct **TSPROFITLOSSGWQUERY**\* pPLGWQuery); | |
| **參數** | bstrLogInID | 登入ID。 |
| pPLGWQuery | SKCOM元件中的[**TSPROFITLOSSGWQUERY**](#_已實現)物件，將未實現損益查詢條件填入該物件後，再帶入此欄位中。 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。錯誤代碼可參考對照表。 | |
| **備註** | 結果由[OnProfitLossGWReport](#_OnProfitLossGWReport)事件回傳。   * 彙總：起始日西元年月日及結束日西元年月日 * 明細：起始日**民國年月日**，不需填結束日 | |

|  |  |  |
| --- | --- | --- |
| 現股當沖損益試算。 | | |
| **宣告** | Long GetProfitLossGWReport ([in] BSTR bstrLogInID, [in] struct **TSPROFITLOSSGWQUERY**\* pPLGWQuery); | |
| **參數** | bstrLogInID | 登入ID。 |
| pPLGWQuery | SKCOM元件中的[**TSPROFITLOSSGWQUERY**](#_現股當沖:(限查閱當日交易))物件，將未實現損益查詢條件填入該物件後，再帶入此欄位中。 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。錯誤代碼可參考對照表。 | |
| **備註** | 結果由[OnProfitLossGWReport](#_現股當沖:(限查閱當日交易))事件回傳。   * 限查當日交易 * 損益回傳格式 1: 彙總 2: 明細 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別ReadForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # comboBoxnTPQueryType 損益          tk.Label(self, text = "種類").grid(row=3, column=1)          self.comboBoxnTPQueryType = ttk.Combobox(self, state='readonly')          self.comboBoxnTPQueryType['values'] = Config.comboBoxnTPQueryType          self.comboBoxnTPQueryType.grid(row=3, column=2)          def on\_comboBoxnTPQueryType(event):              if self.comboBoxnTPQueryType.get() == "未實現損益":                  self.comboBoxnFunc['values'] = Config.comboBoxnFunc0              elif self.comboBoxnTPQueryType.get() == "已實現損益":                  self.comboBoxnFunc['values'] = Config.comboBoxnFunc1              else:                  self.comboBoxnFunc['values'] = Config.comboBoxnFunc2          self.comboBoxnTPQueryType.bind("<<ComboboxSelected>>", on\_comboBoxnTPQueryType)          global comboBoxnTPQueryType          comboBoxnTPQueryType = self.comboBoxnTPQueryType          # comboBoxnFunc          self.comboBoxnFunc = ttk.Combobox(self, state='readonly')          self.comboBoxnFunc.grid(row=3, column=3)          def on\_comboBoxnFunc(event):              if self.comboBoxnTPQueryType.get() == "未實現損益" and self.comboBoxnFunc.get() == "1:明細":                  self.comboBoxbstrTradeType['values'] = Config.comboBoxbstrTradeType0              elif self.comboBoxnTPQueryType.get() == "已實現損益" and self.comboBoxnFunc.get() == "1:明細":                  self.comboBoxbstrTradeType['values'] = Config.comboBoxbstrTradeType1              else:                  self.comboBoxbstrTradeType['values'] = " "            self.comboBoxnFunc.bind("<<ComboboxSelected>>", on\_comboBoxnFunc)          global comboBoxnFunc          comboBoxnFunc = self.comboBoxnFunc          # comboBoxbstrTradeType          self.comboBoxbstrTradeType = ttk.Combobox(self, state='readonly')          self.comboBoxbstrTradeType.grid(row=3, column=4)          global comboBoxbstrTradeType          comboBoxbstrTradeType = self.comboBoxbstrTradeType          # textBoxbstrStockNo3          tk.Label(self, text = "商品代碼").grid(row=3, column=5)              #輸入框          self.textBoxbstrStockNo3 = tk.Entry(self)          self.textBoxbstrStockNo3.grid(row=3, column=6)          global textBoxbstrStockNo3          textBoxbstrStockNo3 = self.textBoxbstrStockNo3          # textBoxbstrStartDate          tk.Label(self, text = "起始日年月日").grid(row=3, column=7)              #輸入框          self.textBoxbstrStartDate = tk.Entry(self)          self.textBoxbstrStartDate.grid(row=3, column=8)          global textBoxbstrStartDate          textBoxbstrStartDate = self.textBoxbstrStartDate          # textBoxbstrEndDate          tk.Label(self, text = "結束日年月日").grid(row=3, column=9)              #輸入框          self.textBoxbstrEndDate = tk.Entry(self)          self.textBoxbstrEndDate.grid(row=3, column=10)          global textBoxbstrEndDate          textBoxbstrEndDate = self.textBoxbstrEndDate          # textBoxbstrBookNo          tk.Label(self, text = "委託書號").grid(row=3, column=11)              #輸入框          self.textBoxbstrBookNo = tk.Entry(self)          self.textBoxbstrBookNo.grid(row=3, column=12)          global textBoxbstrBookNo          textBoxbstrBookNo = self.textBoxbstrBookNo          # textBoxbstrSeqNo          tk.Label(self, text = "委託序號").grid(row=3, column=13)              #輸入框          self.textBoxbstrSeqNo = tk.Entry(self)          self.textBoxbstrSeqNo.grid(row=3, column=14)          global textBoxbstrSeqNo          textBoxbstrSeqNo = self.textBoxbstrSeqNo            # buttonGetProfitLossGWReport          self.buttonGetProfitLossGWReport = tk.Button(self)          self.buttonGetProfitLossGWReport["text"] = "損益試算"          self.buttonGetProfitLossGWReport["command"] = self.buttonGetProfitLossGWReport\_Click          self.buttonGetProfitLossGWReport.grid(row=3, column=15) |

|  |
| --- |
| Config.py |
| comboBoxnTPQueryType = (      "未實現損益", "已實現損益", "現股當沖"  )  comboBoxnFunc0 = (      "0:彙總", "1:明細"  )  comboBoxnFunc1 = (      "0:彙總", "1:明細","2:投資總額", "3:彙總(依股票代號)"  )  comboBoxnFunc2 = (      "1:彙總", "2:明細"  )  comboBoxbstrTradeType0 =(      "0:現股", "3:融資(自)","4:融券(自)", "8:券差", "9:無券賣出"  )  comboBoxbstrTradeType1 =(      "0:現股", "1:融資(代)","2:融券(代)", "3:融資(自)", "4:融券(自)"  ) |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib) |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonGetProfitLossGWReport\_Click(self):          pPLGWQuery = sk.TSPROFITLOSSGWQUERY()          pPLGWQuery.bstrFullAccount = comboBoxAccount.get()          if comboBoxnTPQueryType.get() == "未實現損益":              pPLGWQuery.nTPQueryType = 0          elif comboBoxnTPQueryType.get() == "已實現損益":              pPLGWQuery.nTPQueryType = 1          else:              pPLGWQuery.nTPQueryType = 2            if comboBoxnFunc.get() == "0:彙總":              pPLGWQuery.nFunc = 0          elif comboBoxnFunc.get() == "1:明細":              pPLGWQuery.nFunc = 1          elif comboBoxnFunc.get() == "2:投資總額":              pPLGWQuery.nFunc = 2          elif comboBoxnFunc.get() == "3:彙總(依股票代號)":              pPLGWQuery.nFunc = 3          elif comboBoxnFunc.get() == "1:彙總":              pPLGWQuery.nFunc = 1          elif comboBoxnFunc.get() == "2:明細":              pPLGWQuery.nFunc = 2            pPLGWQuery.bstrStockNo = textBoxbstrStockNo3.get()          if comboBoxbstrTradeType.get() == "0:現股":              pPLGWQuery.bstrTradeType = "0"          elif comboBoxbstrTradeType.get() == "1:融資(代)":              pPLGWQuery.bstrTradeType = "1"          elif comboBoxbstrTradeType.get() == "2:融券(代)":              pPLGWQuery.bstrTradeType = "2"          elif comboBoxbstrTradeType.get() == "3:融資(自)":              pPLGWQuery.bstrTradeType = "3"          elif comboBoxbstrTradeType.get() == "4:融券(自)":              pPLGWQuery.bstrTradeType = "4"          elif comboBoxbstrTradeType.get() == "8:券差":              pPLGWQuery.bstrTradeType = "0"          elif comboBoxbstrTradeType.get() == "9:無券賣出":              pPLGWQuery.bstrTradeType = "0"          pPLGWQuery.bstrStartDate = textBoxbstrStartDate.get()          pPLGWQuery.bstrEndDate = textBoxbstrEndDate.get()          pPLGWQuery.bstrBookNo = textBoxbstrBookNo.get()          pPLGWQuery.bstrSeqNo = textBoxbstrSeqNo.get()            nCode = m\_pSKOrder.GetProfitLossGWReport(comboBoxUserID.get(), pPLGWQuery)          msg = "【GetProfitLossGWReport】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode)          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### GetAvgCost

|  |  |  |
| --- | --- | --- |
| 查詢昨日未沖銷的證券明細。 | | |
| **宣告** | Long GetAvgCost ([in] BSTR bstrLogInID, [in] struct **SKAVGCOST** pOrder,[**out**] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| pOrder | SKCOM元件中的**SKAVGCOST**物件，將昨日未沖銷的證券明細條件填入該物件後，再帶入此欄位中。 |
| bstrMessage | 委託查詢內容。   |  |  | | --- | --- | | 欄位名稱 | 說明 | | 成交日期(民國年) |  | | 單位 | 分公司四碼 | | 帳號 |  | | 股票代號 |  | | 成交序號 |  | | 庫存種類 | 現股/融資/融券 | | 剩餘股數 |  | | 單價 |  | | 手續費 |  | | 融資自備款/融券保證金 |  | | 融資金額/擔保品金額 |  | | 融資/融券利息 |  | | 融券手續費 |  | |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。錯誤代碼可參考對照表。  回傳格式：每一筆資料以「#」分隔，每一個欄位以「,」分隔。  非正常委託內容時  M999: 查詢錯誤  M003: 查無資料 | |
| **備註** | * 一個account在1分鐘內不可超過10次。 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別 AvgCostForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # textBoxAvgCostStockNo  tk.Label(self, text = "商品代碼").grid(row=0, column=1)  #輸入框  self.textBoxAvgCostStockNo = tk.Entry(self)  self.textBoxAvgCostStockNo.grid(row=0, column=2)  global textBoxAvgCostStockNo  textBoxAvgCostStockNo = self.textBoxAvgCostStockNo  # comboBoxGetAvgCost  tk.Label(self, text = "功能").grid(row = 0,column = 3)  self.comboBoxGetAvgCost = ttk.Combobox(self, state='readonly')  self.comboBoxGetAvgCost['values'] = Config.comboBoxGetAvgCost  self.comboBoxGetAvgCost.grid(row=0, column=4)  global comboBoxGetAvgCost  comboBoxGetAvgCost = self.comboBoxGetAvgCost  # buttonGetAvgCost  self.buttonGetAvgCost = tk.Button(self)  self.buttonGetAvgCost["text"] = "查詢"  self.buttonGetAvgCost["command"] = self.buttonGetAvgCost\_Click  self.buttonGetAvgCost.grid(row=1, column=1)  global buttonGetAvgCost  buttonGetAvgCost = self.buttonGetAvgCost |

|  |
| --- |
| Config.py |
| comboBoxGetAvgCost = (      "0:查詢昨日未沖銷明細"  ) |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib) |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonGetAvgCost\_Click(self):  pSKAVGCOST = sk.SKAVGCOST()      pSKAVGCOST.bstrAccount = comboBoxAccount.get()      pSKAVGCOST.bstrStockNo = textBoxAvgCostStockNo.get()      if(comboBoxGetAvgCost.get() == "0:查詢昨日未沖銷明細"):          pSKAVGCOST.bstrFunc = "0"      bstrMessage, nCode = m\_pSKOrder.GetAvgCost(comboBoxUserID.get(), pSKAVGCOST)      msg = "【GetAvgCostData】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode)      richTextBoxMethodMessage.insert('end',  msg + "\n")      strValues = bstrMessage.split('#')      for value in strValues:          if value != "":      value = "【GetAvgCostData】" + value      richTextBoxMessage.insert( 'end',  value + "\n")      richTextBoxMessage.see('end') |

##### GetBalance

|  |  |  |
| --- | --- | --- |
| 一戶通、餘額、可用金額查詢。 | | |
| **宣告** | BSTR GetBalance ([in] BSTR bstrLogInID) | |
| **參數** | bstrLogInID | 登入ID。 |
| **回傳值** | 查詢文字訊息。 | |
| **備註** | -1表示失敗。  每一筆資料以「,」分隔每一個欄位，結尾為「.」。欄位依序為：  EX: true,0,0,0.   |  |  | | --- | --- | | 1 | 是否有開一戶通(true / false) | | 2 | 一戶通餘額 | | 3 | 一戶通可動用金額(目前可出金金額) | | 4 | 一戶通-當日可買進金額 | | |

##### GetT3DueAmt

|  |  |  |
| --- | --- | --- |
| 近三日交割款查詢。 | | |
| **宣告** | BSTR GetT3DueAmt ([in] BSTR bstrLogInID, [in] BSTR bstrAccount) | |
| **參數** | bstrLogInID | 登入ID。 |
| bstrAccount | 證券帳號，分公司四碼＋帳號7碼。 |
| **回傳值** | 查詢文字訊息。 | |
| **備註** | 不同幣別資料以「;」分隔，欄位以「,」分隔，欄位依序為：  EX: 1131111,1131113,-653812,1131108,1131112,-1204594,1131107,1131111,-3262935,臺幣;1131111,1131113,0,1131108,1131112,0,1131107,1131111,0,人民幣;   |  |  | | --- | --- | | 1 | T日交易日期(yyymmdd) | | 2 | T日交割日期(yyymmdd) | | 3 | T日應收付交割款： '-'為客戶應付 '+'為客戶應收。(數字部分左補足9位0 ) \*T日現股盤中即時，信用需補單後 | | 4 | T-1日交易日期(yyymmdd) | | 5 | T-1日交割日期(yyymmdd) | | 6 | T-1日應收付交割款： '-'為客戶應付 '+'為客戶應收。(數字部分左補足9位0) | | 7 | T-2日交易日期(yyymmdd) | | 8 | T-2日交割日期(yyymmdd) | | 9 | T-2日應收付交割款： '-'為客戶應付 '+'為客戶應收。(數字部分左補足9位0) | | 10 | 幣別(中文) | | |

##### GetIPAAmt

|  |  |  |
| --- | --- | --- |
| 台股圈存查詢 | | |
| **宣告** | BSTR GetIPAAmt ([in] BSTR bstrLogInID, [in] BSTR bstrAccount, [in] LONG nMarket) | |
| **參數** | bstrLogInID | 登入ID。 |
| bstrAccount | 證券帳號，分公司四碼＋帳號7碼。 |
| nMarket | 0:上市櫃  1:興櫃 |
| **回傳值** | 欄位以「,」分隔，結尾為「#」，欄位依序為：  有開通一戶通 格式：   |  |  | | --- | --- | | 1 | 可圈存金額 | | 2 | 已圈存金額 | | 3 | 已委託金額 |   EX:  646551,7831,7811#  未開通一戶通 格式：   |  |  | | --- | --- | | 1 |  | | 2 |  | | 3 |  | | |
| **備註** | 沒有一戶通：  EX: 請匯預收款至國泰世華商業銀行(013)群益金鼎證券營業處客戶興櫃結算專戶，再與分公司營業員聯絡 | |

#### **一般下單/刪改單**

##### SendStockOrder

|  |  |  |
| --- | --- | --- |
| 送出證券委託。 | | |
| **宣告** | Long SendStockOrder([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **STOCKORDER**\* pOrder,[**out**] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| bAsyncOrder | 是否為非同步委託。 |
| pOrder | SKCOM元件中的 [**STOCKORDER**](#_STOCKORDER_(證券下單物件)) 物件，將下單條件填入該物件後，再帶入此欄位中。 |
| bstrMessage | **同步委託：**  如果回傳值為 0表示委託成功，訊息內容則為13碼的委託序號。  回傳值非0表示委託失敗，訊息內容為失敗原因。  **非同步委託：**參照[OnAsyncOrder](#_OnAsyncOrder)。 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。  **當錯誤代碼為4碼時，可參考文件「4.下單準備介紹」裡面的【錯誤代碼定義表】，所屬委託物件亦有欄位值說明供參。**  **其他錯誤則由「交易主機」回傳錯誤代碼及錯誤原因。若仍有疑問請洽詢您所屬營業員。** | |
| **備註** | 使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder)取得。  ＊亦適用證券逐筆 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別SendForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖  一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片  自動產生的描述 |
| # checkBoxAsyncOrder          # 是否為非同步委託          self.checkBoxAsyncOrder = tk.Checkbutton(self)          self.var1 = tk.IntVar()          self.checkBoxAsyncOrder["variable"] = self.var1          self.checkBoxAsyncOrder["onvalue"] = True          self.checkBoxAsyncOrder["offvalue"] = False          self.checkBoxAsyncOrder["text"] = "非同步委託"          self.checkBoxAsyncOrder["command"] = self.checkBoxAsyncOrder\_CheckedChanged          self.checkBoxAsyncOrder.grid( row = 0,column = 0)          # textBoxStockID          tk.Label(self, text = "股票代號", fg = "red").grid(row=0, column=1)              #輸入框          self.textBoxStockID = tk.Entry(self)          self.textBoxStockID.grid(row=0, column=2)          global textBoxStockID          textBoxStockID = self.textBoxStockID          # comboBoxPrime          tk.Label(self, text = "上市上櫃/興櫃").grid(row=1, column=1)              #輸入框          self.comboBoxPrime = ttk.Combobox(self, state='readonly')          self.comboBoxPrime['values'] = Config.comboBoxPrime          self.comboBoxPrime.grid(row=1, column=2)          global comboBoxPrime          comboBoxPrime = self.comboBoxPrime          # comboBoxPeriod          tk.Label(self, text = "盤中/盤後/零股").grid(row=2, column=1)              #輸入框          self.comboBoxPeriod = ttk.Combobox(self, state='readonly')          self.comboBoxPeriod['values'] = Config.comboBoxPeriod          self.comboBoxPeriod.grid(row=2, column=2)          global comboBoxPeriod          comboBoxPeriod = self.comboBoxPeriod          # textBoxnQty          tk.Label(self, text = "整股(張)/零股(股數)", fg = "red").grid(row=3, column=1)              #輸入框          self.textBoxnQty = tk.Entry(self)          self.textBoxnQty.grid(row=3, column=2)          global textBoxnQty          textBoxnQty = self.textBoxnQty          # comboBoxFlag          tk.Label(self, text = "現股/融資/融券/無券").grid(row=4, column=1)              #輸入框          self.comboBoxFlag = ttk.Combobox(self, state='readonly')          self.comboBoxFlag['values'] = Config.comboBoxFlag          self.comboBoxFlag.grid(row=4, column=2)          global comboBoxFlag          comboBoxFlag = self.comboBoxFlag          # comboBoxnTradeType          tk.Label(self, text = "ROD/IOC/FOK").grid(row=5, column=1)              #輸入框          self.comboBoxnTradeType = ttk.Combobox(self, state='readonly')          self.comboBoxnTradeType['values'] = Config.comboBoxnTradeType          self.comboBoxnTradeType.grid(row=5, column=2)          global comboBoxnTradeType          comboBoxnTradeType = self.comboBoxnTradeType          # comboBoxnSpecialTradeType          tk.Label(self, text = "市價/限價").grid(row=6, column=1)              #輸入框          self.comboBoxnSpecialTradeType = ttk.Combobox(self, state='readonly')          self.comboBoxnSpecialTradeType['values'] = Config.comboBoxnSpecialTradeType          self.comboBoxnSpecialTradeType.grid(row=6, column=2)          global comboBoxnSpecialTradeType          comboBoxnSpecialTradeType = self.comboBoxnSpecialTradeType          # textBoxbstrPrice          tk.Label(self, text = "委託價", fg = "red").grid(row=7, column=1)              #輸入框          self.textBoxbstrPrice = tk.Entry(self)          self.textBoxbstrPrice.grid(row=7, column=2)          global textBoxbstrPrice          textBoxbstrPrice = self.textBoxbstrPrice            # comboBoxBuySell          tk.Label(self, text = "買進/賣出", fg = "red").grid(row=8, column=1)              #輸入框          self.comboBoxBuySell = ttk.Combobox(self, state='readonly')          self.comboBoxBuySell['values'] = Config.comboBoxBuySell          self.comboBoxBuySell.grid(row=8, column=2)          global comboBoxBuySell          comboBoxBuySell = self.comboBoxBuySell          # buttonSendStockOrder          self.buttonSendStockOrder = tk.Button(self)          self.buttonSendStockOrder["text"] = "證券送出"          self.buttonSendStockOrder["command"] = self.buttonSendStockOrder\_Click          self.buttonSendStockOrder.grid(row=9, column=2) |

|  |
| --- |
| Config.py |
| comboBoxPrime = (      "上市上櫃", "興櫃"  )  comboBoxPeriod = (      "盤中", "盤後", "零股"  )  comboBoxFlag = (      "現股", "融資", "融券", "無券"  )  comboBoxnTradeType = (      "ROD", "IOC", "FOK"  )  comboBoxnSpecialTradeType = (      "市價", "限價"  )  comboBoxBuySell = (      "買進", "賣出"  ) |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib)  # 是否為非同步委託  bAsyncOrder = False |
| Current Block(放置於def createWidgets(self): 的同一層) |
| # checkBoxAsyncOrder      def checkBoxAsyncOrder\_CheckedChanged(self):          global bAsyncOrder          if self.var1.get() == True:              bAsyncOrder = True          else:              bAsyncOrder = False      def buttonSendStockOrder\_Click(self):          pOrder = sk.STOCKORDER()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxStockID.get()          if comboBoxPrime.get() == "上市上櫃":              pOrder.sPrime = 0          elif comboBoxPrime.get() == "興櫃":              pOrder.sPrime = 1            if comboBoxPeriod.get() == "盤中":              pOrder.sPeriod = 0          elif comboBoxPeriod.get() == "盤後":              pOrder.sPeriod = 1          elif comboBoxPeriod.get() == "零股":              pOrder.sPeriod = 2            if comboBoxFlag.get() == "現股":              pOrder.sFlag = 0          elif comboBoxFlag.get() == "融資":              pOrder.sFlag = 1          elif comboBoxFlag.get() == "融券":              pOrder.sFlag = 2          elif comboBoxFlag.get() == "無券":              pOrder.sFlag = 3            if comboBoxBuySell.get() == "買進":              pOrder.sBuySell = 0          elif comboBoxBuySell.get() == "賣出":              pOrder.sBuySell = 1          pOrder.bstrPrice = textBoxbstrPrice.get()          pOrder.nQty = int(textBoxnQty.get())          if comboBoxnTradeType.get() == "ROD":              pOrder.nTradeType = 0          elif comboBoxnTradeType.get() == "IOC":              pOrder.nTradeType = 1          elif comboBoxnTradeType.get() == "FOK":              pOrder.nTradeType = 2          if comboBoxnSpecialTradeType.get() == "市價":              pOrder.nSpecialTradeType = 1          elif comboBoxnSpecialTradeType.get() == "限價":              pOrder.nSpecialTradeType = 2          bstrMessage,nCode= m\_pSKOrder.SendStockOrder(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendStockOrder】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### SendStockOddLotOrder

|  |  |  |
| --- | --- | --- |
| 送出證券盤中零股委託。 | | |
| **宣告** | Long SendStockOddLotOrder([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct STOCKORDER\* pOrder,[out] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| bAsyncOrder | 是否為非同步委託。 |
| pOrder | SKCOM元件中的 [STOCKORDER](#_盤中零股) 物件，將下單條件填入該物件後，再帶入此欄位中。 |
| bstrMessage | 同步委託：如果回傳值為 0表示委託成功，訊息內容則為13碼的委託序號。回傳值非0表示委託失敗，訊息內容為失敗原因。  非同步委託：參照[OnAsyncOrder](#_OnAsyncOrder)。 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。  當錯誤代碼為4碼時，可參考**文件「4.下單準備介紹」裡面的【錯誤代碼定義表】**，所屬委託物件亦有欄位值說明供參。  其他錯誤則由「交易主機」回傳錯誤代碼及錯誤原因，若仍有疑問請洽詢您所屬營業員。 | |
| 備註 | 使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder)取得。 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別SendForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖  其它部分可參考 1.[**SendStockOrder**](#_SendStockOrder)做法 |
| # buttonSendStockOddLotOrder          self.buttonSendStockOddLotOrder = tk.Button(self)          self.buttonSendStockOddLotOrder["text"] = "盤中零股送出"          self.buttonSendStockOddLotOrder["fg"] = "red"  # 設置文字顏色變red          self.buttonSendStockOddLotOrder["command"] = self.buttonSendStockOddLotOrder\_Click          self.buttonSendStockOddLotOrder.grid(row=9, column=1) |

|  |
| --- |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib)  # 是否為非同步委託  bAsyncOrder = False |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonSendStockOddLotOrder\_Click(self):          pOrder = sk.STOCKORDER()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxStockID.get()          pOrder.sPeriod = 4 # 盤中零股          pOrder.sFlag = 0 # 現股          if comboBoxBuySell.get() == "買進":              pOrder.sBuySell = 0          elif comboBoxBuySell.get() == "賣出":              pOrder.sBuySell = 1          pOrder.bstrPrice = textBoxbstrPrice.get()          pOrder.nQty = int(textBoxnQty.get())          bstrMessage,nCode= m\_pSKOrder.SendStockOddLotOrder(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendStockOddLotOrder】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### CorrectPriceBySeqNo

|  |  |  |
| --- | --- | --- |
| 證期權依序號改價。 | | |
| **宣告** | Long CorrectPriceBySeqNo([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] BSTR bstrAccount, [in] BSTR bstrSeqNo, [in] BSTR bstrPrice, [in] LONG nTradeType,[**out**] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| bAsyncOrder | 是否為非同步委託。 |
| bstrAccount | 證券：委託帳號 (BrokerID＋帳號)。  期貨：委託帳號 ( IB＋帳號) 。 |
| bstrSeqNo | 欲改價的委託序號。 |
| bstrPrice | 修改價格。 |
| nTradeType | **證券0:ROD**  期選0:ROD 1:IOC 2:FOK |
| bstrMessage | **同步委託：**如果回傳值為 0表示委託成功，訊息內容則為修改訊息。回傳值非0表示委託失敗，訊息內容為失敗原因。  **非同步委託：**參照[OnAsyncOrder](#_OnAsyncOrder)。 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。 | |
| **備註** | 回傳值0 表示委託伺服器接收成功，詳細委託狀態仍須以委託**改價**回報內容為主。  其他非 0 數值表示異常，詳細原因請參考 Log 內容說明。  使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder)取得。  ＊證券逐筆nTradeType:請設0(ROD).  ＊不適用盤中零股 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別UpdateForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # textBoxSeqNo          tk.Label(self, text = "請輸入委託序號").grid(row=0, column=1)              #輸入框          self.textBoxSeqNo = tk.Entry(self)          self.textBoxSeqNo.grid(row=0, column=2)          global textBoxSeqNo          textBoxSeqNo = self.textBoxSeqNo          # textBoxBookNo          tk.Label(self, text = "請輸入委託書號").grid(row=1, column=1)              #輸入框          self.textBoxBookNo = tk.Entry(self)          self.textBoxBookNo.grid(row=1, column=2)          global textBoxBookNo          textBoxBookNo = self.textBoxBookNo          # textBoxCancelOrderByStockNo          tk.Label(self, text = "請輸入商品代號(空白就刪除所有委託)").grid(row=2, column=1)              #輸入框          self.textBoxCancelOrderByStockNo = tk.Entry(self)          self.textBoxCancelOrderByStockNo.grid(row=2, column=2)          global textBoxCancelOrderByStockNo          textBoxCancelOrderByStockNo = self.textBoxCancelOrderByStockNo          # textBoxStockDecreaseQty          tk.Label(self, text = "請輸入減少數量").grid(row=4, column=1)              #輸入框          self.textBoxStockDecreaseQty = tk.Entry(self)          self.textBoxStockDecreaseQty.grid(row=4, column=2)          global textBoxStockDecreaseQty          textBoxStockDecreaseQty = self.textBoxStockDecreaseQty          # textBoxPrice          tk.Label(self, text = "請輸入修改價格").grid(row=6, column=1)              #輸入框          self.textBoxPrice = tk.Entry(self)          self.textBoxPrice.grid(row=6, column=2)          global textBoxPrice          textBoxPrice = self.textBoxPrice          # comboBoxTradeType          tk.Label(self, text = "ROD/IOC/FOK").grid(row=7, column=1)              #輸入框          self.comboBoxTradeType = ttk.Combobox(self, state='readonly')          self.comboBoxTradeType['values'] = Config.comboBoxnTradeType          self.comboBoxTradeType.grid(row=7, column=2)          global comboBoxTradeType          comboBoxTradeType = self.comboBoxTradeType          # comboBoxMarketSymbol          tk.Label(self, text = "請輸入市場(※書號)").grid(row=8, column=1)              #輸入框          self.comboBoxMarketSymbol = ttk.Combobox(self, state='readonly')          self.comboBoxMarketSymbol['values'] = Config.comboBoxMarketSymbol          self.comboBoxMarketSymbol.grid(row=8, column=2)          global comboBoxMarketSymbol          comboBoxMarketSymbol = self.comboBoxMarketSymbol          # buttonCorrectPriceBySeqNo          self.buttonCorrectPriceBySeqNo = tk.Button(self)          self.buttonCorrectPriceBySeqNo["text"] = "改價(序號)"          self.buttonCorrectPriceBySeqNo["command"] = self.buttonCorrectPriceBySeqNo\_Click          self.buttonCorrectPriceBySeqNo.grid(row=9, column=1) |

|  |
| --- |
| Config.py |
| comboBoxTradeType = (      "0:ROD", "1:IOC", "2:FOK"  )  comboBoxMarketSymbol = (      "TS:證券", "TF:期貨", "TO:選擇權"  ) |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib)  # 是否為非同步委託  bAsyncOrder = False |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonCorrectPriceBySeqNo\_Click(self):          if comboBoxTradeType.get() == "0:ROD":              nTradeType = 0          elif  comboBoxTradeType.get() == "1:IOC":              nTradeType = 1          else:              nTradeType = 2 # FOK          bstrMessage,nCode= m\_pSKOrder.CorrectPriceBySeqNo(comboBoxUserID.get(), bAsyncOrder, comboBoxAccount.get(), textBoxSeqNo.get(), textBoxPrice.get(), nTradeType)          msg = "【CorrectPriceBySeqNo】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### CorrectPriceByBookNo

|  |  |  |
| --- | --- | --- |
| 證期權依書號改價。 | | |
| **宣告** | Long CorrectPriceByBookNo([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] BSTR bstrAccount, [in] BSTR bstrMarketSymbol, [in] BSTR bstrBookNo, [in] BSTR bstrPrice, [in] LONG nTradeType,[**out**] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| bAsyncOrder | 是否為非同步委託。 |
| bstrAccount | 證券：委託帳號 (BrokerID＋帳號)。  期貨：委託帳號 ( IB＋帳號) 。 |
| bstrMarketSymbol | 市場類別：TS: 證券TF:期貨 TO:選擇權 |
| bstrBookNo | 欲改量的委託書號。 |
| bstrPrice | 修改價格。 |
| nTradeType | **證券0:ROD**  期選0:ROD 1:IOC 2:FOK |
| bstrMessage | **同步委託：**如果回傳值為 0表示委託成功，訊息內容則為修改訊息。回傳值非0表示委託失敗，訊息內容為失敗原因。  **非同步委託：**參照[OnAsyncOrder](#_OnAsyncOrder)。 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。 | |
| **備註** | 回傳值0 表示委託伺服器接收成功，詳細委託狀態仍須以委託**改價**回報內容為主。  其他非 0 數值表示異常，詳細原因請參考 Log 內容說明。  使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder)取得。  ＊證券逐筆nTradeType:請設0(ROD).  ＊不適用盤中零股 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別UpdateForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖  其它部分可參考[3. CorrectPriceBySeqNo](#_CorrectPriceBySeqNo_1)做法 |
| # buttonCorrectPriceByBookNo          self.buttonCorrectPriceByBookNo = tk.Button(self)          self.buttonCorrectPriceByBookNo["text"] = "改價(書號)"          self.buttonCorrectPriceByBookNo["command"] = self.buttonCorrectPriceByBookNo\_Click          self.buttonCorrectPriceByBookNo.grid(row=9, column=2) |

|  |
| --- |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib)  # 是否為非同步委託  bAsyncOrder = False |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonCorrectPriceByBookNo\_Click(self):            if comboBoxMarketSymbol.get() == "TS:證券":              bstrMarketSymbol = "TS"          elif  comboBoxMarketSymbol.get() == "TF:期貨":              bstrMarketSymbol = "TF"          else:              bstrMarketSymbol = "TO" # TO:選擇權            if comboBoxTradeType.get() == "0:ROD":              nTradeType = 0          elif  comboBoxTradeType.get() == "1:IOC":              nTradeType = 1          else:              nTradeType = 2 # FOK          bstrMessage,nCode= m\_pSKOrder.CorrectPriceByBookNo(comboBoxUserID.get(), bAsyncOrder, comboBoxAccount.get(), bstrMarketSymbol, textBoxSeqNo.get(), textBoxPrice.get(), nTradeType)          msg = "【CorrectPriceByBookNo】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### DecreaseOrderBySeqNo

|  |  |  |
| --- | --- | --- |
| 委託減量。 | | |
| **宣告** | Long DecreaseOrderBySeqNo([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] BSTR bstrAccount, [in] BSTR bstrSeqNo, [in] LONG nDecreaseQty,[**out**] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| bAsyncOrder | 是否為非同步委託。 |
| bstrAccount | 證券：委託帳號 (BrokerID＋帳號)。  期貨：委託帳號 ( IB＋帳號) 。 |
| bstrSeqNo | 欲改量的委託序號。 |
| nDecreaseQty | 欲減少的數量。 |
| bstrMessage | **同步委託：**如果回傳值為 0表示委託成功，訊息內容則為修改訊息。回傳值非0表示委託失敗，訊息內容為失敗原因。  **非同步委託：**參照[OnAsyncOrder](#_OnAsyncOrder)。 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。  當錯誤代碼為4碼時，可參考**文件「4.下單準備介紹」裡面的【錯誤代碼定義表】**，所屬委託物件亦有欄位值說明供參。  **其他錯誤則由「交易主機」回傳錯誤代碼及錯誤原因。若仍有疑問請洽詢您所屬營業員。** | |
| **備註** | 回傳值0 表示委託伺服器**接收成功**，詳細委託狀態仍須以委託**減量**回報內容為主。  其他非 0 數值表示異常，詳細原因請參考 Log 內容說明。  使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder)取得。  ＊亦適用證券逐筆及擬真平台 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別UpdateForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖  其它部分可參考[3. CorrectPriceBySeqNo](#_CorrectPriceBySeqNo_1)做法 |
| # buttonDecreaseOrderBySeqNo          self.buttonDecreaseOrderBySeqNo = tk.Button(self)          self.buttonDecreaseOrderBySeqNo["text"] = "減量(序號)"          self.buttonDecreaseOrderBySeqNo["command"] = self.buttonDecreaseOrderBySeqNo\_Click          self.buttonDecreaseOrderBySeqNo.grid(row=5, column=1) |

|  |
| --- |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib)  # 是否為非同步委託  bAsyncOrder = False |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonDecreaseOrderBySeqNo\_Click(self):          bstrMessage,nCode= m\_pSKOrder.DecreaseOrderBySeqNo(comboBoxUserID.get(), bAsyncOrder, comboBoxAccount.get(), textBoxSeqNo.get(), int(textBoxStockDecreaseQty.get()))          msg = "【DecreaseOrderBySeqNo】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### CancelOrderBySeqNo

|  |  |  |
| --- | --- | --- |
| 國內委託删單(By委託序號)。 | | |
| **宣告** | Long CancelOrderBySeqNo([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] BSTR bstrAccount, [in] BSTR bstrSeqNo,[**out**] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| bAsyncOrder | 是否為非同步委託。 |
| bstrAccount | 證券：委託帳號 (BrokerID＋帳號)。  期貨：委託帳號 ( IB＋帳號) 。 |
| bstrSeqNo | 欲刪除的委託序號。 |
| bstrMessage | **同步刪單：**如果回傳值為 0表示刪單成功，訊息內容則為刪單訊息。回傳值非0表示刪單失敗，訊息內容為失敗原因。  **非同步刪單：**參照[OnAsyncOrder](#_OnAsyncOrder)。 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。  當錯誤代碼為4碼時，可參考**文件「4.下單準備介紹」裡面的【錯誤代碼定義表】**，所屬委託物件亦有欄位值說明供參。  **其他錯誤則由「交易主機」回傳錯誤代碼及錯誤原因。若仍有疑問請洽詢您所屬營業員。** | |
| **備註** | 回傳值0 表示委託伺服器**接收成功**，詳細委託狀態仍須以委託**刪單**回報內容為主。  其他非 0 數值表示異常，詳細原因請參考 Log 內容說明。  使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder)取得。  ＊亦適用證券逐筆及擬真平台 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別UpdateForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖  其它部分可參考[3. CorrectPriceBySeqNo](#_CorrectPriceBySeqNo_1)做法 |
| # buttonCancelOrderBySeqNo          self.buttonCancelOrderBySeqNo = tk.Button(self)          self.buttonCancelOrderBySeqNo["text"] = "刪單(序號)"          self.buttonCancelOrderBySeqNo["command"] = self.buttonCancelOrderBySeqNo\_Click          self.buttonCancelOrderBySeqNo.grid(row=3, column=1) |

|  |
| --- |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib)  # 是否為非同步委託  bAsyncOrder = False |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonCancelOrderBySeqNo\_Click(self):          bstrMessage,nCode= m\_pSKOrder.CancelOrderBySeqNo(comboBoxUserID.get(), bAsyncOrder, comboBoxAccount.get(), textBoxSeqNo.get())          msg = "【CancelOrderBySeqNo】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### CancelOrderByBookNo

|  |  |  |
| --- | --- | --- |
| 國內委託删單(By書號)。 | | |
| **宣告** | Long CancelOrderByBookNo([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] BSTR bstrAccount, [in] BSTR bstrBookNo,[**out**] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| bAsyncOrder | 是否為非同步委託。 |
| bstrAccount | 證券：委託帳號 (BrokerID＋帳號)。  期貨：委託帳號 ( IB＋帳號) 。 |
| bstrBookNo | 欲刪除的委託書號。 |
| bstrMessage | **同步刪單：**如果回傳值為 0表示刪單成功，訊息內容則為刪單訊息。回傳值非0表示刪單失敗，訊息內容為失敗原因。  **非同步刪單：**參照OnAsyncOrder。 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。 | |
| **備註** | 回傳值0 表示委託伺服器接收成功，詳細委託狀態仍須以委託回報內容為主。  其他非 0 數值表示異常，詳細原因請參考 Log 內容說明。  使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder)取得。 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別UpdateForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖  其它部分可參考[3. CorrectPriceBySeqNo](#_CorrectPriceBySeqNo_1)做法 |
| # buttonCancelOrderByBookNo          self.buttonCancelOrderByBookNo = tk.Button(self)          self.buttonCancelOrderByBookNo["text"] = "刪單(書號)"          self.buttonCancelOrderByBookNo["command"] = self.buttonCancelOrderByBookNo\_Click          self.buttonCancelOrderByBookNo.grid(row=3, column=2) |

|  |
| --- |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib)  # 是否為非同步委託  bAsyncOrder = False |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonCancelOrderByBookNo\_Click(self):          bstrMessage,nCode= m\_pSKOrder.CancelOrderByBookNo(comboBoxUserID.get(), bAsyncOrder, comboBoxAccount.get(), textBoxBookNo.get())          msg = "【CancelOrderByBookNo】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### CancelOrderByStockNo

|  |  |  |
| --- | --- | --- |
| 國內委託删單(By帳號所屬**登入ID**+商品代號)。 | | |
| **宣告** | Long CancelOrderByStockNo([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] BSTR bstrAccount, [in] BSTR bstrStockNo,[**out**] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| bAsyncOrder | 是否為非同步委託。 |
| bstrAccount | 證券：委託帳號 (BrokerID＋帳號)。  期貨：委託帳號 ( IB＋帳號) 。 |
| bstrStockNo | 欲刪除的委託商品代號，請輸入有效商品代碼。 |
| bstrMessage | **同步刪單：**如果回傳值為 0表示刪單成功，訊息內容則為刪單訊息。回傳值非0表示刪單失敗，訊息內容為失敗原因。  **非同步刪單：**參照[OnAsyncOrder](#_OnAsyncOrder)。 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。  當錯誤代碼為4碼時，可參考**文件「4.下單準備介紹」裡面的【錯誤代碼定義表】**，所屬委託物件亦有欄位值說明供參。  **其他錯誤則由「交易主機」回傳錯誤代碼及錯誤原因。若仍有疑問請洽詢您所屬營業員。** | |
| **備註** | 回傳值0 表示委託伺服器**接收成功**，詳細委託狀態仍須以委託**刪單**回報內容為主。  其他非 0 數值表示異常，詳細原因請參考 Log 內容說明。  使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder)取得。  ＊請留意證券代碼，包含刪除盤中零股委託 | |

|  |  |  |
| --- | --- | --- |
| 異動前v2.13.45 | 欄位 | 異動後V2.13.52 |
| 欲刪除的委託商品代號，請輸入有效商品代碼。 | bstrStockNo  欄位說明：  帶空字串，可刪除帳號下所有委託。 | 商品代碼為空，則為刪除全部委託。 |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別UpdateForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖  其它部分可參考[3. CorrectPriceBySeqNo](#_CorrectPriceBySeqNo_1)做法 |
| # buttonCancelOrderByStockNo          self.buttonCancelOrderByStockNo = tk.Button(self)          self.buttonCancelOrderByStockNo["text"] = "刪單(代號)"          self.buttonCancelOrderByStockNo["command"] = self.buttonCancelOrderByStockNo\_Click          self.buttonCancelOrderByStockNo.grid(row=3, column=3) |

|  |
| --- |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib)  # 是否為非同步委託  bAsyncOrder = False |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonCancelOrderByStockNo\_Click(self):          bstrMessage,nCode= m\_pSKOrder.CancelOrderByStockNo(comboBoxUserID.get(), bAsyncOrder, comboBoxAccount.get(), textBoxCancelOrderByStockNo.get())          msg = "【CancelOrderByStockNo】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### CancelOrderByStockNoAdvance

|  |  |  |
| --- | --- | --- |
| 國內委託删單(By帳號所屬**登入ID**+商品代號+買賣別+委託價格)。 | | |
| **宣告** | Long CancelOrderByStockNoAdvance([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] BSTR bstrAccount, [in] BSTR bstrStockNo, [in] LONG nBuySell, [in] BSTR bstrPrice, [**out**] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| bAsyncOrder | 是否為非同步委託。 |
| bstrAccount | 證券：委託帳號 (BrokerID＋帳號)。  期貨：委託帳號 ( IB＋帳號) 。 |
| bstrStockNo | 欲刪除的委託商品代號，請輸入有效商品代碼。 |
| nBuySell | 0:買進 1:賣出 |
| bstrPrice | 委託價格(不使用請帶空字串) |
| bstrMessage | **同步刪單：**如果回傳值為 0表示刪單成功，訊息內容則為刪單訊息。回傳值非0表示刪單失敗，訊息內容為失敗原因。  **非同步刪單：**參照[OnAsyncOrder](#_OnAsyncOrder) |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。  當錯誤代碼為4碼時，可參考**文件「4.下單準備介紹」裡面的【錯誤代碼定義表】**，所屬委託物件亦有欄位值說明供參。  **其他錯誤則由「交易主機」回傳錯誤代碼及錯誤原因。若仍有疑問請洽詢您所屬營業員。** | |
| **備註** | 回傳值0 表示委託伺服器**接收成功**，詳細委託狀態仍須以委託**刪單**回報內容為主。  其他非 0 數值表示異常，詳細原因請參考 Log 內容說明。  使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder)取得。  ＊請留意證券代碼，包含刪除盤中零股委託 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別UpdateForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖    其它部分可參考[3. CorrectPriceBySeqNo](#_CorrectPriceBySeqNo_1)做法 |
| # buttonCancelOrderByStockNo          self.buttonCancelOrderByStockNo = tk.Button(self)          self.buttonCancelOrderByStockNo["text"] = "刪單(代號)"          self.buttonCancelOrderByStockNo["command"] = self.buttonCancelOrderByStockNo\_Click          self.buttonCancelOrderByStockNo.grid(row=3, column=3)  # comboBoxCancelOrderByStockNoAdvanceBuySell  tk.Label(self, text = "買/賣/無").grid(row=2, column=3)  #輸入框  self.comboBoxCancelOrderByStockNoAdvanceBuySell = ttk.Combobox(self, state='readonly')  self.comboBoxCancelOrderByStockNoAdvanceBuySell['values'] = Config.comboBoxCancelOrderByStockNoAdvanceBuySell  self.comboBoxCancelOrderByStockNoAdvanceBuySell.grid(row=2, column=4)  # textBoxCancelOrderByStockNoAdvancePrice  tk.Label(self, text = "委託價格").grid(row=2, column=5)  #輸入框  self.textBoxCancelOrderByStockNoAdvancePrice = tk.Entry(self)  self.textBoxCancelOrderByStockNoAdvancePrice.grid(row=2, column=6)  global textBoxCancelOrderByStockNoAdvancePrice  textBoxCancelOrderByStockNoAdvancePrice = self.textBoxCancelOrderByStockNoAdvancePrice |

|  |
| --- |
| Config.py |
| comboBoxCancelOrderByStockNoAdvanceBuySell =(     "買", "賣", "無"  ) |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib)  # 是否為非同步委託  bAsyncOrder = False |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonCancelOrderByStockNo\_Click(self):  if comboBoxCancelOrderByStockNoAdvanceBuySell.get() == "無":  bstrMessage,nCode= m\_pSKOrder.CancelOrderByStockNo(comboBoxUserID.get(), bAsyncOrder, comboBoxAccount.get(), textBoxCancelOrderByStockNo.get())  msg = "【CancelOrderByStockNo】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage  elif comboBoxCancelOrderByStockNoAdvanceBuySell.get() == "買":  nBuySell = 0  bstrMessage,nCode= m\_pSKOrder.CancelOrderByStockNoAdvance(comboBoxUserID.get(), bAsyncOrder, comboBoxAccount.get(), textBoxCancelOrderByStockNo.get(), nBuySell, textBoxCancelOrderByStockNoAdvancePrice.get())  msg = "【CancelOrderByStockNoAdvance】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage  else:  nBuySell = 1  bstrMessage,nCode= m\_pSKOrder.CancelOrderByStockNoAdvance(comboBoxUserID.get(), bAsyncOrder, comboBoxAccount.get(), textBoxCancelOrderByStockNo.get(), nBuySell, textBoxCancelOrderByStockNoAdvancePrice.get())  msg = "【CancelOrderByStockNoAdvance】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage  richTextBoxMethodMessage.insert('end', msg + "\n")  richTextBoxMethodMessage.see('end') |

#### **Proxy下單/刪改單**

##### SendStockProxyOrder

|  |  |  |
| --- | --- | --- |
| 經由proxy server送出證券委託。 | | |
| **宣告** | long SendStockProxyOrder([in]BSTR bstrLogInID, [in]struct STOCKPROXYORDER\* pSTOCKPROXYORDER, [out] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| pSTOCKPROXYORDER | SKCOM元件中的 [STOCKPROXYORDER](#_Proxy_Server_證券下單) 物件，將下單條件填入該物件後，再帶入此欄位中。 |
| bstrMessage | **委託送出成功：ORKEY**  **委託送出失敗：錯誤訊息** |
| **回傳值** | 0表示連線成功，其餘非0數值都表示失敗。 | |
| **備註** | 透過Proxy Server委託，委託結果請由[OnProxyOrder](#_OnProxyOrder)取得。  連線且成功登入，通知OnProxyStatus為5001時，送至proxy server進行下單。 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別SendProxyForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # textBoxStockIDProxy          tk.Label(self, text = "股票代號").grid(row=0, column=1)              #輸入框          self.textBoxStockIDProxy = tk.Entry(self)          self.textBoxStockIDProxy.grid(row=0, column=2)          global textBoxStockIDProxy          textBoxStockIDProxy = self.textBoxStockIDProxy          # comboBoxPeriodProxy          tk.Label(self, text = "盤中/盤後/零股").grid(row=1, column=1)              #輸入框          self.comboBoxPeriodProxy = ttk.Combobox(self, state='readonly')          self.comboBoxPeriodProxy['values'] = Config.comboBoxPeriodProxy          self.comboBoxPeriodProxy.grid(row=1, column=2)          global comboBoxPeriodProxy          comboBoxPeriodProxy = self.comboBoxPeriodProxy          # textBoxnQtyProxy          tk.Label(self, text = "股數").grid(row=2, column=1)              #輸入框          self.textBoxnQtyProxy = tk.Entry(self)          self.textBoxnQtyProxy.grid(row=2, column=2)          global textBoxnQtyProxy          textBoxnQtyProxy = self.textBoxnQtyProxy          # comboBoxbstrOrderTypeProxy          tk.Label(self, text = "現股/融資/融券/無券").grid(row=3, column=1)              #輸入框          self.comboBoxbstrOrderTypeProxy = ttk.Combobox(self, state='readonly')          self.comboBoxbstrOrderTypeProxy['values'] = Config.comboBoxbstrOrderTypeProxy          self.comboBoxbstrOrderTypeProxy.grid(row=3, column=2)          global comboBoxbstrOrderTypeProxy          comboBoxbstrOrderTypeProxy = self.comboBoxbstrOrderTypeProxy          # comboBoxnTradeTypeProxy          tk.Label(self, text = "ROD/IOC/FOK").grid(row=4, column=1)              #輸入框          self.comboBoxnTradeTypeProxy = ttk.Combobox(self, state='readonly')          self.comboBoxnTradeTypeProxy['values'] = Config.comboBoxnTradeTypeProxy          self.comboBoxnTradeTypeProxy.grid(row=4, column=2)          global comboBoxnTradeTypeProxy          comboBoxnTradeTypeProxy = self.comboBoxnTradeTypeProxy          # comboBoxnSpecialTradeTypeProxy          tk.Label(self, text = "市價/限價").grid(row=5, column=1)              #輸入框          self.comboBoxnSpecialTradeTypeProxy = ttk.Combobox(self, state='readonly')          self.comboBoxnSpecialTradeTypeProxy['values'] = Config.comboBoxnSpecialTradeTypeProxy          self.comboBoxnSpecialTradeTypeProxy.grid(row=5, column=2)          global comboBoxnSpecialTradeTypeProxy          comboBoxnSpecialTradeTypeProxy = self.comboBoxnSpecialTradeTypeProxy          # textBoxbstrPriceProxy          tk.Label(self, text = "委託價").grid(row=6, column=1)              #輸入框          self.textBoxbstrPriceProxy = tk.Entry(self)          self.textBoxbstrPriceProxy.grid(row=6, column=2)          global textBoxbstrPriceProxy          textBoxbstrPriceProxy = self.textBoxbstrPriceProxy          # comboBoxnPriceMarkProxy          tk.Label(self, text = "價格旗標").grid(row=7, column=1)              #輸入框          self.comboBoxnPriceMarkProxy = ttk.Combobox(self, state='readonly')          self.comboBoxnPriceMarkProxy['values'] = Config.comboBoxnPriceMarkProxy          self.comboBoxnPriceMarkProxy.grid(row=7, column=2)          global comboBoxnPriceMarkProxy          comboBoxnPriceMarkProxy = self.comboBoxnPriceMarkProxy          # buttonSendStockProxyOrder          self.buttonSendStockProxyOrder = tk.Button(self)          self.buttonSendStockProxyOrder["text"] = "證券送出"          self.buttonSendStockProxyOrder["command"] = self.buttonSendStockProxyOrder\_Click          self.buttonSendStockProxyOrder.grid(row=8, column=2) |

|  |
| --- |
| Config.py |
| # SendProxyForm  comboBoxPeriodProxy = (      "盤中", "零股", "盤後交易", "盤中零股"  )  comboBoxbstrOrderTypeProxy = (      "現股買進", "現股賣出", "融資買進", "融資賣出", "融券買進", "融券賣出", "無券賣出"  )  comboBoxnTradeTypeProxy =(      "ROD", "IOC", "FOK"  )  comboBoxnSpecialTradeTypeProxy =(      "市價", "限價"  )  comboBoxnPriceMarkProxy =(      "一般定價", "前日收盤價", "漲停", "跌停"  ) |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib)  # 是否為非同步委託  bAsyncOrder = False |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonSendStockProxyOrder\_Click(self):          pSTOCKPROXYORDER = sk.STOCKPROXYORDER()          pSTOCKPROXYORDER.bstrFullAccount = comboBoxAccount.get()          pSTOCKPROXYORDER.bstrStockNo = textBoxStockIDProxy.get()          if comboBoxbstrOrderTypeProxy.get() == "現股買進":              pSTOCKPROXYORDER.bstrOrderType = "1"          elif comboBoxbstrOrderTypeProxy.get() == "現股賣出":              pSTOCKPROXYORDER.bstrOrderType = "2"          elif comboBoxbstrOrderTypeProxy.get() == "融資買進":              pSTOCKPROXYORDER.bstrOrderType = "3"          elif comboBoxbstrOrderTypeProxy.get() == "融資賣出":              pSTOCKPROXYORDER.bstrOrderType = "4"          elif comboBoxbstrOrderTypeProxy.get() == "融券買進":              pSTOCKPROXYORDER.bstrOrderType = "5"          elif comboBoxbstrOrderTypeProxy.get() == "融券賣出":              pSTOCKPROXYORDER.bstrOrderType = "6"          elif comboBoxbstrOrderTypeProxy.get() == "無券賣出":              pSTOCKPROXYORDER.bstrOrderType = "7"          if comboBoxnSpecialTradeTypeProxy.get() == "市價":              pSTOCKPROXYORDER.nSpecialTradeType = 1          elif comboBoxnSpecialTradeTypeProxy.get() == "限價":              pSTOCKPROXYORDER.nSpecialTradeType = 2          if comboBoxPeriodProxy.get() == "盤中":              pSTOCKPROXYORDER.nPeriod = 0          elif comboBoxPeriodProxy.get() == "零股":              pSTOCKPROXYORDER.nPeriod = 1          elif comboBoxPeriodProxy.get() == "盤後交易":              pSTOCKPROXYORDER.nPeriod = 2          elif comboBoxPeriodProxy.get() == "盤中零股":              pSTOCKPROXYORDER.nPeriod = 3          pSTOCKPROXYORDER.bstrPrice = textBoxbstrPriceProxy.get()          pSTOCKPROXYORDER.nQty = int(textBoxnQtyProxy.get())          if comboBoxnTradeTypeProxy.get() == "ROD":              pSTOCKPROXYORDER.nTradeType = 0          elif comboBoxnTradeTypeProxy.get() == "IOC":              pSTOCKPROXYORDER.nTradeType = 1          elif comboBoxnTradeTypeProxy.get() == "FOK":              pSTOCKPROXYORDER.nTradeType = 2            if comboBoxnPriceMarkProxy.get() == "一般定價":              pSTOCKPROXYORDER.nPriceMark = 0          elif comboBoxnPriceMarkProxy.get() == "前日收盤價":              pSTOCKPROXYORDER.nPriceMark = 1          elif comboBoxnPriceMarkProxy.get() == "漲停":              pSTOCKPROXYORDER.nPriceMark = 2          elif comboBoxnPriceMarkProxy.get() == "跌停":              pSTOCKPROXYORDER.nPriceMark = 3            bstrMessage,nCode= m\_pSKOrder.SendStockProxyOrder(comboBoxUserID.get(), pSTOCKPROXYORDER)          if bstrMessage is not None:               msg = "【SendStockProxyOrder】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          else:               msg = "【SendStockProxyOrder】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + " No message"          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### SendStockProxyAlter

|  |  |  |
| --- | --- | --- |
| 經由proxy server送出證券刪改單。 | | |
| **宣告** | long SendStockProxyAlter([in]BSTR bstrLogInID, [in]struct STOCKPROXYORDER\* pSTOCKPROXYORDER, [out] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| pSTOCKPROXYORDER | SKCOM元件中的[STOCKPROXYORDER](#_Proxy_Server_證券刪改單) 物件，將下單條件填入該物件後，再帶入此欄位中。 |
| bstrMessage | **委託送出成功：ORKEY**  **委託送出失敗：錯誤訊息** |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。錯誤代碼可參考對照表。 | |
| **備註** | 透過Proxy Server委託，委託結果請由[OnProxyOrder](#_OnProxyOrder)取得。  連線且成功登入，通知OnProxyStatus為5001時，送至proxy server進行下單。 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py類別SendProxyForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # comboBoxbstrOrderTypeProxyAlter          tk.Label(self, text = "下單類別").grid(row=0, column=1)              #輸入框          self.comboBoxbstrOrderTypeProxyAlter = ttk.Combobox(self, state='readonly')          self.comboBoxbstrOrderTypeProxyAlter['values'] = Config.comboBoxbstrOrderTypeProxyAlter          self.comboBoxbstrOrderTypeProxyAlter.grid(row=0, column=2)          global comboBoxbstrOrderTypeProxyAlter          comboBoxbstrOrderTypeProxyAlter = self.comboBoxbstrOrderTypeProxyAlter          # textBoxSeqNoProxyAlter          tk.Label(self, text = "請輸入委託序號").grid(row=1, column=1)              #輸入框          self.textBoxSeqNoProxyAlter = tk.Entry(self)          self.textBoxSeqNoProxyAlter.grid(row=1, column=2)          global textBoxSeqNoProxyAlter          textBoxSeqNoProxyAlter = self.textBoxSeqNoProxyAlter          # textBoxBookNoProxyAlter          tk.Label(self, text = "請輸入委託書號").grid(row=2, column=1)              #輸入框          self.textBoxBookNoProxyAlter = tk.Entry(self)          self.textBoxBookNoProxyAlter.grid(row=2, column=2)          global textBoxBookNoProxyAlter          textBoxBookNoProxyAlter = self.textBoxBookNoProxyAlter          # textBoxCancelOrderByStockNoProxyAlter          tk.Label(self, text = "委託股票代號").grid(row=3, column=1)              #輸入框          self.textBoxCancelOrderByStockNoProxyAlter = tk.Entry(self)          self.textBoxCancelOrderByStockNoProxyAlter.grid(row=3, column=2)          global textBoxCancelOrderByStockNoProxyAlter          textBoxCancelOrderByStockNoProxyAlter = self.textBoxCancelOrderByStockNoProxyAlter          # comboBoxnSpecialTradeTypeProxyAlter          tk.Label(self, text = "市價/限價").grid(row=4, column=1)              #輸入框          self.comboBoxnSpecialTradeTypeProxyAlter = ttk.Combobox(self, state='readonly')          self.comboBoxnSpecialTradeTypeProxyAlter['values'] = Config.comboBoxnSpecialTradeTypeProxyAlter          self.comboBoxnSpecialTradeTypeProxyAlter.grid(row=4, column=2)          global comboBoxnSpecialTradeTypeProxyAlter          comboBoxnSpecialTradeTypeProxyAlter = self.comboBoxnSpecialTradeTypeProxyAlter          # comboBoxPeriodProxyAlter          tk.Label(self, text = "盤中/盤後/零股").grid(row=5, column=1)              #輸入框          self.comboBoxPeriodProxyAlter = ttk.Combobox(self, state='readonly')          self.comboBoxPeriodProxyAlter['values'] = Config.comboBoxPeriodProxyAlter          self.comboBoxPeriodProxyAlter.grid(row=5, column=2)          global comboBoxPeriodProxyAlter          comboBoxPeriodProxyAlter = self.comboBoxPeriodProxyAlter          # textBoxStockDecreaseQtyProxyAlter          tk.Label(self, text = "請輸入減少數量").grid(row=6, column=1)              #輸入框          self.textBoxStockDecreaseQtyProxyAlter = tk.Entry(self)          self.textBoxStockDecreaseQtyProxyAlter.grid(row=6, column=2)          global textBoxStockDecreaseQtyProxyAlter          textBoxStockDecreaseQtyProxyAlter = self.textBoxStockDecreaseQtyProxyAlter          # textBoxPriceProxyAlter          tk.Label(self, text = "請輸入修改價格").grid(row=7, column=1)              #輸入框          self.textBoxPriceProxyAlter = tk.Entry(self)          self.textBoxPriceProxyAlter.grid(row=7, column=2)          global textBoxPriceProxyAlter          textBoxPriceProxyAlter = self.textBoxPriceProxyAlter          # comboBoxTradeTypeProxyAlter          tk.Label(self, text = "ROD/IOC/FOK").grid(row=8, column=1)              #輸入框          self.comboBoxTradeTypeProxyAlter = ttk.Combobox(self, state='readonly')          self.comboBoxTradeTypeProxyAlter['values'] = Config.comboBoxTradeTypeProxyAlter          self.comboBoxTradeTypeProxyAlter.grid(row=8, column=2)          global comboBoxTradeTypeProxyAlter          comboBoxTradeTypeProxyAlter = self.comboBoxTradeTypeProxyAlter          # comboBoxnPriceMarkProxyAlter          tk.Label(self, text = "價格旗標").grid(row=9, column=1)              #輸入框          self.comboBoxnPriceMarkProxyAlter = ttk.Combobox(self, state='readonly')          self.comboBoxnPriceMarkProxyAlter['values'] = Config.comboBoxnPriceMarkProxyAlter          self.comboBoxnPriceMarkProxyAlter.grid(row=9, column=2)          global comboBoxnPriceMarkProxyAlter          comboBoxnPriceMarkProxyAlter = self.comboBoxnPriceMarkProxyAlter          # buttonSendStockProxyAlter          self.buttonSendStockProxyAlter = tk.Button(self)          self.buttonSendStockProxyAlter["text"] = "刪改單送出"          self.buttonSendStockProxyAlter["command"] = self.buttonSendStockProxyAlter\_Click          self.buttonSendStockProxyAlter.grid(row=10, column=2) |

|  |
| --- |
| Config.py |
| # UpdateProxyForm  comboBoxbstrOrderTypeProxyAlter =(      "刪單", "改量", "改價"  )  comboBoxnSpecialTradeTypeProxyAlter =(      "市價", "限價"  )  comboBoxPeriodProxyAlter =(      "盤中", "零股", "盤後交易", "盤中零股"  )  comboBoxTradeTypeProxyAlter =(      "0:ROD", "1:IOC", "2:FOK"  )  comboBoxnPriceMarkProxyAlter =(      "一般定價", "前日收盤價", "漲停", "跌停"  ) |
| 全域變數 |
| # 群益API元件導入Python code內用的物件宣告  m\_pSKCenter = comtypes.client.CreateObject(sk.SKCenterLib,interface=sk.ISKCenterLib)  m\_pSKOrder = comtypes.client.CreateObject(sk.SKOrderLib,interface=sk.ISKOrderLib)  # 是否為非同步委託  bAsyncOrder = False |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonSendStockProxyAlter\_Click(self):          pSTOCKPROXYORDER = sk.STOCKPROXYORDER()          pSTOCKPROXYORDER.bstrFullAccount = comboBoxAccount.get()          pSTOCKPROXYORDER.bstrStockNo = textBoxCancelOrderByStockNoProxyAlter.get()          if comboBoxbstrOrderTypeProxyAlter.get() == "刪單":              pSTOCKPROXYORDER.bstrOrderType = "0"          elif comboBoxbstrOrderTypeProxyAlter.get() == "改量":              pSTOCKPROXYORDER.bstrOrderType = "1"          elif comboBoxbstrOrderTypeProxyAlter.get() == "改價":              pSTOCKPROXYORDER.bstrOrderType = "2"          if comboBoxnSpecialTradeTypeProxyAlter.get() == "市價":              pSTOCKPROXYORDER.nSpecialTradeType = 1          elif comboBoxnSpecialTradeTypeProxyAlter.get() == "限價":              pSTOCKPROXYORDER.nSpecialTradeType = 2          if comboBoxPeriodProxyAlter.get() == "盤中":              pSTOCKPROXYORDER.nPeriod = 0          elif comboBoxPeriodProxyAlter.get() == "零股":              pSTOCKPROXYORDER.nPeriod = 1          elif comboBoxPeriodProxyAlter.get() == "盤後交易":              pSTOCKPROXYORDER.nPeriod = 2          elif comboBoxPeriodProxyAlter.get() == "盤中零股":              pSTOCKPROXYORDER.nPeriod = 3          pSTOCKPROXYORDER.bstrPrice = textBoxPriceProxyAlter.get()          pSTOCKPROXYORDER.nQty = int(textBoxStockDecreaseQtyProxyAlter.get())          if comboBoxTradeTypeProxyAlter.get() == "0:ROD":              pSTOCKPROXYORDER.nTradeType = 0          elif comboBoxTradeTypeProxyAlter.get() == "1:IOC":              pSTOCKPROXYORDER.nTradeType = 1          elif comboBoxTradeTypeProxyAlter.get() == "2:FOK":              pSTOCKPROXYORDER.nTradeType = 2            if comboBoxnPriceMarkProxyAlter.get() == "一般定價":              pSTOCKPROXYORDER.nPriceMark = 0          elif comboBoxnPriceMarkProxyAlter.get() == "前日收盤價":              pSTOCKPROXYORDER.nPriceMark = 1          elif comboBoxnPriceMarkProxyAlter.get() == "漲停":              pSTOCKPROXYORDER.nPriceMark = 2          elif comboBoxnPriceMarkProxyAlter.get() == "跌停":              pSTOCKPROXYORDER.nPriceMark = 3            pSTOCKPROXYORDER.bstrBookNo = textBoxBookNoProxyAlter.get()          pSTOCKPROXYORDER.bstrSeqNo = textBoxSeqNoProxyAlter.get()          bstrMessage,nCode= m\_pSKOrder.SendStockProxyAlter(comboBoxUserID.get(), pSTOCKPROXYORDER)          if bstrMessage is not None:               msg = "【SendStockProxyAlter】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          else:               msg = "【SendStockProxyAlter】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + " No message"          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### 3.SendStockProxyPreAlter

|  |  |  |
| --- | --- | --- |
| 經由proxy server送出證券特殊刪改單(帶入序號)。 | | |
| **宣告** | long SendStockProxyPreAlter([in] BSTR bstrLogInID, [in]struct STOCKPROXYORDER\* pSTOCKPROXYORDER, [out] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| pSTOCKPROXYORDER | SKCOM元件中的[STOCKPROXYORDER](#_Proxy_Server_證券刪改單) 物件，將下單條件填入該物件後，再帶入此欄位中。 |
| bstrMessage | **委託送出成功：ORKEY**  **委託送出失敗：錯誤訊息** |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。錯誤代碼可參考對照表。 | |
| **備註** | 透過Proxy Server委託，委託結果請由[OnProxyOrder](#_OnProxyOrder)取得。  連線且成功登入，通知OnProxyStatus為5001時，送至proxy server進行下單。 | |

## 事件

### OnRealBalanceReport

|  |  |  |
| --- | --- | --- |
| 證券即時庫存。透過呼叫[GetRealBalanceReport](#_GetRealBalanceReport)後，資訊由該事件回傳。  版本 **v2.13.42 ~ v2.13.54** 間，證券即時庫存查詢「**可資沖 / 可券沖**」欄位回傳順序有誤，兩者內容誤植互換。 自 **v2.13.55** 起已修正此問題，請升級 API 後留意欄位順序。 | | |
| **宣告** | void OnRealBalanceReport([in] BSTR bstrData); | |
| **參數** | bstrData | 每一筆資料以「,」分隔每一個欄位，欄位依序為：   1. 股票代號, 2. 庫存種類, (T:集保 C:融資 L:融券) 3. 資額度(原始), 4. 資額度(可用), 5. 券額度(原始), 6. 券額度(可用), 7. 股數:昨日庫存, 8. 今日委買, 9. 今日委賣, 10. 今日買進成交, 11. 今日賣出成交, 12. 今日資券可回補/集保庫存可賣出, 13. 可資沖股數, 14. 可券沖股數, 15. 即時庫存, 16. **X(此欄請忽略)** 17. **即時個股維持率,** 18. **LOGIN\_ID** 19. **ACCOUNT\_NO** |
| **備註** | 當全部資料已經全部回傳完畢，將回傳一筆以「##」開頭的內容，表示查詢結束。 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py |
| UI程式碼(放置於 def createWidgets(self): 的下一層)/UI示意圖 |
| # richTextBoxMessage          self.richTextBoxMessage = tk.Listbox(self, height=5)          self.richTextBoxMessage.grid(column = 0, row = 10, columnspan=5, sticky = tk.E + tk.W)          global richTextBoxMessage          richTextBoxMessage = self.richTextBoxMessage |

|  |
| --- |
| 事件(可放在事件SKOrderLibEvent註冊物件底下一層) |
| # 證券即時庫存。透過呼叫GetRealBalanceReport後，資訊由該事件回傳。      def OnRealBalanceReport(self, bstrData):          msg = "【OnRealBalanceReport】" + bstrData;          richTextBoxMessage.insert('end', msg + "\n")          richTextBoxMessage.see('end') |

### OnBalanceQuery

|  |  |  |
| --- | --- | --- |
| 集保庫存查詢。透過呼叫 [GetBalanceQuery](#_OnBalanceQuery_1)後，資訊由該事件回傳。 | | |
| **宣告** | void OnBalanceQuery ([in] BSTR bstrData); | |
| **參數** | bstrData | 回傳的內容以「,」分隔每一個欄位，欄位依序為：   |  |  | | --- | --- | | 參數 | 說明 | | 證券帳號 | 分公司4碼+帳號7碼 | | 股票代號 |  | | 股票股數 |  | | 融資股數 |  | | 融資金額 |  | | 融券股數 |  | | 融券金額 |  | | 保證金 |  | | 擔保品 |  | |
| **備註** | 回傳【M003】表示沒有庫存 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py |
| UI程式碼(放置於 def createWidgets(self): 的下一層)/UI示意圖 |
| # richTextBoxMessage          self.richTextBoxMessage = tk.Listbox(self, height=5)          self.richTextBoxMessage.grid(column = 0, row = 10, columnspan=5, sticky = tk.E + tk.W)          global richTextBoxMessage          richTextBoxMessage = self.richTextBoxMessage |

|  |
| --- |
| 事件(可放在事件SKOrderLibEvent註冊物件底下一層) |
| # 集保庫存查詢。透過呼叫 GetBalanceQuery後，資訊由該事件回傳。      def OnBalanceQuery(self, bstrData):          msg = "【OnBalanceQuery】" + bstrData;          richTextBoxMessage.insert('end', msg + "\n")          richTextBoxMessage.see('end') |

### OnMarginPurchaseAmountLimit

|  |  |  |
| --- | --- | --- |
| 資券配額查詢。透過呼叫 [GetMarginPurchaseAmountLimit](#_GetMarginPurchaseAmountLimit_1)後，資訊由該事件回傳。 | | |
| **宣告** | void OnMarginPurchaseAmountLimit ([in] BSTR bstrData); | |
| **參數** | bstrData | 回傳的字串內容【M000】代表成功，接著以「,」分隔每一個欄位，欄位依序為：   |  |  | | --- | --- | | **參數** | **說明** | | 商品代號 |  | | 融資標記 | 0:正常1:停資9:取消信用交易 | | 融資限量 | 0:無配額 9999:無限制 | | 融資比率 |  | | 融券標記 | 0:正常1:停資9:取消信用交易 | | 融券限量 | 0:無配額 9999:無限制 | | 融券比率 |  | | 當沖標記 | 0:不可當沖 1:可當沖 | | 被降成標記 | 0:正常 1:被降成 2:被升成 | | 平盤放空標記 | 0:不可平盤放空 1:可平盤放空 | | 全額交割標記 | 0:正常 1:全額交割 | | 警示標記 | 0:正常1:公司內部警示記號 | | 處置股票標記 | 0:正常1處置股票2:再處置股票  3:彈性處置股票 | | 注意股票標記 | 0:正常1:注意股票 | | 受限股票標記 | 0正常1:證券商委託申報受限股票 | | 異常推介標記 | 0:正常1:異常推介個股 | | 特殊異常標記 | 0:正常1:特殊異常股票 | | 單筆委託張數限制 |  | | 多筆委託張數限制 |  | | 款券預收成數 |  | | **LOGIN\_ID** |  | | **ACCOUNT\_NO** |  | |
| **備註** | 回傳【M003】表示此股票無信用交易資格 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py |
| UI程式碼(放置於 def createWidgets(self): 的下一層)/UI示意圖 |
| # richTextBoxMessage          self.richTextBoxMessage = tk.Listbox(self, height=5)          self.richTextBoxMessage.grid(column = 0, row = 10, columnspan=5, sticky = tk.E + tk.W)          global richTextBoxMessage          richTextBoxMessage = self.richTextBoxMessage |

|  |
| --- |
| 事件(可放在事件SKOrderLibEvent註冊物件底下一層) |
| # 資券配額查詢。透過呼叫 GetMarginPurchaseAmountLimit後，資訊由該事件回傳。      def OnMarginPurchaseAmountLimit(self, bstrData):          msg = "【OnMarginPurchaseAmountLimit】" + bstrData;          richTextBoxMessage.insert('end', msg + "\n")          richTextBoxMessage.see('end') |

### OnProfitLossGWReport

|  |  |  |
| --- | --- | --- |
| 證券新損益查詢結果。透過呼叫[GetProfitLossGWReport](#_GetProfitLossGWReport)後，資訊由該事件回傳。 | | |
| **宣告** | void OnProfitLossGWReport ([in] BSTR bstrData); | |
| **參數** | bstrData | 回傳的字串內容，第一筆為查詢結果。  成功時，成功代碼為「000」與訊息，中間以「,」分隔  失敗時，為錯誤代碼與錯誤訊息，中間以「,」分隔。  成功時：  第二筆資料開始就是資料內容，各筆資料以「,」分隔每一個欄位，欄位依序如下表，請參照下表。 |

|  |
| --- |
| Python範例程式碼\_TSOrder.py |
| UI程式碼(放置於 def createWidgets(self): 的下一層)/UI示意圖 |
| # richTextBoxMessage          self.richTextBoxMessage = tk.Listbox(self, height=5)          self.richTextBoxMessage.grid(column = 0, row = 10, columnspan=5, sticky = tk.E + tk.W)          global richTextBoxMessage          richTextBoxMessage = self.richTextBoxMessage |

|  |
| --- |
| 事件(可放在事件SKOrderLibEvent註冊物件底下一層) |
| # 證券新損益查詢結果。透過呼叫GetProfitLossGWReport後，資訊由該事件回傳。      def OnProfitLossGWReport(self, bstrData):          msg = "【OnProfitLossGWReport】" + bstrData;          richTextBoxMessage.insert('end', msg + "\n")          richTextBoxMessage.see('end') |

#### 未實現

|  |  |  |
| --- | --- | --- |
| 證券新損益查詢結果。透過呼叫 [GetProfitLossGWReport](#_4-2-73_GetProfitLossGWReport) 後，資訊由該事件回傳。  (未實現) | | |
| **宣告** | void OnProfitLossGWReport ([in] BSTR bstrData); | |
| **參數** | bstrData | 回傳的字串內容，第一筆為查詢結果。  成功代碼為「000」與訊息。  成功時：  第二筆資料開始就是資料內容  回傳的內容以「,」分隔每一個欄位，欄位依序為：  (彙總資料格式)   |  |  |  | | --- | --- | --- | |  | 參數 | 說明 | | 1 | 股票名稱 |  | | 2 | 股票代號 |  | | 3 | 幣別 |  | | 4 | 交易種類 |  | | 5 | 庫存股數 |  | | 6 | 市價 |  | | 7 | 今日市價漲跌 |  | | 8 | 市值 |  | | 9 | 淨值 |  | | 10 | 損益 |  | | 11 | 平均買進(券賣)成本 |  | | 12 | 付出成本 |  | | 13 | 成交價金 |  | | 14 | 手續費 |  | | 15 | 預估手續費 |  | | 16 | 交易稅 |  | | 17 | 預估交易稅 |  | | 18 | 融資自備款/融券保證金 |  | | 19 | 融資金/擔保品 |  | | 20 | 預估利息 |  | | 21 | 股息 | 目前此欄無值 | | 22 | 試算報酬率 |  | | 23 | 未知成本股數 |  | | 24 | 備註 |  | | 25 | 詳細資料 | Y:有資料 N:無資料 | | 26 | 排序序號 |  | | 27 | 交易種類代號 | 0:現股 3:融資(自) 4:融券(自) 8:券差 9:無券賣出 | | 28 | 損益兩平點 | 此欄只提供**交易種類為現股**之交易 | | 29 | LOGIN\_ID |  | | 30 | ACCOUNT\_NO |  |   (明細資料格式)   |  |  |  | | --- | --- | --- | |  | 參數 | 說明 | | 1 | 股票名稱 |  | | 2 | 股票代號 |  | | 3 | 成交日期 |  | | 4 | 交易種類 |  | | 5 | 買進股數 |  | | 6 | 可沖股數 |  | | 7 | 單價 |  | | 8 | 成交價金 |  | | 9 | 損益 |  | | 10 | 試算報酬率 |  | | 11 | 手續費 |  | | 12 | 預估手續費 |  | | 13 | 交易稅 |  | | 14 | 預估交易稅 |  | | 15 | 融資自備款/融券保證金 |  | | 16 | 融資金額/擔保品金額 |  | | 17 | 融資/融券利息 |  | | 18 | 融券手續費 |  | | 19 | 收付金額 |  | | 20 | 股息 | 目前此欄無值 | | 21 | 備註 |  | | 22 | 委託書號 |  | | 23 | 交易種類代號 | 配合第4欄交易種類  0:現股 3:融資(自) 4:融券(自) | | 24 | LOGIN\_ID |  | | 25 | ACCOUNT\_NO |  | |
| **備註** | * 當全部資料已經全部回傳完畢，將回傳一筆以「##」開頭的內容，表示查詢結束。 | |

#### 已實現

|  |  |  |
| --- | --- | --- |
| 證券新損益查詢結果。透過呼叫 [GetProfitLossGWReport](#_4-2-73_GetProfitLossGWReport) 後，資訊由該事件回傳。  (已實現) | | |
| **宣告** | void OnProfitLossGWReport ([in] BSTR bstrData); | |
| **參數** | bstrData | 回傳的字串內容，第一筆為查詢結果。  成功代碼為「000」與訊息。  成功時：  第二筆資料開始就是資料內容  回傳的內容以「,」分隔每一個欄位，欄位依序為：  (彙總資料格式)   |  |  |  | | --- | --- | --- | |  | 參數 | 說明 | | 1 | 交易種類 | 現股、 融資(代) 、融券(代)、 融資(自) 、融券(自) 、無券、 現沖 、券差、**股息** | | 2 | 交易種類代號 | 0:現股1:融資(代) 2:融券(代) 3:融資(自) 4:融券(自) 5:無券 6:現沖 8:券差 | | 3 | 日期 | 民國年月日 | | 4 | 營業單位代碼 |  | | 5 | 證券帳號 |  | | 6 | 股票代號 |  | | 7 | 股數 |  | | 8 | 成交價格 |  | | 9 | 損益 |  | | 10 | 報酬率 |  | | 11 | 備註 |  | | 12 | 詳細資料 | Y:有資料 N:無資料 | | 13 | 委託書號(賣) | 交易種類 是股息，此欄為編號 | | 14 | 序號(賣) |  | | 15 | 股票名稱 |  | | 16 | 幣別 |  | | 17 | 投資額 |  | | 18 | LOGIN\_ID |  |   (明細資料格式)   |  |  |  | | --- | --- | --- | |  | 參數 | 說明 | | 1 | 日期 | 民國 | | 2 | 交易種類 |  | | 3 | 股票代號 |  | | 4 | 股票名稱 |  | | 5 | 委託書號 |  | | 6 | 買賣別 |  | | 7 | 股數 |  | | 8 | 單價 |  | | 9 | 價金 |  | | 10 | 手續費 |  | | 11 | 交易稅 |  | | 12 | 融資自備款/融券保證金 |  | | 13 | 融資金額/擔保品金額 |  | | 14 | 融資/融券利息 |  | | 15 | 融券手續費 |  | | 16 | 收付金額 |  | | 17 | 已沖銷股數 |  | | 18 | 備註 |  | | 19 | 交易種類代號 | 0:現股 1:融資(代) 2:融券(代) 3:融資(自) 4:融券(自) | | 20 | LOGIN\_ID |  | | 21 | ACCOUNT\_NO |  |   (投資總額)   |  |  |  | | --- | --- | --- | |  | 參數 | 說明 | | 1 | 營業單位代碼 |  | | 2 | 證券帳號 |  | | 3 | 投資總額 | 台幣 | | 4 | 已實現投資損益 | 台幣 | | 5 | 報酬率 | 台幣 | | 6 | 投資總額 | 人民幣 | | 7 | 已實現投資損益 | 人民幣 | | 8 | 報酬率 | 人民幣 | | 9 | LOGIN\_ID |  |   (彙總-依商品代碼資料格式)   |  |  |  | | --- | --- | --- | |  | 參數 | 說明 | | 1 | 交易種類 |  | | 2 | 交易種類代號 | 配合第1欄交易種類 | | 3 | 營業單位代碼 |  | | 4 | 證券帳號 |  | | 5 | 股票代號 |  | | 6 | 股數 |  | | 7 | 已實現投資損益 | 台幣 | | 8 | 報酬率 | 台幣 | | 9 | 幣別 |  | | 10 | LOGIN\_ID |  | |
| **備註** | * 當全部資料已經全部回傳完畢，將回傳一筆以「##」開頭的內容，表示查詢結束。 | |

#### 現股當沖

|  |  |  |
| --- | --- | --- |
| 證券新損益查詢結果。透過呼叫 [GetProfitLossGWReport](#_4-2-73_GetProfitLossGWReport) 後，資訊由該事件回傳。  (現股當沖) | | |
| **宣告** | void OnProfitLossGWReport ([in] BSTR bstrData); | |
| 參數 | bstrData | 回傳的字串內容，第一筆為查詢結果。  成功代碼為「000」與訊息。  成功時：  第二筆資料開始就是資料內容  回傳的內容以「,」分隔每一個欄位，欄位依序為：  (彙總資料格式)   |  |  |  | | --- | --- | --- | |  | 參數 | 說明 | | 1 | 序號 | 2 : 彙總總計 | | 2 | 股票名稱 |  | | 3 | 股票代號 |  | | 4 | 幣別(中文) |  | | 5 | 已沖銷股數 |  | | 6 | 待沖銷股數 |  | | 7 | 委買中股數 |  | | 8 | 委賣中股數 |  | | 9 | 可沖銷股數 |  | | 10 | 市價 |  | | 11 | 今日市價漲跌 |  | | 12 | 市值 | 序號為2時，市值總計 | | 13 | 淨值 | 序號為2時，淨價總計 | | 14 | 未實現損益 | 序號為2時，未實現損益總計 | | 15 | 已實現損益 | 序號為2時，已實現損益總計 | | 16 | 損益總計 | 序號為2時，損益總計 | | 17 | 平均單價 |  | | 18 | 付出成本 | 序號為2時，付出成本總計 | | 19 | 成交價金 | 序號為2時，成交價金總計 | | 20 | 手續費 | 序號為2時，手續費總計 | | 21 | 預估手續費 | 序號為2時，預估手續費總計 | | 22 | 交易稅 | 序號為2時，交易稅總計 | | 23 | 預估交易稅 | 序號為2時，預估交易稅總計 | | 24 | 試算報酬率 |  | | 25 | 備註 |  | | 26 | 詳細資料 | Y:有資料 N:無資料 | | 27 | 幣別 |  | | 28 | LOGIN\_ID |  | | 29 | ACCOUNT\_NO |  |   (明細資料格式)：   |  |  |  | | --- | --- | --- | |  | 參數 |  | | 1 | 股票名稱 |  | | 2 | 委託書號 |  | | 3 | 買賣股數 |  | | 4 | 已沖股數 |  | | 5 | 單價 |  | | 6 | 成交價金 |  | | 7 | 手續費 |  | | 8 | 預估手續費 |  | | 9 | 交易稅 |  | | 10 | 預估交易稅 |  | | 11 | 收付金額 |  | | 12 | 股票代號 |  | | **13** | **買賣別 (B:買; S:賣)** | **V2.13.37起新增** | | 14 | LOGIN\_ID |  | | 15 | ACCOUNT\_NO |  | |

### OnAsyncOrder

|  |  |  |
| --- | --- | --- |
| 非同步委託結果。 | | |
| **宣告** | void OnAsyncOrder([in] LONG nThreadID, [in] LONG nCode, [in] BSTR bstrMessage); | |
| **參數** | nThreadID | Thread ID |
| nCode | 收單結果代碼。 |
| bstrMessage | 收單回傳訊息。 |
| **備註** | 送單函式中使用非同步下單時會取得一個Thread ID，當此事件觸發時可藉由nThreadID對應下單來源。   * 委託成功：收單回傳訊息為 委託序號 * 委託失敗：收單回傳訊息為 委託失敗原因 | |
| 期貨智慧單: 如果回傳值為 0表示委託成功，訊息內容則為委託日期、智慧單序號、委託書號、委託內容(帳號等)，以實際回傳內容為主。  回傳成功訊息範例:  EX1: 000,20220419,智慧條件單已送出 條件單號：568157,無委託書號,568157,1689200000003  EX2: 000,20220420,智慧條件單已送出 條件單號：568353,y0004,568353,1688000000001 | |

|  |
| --- |
| Python範例程式碼\_TSOrder.py |
| UI程式碼(放置於 def createWidgets(self): 的下一層)/UI示意圖 |
| # richTextBoxMessage          self.richTextBoxMessage = tk.Listbox(self, height=5)          self.richTextBoxMessage.grid(column = 0, row = 10, columnspan=5, sticky = tk.E + tk.W)          global richTextBoxMessage          richTextBoxMessage = self.richTextBoxMessage |

|  |
| --- |
| 事件(可放在事件SKOrderLibEvent註冊物件底下一層) |
| # 非同步委託結果。      def OnAsyncOrder(self, nThreadID, nCode, bstrMessage):          msg = "【OnAsyncOrder】" + str(nThreadID) + str(nCode) + bstrMessage;          richTextBoxMessage.insert('end', msg + "\n")          richTextBoxMessage.see('end') |

### OnProxyOrder

|  |  |  |
| --- | --- | --- |
| Proxy委託結果。 | | |
| **宣告** | void OnProxyOrder([in] LONG nStampID, [in] LONG nCode, [in] BSTR bstrMessage); | |
| **參數** | nStampID | StampID |
| nCode | Proxy收單結果代碼。 |
| bstrMessage | Proxy收單回傳訊息。 |
| **備註** | 送出委託後會回傳二筆通知   |  |  | | --- | --- | | 第一筆 | | | nCode為0代表成功;其餘非0 代表失敗. | bstrMessage  Pending Order送出委託  ORKEY:為Time Stamp供比對第二筆通知使用 | | 第二筆 | | | nCode 為0代表成功;其餘非0 代表失敗. | bstrMessage  **Proxy回傳訊息欄位說明**  SeqNo:代表原始委託13碼  OrderNo:代表委託書號  ORKEY:為Time Stamp供比對第一筆通知使用  (下列兩欄僅證券下單才回)  Action: 處理動作，'S'：需圈股票、'A'：需圈錢，其餘空白  BlockNum: 需圈存的金額或股數  Action為'S'或'A'才有值，其餘空白 | | |

\*v2.13.47異動：補回傳參數nStampID文字說明

\*v2.13.48異動：新增證券下單回傳欄位

|  |
| --- |
| Python範例程式碼\_TSOrder.py |
| UI程式碼(放置於 def createWidgets(self): 的下一層)/UI示意圖 |
| # richTextBoxMessage          self.richTextBoxMessage = tk.Listbox(self, height=5)          self.richTextBoxMessage.grid(column = 0, row = 10, columnspan=5, sticky = tk.E + tk.W)          global richTextBoxMessage          richTextBoxMessage = self.richTextBoxMessage |

|  |
| --- |
| 事件(可放在事件SKOrderLibEvent註冊物件底下一層) |
| # Proxy委託結果。      def OnProxyOrder(self, nStampID, nCode, bstrMessage):          msg = "【OnProxyOrder】" + str(nStampID) + str(nCode) + bstrMessage;          richTextBoxMessage.insert('end', msg + "\n")          richTextBoxMessage.see('end') |

## Struct 結構物件

### TSPROFITLOSSGWQUERY(新損益試算查詢物件)

#### **未實現**

struct TSPROFITLOSSGWQUERY

{

BSTR bstrFullAccount; //證券帳號，分公司四碼＋帳號7碼

LONG nTPQueryType; //損益類別 0: 未實現損益；1: 已實現損益 2:現股當沖

LONG nFunc; //損益回傳格式 //(未實現) 0:彙總 ；1:明細

BSTR bstrStockNo; //當nFunc 為 1:明細 時,指定商品代碼

BSTR bstrTradeType; //當nFunc 為 1:明細 時,指定交易類別

//0:現股 3:融資(自) 4:融券(自) 8:券差 9:無券賣出

};

#### **已實現**

struct TSPROFITLOSSGWQUERY

{

BSTR bstrFullAccount; //證券帳號，分公司四碼＋帳號7碼

LONG nTPQueryType; //損益類別 1: 已實現損益

LONG nFunc; //損益回傳格式

// 0:彙總；1:明細; 2:投資總額; 3:彙總(依股票代號)

BSTR bstrStockNo; 　//當nFunc 為 1:明細 及3: 彙總依股票代號 時，指定商品代碼

　　BSTR bstrTradeType; //當nFunc 為 1:明細，指定交易類別

//0:現股 1:融資(代) 2:融券(代) 3:融資(自) 4:融券(自) 6:現沖 7.信用當沖 8:券差

BSTR bstrStartDate;

//當nFunc 為 0:彙總 時，起始日**西元年月日**//YYYYMMDD

//當nFunc 為 1:明細 及 3:彙總依股票代號 時，起始日**民國年月日**

BSTR bstrEndDate; //當nFunc 為 0:彙總 時，結束日**西元年月日**//YYYYMMDD

//當nFunc 為 1:明細格式時,**不**需填結束日

//當nFunc 3:彙總依股票代號 時，結束日 **民國年月日**

BSTR bstrBookNo; //當nFunc 為 1:明細 時,指定委託書號

BSTR bstrSeqNo; //當nFunc 為 1:明細 時,指定序號

};

#### **現股當沖:(限查閱當日交易)**

struct TSPROFITLOSSGWQUERY

{

BSTR bstrFullAccount; //證券帳號，分公司四碼＋帳號7碼

LONG nTPQueryType; //損益類別 2:現股當沖

LONG nFunc; //損益回傳格式 //(現股當沖) 1:彙總 2:明細

BSTR bstrStockNo; //當nFunc 為 2時:明細時,需指定商品代碼

};

### STOCKORDER (證券下單物件)

struct STOCKORDER

{

BSTR bstrFullAccount; //證券帳號，分公司代碼＋帳號7碼

BSTR bstrStockNo; //委託股票代號

SHORT sPrime; //0:上市上櫃1:興櫃

SHORT sPeriod; //0:盤中 1:盤後 2:零股

SHORT sFlag; //0:現股 1:融資 2:融券 3:無券

SHORT sBuySell; //0:買進 1:賣出

BSTR bstrPrice; //委託價格，「M」表示參考價（昨收價）、「H」表示漲停價、「L」表示跌停價

LONG nQty; //整股交易為張數，如果是零股則為股數

LONG nTradeType; //[證券逐筆交易]**0:ROD ; 1:IOC ; 2:FOK**

LONG nSpecialTradeType; //[證券逐筆交易]**1:市價; 2:限價**

(市價單之委託價格Price請給0; 限價單之委託價格Price 不可為0)

};

#### **盤中零股**

struct STOCKORDER

{

BSTR bstrFullAccount; //證券帳號，分公司代碼＋帳號7碼

BSTR bstrStockNo; //委託股票代號

SHORT sPeriod; // 4:盤中零股

SHORT sFlag; //0:現股

SHORT sBuySell; //0:買進 1:賣出

BSTR bstrPrice; //委託價格

LONG nQty; //零股數量為股數1~999股

};

### SKAVGCOST (昨日未沖銷的證券明細查詢物件)

struct SKAVGCOST

{

BSTR bstrAccount; //證券帳號，分公司四碼＋帳號7碼

BSTR bstrFunc; //功能(0:查詢昨日未沖銷明細)

BSTR bstrStockNo; //股票代號(空白:所有股票代號)

}

### Proxy Server 證券

#### **Proxy Server 證券下單**

struct STOCKPROXYORDER

{

BSTR bstrFullAccount; //證券帳號，分公司代碼＋帳號7碼

BSTR bstrStockNo; //委託股票代號

BSTR bstrOrderType; //下單類別

(1: 現股買進。2: 現股賣出。3: 融資買進。4: 融資賣出。5: 融券買進。6: 融券賣出。7: 無券賣出)

LONG nSpecialTradeType; // 1:市價 2:限價

LONG nPeriod; //0:盤中 1:零股 2:盤後交易 3:盤中零股

BSTR bstrPrice; //委託價格

LONG nQty; //股數

LONG nTradeType; //0:ROD ; 1:IOC ; 2:FOK

LONG nPriceMark; //價格旗標 0:一般定價 1:前日收盤價 2:漲停 3:跌停

};

#### **Proxy Server 證券刪改單**

struct STOCKPROXYORDER

{

BSTR bstrFullAccount; //證券帳號，分公司代碼＋帳號7碼

BSTR bstrStockNo; //委託股票代號

BSTR bstrOrderType; //下單類別 (0:刪單。1:改量。2:改價。)

LONG nSpecialTradeType; // 1:市價 2:限價

LONG nPeriod; //0:盤中 1:零股 2:盤後交易 3:盤中零股

BSTR bstrPrice; //委託價格

(改價:必填，帶””表示市價。減量:帶0，刪單:帶””或是原始價)

LONG nQty; //股數

(改價:帶0。減量:帶要減的量。刪單:帶原始量)

LONG nTradeType; //0:ROD ; 1:IOC ; 2:FOK

LONG nPriceMark; //價格旗標 0:一般定價 1:前日收盤價 2:漲停 3:跌停

BSTR bstrBookNo; //委託書號

BSTR bstrSeqNo; //委託序號

};

#### Proxy Server 證券特殊刪改單(帶入序號刪單)

struct STOCKPROXYORDER

{

BSTR bstrFullAccount; //證券帳號，分公司代碼＋帳號7碼

BSTR bstrOrderType; //下單類別，0:刪單1:改量2:改價

BSTR bstrSeqNo; //委託序號

(序號帶入 ”” 或 不帶入，會刪除證券市場所有委託)

};

#### Proxy Server 證券特殊刪改單

struct STOCKPROXYORDER

{

BSTR bstrFullAccount; //證券帳號，分公司代碼＋帳號7碼

BSTR bstrOrderType; //下單類別，0:刪單1:改量2:改價

BSTR bstrPrice; //委託價格

(改價 : 帶要改的價，改量:帶0，也可以帶""，刪單 : 帶原始價, 也可帶"")

LONG nPriceMark; //價格旗標(改價可用) 0:一般定價 1:前日收盤價 2:漲停 3:跌停

LONG nQty; //股數(改量 : 帶要減的量)

BSTR bstrBookNo; //委託書號

BSTR bstrSeqNo; //委託序號

(書號、序號、商品代號擇一填寫，都帶入””或不帶入，會刪除證券市場所有委託)

BSTR bstrStockNo; //委託股票代號(用於商品代號刪單情境)

BSTR bstrPrice\_forD; //原委託單價格(用於商品代號刪單情境)

**BSTR** bstrBuySell\_forD; //原委託單買賣別(用於商品代號刪單情境) 0是BUY ; 1 是 SELL

};

|  |
| --- |
| 支持股號刪單，可刪除此id委託此股號的尚未成交的所有委託單  支持刪除股號+指定刪特定價格和買賣別的委託單  支持刪除股號+指定買賣別的委託單 |
| 股號刪單：帶入 bstrStockNo |
| 刪除股號+指定刪特定價格和買賣別：帶入bstrStockNo + bstrPrice\_forD + bstrBuySell\_forD |
| 刪除股號+指定買賣別：帶入bstrStockNo + bstrBuySell\_forD |