策略王COM元件使用說明

-下單-證券智慧單-

## 函式範例

### SKOrderLib(國內證券)

*函式*

1. 智慧單

|  |  |  |
| --- | --- | --- |
| **功能** | **函式名稱** | **備註** |
| 1. **證券智慧單當沖條件委託** | [SendStockStrategyDayTrade](#_SendStockStrategyDayTrade) | 簽署證券智慧單風險預告書 |
| 1. **證券智慧單出清條件委託** | [SendStockStrategyClear](#_SendStockStrategyClear_1) |  |
| 1. **證券智慧單MIT條件委託** | [SendStockStrategyMIT](#_SendStockStrategyMIT_2) |  |
| 1. **證券智慧單OCO條件委託** | [SendStockStrategyOCO](#_SendStockStrategyOCO_1) |  |
| 1. **證券智慧單MIOC條件委託** | [SendStockStrategyMIOC](#_SendStockStrategyMIOC_1) |  |
| 1. **證券智慧單MST條件委託** | [SendStockStrategyMST](#_SendStockStrategyMST_1) |  |
| 1. **證券AB單委託** | [SendStockStrategyAB](#_SendStockStrategyAB_1) |  |
| 1. **證券自組單委託** | [SendStockStrategyCB](#_SendStockStrategyCB_1) |  |
| 1. **取消證券智慧單委託** | |  | | --- | | **新版**[CancelTSStrategyOrderV1](#_CancelTSStrategyOrderV1_1) | | [CancelTSStrategyOrder](#_CancelTSStrategyOrder_1) | |  |
| 1. **取消多筆智慧單委託** | [CancelStrategyList](#_CancelStrategyList) |  |
| 1. **證券智慧單被動查詢** | [GetTSSmartStrategyReport](#_GetTSSmartStrategyReport) |  |

*事件*

|  |  |  |
| --- | --- | --- |
| **功能** | **事件名稱** | **備註** |
| 1. **證券智慧單資料** | [OnTSSmartStrategyReport](#_OnTSSmartStrategyReport) |  |
| 1. **非同步委託結果** | [OnAsyncOrder](#_OnAsyncOrder_1) |  |

#### 智慧單

##### SendStockStrategyDayTrade

|  |  |  |
| --- | --- | --- |
| 送出證券智慧單當沖條件委託。  智慧單用戶請注意備註。 | | |
| **宣告** | Long SendStockStrategyDayTrade ([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **STOCKSTRATEGYORDER**\* 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[**STOCKSTRATEGYORDER**](#_證券當沖)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，訊息內容則為委託日期、條件單號 (智慧單序號)等委託資料。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。 |
| **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。**  **其他錯誤由「智慧單主機」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_1)取得。  需簽署證券智慧單風險預告書及未簽署者可至金融網－同意書簽署專區申請。  ［智慧單用戶請注意］：智慧單為獨立運作機制，若盤中已自行出清或回補，請務必於盤中自行取消已設定的智慧單，避免條件觸發後重覆成交。 | |

|  |
| --- |
| Python範例程式碼\_TSStrategyOrder.py類別DayTradeForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # 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)          # textBoxbstrStockNoDT          tk.Label(self, text = "委託股票代號").grid(row=0, column=1)          #輸入框          self.textBoxbstrStockNoDT = tk.Entry(self)          self.textBoxbstrStockNoDT.grid(row=0, column=2)          global textBoxbstrStockNoDT          textBoxbstrStockNoDT = self.textBoxbstrStockNoDT          # comboBoxnBuySellDT          tk.Label(self, text = "0:現股買/1:無券賣出").grid(row=0, column=3)              #輸入框          self.comboBoxnBuySellDT = ttk.Combobox(self, state='readonly')          self.comboBoxnBuySellDT['values'] = Config.comboBoxnBuySellDT          self.comboBoxnBuySellDT.grid(row=0, column=4)          global comboBoxnBuySellDT          comboBoxnBuySellDT = self.comboBoxnBuySellDT          # textBoxnQtyDT          tk.Label(self, text = "委託張數").grid(row=1, column=1)          #輸入框          self.textBoxnQtyDT = tk.Entry(self)          self.textBoxnQtyDT.grid(row=1, column=2)          global textBoxnQtyDT          textBoxnQtyDT = self.textBoxnQtyDT          # comboBoxnOrderPriceCondDT          tk.Label(self, text = "委託時效(0:ROD,3:IOC,4:FOK)").grid(row=1, column=3)              #輸入框          self.comboBoxnOrderPriceCondDT = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderPriceCondDT['values'] = Config.comboBoxnOrderPriceCondDT          self.comboBoxnOrderPriceCondDT.grid(row=1, column=4)          global comboBoxnOrderPriceCondDT          comboBoxnOrderPriceCondDT = self.comboBoxnOrderPriceCondDT          # textBoxbstrOrderPriceDT          tk.Label(self, text = "當沖單進場委託價").grid(row=2, column=1)          #輸入框          self.textBoxbstrOrderPriceDT = tk.Entry(self)          self.textBoxbstrOrderPriceDT.grid(row=2, column=2)          global textBoxbstrOrderPriceDT          textBoxbstrOrderPriceDT = self.textBoxbstrOrderPriceDT          # comboBoxnOrderPriceTypeDT          tk.Label(self, text = "委託價類別(1:市價; 2:限價)").grid(row=2, column=3)              #輸入框          self.comboBoxnOrderPriceTypeDT = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderPriceTypeDT['values'] = Config.comboBoxnOrderPriceTypeDT          self.comboBoxnOrderPriceTypeDT.grid(row=2, column=4)          global comboBoxnOrderPriceTypeDT          comboBoxnOrderPriceTypeDT = self.comboBoxnOrderPriceTypeDT          # comboBoxnInnerOrderIsMITDT          tk.Label(self, text = "進場是否MIT( 0: N; 1:Y)").grid(row=3, column=1)              #輸入框          self.comboBoxnInnerOrderIsMITDT = ttk.Combobox(self, state='readonly')          self.comboBoxnInnerOrderIsMITDT['values'] = Config.comboBoxnInnerOrderIsMITDT          self.comboBoxnInnerOrderIsMITDT.grid(row=3, column=2)          global comboBoxnInnerOrderIsMITDT          comboBoxnInnerOrderIsMITDT = self.comboBoxnInnerOrderIsMITDT          # comboBoxnMITDirDT          tk.Label(self, text = "進場MIT觸價方向(0:未啟用MIT,1: 向上觸發(大於等於) , 2:向下觸發(小於等於))").grid(row=3, column=3)              #輸入框          self.comboBoxnMITDirDT = ttk.Combobox(self, state='readonly')          self.comboBoxnMITDirDT['values'] = Config.comboBoxnMITDirDT          self.comboBoxnMITDirDT.grid(row=3, column=4)          global comboBoxnMITDirDT          comboBoxnMITDirDT = self.comboBoxnMITDirDT          # textBoxbstrMITTriggerPriceDT          tk.Label(self, text = "進場MIT觸發價(若未啟用MIT,請填0)").grid(row=3, column=5)          #輸入框          self.textBoxbstrMITTriggerPriceDT = tk.Entry(self)          self.textBoxbstrMITTriggerPriceDT.grid(row=3, column=6)          global textBoxbstrMITTriggerPriceDT          textBoxbstrMITTriggerPriceDT = self.textBoxbstrMITTriggerPriceDT          # textBoxbstrMITDealPriceDT          tk.Label(self, text = "進場MIT當下市價(若未啟用MIT,請填0)").grid(row=3, column=7)          #輸入框          self.textBoxbstrMITDealPriceDT = tk.Entry(self)          self.textBoxbstrMITDealPriceDT.grid(row=3, column=8)          global textBoxbstrMITDealPriceDT          textBoxbstrMITDealPriceDT = self.textBoxbstrMITDealPriceDT          # comboBoxnStopLossFlagDT          tk.Label(self, text = "出場停損條件(0:否, 1:是)").grid(row=4, column=1)              #輸入框          self.comboBoxnStopLossFlagDT = ttk.Combobox(self, state='readonly')          self.comboBoxnStopLossFlagDT['values'] = Config.comboBoxnStopLossFlagDT          self.comboBoxnStopLossFlagDT.grid(row=4, column=2)          global comboBoxnStopLossFlagDT          comboBoxnStopLossFlagDT = self.comboBoxnStopLossFlagDT          # comboBoxnRDOSLPercent          tk.Label(self, text = "停損類型 0:觸發價；1:漲跌幅").grid(row=4, column=3)              #輸入框          self.comboBoxnRDOSLPercent = ttk.Combobox(self, state='readonly')          self.comboBoxnRDOSLPercent['values'] = Config.comboBoxnRDOSLPercent          self.comboBoxnRDOSLPercent.grid(row=4, column=4)          global comboBoxnRDOSLPercent          comboBoxnRDOSLPercent = self.comboBoxnRDOSLPercent          # textBoxbstrSLTrigger          tk.Label(self, text = "停損觸發價").grid(row=4, column=5)          #輸入框          self.textBoxbstrSLTrigger = tk.Entry(self)          self.textBoxbstrSLTrigger.grid(row=4, column=6)          global textBoxbstrSLTrigger          textBoxbstrSLTrigger = self.textBoxbstrSLTrigger          # textBoxbstrSLPercentDT          tk.Label(self, text = "停損百分比").grid(row=4, column=7)          #輸入框          self.textBoxbstrSLPercentDT = tk.Entry(self)          self.textBoxbstrSLPercentDT.grid(row=4, column=8)          global textBoxbstrSLPercentDT          textBoxbstrSLPercentDT = self.textBoxbstrSLPercentDT          # textBoxbstrSLOrderPriceDT          tk.Label(self, text = "停損委託價").grid(row=4, column=9)          #輸入框          self.textBoxbstrSLOrderPriceDT = tk.Entry(self)          self.textBoxbstrSLOrderPriceDT.grid(row=4, column=10)          global textBoxbstrSLOrderPriceDT          textBoxbstrSLOrderPriceDT = self.textBoxbstrSLOrderPriceDT          # comboBoxnRDSLMarketPriceType          tk.Label(self, text = "停損委託價方式1:市價;2:限價").grid(row=4, column=11)              #輸入框          self.comboBoxnRDSLMarketPriceType = ttk.Combobox(self, state='readonly')          self.comboBoxnRDSLMarketPriceType['values'] = Config.comboBoxnRDSLMarketPriceType          self.comboBoxnRDSLMarketPriceType.grid(row=4, column=12)          global comboBoxnRDSLMarketPriceType          comboBoxnRDSLMarketPriceType = self.comboBoxnRDSLMarketPriceType          # comboBoxnStopLossOrderCond          tk.Label(self, text = "停損出場時效 0:ROD;3:IOC;4:FOK").grid(row=4, column=13)              #輸入框          self.comboBoxnStopLossOrderCond = ttk.Combobox(self, state='readonly')          self.comboBoxnStopLossOrderCond['values'] = Config.comboBoxnStopLossOrderCond          self.comboBoxnStopLossOrderCond.grid(row=4, column=14)          global comboBoxnStopLossOrderCond          comboBoxnStopLossOrderCond = self.comboBoxnStopLossOrderCond          # comboBoxnTakeProfitFlagDT          tk.Label(self, text = "出場停利條件0:否；1:是").grid(row=5, column=1)              #輸入框          self.comboBoxnTakeProfitFlagDT = ttk.Combobox(self, state='readonly')          self.comboBoxnTakeProfitFlagDT['values'] = Config.comboBoxnTakeProfitFlagDT          self.comboBoxnTakeProfitFlagDT.grid(row=5, column=2)          global comboBoxnTakeProfitFlagDT          comboBoxnTakeProfitFlagDT = self.comboBoxnTakeProfitFlagDT          # comboBoxnRDOTPPercent          tk.Label(self, text = "停利類型 0:觸發價；1:漲幅").grid(row=5, column=3)              #輸入框          self.comboBoxnRDOTPPercent = ttk.Combobox(self, state='readonly')          self.comboBoxnRDOTPPercent['values'] = Config.comboBoxnRDOTPPercent          self.comboBoxnRDOTPPercent.grid(row=5, column=4)          global comboBoxnRDOTPPercent          comboBoxnRDOTPPercent = self.comboBoxnRDOTPPercent          # textBoxbstrTPTrigger          tk.Label(self, text = "停利觸發價").grid(row=5, column=5)          #輸入框          self.textBoxbstrTPTrigger = tk.Entry(self)          self.textBoxbstrTPTrigger.grid(row=5, column=6)          global textBoxbstrTPTrigger          textBoxbstrTPTrigger = self.textBoxbstrTPTrigger          # textBoxbstrTPPercent          tk.Label(self, text = "停利百分比").grid(row=5, column=7)          #輸入框          self.textBoxbstrTPPercent = tk.Entry(self)          self.textBoxbstrTPPercent.grid(row=5, column=8)          global textBoxbstrTPPercent          textBoxbstrTPPercent = self.textBoxbstrTPPercent          # textBoxbstrTPOrderPriceDT          tk.Label(self, text = "停利委託價").grid(row=5, column=9)          #輸入框          self.textBoxbstrTPOrderPriceDT = tk.Entry(self)          self.textBoxbstrTPOrderPriceDT.grid(row=5, column=10)          global textBoxbstrTPOrderPriceDT          textBoxbstrTPOrderPriceDT = self.textBoxbstrTPOrderPriceDT          # comboBoxnRDTPMarketPriceType          tk.Label(self, text = "停利委託價方式1:市價;2:限價").grid(row=5, column=11)              #輸入框          self.comboBoxnRDTPMarketPriceType = ttk.Combobox(self, state='readonly')          self.comboBoxnRDTPMarketPriceType['values'] = Config.comboBoxnRDTPMarketPriceType          self.comboBoxnRDTPMarketPriceType.grid(row=5, column=12)          global comboBoxnRDTPMarketPriceType          comboBoxnRDTPMarketPriceType = self.comboBoxnRDTPMarketPriceType          # comboBoxnTakeProfitOrderCond          tk.Label(self, text = "停利出場時效 0:ROD;3:IOC;4:FOK").grid(row=5, column=13)              #輸入框          self.comboBoxnTakeProfitOrderCond = ttk.Combobox(self, state='readonly')          self.comboBoxnTakeProfitOrderCond['values'] = Config.comboBoxnTakeProfitOrderCond          self.comboBoxnTakeProfitOrderCond.grid(row=5, column=14)          global comboBoxnTakeProfitOrderCond          comboBoxnTakeProfitOrderCond = self.comboBoxnTakeProfitOrderCond          # comboBoxnClearAllFlagDT          tk.Label(self, text = "執行時間出清0:否；1:是").grid(row=6, column=1)              #輸入框          self.comboBoxnClearAllFlagDT = ttk.Combobox(self, state='readonly')          self.comboBoxnClearAllFlagDT['values'] = Config.comboBoxnClearAllFlagDT          self.comboBoxnClearAllFlagDT.grid(row=6, column=2)          global comboBoxnClearAllFlagDT          comboBoxnClearAllFlagDT = self.comboBoxnClearAllFlagDT          # textBoxbstrClearCancelTimeDT          tk.Label(self, text = "出清時間-時＋分（hhmm）{每日13:20截止)").grid(row=6, column=3)          #輸入框          self.textBoxbstrClearCancelTimeDT = tk.Entry(self)          self.textBoxbstrClearCancelTimeDT.grid(row=6, column=4)          global textBoxbstrClearCancelTimeDT          textBoxbstrClearCancelTimeDT = self.textBoxbstrClearCancelTimeDT          # textBoxbstrClearAllOrderPriceDT          tk.Label(self, text = "出清委託價").grid(row=6, column=5)          #輸入框          self.textBoxbstrClearAllOrderPriceDT = tk.Entry(self)          self.textBoxbstrClearAllOrderPriceDT.grid(row=6, column=6)          global textBoxbstrClearAllOrderPriceDT          textBoxbstrClearAllOrderPriceDT = self.textBoxbstrClearAllOrderPriceDT          # comboBoxnClearAllPriceTypeDT          tk.Label(self, text = "出清委託價方式= 1:市價;2:限價").grid(row=6, column=7)              #輸入框          self.comboBoxnClearAllPriceTypeDT = ttk.Combobox(self, state='readonly')          self.comboBoxnClearAllPriceTypeDT['values'] = Config.comboBoxnClearAllPriceTypeDT          self.comboBoxnClearAllPriceTypeDT.grid(row=6, column=8)          global comboBoxnClearAllPriceTypeDT          comboBoxnClearAllPriceTypeDT = self.comboBoxnClearAllPriceTypeDT          # comboBoxnClearOrderCondDT          tk.Label(self, text = "出清出場時效 0:ROD;3:IOC;4:FOK").grid(row=6, column=9)              #輸入框          self.comboBoxnClearOrderCondDT = ttk.Combobox(self, state='readonly')          self.comboBoxnClearOrderCondDT['values'] = Config.comboBoxnClearOrderCondDT          self.comboBoxnClearOrderCondDT.grid(row=6, column=10)          global comboBoxnClearOrderCondDT          comboBoxnClearOrderCondDT = self.comboBoxnClearOrderCondDT          # comboBoxnFinalClearFlagDT          tk.Label(self, text = "盤後定盤交易 0:否；1:是").grid(row=7, column=1)              #輸入框          self.comboBoxnFinalClearFlagDT = ttk.Combobox(self, state='readonly')          self.comboBoxnFinalClearFlagDT['values'] = Config.comboBoxnFinalClearFlagDT          self.comboBoxnFinalClearFlagDT.grid(row=7, column=2)          global comboBoxnFinalClearFlagDT          comboBoxnFinalClearFlagDT = self.comboBoxnFinalClearFlagDT          # buttonSendStockStrategyDayTrade          self.buttonSendStockStrategyDayTrade = tk.Button(self)          self.buttonSendStockStrategyDayTrade["text"] = "當沖送出"          self.buttonSendStockStrategyDayTrade["command"] = self.buttonSendStockStrategyDayTrade\_Click          self.buttonSendStockStrategyDayTrade.grid(row=8, column=1) |

|  |
| --- |
| Config.py |
| # DayTradeForm  comboBoxnBuySellDT =(      "0:現股買", "1:無券賣出"  )  comboBoxnOrderPriceCondDT =(      "0:ROD", "3:IOC", "4:FOK"  )  comboBoxnOrderPriceTypeDT =(      "1:市價", "2:限價"  )  comboBoxnInnerOrderIsMITDT =(      "0:N", "1:Y"  )  comboBoxnMITDirDT =(      "0:未啟用MIT", "1:向上觸發(大於等於)", "2:向下觸發(小於等於)"  )  comboBoxnStopLossFlagDT =(      "0:否", "1:是"  )  comboBoxnRDOSLPercent =(      "0:觸發價", "1:漲跌幅"  )  comboBoxnRDSLMarketPriceType =(      "1:市價", "2:限價"  )  comboBoxnStopLossOrderCond =(      "0:ROD", "3:IOC", "4:FOK"  )  comboBoxnTakeProfitFlagDT =(      "0:否", "1:是"  )  comboBoxnRDOTPPercent =(      "0:觸發價", "1:漲幅"  )  comboBoxnRDTPMarketPriceType =(      "1:市價", "2:限價"  )  comboBoxnTakeProfitOrderCond =(      "0:ROD", "3:IOC", "4:FOK"  )  comboBoxnClearAllFlagDT =(      "0:否", "1:是"  )  comboBoxnClearAllPriceTypeDT =(      "1:市價", "2:限價"  )  comboBoxnClearOrderCondDT =(      "0:ROD", "3:IOC", "4:FOK"  )  comboBoxnFinalClearFlagDT =(      "0:否", "1:是"  ) |
| 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 buttonSendStockStrategyDayTrade\_Click(self):            pOrder = sk.STOCKSTRATEGYORDER()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNoDT.get()          pOrder.nQty = int(textBoxnQtyDT.get())          pOrder.bstrOrderPrice = textBoxbstrOrderPriceDT.get()          if (comboBoxnBuySellDT.get() == "0:現股買") :              pOrder.nBuySell = 0          elif (comboBoxnBuySellDT.get() == "1:無券賣出"):              pOrder.nBuySell = 1          if (comboBoxnOrderPriceCondDT.get() == "0:ROD"):              pOrder.nOrderPriceCond = 0          elif (comboBoxnOrderPriceCondDT.get() == "3:IOC"):              pOrder.nOrderPriceCond = 3          elif (comboBoxnOrderPriceCondDT.get() == "4:FOK"):              pOrder.nOrderPriceCond = 4          if (comboBoxnOrderPriceTypeDT.get() == "1:市價"):              pOrder.nOrderPriceType = 1          elif (comboBoxnOrderPriceTypeDT.get() == "2:限價"):              pOrder.nOrderPriceType = 2          if (comboBoxnInnerOrderIsMITDT.get() == "0:N"):              pOrder.nInnerOrderIsMIT = 0          elif (comboBoxnInnerOrderIsMITDT.get() == "1:Y"):              pOrder.nInnerOrderIsMIT = 1          if (comboBoxnMITDirDT.get() == "0:未啟用MIT"):              pOrder.nMITDir = 0          elif (comboBoxnMITDirDT.get() == "1:向上觸發(大於等於)"):              pOrder.nMITDir = 1          elif (comboBoxnMITDirDT.get() == "2:向下觸發(小於等於)"):              pOrder.nMITDir = 2          pOrder.bstrMITTriggerPrice = textBoxbstrMITTriggerPriceDT.get()          pOrder.bstrMITDealPrice = textBoxbstrMITDealPriceDT.get()          if (comboBoxnClearAllFlagDT.get() == "0:否"):              pOrder.nClearAllFlag = 0          elif (comboBoxnClearAllFlagDT.get() == "1:是"):              pOrder.nClearAllFlag = 1          pOrder.bstrClearCancelTime = textBoxbstrClearCancelTimeDT.get()          if (comboBoxnClearAllPriceTypeDT.get() == "1:市價"):              pOrder.nClearAllPriceType = 1          elif (comboBoxnClearAllPriceTypeDT.get() == "2:限價"):              pOrder.nClearAllPriceType = 2          pOrder.bstrClearAllOrderPrice = textBoxbstrClearAllOrderPriceDT.get()          if (comboBoxnFinalClearFlagDT.get() == "0:否"):              pOrder.nFinalClearFlag = 0          elif (comboBoxnFinalClearFlagDT.get() == "1:是"):              pOrder.nFinalClearFlag = 1          if (comboBoxnTakeProfitFlagDT.get() == "0:否"):              pOrder.nTakeProfitFlag = 0          elif (comboBoxnTakeProfitFlagDT.get() == "1:是"):              pOrder.nTakeProfitFlag = 1          if (comboBoxnRDOTPPercent.get() == "0:觸發價"):              pOrder.nRDOTPPercent = 0          elif (comboBoxnRDOTPPercent.get() == "1:漲幅"):              pOrder.nRDOTPPercent = 1          pOrder.bstrTPPercent = textBoxbstrTPPercent.get()          pOrder.bstrTPTrigger = textBoxbstrTPTrigger.get()          if (comboBoxnRDTPMarketPriceType.get() == "1:市價"):              pOrder.nRDTPMarketPriceType = 1          elif (comboBoxnRDTPMarketPriceType.get() == "2:限價"):              pOrder.nRDTPMarketPriceType = 2          pOrder.bstrTPOrderPrice = textBoxbstrTPOrderPriceDT.get()          if (comboBoxnStopLossFlagDT.get() == "0:否"):              pOrder.nStopLossFlag = 0          elif (comboBoxnStopLossFlagDT.get() == "1:是"):              pOrder.nStopLossFlag = 1          if (comboBoxnRDOSLPercent.get() == "0:觸發價"):              pOrder.nRDOSLPercent = 0          elif (comboBoxnRDOSLPercent.get() == "1:漲跌幅"):              pOrder.nRDOSLPercent = 1          pOrder.bstrSLPercent = textBoxbstrSLPercentDT.get()          pOrder.bstrSLTrigger = textBoxbstrSLTrigger.get()          if (comboBoxnRDSLMarketPriceType.get() == "1:市價"):              pOrder.nRDSLMarketPriceType = 1          elif (comboBoxnRDSLMarketPriceType.get() == "2:限價"):              pOrder.nRDSLMarketPriceType = 2          pOrder.bstrSLOrderPrice = textBoxbstrSLOrderPriceDT.get()          if (comboBoxnTakeProfitOrderCond.get() == "0:ROD"):              pOrder.nTakeProfitOrderCond = 0          elif (comboBoxnTakeProfitOrderCond.get() == "3:IOC"):              pOrder.nTakeProfitOrderCond = 3          elif (comboBoxnTakeProfitOrderCond.get() == "4:FOK"):              pOrder.nTakeProfitOrderCond = 4          if (comboBoxnStopLossOrderCond.get() == "0:ROD"):              pOrder.nStopLossOrderCond = 0          elif (comboBoxnStopLossOrderCond.get() == "3:IOC"):              pOrder.nStopLossOrderCond = 3          elif (comboBoxnStopLossOrderCond.get() == "4:FOK"):              pOrder.nStopLossOrderCond = 4          if (comboBoxnClearOrderCondDT.get() == "0:ROD"):              pOrder.nClearOrderCond = 0          elif (comboBoxnClearOrderCondDT.get() == "3:IOC"):              pOrder.nClearOrderCond = 3          elif (comboBoxnClearOrderCondDT.get() == "4:FOK"):              pOrder.nClearOrderCond = 4          # 送出證券智慧單當沖條件委託。          bstrMessage,nCode= m\_pSKOrder.SendStockStrategyDayTrade(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendStockStrategyDayTrade】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### SendStockStrategyClear

|  |  |  |
| --- | --- | --- |
| 送出證券智慧單出清條件委託。  智慧單用戶請注意備註。 | | |
| **宣告** | Long SendStockStrategyClear ([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **STOCKSTRATEGYORDEROUT**\* 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[**STOCKSTRATEGYORDEROUT**](#_STOCKSTRATEGYORDEROUT(證券智慧單物件-出清)_1)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，訊息內容則為委託日期、條件單號 (智慧單序號)等委託資料。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。 |
| **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** | 使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder_1)取得。  需簽署證券智慧單風險預告書及未簽署者可至金融網－同意書簽署專區申請。  ［智慧單用戶請注意］：智慧單為獨立運作機制，若盤中已自行出清或回補，請務必於盤中自行取消已設定的智慧單，避免條件觸發後重覆成交。 | |

|  |
| --- |
| Python範例程式碼\_TSStrategyOrder.py類別ClearForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # 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)          # textBoxbstrStockNoC          tk.Label(self, text = "委託股票代號").grid(row=0, column=1)          #輸入框          self.textBoxbstrStockNoC = tk.Entry(self)          self.textBoxbstrStockNoC.grid(row=0, column=2)          global textBoxbstrStockNoC          textBoxbstrStockNoC = self.textBoxbstrStockNoC          # comboBoxnBuySellC          tk.Label(self, text = "0買;1賣").grid(row=0, column=3)              #輸入框          self.comboBoxnBuySellC = ttk.Combobox(self, state='readonly')          self.comboBoxnBuySellC['values'] = Config.comboBoxnBuySellC          self.comboBoxnBuySellC.grid(row=0, column=4)          global comboBoxnBuySellC          comboBoxnBuySellC = self.comboBoxnBuySellC          # comboBoxnOrderTypeC          tk.Label(self, text = "出清進場委託交易別(0:現股, 3:融資, 4:融券)").grid(row=0, column=5)              #輸入框          self.comboBoxnOrderTypeC = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderTypeC['values'] = Config.comboBoxnOrderTypeC          self.comboBoxnOrderTypeC.grid(row=0, column=6)          global comboBoxnOrderTypeC          comboBoxnOrderTypeC = self.comboBoxnOrderTypeC          # textBoxnQtyC          tk.Label(self, text = "委託張數").grid(row=1, column=1)          #輸入框          self.textBoxnQtyC = tk.Entry(self)          self.textBoxnQtyC.grid(row=1, column=2)          global textBoxnQtyC          textBoxnQtyC = self.textBoxnQtyC          # comboBoxnGTEFlag          tk.Label(self, text = "出場成交價大於指定價後觸發0:否；1:是").grid(row=2, column=1)              #輸入框          self.comboBoxnGTEFlag = ttk.Combobox(self, state='readonly')          self.comboBoxnGTEFlag['values'] = Config.comboBoxnGTEFlag          self.comboBoxnGTEFlag.grid(row=2, column=2)          global comboBoxnGTEFlag          comboBoxnGTEFlag = self.comboBoxnGTEFlag          # textBoxbstrGTETriggerPrice          tk.Label(self, text = "指定價").grid(row=2, column=3)          #輸入框          self.textBoxbstrGTETriggerPrice = tk.Entry(self)          self.textBoxbstrGTETriggerPrice.grid(row=2, column=4)          global textBoxbstrGTETriggerPrice          textBoxbstrGTETriggerPrice = self.textBoxbstrGTETriggerPrice          # textBoxbstrGTEOrderPrice          tk.Label(self, text = "委託價").grid(row=2, column=5)          #輸入框          self.textBoxbstrGTEOrderPrice = tk.Entry(self)          self.textBoxbstrGTEOrderPrice.grid(row=2, column=6)          global textBoxbstrGTEOrderPrice          textBoxbstrGTEOrderPrice = self.textBoxbstrGTEOrderPrice          # comboBoxnGTEMarketPrice          tk.Label(self, text = "(大於條件)以1:市價;2:限價 做為出場價格").grid(row=2, column=7)              #輸入框          self.comboBoxnGTEMarketPrice = ttk.Combobox(self, state='readonly')          self.comboBoxnGTEMarketPrice['values'] = Config.comboBoxnGTEMarketPrice          self.comboBoxnGTEMarketPrice.grid(row=2, column=8)          global comboBoxnGTEMarketPrice          comboBoxnGTEMarketPrice = self.comboBoxnGTEMarketPrice          # comboBoxnGTEOrderCond          tk.Label(self, text = "大於條件出場時效0:ROD;3:IOC;4:FOK").grid(row=2, column=9)              #輸入框          self.comboBoxnGTEOrderCond = ttk.Combobox(self, state='readonly')          self.comboBoxnGTEOrderCond['values'] = Config.comboBoxnGTEOrderCond          self.comboBoxnGTEOrderCond.grid(row=2, column=10)          global comboBoxnGTEOrderCond          comboBoxnGTEOrderCond = self.comboBoxnGTEOrderCond          # comboBoxnLTEFlag          tk.Label(self, text = "出場成交價小於指定價後觸發0:否；1:是").grid(row=3, column=1)              #輸入框          self.comboBoxnLTEFlag = ttk.Combobox(self, state='readonly')          self.comboBoxnLTEFlag['values'] = Config.comboBoxnLTEFlag          self.comboBoxnLTEFlag.grid(row=3, column=2)          global comboBoxnLTEFlag          comboBoxnLTEFlag = self.comboBoxnLTEFlag          # textBoxbstrLTETriggerPrice          tk.Label(self, text = "指定價").grid(row=3, column=3)          #輸入框          self.textBoxbstrLTETriggerPrice = tk.Entry(self)          self.textBoxbstrLTETriggerPrice.grid(row=3, column=4)          global textBoxbstrLTETriggerPrice          textBoxbstrLTETriggerPrice = self.textBoxbstrLTETriggerPrice          # textBoxbstrLTEOrderPrice          tk.Label(self, text = "委託價").grid(row=3, column=5)          #輸入框          self.textBoxbstrLTEOrderPrice = tk.Entry(self)          self.textBoxbstrLTEOrderPrice.grid(row=3, column=6)          global textBoxbstrLTEOrderPrice          textBoxbstrLTEOrderPrice = self.textBoxbstrLTEOrderPrice          # comboBoxnLTEMarketPrice          tk.Label(self, text = "(小於條件)以1:市價;2:限價 做為出場價格").grid(row=3, column=7)              #輸入框          self.comboBoxnLTEMarketPrice = ttk.Combobox(self, state='readonly')          self.comboBoxnLTEMarketPrice['values'] = Config.comboBoxnLTEMarketPrice          self.comboBoxnLTEMarketPrice.grid(row=3, column=8)          global comboBoxnLTEMarketPrice          comboBoxnLTEMarketPrice = self.comboBoxnLTEMarketPrice          # comboBoxnLTEOrderCond          tk.Label(self, text = "小於條件出場時效0:ROD;3:IOC;4:FOK").grid(row=3, column=9)              #輸入框          self.comboBoxnLTEOrderCond = ttk.Combobox(self, state='readonly')          self.comboBoxnLTEOrderCond['values'] = Config.comboBoxnLTEOrderCond          self.comboBoxnLTEOrderCond.grid(row=3, column=10)          global comboBoxnLTEOrderCond          comboBoxnLTEOrderCond = self.comboBoxnLTEOrderCond          # comboBoxnClearAllFlagC          tk.Label(self, text = "執行時間出清0:否；1:是").grid(row=4, column=1)              #輸入框          self.comboBoxnClearAllFlagC = ttk.Combobox(self, state='readonly')          self.comboBoxnClearAllFlagC['values'] = Config.comboBoxnClearAllFlagC          self.comboBoxnClearAllFlagC.grid(row=4, column=2)          global comboBoxnClearAllFlagC          comboBoxnClearAllFlagC = self.comboBoxnClearAllFlagC          # textBoxbstrClearCancelTimeC          tk.Label(self, text = "出清時間-時＋分（hhmm）{每日13:20截止)").grid(row=4, column=3)          #輸入框          self.textBoxbstrClearCancelTimeC = tk.Entry(self)          self.textBoxbstrClearCancelTimeC.grid(row=4, column=4)          global textBoxbstrClearCancelTimeC          textBoxbstrClearCancelTimeC = self.textBoxbstrClearCancelTimeC          # textBoxbstrClearAllOrderPriceC          tk.Label(self, text = "出清委託價").grid(row=4, column=5)          #輸入框          self.textBoxbstrClearAllOrderPriceC = tk.Entry(self)          self.textBoxbstrClearAllOrderPriceC.grid(row=4, column=6)          global textBoxbstrClearAllOrderPriceC          textBoxbstrClearAllOrderPriceC = self.textBoxbstrClearAllOrderPriceC          # comboBoxnClearAllPriceTypeC          tk.Label(self, text = "出清委託價方式= 1:市價;2:限價").grid(row=4, column=7)              #輸入框          self.comboBoxnClearAllPriceTypeC = ttk.Combobox(self, state='readonly')          self.comboBoxnClearAllPriceTypeC['values'] = Config.comboBoxnClearAllPriceTypeC          self.comboBoxnClearAllPriceTypeC.grid(row=4, column=8)          global comboBoxnClearAllPriceTypeC          comboBoxnClearAllPriceTypeC = self.comboBoxnClearAllPriceTypeC          # comboBoxnClearOrderCondC          tk.Label(self, text = "出清出場時效 0:ROD;3:IOC;4:FOK").grid(row=4, column=9)              #輸入框          self.comboBoxnClearOrderCondC = ttk.Combobox(self, state='readonly')          self.comboBoxnClearOrderCondC['values'] = Config.comboBoxnClearOrderCondC          self.comboBoxnClearOrderCondC.grid(row=4, column=10)          global comboBoxnClearOrderCondC          comboBoxnClearOrderCondC = self.comboBoxnClearOrderCondC          # comboBoxnFinalClearFlagC          tk.Label(self, text = "盤後定盤交易 0:否；1:是").grid(row=5, column=1)              #輸入框          self.comboBoxnFinalClearFlagC = ttk.Combobox(self, state='readonly')          self.comboBoxnFinalClearFlagC['values'] = Config.comboBoxnFinalClearFlagC          self.comboBoxnFinalClearFlagC.grid(row=5, column=2)          global comboBoxnFinalClearFlagC          comboBoxnFinalClearFlagC = self.comboBoxnFinalClearFlagC          # buttonSendStockStrategyClear          self.buttonSendStockStrategyClear = tk.Button(self)          self.buttonSendStockStrategyClear["text"] = "出清送出"          self.buttonSendStockStrategyClear["command"] = self.buttonSendStockStrategyClear\_Click          self.buttonSendStockStrategyClear.grid(row=6, column=1) |

|  |
| --- |
| Config.py |
| # ClearForm  comboBoxnBuySellC =(      "0:買", "1:賣"  )  comboBoxnOrderTypeC =(      "0:現股", "3:融資", "4:融券"  )  comboBoxnGTEFlag =(      "0:否", "1:是"  )  comboBoxnGTEMarketPrice =(      "1:市價", "2:限價"  )  comboBoxnGTEOrderCond =(      "0:ROD", "3:IOC", "4:FOK"  )  comboBoxnLTEFlag =(      "0:否", "1:是"  )  comboBoxnLTEMarketPrice =(      "1:市價", "2:限價"  )  comboBoxnLTEOrderCond =(      "0:ROD", "3:IOC", "4:FOK"  )  comboBoxnClearAllFlagC =(      "0:否", "1:是"  )  comboBoxnClearAllPriceTypeC =(      "1:市價", "2:限價"  )  comboBoxnClearOrderCondC =(      "0:ROD", "3:IOC", "4:FOK"  )  comboBoxnFinalClearFlagC =(      "0:否", "1:是"  ) |
| 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 buttonSendStockStrategyClear\_Click(self):            pOrder = sk.STOCKSTRATEGYORDEROUT()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNoC.get()          pOrder.bstrClearCancelTime = textBoxbstrClearCancelTimeC.get()          pOrder.bstrClearAllOrderPrice = textBoxbstrClearAllOrderPriceC.get()          pOrder.bstrLTETriggerPrice = textBoxbstrLTETriggerPrice.get()          pOrder.bstrLTEOrderPrice = textBoxbstrLTEOrderPrice.get()          pOrder.bstrGTETriggerPrice = textBoxbstrGTETriggerPrice.get()          pOrder.bstrGTEOrderPrice = textBoxbstrGTEOrderPrice.get()          pOrder.nQty = int(textBoxnQtyC.get())          if (comboBoxnBuySellC.get() == "0:買"):              pOrder.nBuySell = 0          elif (comboBoxnBuySellC.get() == "1:賣"):              pOrder.nBuySell = 1          if (comboBoxnClearAllFlagC.get() == "0:否"):              pOrder.nClearAllFlag = 0          elif (comboBoxnClearAllFlagC.get() == "1:是"):              pOrder.nClearAllFlag = 1          if (comboBoxnClearAllPriceTypeC.get() == "1:市價"):              pOrder.nClearAllPriceType = 1          elif (comboBoxnClearAllPriceTypeC.get() == "2:限價"):              pOrder.nClearAllPriceType = 2          if (comboBoxnFinalClearFlagC.get() == "0:否"):              pOrder.nFinalClearFlag = 0          elif (comboBoxnFinalClearFlagC.get() == "1:是"):              pOrder.nFinalClearFlag = 1          if (comboBoxnOrderTypeC.get() == "0:現股"):              pOrder.nOrderType = 0          elif (comboBoxnOrderTypeC.get() == "3:融資"):              pOrder.nOrderType = 3          elif (comboBoxnOrderTypeC.get() == "4:融券"):              pOrder.nOrderType = 4          if (comboBoxnLTEFlag.get() == "0:否"):              pOrder.nLTEFlag = 0          elif (comboBoxnLTEFlag.get() == "1:是"):              pOrder.nLTEFlag = 1          if (comboBoxnLTEMarketPrice.get() == "1:市價"):              pOrder.nLTEMarketPrice = 1          elif (comboBoxnLTEMarketPrice.get() == "2:限價"):              pOrder.nLTEMarketPrice = 2          if (comboBoxnGTEFlag.get() == "0:否"):              pOrder.nGTEFlag = 0          elif (comboBoxnGTEFlag.get() == "1:是"):              pOrder.nGTEFlag = 1          if (comboBoxnGTEMarketPrice.get() == "1:市價"):              pOrder.nGTEMarketPrice = 1          elif (comboBoxnGTEMarketPrice.get() == "2:限價"):              pOrder.nGTEMarketPrice = 2          if (comboBoxnGTEOrderCond.get() == "0:ROD"):              pOrder.nGTEOrderCond = 0          elif (comboBoxnGTEOrderCond.get() == "3:IOC"):              pOrder.nGTEOrderCond = 3          elif (comboBoxnGTEOrderCond.get() == "4:FOK"):              pOrder.nGTEOrderCond = 4          if (comboBoxnLTEOrderCond.get() == "0:ROD"):              pOrder.nLTEOrderCond = 0          elif (comboBoxnLTEOrderCond.get() == "3:IOC"):              pOrder.nLTEOrderCond = 3          elif (comboBoxnLTEOrderCond.get() == "4:FOK"):              pOrder.nLTEOrderCond = 4          if (comboBoxnClearOrderCondC.get() == "0:ROD"):              pOrder.nClearOrderCond = 0          elif (comboBoxnClearOrderCondC.get() == "3:IOC"):              pOrder.nClearOrderCond = 3          elif (comboBoxnClearOrderCondC.get() == "4:FOK"):              pOrder.nClearOrderCond = 4          # 送出證券智慧單出清條件委託。          bstrMessage,nCode= m\_pSKOrder.SendStockStrategyClear(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendStockStrategyClear】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### SendStockStrategyMIT

|  |  |  |
| --- | --- | --- |
| 送出證券智慧單MIT條件委託。  智慧單用戶請注意備註。 | | |
| **宣告** | Long SendStockStrategyMIT ([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **STOCKSTRATEGYORDERMIT**\* 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[**STOCKSTRATEGYORDERMIT**](#_MIT單)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，訊息內容則為委託日期、條件單號 (智慧單序號)等委託資料。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。 |
| **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** | 使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder_1)取得。  需簽署證券智慧單風險預告書及未簽署者可至金融網－同意書簽署專區申請。   * 自V2.13.30(含)版本起，客戶可自設觸價方向(物件[**STOCKSTRATEGYORDERMIT**](#_MIT單))。 * 相關回報部分：被動回報[GetTSSmartStrategyReport](#_GetTSSmartStrategyReport)，主動回報請參考回報文件。 * ［智慧單用戶請注意］：智慧單為獨立運作機制，若盤中已自行出清或回補，請務必於盤中自行取消已設定的智慧單，避免條件觸發後重覆成交。 * 預風控說明：此功能將於主機收到智慧單委託後立刻進行風控計算，可以大幅縮短觸價至送出委託的所耗時間。 * 預風控注意事項：此功能會在觸發前預先扣除條件中委託所使用之額度(包括能夠進行操作的庫存)， 使用時請務必留意是否有進行中的MIT智慧單與委託，在未達成條件前取消智慧單委託可釋放占用額度。 | |

|  |
| --- |
| Python範例程式碼\_TSStrategyOrder.py類別MITForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # 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)          # textBoxbstrStockNoMIT          tk.Label(self, text = "委託股票代號").grid(row=0, column=1)          #輸入框          self.textBoxbstrStockNoMIT = tk.Entry(self)          self.textBoxbstrStockNoMIT.grid(row=0, column=2)          global textBoxbstrStockNoMIT          textBoxbstrStockNoMIT = self.textBoxbstrStockNoMIT          # textBoxbstrDealPriceMIT          tk.Label(self, text = "成交價(當下市價)，洗價機留存用").grid(row=0, column=3)          #輸入框          self.textBoxbstrDealPriceMIT = tk.Entry(self)          self.textBoxbstrDealPriceMIT.grid(row=0, column=4)          global textBoxbstrDealPriceMIT          textBoxbstrDealPriceMIT = self.textBoxbstrDealPriceMIT          # comboBoxnOrderTypeMIT          tk.Label(self, text = "委託交易別(0:現股, 3:融資, 4:融券, 8:無券普賣)").grid(row=1, column=1)              #輸入框          self.comboBoxnOrderTypeMIT = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderTypeMIT['values'] = Config.comboBoxnOrderTypeMIT          self.comboBoxnOrderTypeMIT.grid(row=1, column=2)          global comboBoxnOrderTypeMIT          comboBoxnOrderTypeMIT = self.comboBoxnOrderTypeMIT          # comboBoxnOrderCondMIT          tk.Label(self, text = "0:ROD;3:IOC;4:FOK:").grid(row=1, column=3)              #輸入框          self.comboBoxnOrderCondMIT = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderCondMIT['values'] = Config.comboBoxnOrderCondMIT          self.comboBoxnOrderCondMIT.grid(row=1, column=4)          global comboBoxnOrderCondMIT          comboBoxnOrderCondMIT = self.comboBoxnOrderCondMIT          # comboBoxnBuySellMIT          tk.Label(self, text = "0買;1賣").grid(row=1, column=5)              #輸入框          self.comboBoxnBuySellMIT = ttk.Combobox(self, state='readonly')          self.comboBoxnBuySellMIT['values'] = Config.comboBoxnBuySellMIT          self.comboBoxnBuySellMIT.grid(row=1, column=6)          global comboBoxnBuySellMIT          comboBoxnBuySellMIT = self.comboBoxnBuySellMIT          # comboBoxnTriggerDirMIT          tk.Label(self, text = "觸價方向(1:GTE大於等於,2:LTE小於等於)").grid(row=2, column=1)              #輸入框          self.comboBoxnTriggerDirMIT = ttk.Combobox(self, state='readonly')          self.comboBoxnTriggerDirMIT['values'] = Config.comboBoxnTriggerDirMIT          self.comboBoxnTriggerDirMIT.grid(row=2, column=2)          global comboBoxnTriggerDirMIT          comboBoxnTriggerDirMIT = self.comboBoxnTriggerDirMIT          # textBoxbstrTriggerPriceMIT          tk.Label(self, text = "觸發價").grid(row=2, column=3)          #輸入框          self.textBoxbstrTriggerPriceMIT = tk.Entry(self)          self.textBoxbstrTriggerPriceMIT.grid(row=2, column=4)          global textBoxbstrTriggerPriceMIT          textBoxbstrTriggerPriceMIT = self.textBoxbstrTriggerPriceMIT          # textBoxnQtyMIT          tk.Label(self, text = "委託張數").grid(row=2, column=5)          #輸入框          self.textBoxnQtyMIT = tk.Entry(self)          self.textBoxnQtyMIT.grid(row=2, column=6)          global textBoxnQtyMIT          textBoxnQtyMIT = self.textBoxnQtyMIT          # comboBoxnOrderPriceTypeMIT          tk.Label(self, text = "委託價類別(1:市價; 2:限價)").grid(row=2, column=7)              #輸入框          self.comboBoxnOrderPriceTypeMIT = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderPriceTypeMIT['values'] = Config.comboBoxnOrderPriceTypeMIT          self.comboBoxnOrderPriceTypeMIT.grid(row=2, column=8)          global comboBoxnOrderPriceTypeMIT          comboBoxnOrderPriceTypeMIT = self.comboBoxnOrderPriceTypeMIT          # textBoxbstrOrderPriceMIT          tk.Label(self, text = "委託價(若為市價單, 委託價請填0)").grid(row=2, column=9)          #輸入框          self.textBoxbstrOrderPriceMIT = tk.Entry(self)          self.textBoxbstrOrderPriceMIT.grid(row=2, column=10)          global textBoxbstrOrderPriceMIT          textBoxbstrOrderPriceMIT = self.textBoxbstrOrderPriceMIT          # comboBoxnLongActionFlagMIT          tk.Label(self, text = "是否為長效單(0:否, 1:是):").grid(row=3, column=1)              #輸入框          self.comboBoxnLongActionFlagMIT = ttk.Combobox(self, state='readonly')          self.comboBoxnLongActionFlagMIT['values'] = Config.comboBoxnLongActionFlagMIT          self.comboBoxnLongActionFlagMIT.grid(row=3, column=2)          global comboBoxnLongActionFlagMIT          comboBoxnLongActionFlagMIT = self.comboBoxnLongActionFlagMIT          # textBoxbstrLongEndDateMIT          tk.Label(self, text = "長效單結束日期(YYYYMMDD共8碼, EX:20220630)").grid(row=3, column=3)          #輸入框          self.textBoxbstrLongEndDateMIT = tk.Entry(self)          self.textBoxbstrLongEndDateMIT.grid(row=3, column=4)          global textBoxbstrLongEndDateMIT          textBoxbstrLongEndDateMIT = self.textBoxbstrLongEndDateMIT            # comboBoxnLATypeMIT          tk.Label(self, text = "觸發結束條件(1:效期內觸發即失效, 3:效期內完全成交即失效):").grid(row=3, column=5)              #輸入框          self.comboBoxnLATypeMIT = ttk.Combobox(self, state='readonly')          self.comboBoxnLATypeMIT['values'] = Config.comboBoxnLATypeMIT          self.comboBoxnLATypeMIT.grid(row=3, column=6)          global comboBoxnLATypeMIT          comboBoxnLATypeMIT = self.comboBoxnLATypeMIT          # comboBoxnPreRiskFlag          tk.Label(self, text = "預風控功能 0:關閉預風控 ;1:開啟預風控(不支援信用交易)").grid(row=4, column=1)              #輸入框          self.comboBoxnPreRiskFlag = ttk.Combobox(self, state='readonly')          self.comboBoxnPreRiskFlag['values'] = Config.comboBoxnPreRiskFlag          self.comboBoxnPreRiskFlag.grid(row=4, column=2)          global comboBoxnPreRiskFlag          comboBoxnPreRiskFlag = self.comboBoxnPreRiskFlag          # buttonSendStockStrategyMIT          self.buttonSendStockStrategyMIT = tk.Button(self)          self.buttonSendStockStrategyMIT["text"] = "MIT送出"          self.buttonSendStockStrategyMIT["command"] = self.buttonSendStockStrategyMIT\_Click          self.buttonSendStockStrategyMIT.grid(row=5, column=1) |

|  |
| --- |
| Config.py |
| # MITForm  comboBoxnOrderTypeMIT =(      "0:現股", "3:融資", "4:融券" , "8:無券普賣"  )  comboBoxnOrderCondMIT =(      "0:ROD", "3:IOC", "4:FOK"  )  comboBoxnBuySellMIT =(      "0:買", "1:賣"  )  comboBoxnTriggerDirMIT =(      "1:GTE大於等於", "2:LTE小於等於"  )  comboBoxnOrderPriceTypeMIT =(      "1:市價", "2:限價"  )  comboBoxnLongActionFlagMIT =(      "0:否", "1:是"  )  comboBoxnLATypeMIT =(      "0:非長效單", "1:效期內觸發即失效", "3:效期內完全成交即失效"  )  comboBoxnPreRiskFlag =(      "0:關閉預風控", "1:開啟預風控"  ) |
| 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 buttonSendStockStrategyMIT\_Click(self):            pOrder = sk.STOCKSTRATEGYORDERMIT()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNoMIT.get()          pOrder.bstrOrderPrice = textBoxbstrOrderPriceMIT.get()          pOrder.bstrTriggerPrice = textBoxbstrTriggerPriceMIT.get()          pOrder.bstrDealPrice = textBoxbstrDealPriceMIT.get()          pOrder.bstrLongEndDate = textBoxbstrLongEndDateMIT.get()          pOrder.nQty = int(textBoxnQtyMIT.get())          if (comboBoxnOrderTypeMIT.get() == "0:現股"):              pOrder.nOrderType = 0          elif (comboBoxnOrderTypeMIT.get() == "3:融資"):              pOrder.nOrderType = 3          elif (comboBoxnOrderTypeMIT.get() == "4:融券"):              pOrder.nOrderType = 4          elif (comboBoxnOrderTypeMIT.get() == "8:無券普賣"):              pOrder.nOrderType = 8          if (comboBoxnBuySellMIT.get() == "0:買"):              pOrder.nBuySell = 0          elif (comboBoxnBuySellMIT.get() == "1:賣"):              pOrder.nBuySell = 1          if (comboBoxnOrderPriceTypeMIT.get() == "1:市價"):              pOrder.nOrderPriceType = 1          elif (comboBoxnOrderPriceTypeMIT.get() == "2:限價"):              pOrder.nOrderPriceType = 2          if (comboBoxnOrderCondMIT.get() == "0:ROD"):              pOrder.nOrderCond = 0          elif (comboBoxnOrderCondMIT.get() == "3:IOC"):              pOrder.nOrderCond = 3          elif (comboBoxnOrderCondMIT.get() == "4:FOK"):              pOrder.nOrderCond = 4          if (comboBoxnTriggerDirMIT.get() == "1:GTE大於等於"):              pOrder.nTriggerDir = 1          elif (comboBoxnTriggerDirMIT.get() == "2:LTE小於等於"):              pOrder.nTriggerDir = 2          if (comboBoxnPreRiskFlag.get() == "0:關閉預風控"):              pOrder.nPreRiskFlag = 0          elif (comboBoxnPreRiskFlag.get() == "1:開啟預風控"):              pOrder.nPreRiskFlag = 1          if (comboBoxnLongActionFlagMIT.get() == "0:否"):              pOrder.nLongActionFlag = 0          elif (comboBoxnLongActionFlagMIT.get() == "1:是"):              pOrder.nLongActionFlag = 1          if (comboBoxnLATypeMIT.get() == "1:效期內觸發即失效"):              pOrder.nLAType = 1          elif (comboBoxnLATypeMIT.get() == "3:效期內完全成交即失效"):              pOrder.nLAType = 3          # 送出證券智慧單MIT條件委託。          bstrMessage,nCode= m\_pSKOrder.SendStockStrategyMIT(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendStockStrategyMIT】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### SendStockStrategyOCO

|  |  |  |
| --- | --- | --- |
| 送出證券智慧單二擇一OCO條件委託。  智慧單用戶請注意備註。 | | |
| **宣告** | Long SendStockStrategyOCO ([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **STOCKSTRATEGYORDEROCO**\* 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[**STOCKSTRATEGYORDEROCO**](#_STOCKSTRATEGYORDEROCO(證券智慧單物件-OCO))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，訊息內容則為委託日期、條件單號 (智慧單序號)等委託資料。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。 |
| **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** | 使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder_1)取得。  需簽署證券智慧單風險預告書及未簽署者可至金融網－同意書簽署專區申請。  ［智慧單用戶請注意］：智慧單為獨立運作機制，若盤中已自行出清或回補，請務必於盤中自行取消已設定的智慧單，避免條件觸發後重覆成交。 | |

|  |
| --- |
| Python範例程式碼\_TSStrategyOrder.py類別OCOForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # 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)          # textBoxbstrStockNoOCO          tk.Label(self, text = "委託股票代號").grid(row=0, column=1)          #輸入框          self.textBoxbstrStockNoOCO = tk.Entry(self)          self.textBoxbstrStockNoOCO.grid(row=0, column=2)          global textBoxbstrStockNoOCO          textBoxbstrStockNoOCO = self.textBoxbstrStockNoOCO          # textBoxbstrTriggerUp          tk.Label(self, text = "第一腳 觸發價").grid(row=1, column=1)          #輸入框          self.textBoxbstrTriggerUp = tk.Entry(self)          self.textBoxbstrTriggerUp.grid(row=1, column=2)          global textBoxbstrTriggerUp          textBoxbstrTriggerUp = self.textBoxbstrTriggerUp          # comboBoxnBuySellUp          tk.Label(self, text = "第一腳 0:買;1:賣").grid(row=1, column=3)              #輸入框          self.comboBoxnBuySellUp = ttk.Combobox(self, state='readonly')          self.comboBoxnBuySellUp['values'] = Config.comboBoxnBuySellUp          self.comboBoxnBuySellUp.grid(row=1, column=4)          global comboBoxnBuySellUp          comboBoxnBuySellUp = self.comboBoxnBuySellUp          # comboBoxnOrderCondUp          tk.Label(self, text = "第一腳 委託時效(0:ROD,3:IOC,4:FOK)").grid(row=1, column=5)              #輸入框          self.comboBoxnOrderCondUp = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderCondUp['values'] = Config.comboBoxnOrderCondUp          self.comboBoxnOrderCondUp.grid(row=1, column=6)          global comboBoxnOrderCondUp          comboBoxnOrderCondUp = self.comboBoxnOrderCondUp          # comboBoxnOrderTypeUp          tk.Label(self, text = "第一腳 委託交易別(0:現股, 3:融資, 4:融券)").grid(row=1, column=7)              #輸入框          self.comboBoxnOrderTypeUp = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderTypeUp['values'] = Config.comboBoxnOrderTypeUp          self.comboBoxnOrderTypeUp.grid(row=1, column=8)          global comboBoxnOrderTypeUp          comboBoxnOrderTypeUp = self.comboBoxnOrderTypeUp          # textBoxbstrOrderPriceOCO          tk.Label(self, text = "第一腳 委託價").grid(row=1, column=9)          #輸入框          self.textBoxbstrOrderPriceOCO = tk.Entry(self)          self.textBoxbstrOrderPriceOCO.grid(row=1, column=10)          global textBoxbstrOrderPriceOCO          textBoxbstrOrderPriceOCO = self.textBoxbstrOrderPriceOCO          # comboBoxnOrderPriceTypeUp          tk.Label(self, text = "第一腳 委託價類別(1:市價; 2:限價)").grid(row=1, column=11)              #輸入框          self.comboBoxnOrderPriceTypeUp = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderPriceTypeUp['values'] = Config.comboBoxnOrderPriceTypeUp          self.comboBoxnOrderPriceTypeUp.grid(row=1, column=12)          global comboBoxnOrderPriceTypeUp          comboBoxnOrderPriceTypeUp = self.comboBoxnOrderPriceTypeUp          # textBoxbstrTriggerDown          tk.Label(self, text = "第二腳 觸發價").grid(row=2, column=1)          #輸入框          self.textBoxbstrTriggerDown = tk.Entry(self)          self.textBoxbstrTriggerDown.grid(row=2, column=2)          global textBoxbstrTriggerDown          textBoxbstrTriggerDown = self.textBoxbstrTriggerDown          # comboBoxnBuySellDown          tk.Label(self, text = "第二腳 0:買;1:賣").grid(row=2, column=3)              #輸入框          self.comboBoxnBuySellDown = ttk.Combobox(self, state='readonly')          self.comboBoxnBuySellDown['values'] = Config.comboBoxnBuySellDown          self.comboBoxnBuySellDown.grid(row=2, column=4)          global comboBoxnBuySellDown          comboBoxnBuySellDown = self.comboBoxnBuySellDown          # comboBoxnOrderCondDown          tk.Label(self, text = "第二腳 委託時效(0:ROD,3:IOC,4:FOK)").grid(row=2, column=5)              #輸入框          self.comboBoxnOrderCondDown = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderCondDown['values'] = Config.comboBoxnOrderCondDown          self.comboBoxnOrderCondDown.grid(row=2, column=6)          global comboBoxnOrderCondDown          comboBoxnOrderCondDown = self.comboBoxnOrderCondDown          # comboBoxnOrderTypeDown          tk.Label(self, text = "第二腳 委託交易別(0:現股, 3:融資, 4:融券)").grid(row=2, column=7)              #輸入框          self.comboBoxnOrderTypeDown = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderTypeDown['values'] = Config.comboBoxnOrderTypeDown          self.comboBoxnOrderTypeDown.grid(row=2, column=8)          global comboBoxnOrderTypeDown          comboBoxnOrderTypeDown = self.comboBoxnOrderTypeDown          # textBoxbstrOrderPrice2          tk.Label(self, text = "第二腳 委託價").grid(row=2, column=9)          #輸入框          self.textBoxbstrOrderPrice2 = tk.Entry(self)          self.textBoxbstrOrderPrice2.grid(row=2, column=10)          global textBoxbstrOrderPrice2          textBoxbstrOrderPrice2 = self.textBoxbstrOrderPrice2          # comboBoxnOrderPriceTypeDown          tk.Label(self, text = "第二腳 委託價類別(1:市價; 2:限價)").grid(row=2, column=11)              #輸入框          self.comboBoxnOrderPriceTypeDown = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderPriceTypeDown['values'] = Config.comboBoxnOrderPriceTypeDown          self.comboBoxnOrderPriceTypeDown.grid(row=2, column=12)          global comboBoxnOrderPriceTypeDown          comboBoxnOrderPriceTypeDown = self.comboBoxnOrderPriceTypeDown          # textBoxnQtyOCO          tk.Label(self, text = "委託張數").grid(row=3, column=1)          #輸入框          self.textBoxnQtyOCO = tk.Entry(self)          self.textBoxnQtyOCO.grid(row=3, column=2)          global textBoxnQtyOCO          textBoxnQtyOCO = self.textBoxnQtyOCO          # buttonSendStockStrategyOCO          self.buttonSendStockStrategyOCO = tk.Button(self)          self.buttonSendStockStrategyOCO["text"] = "OCO送出"          self.buttonSendStockStrategyOCO["command"] = self.buttonSendStockStrategyOCO\_Click          self.buttonSendStockStrategyOCO.grid(row=4, column=1) |

|  |
| --- |
| Config.py |
| # OCOForm  comboBoxnBuySellUp =(      "0:買", "1:賣"  )  comboBoxnOrderCondUp =(      "0:ROD", "3:IOC", "4:FOK"  )  comboBoxnOrderTypeUp =(      "0:現股", "3:融資", "4:融券", "8:無券普賣"  )  comboBoxnOrderPriceTypeUp =(      "1:市價", "2:限價"  )  comboBoxnBuySellDown =(      "0:買", "1:賣"  )  comboBoxnOrderCondDown =(      "0:ROD", "3:IOC", "4:FOK"  )  comboBoxnOrderTypeDown =(      "0:現股", "3:融資", "4:融券", "8:無券普賣"  )  comboBoxnOrderPriceTypeDown =(      "1:市價", "2:限價"  ) |
| 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 buttonSendStockStrategyOCO\_Click(self):            pOrder = sk.STOCKSTRATEGYORDEROCO()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNoOCO.get()          pOrder.nQty = int(textBoxnQtyOCO.get())          pOrder.bstrOrderPrice = textBoxbstrOrderPriceOCO.get()          pOrder.bstrTriggerUp = textBoxbstrTriggerUp.get()          pOrder.bstrOrderPrice2 = textBoxbstrOrderPrice2.get()          pOrder.bstrTriggerDown = textBoxbstrTriggerDown.get()          if (comboBoxnBuySellUp.get() == "0:買"):              pOrder.nBuySellUp = 0          elif (comboBoxnBuySellUp.get() == "1:賣"):              pOrder.nBuySellUp = 1          if (comboBoxnBuySellDown.get() == "0:買"):              pOrder.nBuySellDown = 0          elif (comboBoxnBuySellDown.get() == "1:賣"):              pOrder.nBuySellDown = 1          if (comboBoxnOrderCondUp.get() == "0:ROD"):              pOrder.nOrderCondUp = 0          elif (comboBoxnOrderCondUp.get() == "3:IOC"):              pOrder.nOrderCondUp = 3          elif (comboBoxnOrderCondUp.get() == "4:FOK"):              pOrder.nOrderCondUp = 4          if (comboBoxnOrderCondDown.get() == "0:ROD"):              pOrder.nOrderCondDown = 0          elif (comboBoxnOrderCondDown.get() == "3:IOC"):              pOrder.nOrderCondDown = 3          elif (comboBoxnOrderCondDown.get() == "4:FOK"):              pOrder.nOrderCondDown = 4          if (comboBoxnOrderTypeUp.get() == "0:現股"):              pOrder.nOrderTypeUp = 0          elif (comboBoxnOrderTypeUp.get() == "3:融資"):              pOrder.nOrderTypeUp = 3          elif (comboBoxnOrderTypeUp.get() == "4:融券"):              pOrder.nOrderTypeUp = 4          elif (comboBoxnOrderTypeUp.get() == "8:無券普賣"):              pOrder.nOrderTypeUp = 8          if (comboBoxnOrderTypeDown.get() == "0:現股"):              pOrder.nOrderTypeDown = 0          elif (comboBoxnOrderTypeDown.get() == "3:融資"):              pOrder.nOrderTypeDown = 3          elif (comboBoxnOrderTypeDown.get() == "4:融券"):              pOrder.nOrderTypeDown = 4          elif (comboBoxnOrderTypeDown.get() == "8:無券普賣"):              pOrder.nOrderTypeDown = 8          if (comboBoxnOrderPriceTypeUp.get() == "1:市價"):              pOrder.nOrderPriceTypeUp = 1          elif (comboBoxnOrderPriceTypeUp.get() == "2:限價"):              pOrder.nOrderPriceTypeUp = 2          if (comboBoxnOrderPriceTypeDown.get() == "1:市價"):              pOrder.nOrderPriceTypeDown = 1          elif (comboBoxnOrderPriceTypeDown.get() == "2:限價"):              pOrder.nOrderPriceTypeDown = 2          # 送出證券智慧單二擇一OCO條件委託。          bstrMessage,nCode= m\_pSKOrder.SendStockStrategyOCO(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendStockStrategyOCO】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### SendStockStrategyMIOC

|  |  |  |
| --- | --- | --- |
| 送出證券智慧單多次IOC條件委託。  智慧單用戶請注意備註。 | | |
| **宣告** | Long SendStockStrategyMIOC ([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **STOCKSTRATEGYORDERMIOC**\* 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[**STOCKSTRATEGYORDERMIOC**](#_STOCKSTRATEGYORDERMIOC(證券智慧單物件-多次IO_1)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，訊息內容則為委託日期、條件單號 (智慧單序號)等委託資料。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。 |
| **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** | 使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder_1)取得。  需簽署證券智慧單風險預告書及未簽署者可至金融網－同意書簽署專區申請。  ［智慧單用戶請注意］：智慧單為獨立運作機制，若盤中已自行出清或回補，請務必於盤中自行取消已設定的智慧單，避免條件觸發後重覆成交。 | |

|  |
| --- |
| Python範例程式碼\_TSStrategyOrder.py類別 MIOCForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # 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)          # textBoxbstrStockNoMIOC          tk.Label(self, text = "委託股票代號").grid(row=0, column=1)          #輸入框          self.textBoxbstrStockNoMIOC = tk.Entry(self)          self.textBoxbstrStockNoMIOC.grid(row=0, column=2)          global textBoxbstrStockNoMIOC          textBoxbstrStockNoMIOC = self.textBoxbstrStockNoMIOC          # comboBoxnPrimeMIOC          tk.Label(self, text = "0:上市; 1:上櫃").grid(row=0, column=3)              #輸入框          self.comboBoxnPrimeMIOC = ttk.Combobox(self, state='readonly')          self.comboBoxnPrimeMIOC['values'] = Config.comboBoxnPrimeMIOC          self.comboBoxnPrimeMIOC.grid(row=0, column=4)          global comboBoxnPrimeMIOC          comboBoxnPrimeMIOC = self.comboBoxnPrimeMIOC          # comboBoxnOrderTypeMIOC          tk.Label(self, text = "委託交易別(0:現股, 3:融資, 4:融券, 8:無券普賣)").grid(row=1, column=1)              #輸入框          self.comboBoxnOrderTypeMIOC = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderTypeMIOC['values'] = Config.comboBoxnOrderTypeMIOC          self.comboBoxnOrderTypeMIOC.grid(row=1, column=2)          global comboBoxnOrderTypeMIOC          comboBoxnOrderTypeMIOC = self.comboBoxnOrderTypeMIOC          # comboBoxnBuySellMIOC1          tk.Label(self, text = "1:買 2:賣").grid(row=1, column=3)              #輸入框          self.comboBoxnBuySellMIOC1 = ttk.Combobox(self, state='readonly')          self.comboBoxnBuySellMIOC1['values'] = Config.comboBoxnBuySellMIOC1          self.comboBoxnBuySellMIOC1.grid(row=1, column=4)          global comboBoxnBuySellMIOC1          comboBoxnBuySellMIOC1 = self.comboBoxnBuySellMIOC1          # comboBoxnOrderPriceTypeMIOC          tk.Label(self, text = "0:市價; 1:(買單)委賣價或(賣單)委買價; 委買、賣價實際價格由中台決定").grid(row=2, column=1)              #輸入框          self.comboBoxnOrderPriceTypeMIOC = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderPriceTypeMIOC['values'] = Config.comboBoxnOrderPriceTypeMIOC          self.comboBoxnOrderPriceTypeMIOC.grid(row=2, column=2)          global comboBoxnOrderPriceTypeMIOC          comboBoxnOrderPriceTypeMIOC = self.comboBoxnOrderPriceTypeMIOC          # textBoxnOneceQtyLimit          tk.Label(self, text = "單次交易張數上限").grid(row=2, column=3)          #輸入框          self.textBoxnOneceQtyLimit = tk.Entry(self)          self.textBoxnOneceQtyLimit.grid(row=2, column=4)          global textBoxnOneceQtyLimit          textBoxnOneceQtyLimit = self.textBoxnOneceQtyLimit          # textBoxbstrOrderPriceUp          tk.Label(self, text = "委託價上限").grid(row=3, column=1)          #輸入框          self.textBoxbstrOrderPriceUp = tk.Entry(self)          self.textBoxbstrOrderPriceUp.grid(row=3, column=2)          global textBoxbstrOrderPriceUp          textBoxbstrOrderPriceUp = self.textBoxbstrOrderPriceUp          # textBoxbstrOrderPriceDown          tk.Label(self, text = "委託價下限").grid(row=3, column=3)          #輸入框          self.textBoxbstrOrderPriceDown = tk.Entry(self)          self.textBoxbstrOrderPriceDown.grid(row=3, column=4)          global textBoxbstrOrderPriceDown          textBoxbstrOrderPriceDown = self.textBoxbstrOrderPriceDown          # textBoxnTotalQty          tk.Label(self, text = "總委張數").grid(row=3, column=5)          #輸入框          self.textBoxnTotalQty = tk.Entry(self)          self.textBoxnTotalQty.grid(row=3, column=6)          global textBoxnTotalQty          textBoxnTotalQty = self.textBoxnTotalQty          # buttonSendStockStrategyMIOC          self.buttonSendStockStrategyMIOC = tk.Button(self)          self.buttonSendStockStrategyMIOC["text"] = "MIOC送出"          self.buttonSendStockStrategyMIOC["command"] = self.buttonSendStockStrategyMIOC\_Click          self.buttonSendStockStrategyMIOC.grid(row=4, column=1) |

|  |
| --- |
| Config.py |
| # MIOCForm  comboBoxnPrimeMIOC =(      "0:上市", "1:上櫃"  )  comboBoxnOrderTypeMIOC =(      "0:現股", "3:融資", "4:融券" , "8:無券普賣"  )  comboBoxnBuySellMIOC1 =(      "1:買", "2:賣"  )  comboBoxnOrderPriceTypeMIOC =(      "0:市價", "1:(買單)委賣價或(賣單)委買價"  ) |
| 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 buttonSendStockStrategyMIOC\_Click(self):            pOrder = sk.STOCKSTRATEGYORDERMIOC()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNoMIOC.get()          pOrder.bstrOrderPriceUp = textBoxbstrOrderPriceUp.get()          pOrder.bstrOrderPriceDown = textBoxbstrOrderPriceDown.get()          pOrder.nOneceQtyLimit = int(textBoxnOneceQtyLimit.get())          pOrder.nTotalQty = int(textBoxnTotalQty.get())          if (comboBoxnPrimeMIOC.get() == "0:上市"):              pOrder.nPrime = 0          elif (comboBoxnPrimeMIOC.get() == "1:上櫃"):              pOrder.nPrime = 1          if (comboBoxnBuySellMIOC1.get() == "1:買"):              pOrder.nBuySell = 1          elif (comboBoxnBuySellMIOC1.get() == "2:賣"):              pOrder.nBuySell = 2          if (comboBoxnOrderTypeMIOC.get() == "0:現股"):              pOrder.nOrderType = 0          elif (comboBoxnOrderTypeMIOC.get() == "3:融資"):              pOrder.nOrderType = 3          elif (comboBoxnOrderTypeMIOC.get() == "4:融券"):              pOrder.nOrderType = 4          elif (comboBoxnOrderTypeMIOC.get() == "8:無券普賣"):              pOrder.nOrderType = 8          if (comboBoxnOrderPriceTypeMIOC.get() == "0:市價"):              pOrder.nOrderPriceType = 1          elif (comboBoxnOrderPriceTypeMIOC.get() == "1:(買單)委賣價或(賣單)委買價"):              pOrder.nOrderPriceType = 2          # 送出證券智慧單多次IOC條件委託。          bstrMessage,nCode= m\_pSKOrder.SendStockStrategyMIOC(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendStockStrategyMIOC】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### SendStockStrategyMST

|  |  |  |
| --- | --- | --- |
| 送出證券智慧單移動停損條件委託。  智慧單用戶請注意備註。 | | |
| **宣告** | Long SendStockStrategyMST ([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **STOCKSTRATEGYORDERMIT**\* 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[**STOCKSTRATEGYORDERMIT**](#_MST單)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，訊息內容則為委託日期、條件單號 (智慧單序號)等委託資料。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。 |
| **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** | 使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder_1)取得。  需簽署證券智慧單風險預告書及未簽署者可至金融網－同意書簽署專區申請。   * 自V2.13.30(含)版本起，客戶可選擇是否自訂啟動價格、自訂啟動觸價方向等條件(詳參考物件[**STOCKSTRATEGYORDERMIT**](#_MST單))。 * 相關回報部分：被動回報[GetTSSmartStrategyReport](#_GetTSSmartStrategyReport)，主動回報請參考回報文件。 * ［智慧單用戶請注意］：智慧單為獨立運作機制，若盤中已自行出清或回補，請務必於盤中自行取消已設定的智慧單，避免條件觸發後重覆成交。 | |

|  |
| --- |
| Python範例程式碼\_TSStrategyOrder.py類別MSTForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # 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)          # textBoxbstrStockNoMST          tk.Label(self, text = "委託股票代號").grid(row=0, column=1)          #輸入框          self.textBoxbstrStockNoMST = tk.Entry(self)          self.textBoxbstrStockNoMST.grid(row=0, column=2)          global textBoxbstrStockNoMST          textBoxbstrStockNoMST = self.textBoxbstrStockNoMST          # comboBoxnPrimeMST          tk.Label(self, text = "0:上市; 1:上櫃").grid(row=0, column=3)              #輸入框          self.comboBoxnPrimeMST = ttk.Combobox(self, state='readonly')          self.comboBoxnPrimeMST['values'] = Config.comboBoxnPrimeMST          self.comboBoxnPrimeMST.grid(row=0, column=4)          global comboBoxnPrimeMST          comboBoxnPrimeMST = self.comboBoxnPrimeMST          # comboBoxnOrderTypeMST          tk.Label(self, text = "委託交易別(0:現股, 3:融資, 4:融券, 8:無券普賣)").grid(row=1, column=1)              #輸入框          self.comboBoxnOrderTypeMST = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderTypeMST['values'] = Config.comboBoxnOrderTypeMST          self.comboBoxnOrderTypeMST.grid(row=1, column=2)          global comboBoxnOrderTypeMST          comboBoxnOrderTypeMST = self.comboBoxnOrderTypeMST          # comboBoxnOrderCondMST          tk.Label(self, text = "委託時效(0:ROD,3:IOC,4:FOK)").grid(row=1, column=3)              #輸入框          self.comboBoxnOrderCondMST = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderCondMST['values'] = Config.comboBoxnOrderCondMST          self.comboBoxnOrderCondMST.grid(row=1, column=4)          global comboBoxnOrderCondMST          comboBoxnOrderCondMST = self.comboBoxnOrderCondMST          # comboBoxnBuySellMST          tk.Label(self, text = "0:買; 1:賣").grid(row=1, column=5)              #輸入框          self.comboBoxnBuySellMST = ttk.Combobox(self, state='readonly')          self.comboBoxnBuySellMST['values'] = Config.comboBoxnBuySellMST          self.comboBoxnBuySellMST.grid(row=1, column=6)          global comboBoxnBuySellMST          comboBoxnBuySellMST = self.comboBoxnBuySellMST          # comboBoxnTriggerMethod          tk.Label(self, text = "啟動條件(0：以市價立即啟動, 1：以自訂價格啟動)").grid(row=2, column=1)              #輸入框          self.comboBoxnTriggerMethod = ttk.Combobox(self, state='readonly')          self.comboBoxnTriggerMethod['values'] = Config.comboBoxnTriggerMethod          self.comboBoxnTriggerMethod.grid(row=2, column=2)          global comboBoxnTriggerMethod          comboBoxnTriggerMethod = self.comboBoxnTriggerMethod          # comboBoxnTriggerDirMST          tk.Label(self, text = "觸價方向(1:GTE大於等於,2:LTE小於等於)").grid(row=2, column=3)              #輸入框          self.comboBoxnTriggerDirMST = ttk.Combobox(self, state='readonly')          self.comboBoxnTriggerDirMST['values'] = Config.comboBoxnTriggerDirMST          self.comboBoxnTriggerDirMST.grid(row=2, column=4)          global comboBoxnTriggerDirMST          comboBoxnTriggerDirMST = self.comboBoxnTriggerDirMST          # textBoxbstrStartPriceMST          tk.Label(self, text = "[自訂價格啟動適用]觸發價").grid(row=2, column=5)          #輸入框          self.textBoxbstrStartPriceMST = tk.Entry(self)          self.textBoxbstrStartPriceMST.grid(row=2, column=6)          global textBoxbstrStartPriceMST          textBoxbstrStartPriceMST = self.textBoxbstrStartPriceMST          # textBoxbstrDealPriceMST          tk.Label(self, text = "[市價啟動適用]請填市價").grid(row=2, column=7)          #輸入框          self.textBoxbstrDealPriceMST = tk.Entry(self)          self.textBoxbstrDealPriceMST.grid(row=2, column=8)          global textBoxbstrDealPriceMST          textBoxbstrDealPriceMST = self.textBoxbstrDealPriceMST          # textBoxbstrMovePoint          tk.Label(self, text = "移動點數").grid(row=3, column=1)          #輸入框          self.textBoxbstrMovePoint = tk.Entry(self)          self.textBoxbstrMovePoint.grid(row=3, column=2)          global textBoxbstrMovePoint          textBoxbstrMovePoint = self.textBoxbstrMovePoint          # textBoxnQtyMST          tk.Label(self, text = "委託張數").grid(row=4, column=1)          #輸入框          self.textBoxnQtyMST = tk.Entry(self)          self.textBoxnQtyMST.grid(row=4, column=2)          global textBoxnQtyMST          textBoxnQtyMST = self.textBoxnQtyMST          # comboBoxnOrderPriceTypeMST          tk.Label(self, text = "委託價類別(1:市價; 2:限價)").grid(row=4, column=3)              #輸入框          self.comboBoxnOrderPriceTypeMST = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderPriceTypeMST['values'] = Config.comboBoxnOrderPriceTypeMST          self.comboBoxnOrderPriceTypeMST.grid(row=4, column=4)          global comboBoxnOrderPriceTypeMST          comboBoxnOrderPriceTypeMST = self.comboBoxnOrderPriceTypeMST          # textBoxbstrOrderPriceMST          tk.Label(self, text = "委託價").grid(row=4, column=5)          #輸入框          self.textBoxbstrOrderPriceMST = tk.Entry(self)          self.textBoxbstrOrderPriceMST.grid(row=4, column=6)          global textBoxbstrOrderPriceMST          textBoxbstrOrderPriceMST = self.textBoxbstrOrderPriceMST          # comboBoxnLongActionFlagMST          tk.Label(self, text = "是否為長效單(0:否, 1:是):").grid(row=5, column=1)              #輸入框          self.comboBoxnLongActionFlagMST = ttk.Combobox(self, state='readonly')          self.comboBoxnLongActionFlagMST['values'] = Config.comboBoxnLongActionFlagMST          self.comboBoxnLongActionFlagMST.grid(row=5, column=2)          global comboBoxnLongActionFlagMST          comboBoxnLongActionFlagMST = self.comboBoxnLongActionFlagMST          # textBoxbstrLongEndDateMST          tk.Label(self, text = "長效單結束日期(YYYYMMDD共8碼, EX:20220630)").grid(row=5, column=3)          #輸入框          self.textBoxbstrLongEndDateMST = tk.Entry(self)          self.textBoxbstrLongEndDateMST.grid(row=5, column=4)          global textBoxbstrLongEndDateMST          textBoxbstrLongEndDateMST = self.textBoxbstrLongEndDateMST          # comboBoxnLATypeMST          tk.Label(self, text = "觸發結束條件(1:效期內觸發即失效, 3:效期內完全成交即失效):").grid(row=5, column=5)              #輸入框          self.comboBoxnLATypeMST = ttk.Combobox(self, state='readonly')          self.comboBoxnLATypeMST['values'] = Config.comboBoxnLATypeMST          self.comboBoxnLATypeMST.grid(row=5, column=6)          global comboBoxnLATypeMST          comboBoxnLATypeMST = self.comboBoxnLATypeMST          # buttonSendStockStrategyMST          self.buttonSendStockStrategyMST = tk.Button(self)          self.buttonSendStockStrategyMST["text"] = "MST送出"          self.buttonSendStockStrategyMST["command"] = self.buttonSendStockStrategyMST\_Click          self.buttonSendStockStrategyMST.grid(row=6, column=1) |

|  |
| --- |
| Config.py |
| # MSTForm  comboBoxnPrimeMST =(      "0:上市", "1:上櫃"  )  comboBoxnOrderTypeMST =(      "0:現股", "3:融資", "4:融券" , "8:無券普賣"  )  comboBoxnOrderCondMST =(      "0:ROD", "3:IOC", "4:FOK"  )  comboBoxnBuySellMST =(      "0:買", "1:賣"  )  comboBoxnTriggerMethod =(      "0:否,由市價啟動", "1:是,由自訂價格啟動"  )  comboBoxnTriggerDirMST =(      "1:GTE大於等於", "2:LTE小於等於"  )  comboBoxnOrderPriceTypeMST =(      "1:市價", "2:限價"  )  comboBoxnLongActionFlagMST =(      "0:否", "1:是"  )  comboBoxnLATypeMST =(      "0:非長效單", "1:效期內觸發即失效", "3:效期內完全成交即失效"  ) |
| 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 buttonSendStockStrategyMST\_Click(self):            pOrder = sk.STOCKSTRATEGYORDERMIT()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNoMST.get()          pOrder.bstrOrderPrice = textBoxbstrOrderPriceMST.get()          pOrder.bstrMovePoint = textBoxbstrMovePoint.get()          pOrder.bstrDealPrice = textBoxbstrDealPriceMST.get()          pOrder.bstrStartPrice = textBoxbstrStartPriceMST.get()          pOrder.nQty = int(textBoxnQtyMST.get())          pOrder.bstrLongEndDate = textBoxbstrLongEndDateMST.get()          if (comboBoxnOrderTypeMST.get() == "0:現股"):              pOrder.nOrderType = 0          elif (comboBoxnOrderTypeMST.get() == "3融資"):              pOrder.nOrderType = 3          elif (comboBoxnOrderTypeMST.get() == "4:融券"):              pOrder.nOrderType = 4          elif (comboBoxnOrderTypeMST.get() == "8:無券普賣"):              pOrder.nOrderType = 8          if (comboBoxnBuySellMST.get() == "0:買"):              pOrder.nBuySell = 0          elif (comboBoxnBuySellMST.get() == "1:賣"):              pOrder.nBuySell = 1          if (comboBoxnOrderPriceTypeMST.get() == "1:市價"):              pOrder.nOrderPriceType = 1          elif (comboBoxnOrderPriceTypeMST.get() == "2:限價"):              pOrder.nOrderPriceType = 2          if (comboBoxnOrderCondMST.get() == "0:ROD"):              pOrder.nOrderCond = 0          elif (comboBoxnOrderCondMST.get() == "3:IOC"):              pOrder.nOrderCond = 3          elif (comboBoxnOrderCondMST.get() == "4:FOK"):              pOrder.nOrderCond = 4          if (comboBoxnTriggerDirMST.get() == "1:GTE大於等於"):              pOrder.nTriggerDir = 1          elif (comboBoxnTriggerDirMST.get() == "2:LTE小於等於"):              pOrder.nTriggerDir = 2          if (comboBoxnTriggerMethod.get() == "0:否,由市價啟動"):              pOrder.nTriggerMethod = 0          elif (comboBoxnTriggerMethod.get() == "1:是,由自訂價格啟動"):              pOrder.nTriggerMethod = 1          if (comboBoxnPrimeMST.get() == "0:上市"):              pOrder.nPrime = 0          elif (comboBoxnPrimeMST.get() == "1:上櫃"):              pOrder.nPrime = 1          if (comboBoxnLongActionFlagMST.get() == "0:否"):              pOrder.nLongActionFlag = 0          elif (comboBoxnLongActionFlagMST.get() == "1:是"):              pOrder.nLongActionFlag = 1          if (comboBoxnLATypeMST.get() == "0:非長效單"):              pOrder.nLAType = 0          elif (comboBoxnLATypeMST.get() == "1:效期內觸發即失效"):              pOrder.nLAType = 1          # 送出證券智慧單移動停損條件委託。          bstrMessage,nCode= m\_pSKOrder.SendStockStrategyMST(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendStockStrategyMST】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### SendStockStrategyAB

|  |  |  |
| --- | --- | --- |
| 送出證券智慧單看A下B單委託。  智慧單用戶請注意備註。 | | |
| **宣告** | Long SendStockStrategyAB ([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **STOCKSTRATEGYORDERMIT**\* 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[**STOCKSTRATEGYORDERMIT**](#_AB單_1)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，訊息內容則為委託日期、條件單號 (智慧單序號)等委託資料。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。 |
| **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** | 使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder_1)取得。  需簽署證券智慧單風險預告書及未簽署者可至金融網－同意書簽署專區申請。   * 相關回報部分：被動回報[GetTSSmartStrategyReport](#_GetTSSmartStrategyReport)，主動回報請參考回報文件。 * ［智慧單用戶請注意］：智慧單為獨立運作機制，若盤中已自行出清或回補，請務必於盤中自行取消已設定的智慧單，避免條件觸發後重覆成交。 | |

|  |
| --- |
| Python範例程式碼\_TSStrategyOrder.py類別ABForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # 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)          # textBoxbstrStockNo2          tk.Label(self, text = "A商品 委託股票代號").grid(row=0, column=1)          #輸入框          self.textBoxbstrStockNo2 = tk.Entry(self)          self.textBoxbstrStockNo2.grid(row=0, column=2)          global textBoxbstrStockNo2          textBoxbstrStockNo2 = self.textBoxbstrStockNo2          # comboBoxnMarketNo          tk.Label(self, text = "A商品 市場編號(1:國內證, 2:國內期, 3:國外證, 4:國外期)").grid(row=0, column=3)              #輸入框          self.comboBoxnMarketNo = ttk.Combobox(self, state='readonly')          self.comboBoxnMarketNo['values'] = Config.comboBoxnMarketNo          self.comboBoxnMarketNo.grid(row=0, column=4)          global comboBoxnMarketNo          comboBoxnMarketNo = self.comboBoxnMarketNo          # textBoxbstrExchangeNo          tk.Label(self, text = "A商品 交易所代碼(EX: TSE、TAIFEX、CME)").grid(row=0, column=5)          #輸入框          self.textBoxbstrExchangeNo = tk.Entry(self)          self.textBoxbstrExchangeNo.grid(row=0, column=6)          global textBoxbstrExchangeNo          textBoxbstrExchangeNo = self.textBoxbstrExchangeNo          # textBoxbstrStartPriceAB          tk.Label(self, text = "A商品 成交價").grid(row=0, column=7)          #輸入框          self.textBoxbstrStartPriceAB = tk.Entry(self)          self.textBoxbstrStartPriceAB.grid(row=0, column=8)          global textBoxbstrStartPriceAB          textBoxbstrStartPriceAB = self.textBoxbstrStartPriceAB          # comboBoxnTriggerDirAB          tk.Label(self, text = "A商品 觸價方向(1:GTE大於等於,2:LTE小於等於)").grid(row=1, column=1)              #輸入框          self.comboBoxnTriggerDirAB = ttk.Combobox(self, state='readonly')          self.comboBoxnTriggerDirAB['values'] = Config.comboBoxnTriggerDirAB          self.comboBoxnTriggerDirAB.grid(row=1, column=2)          global comboBoxnTriggerDirAB          comboBoxnTriggerDirAB = self.comboBoxnTriggerDirAB          # textBoxbstrTriggerPriceAB          tk.Label(self, text = "A商品 觸發價").grid(row=1, column=3)          #輸入框          self.textBoxbstrTriggerPriceAB = tk.Entry(self)          self.textBoxbstrTriggerPriceAB.grid(row=1, column=4)          global textBoxbstrTriggerPriceAB          textBoxbstrTriggerPriceAB = self.textBoxbstrTriggerPriceAB          # comboBoxnBuySellAB          tk.Label(self, text = "B商品 買賣別(0:買, 1:賣)").grid(row=2, column=1)              #輸入框          self.comboBoxnBuySellAB = ttk.Combobox(self, state='readonly')          self.comboBoxnBuySellAB['values'] = Config.comboBoxnBuySellAB          self.comboBoxnBuySellAB.grid(row=2, column=2)          global comboBoxnBuySellAB          comboBoxnBuySellAB = self.comboBoxnBuySellAB          # comboBoxnOrderTypeAB          tk.Label(self, text = "B商品 委託交易別(0:現股, 3:融資, 4:融券, 8:無券普賣)").grid(row=2, column=3)              #輸入框          self.comboBoxnOrderTypeAB = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderTypeAB['values'] = Config.comboBoxnOrderTypeAB          self.comboBoxnOrderTypeAB.grid(row=2, column=4)          global comboBoxnOrderTypeAB          comboBoxnOrderTypeAB = self.comboBoxnOrderTypeAB          # comboBoxnOrderCondAB          tk.Label(self, text = "B商品 委託時效(0:ROD,3:IOC,4:FOK)").grid(row=2, column=5)              #輸入框          self.comboBoxnOrderCondAB = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderCondAB['values'] = Config.comboBoxnOrderCondAB          self.comboBoxnOrderCondAB.grid(row=2, column=6)          global comboBoxnOrderCondAB          comboBoxnOrderCondAB = self.comboBoxnOrderCondAB          # textBoxbstrStockNoAB          tk.Label(self, text = "B商品 委託股票代號").grid(row=3, column=1)          #輸入框          self.textBoxbstrStockNoAB = tk.Entry(self)          self.textBoxbstrStockNoAB.grid(row=3, column=2)          global textBoxbstrStockNoAB          textBoxbstrStockNoAB = self.textBoxbstrStockNoAB          # comboBoxnPrimeAB          tk.Label(self, text = "B商品 0:上市,1:上櫃").grid(row=3, column=3)              #輸入框          self.comboBoxnPrimeAB = ttk.Combobox(self, state='readonly')          self.comboBoxnPrimeAB['values'] = Config.comboBoxnPrimeAB          self.comboBoxnPrimeAB.grid(row=3, column=4)          global comboBoxnPrimeAB          comboBoxnPrimeAB = self.comboBoxnPrimeAB          # textBoxnQtyAB          tk.Label(self, text = "B商品 委託張數").grid(row=3, column=5)          #輸入框          self.textBoxnQtyAB = tk.Entry(self)          self.textBoxnQtyAB.grid(row=3, column=6)          global textBoxnQtyAB          textBoxnQtyAB = self.textBoxnQtyAB          # comboBoxnOrderPriceTypeAB          tk.Label(self, text = "B商品委託價類別(1:市價; 2:限價)").grid(row=3, column=7)              #輸入框          self.comboBoxnOrderPriceTypeAB = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderPriceTypeAB['values'] = Config.comboBoxnOrderPriceTypeAB          self.comboBoxnOrderPriceTypeAB.grid(row=3, column=8)          global comboBoxnOrderPriceTypeAB          comboBoxnOrderPriceTypeAB = self.comboBoxnOrderPriceTypeAB          # textBoxbstrOrderPriceAB          tk.Label(self, text = "B商品 委託價").grid(row=3, column=9)          #輸入框          self.textBoxbstrOrderPriceAB = tk.Entry(self)          self.textBoxbstrOrderPriceAB.grid(row=3, column=10)          global textBoxbstrOrderPriceAB          textBoxbstrOrderPriceAB = self.textBoxbstrOrderPriceAB          # comboBoxnReserved          tk.Label(self, text = "是否為預約單(0:否, 1:是)A商品為國內期選商品時可選預約單，其餘市場為非預約單:").grid(row=4, column=1)              #輸入框          self.comboBoxnReserved = ttk.Combobox(self, state='readonly')          self.comboBoxnReserved['values'] = Config.comboBoxnReserved          self.comboBoxnReserved.grid(row=4, column=2)          global comboBoxnReserved          comboBoxnReserved = self.comboBoxnReserved          # buttonSendStockStrategyAB          self.buttonSendStockStrategyAB = tk.Button(self)          self.buttonSendStockStrategyAB["text"] = "AB送出"          self.buttonSendStockStrategyAB["command"] = self.buttonSendStockStrategyAB\_Click          self.buttonSendStockStrategyAB.grid(row=5, column=1) |

|  |
| --- |
| Config.py |
| # ABForm  comboBoxnMarketNo =(      "1:國內證", "2:國內期", "3:國外證", "4:國外期"  )  comboBoxnTriggerDirAB =(      "1:GTE大於等於", "2:LTE小於等於"  )  comboBoxnBuySellAB =(      "0:買", "1:賣"  )  comboBoxnOrderTypeAB =(      "0:現股", "3:融資", "4:融券" , "8:無券普賣"  )  comboBoxnOrderCondAB =(      "0:ROD", "3:IOC", "4:FOK"  )  comboBoxnPrimeAB =(      "0:上市", "1:上櫃"  )  comboBoxnOrderPriceTypeAB =(      "1:市價", "2:限價"  )  comboBoxnReserved =(      "0:否", "1:是"  ) |
| 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 buttonSendStockStrategyAB\_Click(self):            pOrder = sk.STOCKSTRATEGYORDERMIT()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNoAB.get()          pOrder.bstrOrderPrice = textBoxbstrOrderPriceAB.get()          pOrder.bstrTriggerPrice = textBoxbstrTriggerPriceAB.get()          pOrder.bstrExchangeNo = textBoxbstrExchangeNo.get()          pOrder.bstrStockNo2 = textBoxbstrStockNo2.get()          pOrder.bstrStartPrice = textBoxbstrStartPriceAB.get()          pOrder.nQty = int(textBoxnQtyAB.get())          if (comboBoxnOrderTypeAB.get() == "0:現股"):              pOrder.nOrderType = 0          elif (comboBoxnOrderTypeAB.get() == "3融資"):              pOrder.nOrderType = 3          elif (comboBoxnOrderTypeAB.get() == "4:融券"):              pOrder.nOrderType = 4          elif (comboBoxnOrderTypeAB.get() == "8:無券普賣"):              pOrder.nOrderType = 8          if (comboBoxnBuySellAB.get() == "0:買"):              pOrder.nBuySell = 0          elif (comboBoxnBuySellAB.get() == "1:賣"):              pOrder.nBuySell = 1          if (comboBoxnOrderPriceTypeAB.get() == "1:市價"):              pOrder.nOrderPriceType = 1          elif (comboBoxnOrderPriceTypeAB.get() == "2:限價"):              pOrder.nOrderPriceType = 2          if (comboBoxnOrderCondAB.get() == "0:ROD"):              pOrder.nOrderCond = 0          elif (comboBoxnOrderCondAB.get() == "3:IOC"):              pOrder.nOrderCond = 3          elif (comboBoxnOrderCondAB.get() == "4:FOK"):              pOrder.nOrderCond = 4          if (comboBoxnTriggerDirAB.get() == "1:GTE大於等於"):              pOrder.nTriggerDir = 1          elif (comboBoxnTriggerDirAB.get() == "2:LTE小於等於"):              pOrder.nTriggerDir = 2          if (comboBoxnMarketNo.get() == "1:國內證"):              pOrder.nMarketNo = 1          elif (comboBoxnMarketNo.get() == "2:國內期"):              pOrder.nMarketNo = 2          elif (comboBoxnMarketNo.get() == "3:國外證"):              pOrder.nMarketNo = 3          elif (comboBoxnMarketNo.get() == "4:國外期"):              pOrder.nMarketNo = 4          if (comboBoxnReserved.get() == "0:否"):              pOrder.nReserved = 0          elif (comboBoxnReserved.get() == "1:是"):              pOrder.nReserved = 1          if (comboBoxnPrimeAB.get() == "0:上市"):              pOrder.nPrime = 1          elif (comboBoxnPrimeAB.get() == "1:上櫃"):              pOrder.nPrime = 2          # 送出證券智慧單看A下B單委託。          bstrMessage,nCode= m\_pSKOrder.SendStockStrategyAB(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendStockStrategyAB】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### SendStockStrategyCB

|  |  |  |
| --- | --- | --- |
| 送出證券智慧單自組單委託。  智慧單用戶請注意備註。 | | |
| **宣告** | Long SendStockStrategyCB ([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **STOCKSTRATEGYORDER\*** 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[**STOCKSTRATEGYORDER**](#_證券自組單(CB)_1)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，訊息內容則為委託日期、條件單號 (智慧單序號)等委託資料。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。 |
| **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** | 使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder_1)取得。  需簽署證券智慧單風險預告書及未簽署者可至金融網－同意書簽署專區申請。   * 相關回報部分：被動回報[GetTSSmartStrategyReport](#_GetTSSmartStrategyReport)，主動回報請參考回報文件。 * ［智慧單用戶請注意］：智慧單為獨立運作機制，若盤中已自行出清或回補，請務必於盤中自行取消已設定的智慧單，避免條件觸發後重覆成交。 | |

|  |
| --- |
| Python範例程式碼\_TSStrategyOrder.py類別CBForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # 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)          # textBoxbstrStockNoCB          tk.Label(self, text = "委託股票代號").grid(row=0, column=1)          #輸入框          self.textBoxbstrStockNoCB = tk.Entry(self)          self.textBoxbstrStockNoCB.grid(row=0, column=2)          global textBoxbstrStockNoCB          textBoxbstrStockNoCB = self.textBoxbstrStockNoCB          # comboBoxnOrderTypeCB          tk.Label(self, text = "委託交易別(0:現股, 3:融資, 4:融券, 8:無券普賣)").grid(row=1, column=1)              #輸入框          self.comboBoxnOrderTypeCB = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderTypeCB['values'] = Config.comboBoxnOrderTypeCB          self.comboBoxnOrderTypeCB.grid(row=1, column=2)          global comboBoxnOrderTypeCB          comboBoxnOrderTypeCB = self.comboBoxnOrderTypeCB          # comboBoxnOrderPriceCondCB          tk.Label(self, text = "委託時效(0:ROD,3:IOC,4:FOK)").grid(row=1, column=3)              #輸入框          self.comboBoxnOrderPriceCondCB = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderPriceCondCB['values'] = Config.comboBoxnOrderPriceCondCB          self.comboBoxnOrderPriceCondCB.grid(row=1, column=4)          global comboBoxnOrderPriceCondCB          comboBoxnOrderPriceCondCB = self.comboBoxnOrderPriceCondCB          # comboBoxnBuySellCB          tk.Label(self, text = "0買;1賣").grid(row=1, column=5)              #輸入框          self.comboBoxnBuySellCB = ttk.Combobox(self, state='readonly')          self.comboBoxnBuySellCB['values'] = Config.comboBoxnBuySellCB          self.comboBoxnBuySellCB.grid(row=1, column=6)          global comboBoxnBuySellCB          comboBoxnBuySellCB = self.comboBoxnBuySellCB          # comboBoxnInnerOrderIsMITCB          tk.Label(self, text = "觸發條件是否為成交價(0:否, 1:是)").grid(row=2, column=1)              #輸入框          self.comboBoxnInnerOrderIsMITCB = ttk.Combobox(self, state='readonly')          self.comboBoxnInnerOrderIsMITCB['values'] = Config.comboBoxnInnerOrderIsMITCB          self.comboBoxnInnerOrderIsMITCB.grid(row=2, column=2)          global comboBoxnInnerOrderIsMITCB          comboBoxnInnerOrderIsMITCB = self.comboBoxnInnerOrderIsMITCB          # comboBoxnMITDirCB          tk.Label(self, text = "成交價觸價方向(0:None, 1:GTE大於等於, 2:LTE小於等於)").grid(row=2, column=3)              #輸入框          self.comboBoxnMITDirCB = ttk.Combobox(self, state='readonly')          self.comboBoxnMITDirCB['values'] = Config.comboBoxnMITDirCB          self.comboBoxnMITDirCB.grid(row=2, column=4)          global comboBoxnMITDirCB          comboBoxnMITDirCB = self.comboBoxnMITDirCB          # textBoxbstrMITTriggerPriceCB          tk.Label(self, text = "成交價").grid(row=2, column=5)          #輸入框          self.textBoxbstrMITTriggerPriceCB = tk.Entry(self)          self.textBoxbstrMITTriggerPriceCB.grid(row=2, column=6)          global textBoxbstrMITTriggerPriceCB          textBoxbstrMITTriggerPriceCB = self.textBoxbstrMITTriggerPriceCB          # comboBoxnTakeProfitFlagCB          tk.Label(self, text = "觸發條件是否為委買價 0:否；1:是").grid(row=3, column=1)              #輸入框          self.comboBoxnTakeProfitFlagCB = ttk.Combobox(self, state='readonly')          self.comboBoxnTakeProfitFlagCB['values'] = Config.comboBoxnTakeProfitFlagCB          self.comboBoxnTakeProfitFlagCB.grid(row=3, column=2)          global comboBoxnTakeProfitFlagCB          comboBoxnTakeProfitFlagCB = self.comboBoxnTakeProfitFlagCB          # comboBoxnTPDir          tk.Label(self, text = "委買價觸價方向 (0:None, 1:GTE大於等於, 2:LTE小於等於)").grid(row=3, column=3)              #輸入框          self.comboBoxnTPDir = ttk.Combobox(self, state='readonly')          self.comboBoxnTPDir['values'] = Config.comboBoxnTPDir          self.comboBoxnTPDir.grid(row=3, column=4)          global comboBoxnTPDir          comboBoxnTPDir = self.comboBoxnTPDir          # textBoxbstrTPOrderPriceCB          tk.Label(self, text = "委買價").grid(row=3, column=5)          #輸入框          self.textBoxbstrTPOrderPriceCB = tk.Entry(self)          self.textBoxbstrTPOrderPriceCB.grid(row=3, column=6)          global textBoxbstrTPOrderPriceCB          textBoxbstrTPOrderPriceCB = self.textBoxbstrTPOrderPriceCB          # comboBoxnUpDownFlag          tk.Label(self, text = "是否為漲跌幅 (0:否, 1:是)").grid(row=4, column=1)              #輸入框          self.comboBoxnUpDownFlag = ttk.Combobox(self, state='readonly')          self.comboBoxnUpDownFlag['values'] = Config.comboBoxnUpDownFlag          self.comboBoxnUpDownFlag.grid(row=4, column=2)          global comboBoxnUpDownFlag          comboBoxnUpDownFlag = self.comboBoxnUpDownFlag          # comboBoxnUpDownDir          tk.Label(self, text = "漲跌幅觸價方向(0:None, 1:GTE大於等於, 2:LTE小於等於)").grid(row=4, column=3)              #輸入框          self.comboBoxnUpDownDir = ttk.Combobox(self, state='readonly')          self.comboBoxnUpDownDir['values'] = Config.comboBoxnUpDownDir          self.comboBoxnUpDownDir.grid(row=4, column=4)          global comboBoxnUpDownDir          comboBoxnUpDownDir = self.comboBoxnUpDownDir          # textBoxbstrSLPercentCB          tk.Label(self, text = "漲跌幅數").grid(row=4, column=5)          #輸入框          self.textBoxbstrSLPercentCB = tk.Entry(self)          self.textBoxbstrSLPercentCB.grid(row=4, column=6)          global textBoxbstrSLPercentCB          textBoxbstrSLPercentCB = self.textBoxbstrSLPercentCB          # comboBoxnPreQtyFlagCB          tk.Label(self, text = "是否執行單量 (0:否, 1:是)").grid(row=5, column=1)              #輸入框          self.comboBoxnPreQtyFlagCB = ttk.Combobox(self, state='readonly')          self.comboBoxnPreQtyFlagCB['values'] = Config.comboBoxnPreQtyFlagCB          self.comboBoxnPreQtyFlagCB.grid(row=5, column=2)          global comboBoxnPreQtyFlagCB          comboBoxnPreQtyFlagCB = self.comboBoxnPreQtyFlagCB          # comboBoxnPreQtyDirCB          tk.Label(self, text = "單量觸價方向(0:None, 1:GTE大於等於, 2:LTE小於等於)").grid(row=5, column=3)              #輸入框          self.comboBoxnPreQtyDirCB = ttk.Combobox(self, state='readonly')          self.comboBoxnPreQtyDirCB['values'] = Config.comboBoxnPreQtyDirCB          self.comboBoxnPreQtyDirCB.grid(row=5, column=4)          global comboBoxnPreQtyDirCB          comboBoxnPreQtyDirCB = self.comboBoxnPreQtyDirCB          # textBoxbstrPreQtyCB          tk.Label(self, text = "單量").grid(row=5, column=5)          #輸入框          self.textBoxbstrPreQtyCB = tk.Entry(self)          self.textBoxbstrPreQtyCB.grid(row=5, column=6)          global textBoxbstrPreQtyCB          textBoxbstrPreQtyCB = self.textBoxbstrPreQtyCB          # comboBoxnTickFlagCB          tk.Label(self, text = "是否為tick觸發(0:否, 1:是)").grid(row=6, column=1)              #輸入框          self.comboBoxnTickFlagCB = ttk.Combobox(self, state='readonly')          self.comboBoxnTickFlagCB['values'] = Config.comboBoxnTickFlagCB          self.comboBoxnTickFlagCB.grid(row=6, column=2)          global comboBoxnTickFlagCB          comboBoxnTickFlagCB = self.comboBoxnTickFlagCB          # comboBoxnTickDir          tk.Label(self, text = "tick觸發方向 (0:None, 1:GTE大於等於, 2:LTE小於等於)").grid(row=6, column=3)              #輸入框          self.comboBoxnTickDir = ttk.Combobox(self, state='readonly')          self.comboBoxnTickDir['values'] = Config.comboBoxnTickDir          self.comboBoxnTickDir.grid(row=6, column=4)          global comboBoxnTickDir          comboBoxnTickDir = self.comboBoxnTickDir          # textBoxbstrTickCB          tk.Label(self, text = "tick數").grid(row=6, column=5)          #輸入框          self.textBoxbstrTickCB = tk.Entry(self)          self.textBoxbstrTickCB.grid(row=6, column=6)          global textBoxbstrTickCB          textBoxbstrTickCB = self.textBoxbstrTickCB          # comboBoxnStopLossFlagCB          tk.Label(self, text = "是否為委賣價(0:否, 1:是)").grid(row=7, column=1)              #輸入框          self.comboBoxnStopLossFlagCB = ttk.Combobox(self, state='readonly')          self.comboBoxnStopLossFlagCB['values'] = Config.comboBoxnStopLossFlagCB          self.comboBoxnStopLossFlagCB.grid(row=7, column=2)          global comboBoxnStopLossFlagCB          comboBoxnStopLossFlagCB = self.comboBoxnStopLossFlagCB          # comboBoxnSLDirCB          tk.Label(self, text = "委賣價觸價方向(0:None, 1:GTE大於等於, 2:LTE小於等於").grid(row=7, column=3)              #輸入框          self.comboBoxnSLDirCB = ttk.Combobox(self, state='readonly')          self.comboBoxnSLDirCB['values'] = Config.comboBoxnSLDirCB          self.comboBoxnSLDirCB.grid(row=7, column=4)          global comboBoxnSLDirCB          comboBoxnSLDirCB = self.comboBoxnSLDirCB          # textBoxbstrSLOrderPriceCB          tk.Label(self, text = "委賣價").grid(row=7, column=5)          #輸入框          self.textBoxbstrSLOrderPriceCB = tk.Entry(self)          self.textBoxbstrSLOrderPriceCB.grid(row=7, column=6)          global textBoxbstrSLOrderPriceCB          textBoxbstrSLOrderPriceCB = self.textBoxbstrSLOrderPriceCB          # comboBoxnSumQtyFlagCB          tk.Label(self, text = "觸發條件是否為總量/(0:否, 1:是)").grid(row=8, column=1)              #輸入框          self.comboBoxnSumQtyFlagCB = ttk.Combobox(self, state='readonly')          self.comboBoxnSumQtyFlagCB['values'] = Config.comboBoxnSumQtyFlagCB          self.comboBoxnSumQtyFlagCB.grid(row=8, column=2)          global comboBoxnSumQtyFlagCB          comboBoxnSumQtyFlagCB = self.comboBoxnSumQtyFlagCB          # comboBoxnSumQtyDir          tk.Label(self, text = "總量觸價方向(0:None, 1:GTE大於等於, 2:LTE小於等於)").grid(row=8, column=3)              #輸入框          self.comboBoxnSumQtyDir = ttk.Combobox(self, state='readonly')          self.comboBoxnSumQtyDir['values'] = Config.comboBoxnSumQtyDir          self.comboBoxnSumQtyDir.grid(row=8, column=4)          global comboBoxnSumQtyDir          comboBoxnSumQtyDir = self.comboBoxnSumQtyDir          # textBoxbstrSumQtyCB          tk.Label(self, text = "總量").grid(row=8, column=5)          #輸入框          self.textBoxbstrSumQtyCB = tk.Entry(self)          self.textBoxbstrSumQtyCB.grid(row=8, column=6)          global textBoxbstrSumQtyCB          textBoxbstrSumQtyCB = self.textBoxbstrSumQtyCB          # comboBoxnClearAllFlagCB          tk.Label(self, text = "是否為自訂啟動時間(0:立即,1:自訂)").grid(row=9, column=1)              #輸入框          self.comboBoxnClearAllFlagCB = ttk.Combobox(self, state='readonly')          self.comboBoxnClearAllFlagCB['values'] = Config.comboBoxnClearAllFlagCB          self.comboBoxnClearAllFlagCB.grid(row=9, column=2)          global comboBoxnClearAllFlagCB          comboBoxnClearAllFlagCB = self.comboBoxnClearAllFlagCB          # textBoxbstrClearCancelTimeCB          tk.Label(self, text = "自訂啟動時間(14天內) 格式：hhmmss (時時分分秒秒)").grid(row=9, column=3)          #輸入框          self.textBoxbstrClearCancelTimeCB = tk.Entry(self)          self.textBoxbstrClearCancelTimeCB.grid(row=9, column=4)          global textBoxbstrClearCancelTimeCB          textBoxbstrClearCancelTimeCB = self.textBoxbstrClearCancelTimeCB          # comboBoxnFinalClearFlagCB          tk.Label(self, text = "條件是否為全部成立 (0:任一成立, 1:全部成立)").grid(row=9, column=5)              #輸入框          self.comboBoxnFinalClearFlagCB = ttk.Combobox(self, state='readonly')          self.comboBoxnFinalClearFlagCB['values'] = Config.comboBoxnFinalClearFlagCB          self.comboBoxnFinalClearFlagCB.grid(row=9, column=6)          global comboBoxnFinalClearFlagCB          comboBoxnFinalClearFlagCB = self.comboBoxnFinalClearFlagCB          # textBoxnQtyCB          tk.Label(self, text = "委託張數").grid(row=10, column=1)          #輸入框          self.textBoxnQtyCB = tk.Entry(self)          self.textBoxnQtyCB.grid(row=10, column=2)          global textBoxnQtyCB          textBoxnQtyCB = self.textBoxnQtyCB          # comboBoxnOrderPriceTypeCB          tk.Label(self, text = "委託價類別(1:市價; 2:限價)").grid(row=11, column=1)              #輸入框          self.comboBoxnOrderPriceTypeCB = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderPriceTypeCB['values'] = Config.comboBoxnOrderPriceTypeCB          self.comboBoxnOrderPriceTypeCB.grid(row=11, column=2)          global comboBoxnOrderPriceTypeCB          comboBoxnOrderPriceTypeCB = self.comboBoxnOrderPriceTypeCB          # textBoxbstrOrderPriceCB          tk.Label(self, text = "委託價").grid(row=11, column=3)          #輸入框          self.textBoxbstrOrderPriceCB = tk.Entry(self)          self.textBoxbstrOrderPriceCB.grid(row=11, column=4)          global textBoxbstrOrderPriceCB          textBoxbstrOrderPriceCB = self.textBoxbstrOrderPriceCB          # buttonSendStockStrategyCB          self.buttonSendStockStrategyCB = tk.Button(self)          self.buttonSendStockStrategyCB["text"] = "CB送出"          self.buttonSendStockStrategyCB["command"] = self.buttonSendStockStrategyCB\_Click          self.buttonSendStockStrategyCB.grid(row=12, column=1) |

|  |
| --- |
| Config.py |
| # CBForm  comboBoxnOrderTypeCB =(      "0:現", "3:(自)融資", "4:(自)融券" , "8:無券賣出"  )  comboBoxnOrderPriceCondCB =(      "0:ROD", "3:IOC", "4:FOK"  )  comboBoxnBuySellCB =(      "0:現股買", "1:無券賣出"  )  comboBoxnInnerOrderIsMITCB =(      "0:N", "1:Y"  )  comboBoxnMITDirCB =(      "0:None", "1:GTE大於等於", "2:LTE小於等於"  )  comboBoxnTakeProfitFlagCB =(      "0:否", "1:是"  )  comboBoxnTPDir =(      "0:None", "1:GTE大於等於", "2:LTE小於等於"  )  comboBoxnUpDownFlag =(      "0:否", "1:是"  )  comboBoxnUpDownDir =(      "0:None", "1:GTE大於等於", "2:LTE小於等於"  )  comboBoxnPreQtyFlagCB =(      "0:否", "1:是"  )  comboBoxnPreQtyDirCB =(      "0:None", "1:GTE大於等於", "2:LTE小於等於"  )  comboBoxnTickFlagCB =(      "0:否", "1:是"  )  comboBoxnTickDir =(      "0:None", "1:GTE大於等於", "2:LTE小於等於"  )  comboBoxnStopLossFlagCB =(      "0:否", "1:是"  )  comboBoxnSLDirCB =(      "0:None", "1:GTE大於等於", "2:LTE小於等於"  )  comboBoxnSumQtyFlagCB =(      "0:否", "1:是"  )  comboBoxnSumQtyDir =(      "0:None", "1:GTE大於等於", "2:LTE小於等於"  )  comboBoxnClearAllFlagCB =(      "0:否", "1:是"  )  comboBoxnFinalClearFlagCB =(      "0:否", "1:是"  )  comboBoxnOrderPriceTypeCB =(      "1:市價", "2:限價"  ) |
| 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 buttonSendStockStrategyCB\_Click(self):            pOrder = sk.STOCKSTRATEGYORDER()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNoCB.get()          pOrder.nQty = int(textBoxnQtyCB.get())          pOrder.bstrOrderPrice = textBoxbstrOrderPriceCB.get()          pOrder.bstrMITTriggerPrice = textBoxbstrMITTriggerPriceCB.get()          pOrder.bstrTPOrderPrice = textBoxbstrTPOrderPriceCB.get()          pOrder.bstrSLOrderPrice = textBoxbstrSLOrderPriceCB.get()          pOrder.bstrTick = textBoxbstrTickCB.get()          pOrder.bstrSLPercent = textBoxbstrSLPercentCB.get()          pOrder.bstrPreQty = textBoxbstrPreQtyCB.get()          pOrder.bstrSumQty = textBoxbstrSumQtyCB.get()          pOrder.bstrClearCancelTime = textBoxbstrClearCancelTimeCB.get()          if (comboBoxnOrderTypeCB.get() == "0:現股"):              pOrder.nOrderType = 0          elif (comboBoxnOrderTypeCB.get() == "3:融資"):              pOrder.nOrderType = 3          elif (comboBoxnOrderTypeCB.get() == "4:融券"):              pOrder.nOrderType = 4          elif (comboBoxnOrderTypeCB.get() == "8:無券普賣"):              pOrder.nOrderType = 8          if (comboBoxnOrderPriceCondCB.get() == "0:ROD"):              pOrder.nOrderPriceCond = 0          elif (comboBoxnOrderPriceCondCB.get() == "3:IOC"):              pOrder.nOrderPriceCond = 3          elif (comboBoxnOrderPriceCondCB.get() == "4:FOK"):              pOrder.nOrderPriceCond = 4          if (comboBoxnBuySellCB.get() == "0:買"):              pOrder.nBuySell = 0          elif (comboBoxnBuySellCB.get() == "1:賣"):              pOrder.nBuySell = 1          if (comboBoxnOrderPriceTypeCB.get() == "1:市價"):              pOrder.nOrderPriceType = 1          elif (comboBoxnOrderPriceTypeCB.get() == "2:限價"):              pOrder.nOrderPriceType = 2          if (comboBoxnInnerOrderIsMITCB.get() == "0:否"):              pOrder.nInnerOrderIsMIT = 0          elif (comboBoxnInnerOrderIsMITCB.get() == "1:是"):              pOrder.nInnerOrderIsMIT = 1          if (comboBoxnTPDir.get() == "0:None"):              pOrder.nTPDir = 0          elif (comboBoxnTPDir.get() == "1:GTE大於等於"):              pOrder.nTPDir = 1          elif (comboBoxnTPDir.get() == "2:LTE小於等於"):              pOrder.nTPDir = 2          if (comboBoxnStopLossFlagCB.get() == "0:否"):              pOrder.nStopLossFlag = 0          elif (comboBoxnStopLossFlagCB.get() == "1:是"):              pOrder.nStopLossFlag = 1          if (comboBoxnSLDirCB.get() == "0:None"):              pOrder.nSLDir = 0          elif (comboBoxnSLDirCB.get() == "1:GTE大於等於"):              pOrder.nSLDir = 1          elif (comboBoxnSLDirCB.get() == "2:LTE小於等於"):              pOrder.nSLDir = 2          if (comboBoxnTickFlagCB.get() == "0:否"):              pOrder.nTickFlag = 0          elif (comboBoxnTickFlagCB.get() == "1:是"):              pOrder.nTickFlag = 1          if (comboBoxnTickDir.get() == "0:None"):              pOrder.nTickDir = 0          elif (comboBoxnTickDir.get() == "1:GTE大於等於"):              pOrder.nTickDir = 1          elif (comboBoxnTickDir.get() == "2:LTE小於等於"):              pOrder.nTickDir = 2          if (comboBoxnUpDownFlag.get() == "0:否"):              pOrder.nUpDownFlag = 0          elif (comboBoxnUpDownFlag.get() == "1:是"):              pOrder.nUpDownFlag = 1          if (comboBoxnUpDownDir.get() == "0:None"):              pOrder.nUpDownDir = 0          elif (comboBoxnUpDownDir.get() == "1:GTE大於等於"):              pOrder.nUpDownDir = 1          elif (comboBoxnUpDownDir.get() == "2:LTE小於等於"):              pOrder.nUpDownDir = 2          if (comboBoxnPreQtyFlagCB.get() == "0:否"):              pOrder.nPreQtyFlag = 0          elif (comboBoxnPreQtyFlagCB.get() == "1:是"):              pOrder.nPreQtyFlag = 1          if (comboBoxnPreQtyDirCB.get() == "0:None"):              pOrder.nPreQtyDir = 0          elif (comboBoxnPreQtyDirCB.get() == "1:GTE大於等於"):              pOrder.nPreQtyDir = 1          elif (comboBoxnPreQtyDirCB.get() == "2:LTE小於等於"):              pOrder.nPreQtyDir = 2          if (comboBoxnSumQtyFlagCB.get() == "0:否"):              pOrder.nSumQtyFlag = 0          elif (comboBoxnSumQtyFlagCB.get() == "1:是"):              pOrder.nSumQtyFlag = 1          if (comboBoxnSumQtyDir.get() == "0:None"):              pOrder.nSumQtyDir = 0          elif (comboBoxnSumQtyDir.get() == "1:GTE大於等於"):              pOrder.nSumQtyDir = 1          elif (comboBoxnSumQtyDir.get() == "2:LTE小於等於"):              pOrder.nSumQtyDir = 2          if (comboBoxnClearAllFlagCB.get() == "0:否"):              pOrder.nClearAllFlag = 0          elif (comboBoxnClearAllFlagCB.get() == "1:是"):              pOrder.nClearAllFlag = 1          if (comboBoxnFinalClearFlagCB.get() == "0:否"):              pOrder.nFinalClearFlag = 0          elif (comboBoxnFinalClearFlagCB.get() == "1:是"):              pOrder.nFinalClearFlag = 1          if (comboBoxnMITDirCB.get() == "0:None"):              pOrder.nMITDir = 0          elif (comboBoxnMITDirCB.get() == "1:GTE大於等於"):              pOrder.nMITDir = 1          elif (comboBoxnMITDirCB.get() == "2:LTE小於等於"):              pOrder.nMITDir = 2          if (comboBoxnTakeProfitFlagCB.get() == "0:否"):              pOrder.nTakeProfitFlag = 0          elif (comboBoxnTakeProfitFlagCB.get() == "1:是"):              pOrder.nTakeProfitFlag = 1          # 送出證券智慧單自組單委託。          bstrMessage,nCode= m\_pSKOrder.SendStockStrategyCB(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendStockStrategyCB】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### 取消證券智慧單委託

###### CancelTSStrategyOrderV1

|  |  |  |
| --- | --- | --- |
| 取消證券智慧單委託新版本。刪單欄位請參考GetTSSmartStrategyReport回傳的內容。 | | |
| **宣告** | Long CancelTSStrategyOrderV1 ([in] BSTR bstrLogInID, [in] struct CANCELSTRATEGYORDER\* pCancelOrder, [**out**] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| pCancelOrder | 智慧單刪單物件([CANCELSTRATEGYORDER](#_CANCELSTRATEGYORDER(智慧單刪單物件))) |
| bstrMessage | **非同步刪單：**  參照OnAsyncOrder。  如果回傳值為 0表示刪單成功，訊息內容則為刪單之Thread ID。  回傳值非0表示刪單失敗，訊息內容為失敗原因。 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。 | |
| **備註** | 使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder_1)取得。  需簽署證券智慧單風險預告書及未簽署者可至金融網－同意書簽署專區申請。  刪單後，請透過智慧單被動回報確認狀態。 | |

###### CancelTSStrategyOrder

|  |  |  |
| --- | --- | --- |
| 取消證券智慧單委託。欄位請參考GetTSSmartStrategyReport 回傳的內容。  注意，當已經觸發的智慧單，將無法取消委託。 | | |
| **宣告** | Long CancelTSStrategyOrder ([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] BSTR bstrAccount, [in] BSTR bstrSmartKey, [in] Long nTradeKind, [**out**] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| bAsyncOrder | 是否為非同步委託。 |
| bstrAccount | 委託帳號 ( BrokerID＋帳號) 。 |
| bstrSmartKey | 智慧單序號或進場單號。 |
| nTradeKind | 智慧單類型  6:MIOC;7:MST; **8:MIT**。 |
| bstrMessage | **同步刪單：**如果回傳值為 0表示刪單成功，訊息內容則為智慧單號刪單訊息。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** | 使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder_1)取得。  需簽署證券智慧單風險預告書及未簽署者可至金融網－同意書簽署專區申請。  刪單後，請透過智慧單被動回報確認狀態。  \*當沖單/出清單/OCO二擇一: 請用[CancelTSStrategyOrderV1](#_CancelTSStrategyOrderV1_1) | |

|  |
| --- |
| Python範例程式碼\_TSStrategyOrder.py類別CancelForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # 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)          # textBoxbstrSmartKey          tk.Label(self, text = "智慧單號").grid(row=0, column=1)          #輸入框          self.textBoxbstrSmartKey = tk.Entry(self)          self.textBoxbstrSmartKey.grid(row=0, column=2)          global textBoxbstrSmartKey          textBoxbstrSmartKey = self.textBoxbstrSmartKey          # comboBoxnMarket          tk.Label(self, text = "市場別(AB單需選欲刪單之A商品市場)").grid(row=1, column=1)              #輸入框          self.comboBoxnMarket = ttk.Combobox(self, state='readonly')          self.comboBoxnMarket['values'] = Config.comboBoxnMarket          self.comboBoxnMarket.grid(row=1, column=2)          global comboBoxnMarket          comboBoxnMarket = self.comboBoxnMarket          # textBoxbstrSeqNo          tk.Label(self, text = "委託序號 (預約單可忽略)").grid(row=2, column=1)          #輸入框          self.textBoxbstrSeqNo = tk.Entry(self)          self.textBoxbstrSeqNo.grid(row=2, column=2)          global textBoxbstrSeqNo          textBoxbstrSeqNo = self.textBoxbstrSeqNo          # textBoxbstrOrderNo          tk.Label(self, text = "委託書號（若觸發，需給書號）").grid(row=3, column=1)          #輸入框          self.textBoxbstrOrderNo = tk.Entry(self)          self.textBoxbstrOrderNo.grid(row=3, column=2)          global textBoxbstrOrderNo          textBoxbstrOrderNo = self.textBoxbstrOrderNo          # textBoxbstrLongActionKey          tk.Label(self, text = "長效單號(非長效單可忽略)").grid(row=4, column=1)          #輸入框          self.textBoxbstrLongActionKey = tk.Entry(self)          self.textBoxbstrLongActionKey.grid(row=4, column=2)          global textBoxbstrLongActionKey          textBoxbstrLongActionKey = self.textBoxbstrLongActionKey          # textBoxbstrParentSmartKey          tk.Label(self, text = "(當沖)智慧母單號").grid(row=5, column=1)          #輸入框          self.textBoxbstrParentSmartKey = tk.Entry(self)          self.textBoxbstrParentSmartKey.grid(row=5, column=2)          global textBoxbstrParentSmartKey          textBoxbstrParentSmartKey = self.textBoxbstrParentSmartKey          # comboBoxnDeletType          tk.Label(self, text = "(當沖)刪單類型").grid(row=6, column=1)              #輸入框          self.comboBoxnDeletType = ttk.Combobox(self, state='readonly')          self.comboBoxnDeletType['values'] = Config.comboBoxnDeletType          self.comboBoxnDeletType.grid(row=6, column=2)          global comboBoxnDeletType          comboBoxnDeletType = self.comboBoxnDeletType          # textBoxbstrSmartKeyOut          tk.Label(self, text = "(當沖)出場單號").grid(row=7, column=1)          #輸入框          self.textBoxbstrSmartKeyOut = tk.Entry(self)          self.textBoxbstrSmartKeyOut.grid(row=7, column=2)          global textBoxbstrSmartKeyOut          textBoxbstrSmartKeyOut = self.textBoxbstrSmartKeyOut          # comboBoxnTradeKind          tk.Label(self, text = "舊版:6:MIOC;7:MST; 8:MIT").grid(row=8, column=1)              #輸入框          self.comboBoxnTradeKind = ttk.Combobox(self, state='readonly')          self.comboBoxnTradeKind['values'] = Config.comboBoxnTradeKind          self.comboBoxnTradeKind.grid(row=8, column=2)          global comboBoxnTradeKind          comboBoxnTradeKind = self.comboBoxnTradeKind          # buttonCancelTSStrategyOrder          self.buttonCancelTSStrategyOrder = tk.Button(self)          self.buttonCancelTSStrategyOrder["text"] = "(6:MIOC;7:MST; 8:MIT)舊版刪單送出"          self.buttonCancelTSStrategyOrder["command"] = self.buttonCancelTSStrategyOrder\_Click          self.buttonCancelTSStrategyOrder.grid(row=8, column=3)          # comboBoxnTradeKindV1          tk.Label(self, text = "V1版本:3:OCO; 8:MIT;9:MST;10:AB;11:當沖;17:出清;27：CB").grid(row=9, column=1)              #輸入框          self.comboBoxnTradeKindV1 = ttk.Combobox(self, state='readonly')          self.comboBoxnTradeKindV1['values'] = Config.comboBoxnTradeKindV1          self.comboBoxnTradeKindV1.grid(row=9, column=2)          global comboBoxnTradeKindV1          comboBoxnTradeKindV1 = self.comboBoxnTradeKindV1          # buttonCancelTSStrategyOrderV1          self.buttonCancelTSStrategyOrderV1 = tk.Button(self)          self.buttonCancelTSStrategyOrderV1["text"] = "新版刪單送出"          self.buttonCancelTSStrategyOrderV1["command"] = self.buttonCancelTSStrategyOrderV1\_Click          self.buttonCancelTSStrategyOrderV1.grid(row=9, column=3) |

|  |
| --- |
| Config.py |
| # CancelForm  comboBoxnMarket =(      "1:國內證", "2:國內期", "3:國外證", "4:國外期"  )  comboBoxnDeletType =(      "1:全部", "2:進場單", "3:出場單"  )  comboBoxnTradeKind =(      "6:MIOC", "7:MST", "8:MIT"  )  comboBoxnTradeKindV1 =(      "3:OCO", "8:MIT", "9:MST","10:AB", "11:當沖", "17:出清","27：CB"  ) |
| 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 buttonCancelTSStrategyOrder\_Click(self):            if (comboBoxnTradeKind.get() == "6:MIOC"):              nTradeKind = 6          elif (comboBoxnTradeKind.get() == "7:MST"):              nTradeKind = 7          elif (comboBoxnTradeKind.get() == "8:MIT"):              nTradeKind = 8          # 取消證券智慧單委託。欄位請參考GetTSStrategyOrder 回傳的內容。注意，當已經觸發的智慧單，將無法取消委託。          bstrMessage,nCode= m\_pSKOrder.CancelTSStrategyOrder(comboBoxUserID.get(), bAsyncOrder, comboBoxAccount.get(), textBoxbstrSmartKey.get(), nTradeKind)          msg = "【CancelTSStrategyOrder】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end')        def buttonCancelTSStrategyOrderV1\_Click(self):            pCancelOrder = sk.CANCELSTRATEGYORDER()          pCancelOrder.bstrFullAccount = comboBoxAccount.get()          if (comboBoxnMarket.get() == "1：國內證"):              pCancelOrder.nMarket = 1          elif (comboBoxnMarket.get() == "2：國內期"):              pCancelOrder.nMarket = 2          elif (comboBoxnMarket.get() == "3：國外證"):              pCancelOrder.nMarket = 3          elif (comboBoxnMarket.get() == "4：國外期"):              pCancelOrder.nMarket = 4          pCancelOrder.bstrParentSmartKey = textBoxbstrParentSmartKey.get()          pCancelOrder.bstrSmartKey = textBoxbstrSmartKey.get()          if (comboBoxnTradeKindV1.get() == "3:OCO"):              pCancelOrder.nTradeKind = 3          elif (comboBoxnTradeKindV1.get() == "8:MIT"):              pCancelOrder.nTradeKind = 8          elif (comboBoxnTradeKindV1.get() == "9:MST"):              pCancelOrder.nTradeKind = 9          elif (comboBoxnTradeKindV1.get() == "10:AB"):              pCancelOrder.nTradeKind = 10          elif (comboBoxnTradeKindV1.get() == "11:當沖"):              pCancelOrder.nTradeKind = 11          elif (comboBoxnTradeKindV1.get() == "17:出清"):              pCancelOrder.nTradeKind = 17          elif (comboBoxnTradeKindV1.get() == "27：CB"):              pCancelOrder.nTradeKind = 27          else:              pCancelOrder.nTradeKind = 0          if (comboBoxnDeletType.get() == "1:全部"):              pCancelOrder.nDeleteType = 1          elif (comboBoxnDeletType.get() == "2:進場單"):              pCancelOrder.nDeleteType = 2          elif (comboBoxnDeletType.get() == "3:出場單"):              pCancelOrder.nDeleteType = 3          pCancelOrder.bstrSeqNo = textBoxbstrSeqNo.get()          pCancelOrder.bstrOrderNo = textBoxbstrOrderNo.get()          pCancelOrder.bstrSmartKeyOut = textBoxbstrSmartKeyOut.get()          # 取消證券智慧單委託。欄位請參考GetTSStrategyOrder 回傳的內容。注意，當已經觸發的智慧單，將無法取消委託。          bstrMessage,nCode= m\_pSKOrder.CancelTSStrategyOrderV1(comboBoxUserID.get(), pCancelOrder)          msg = "【CancelTSStrategyOrderV1】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### CancelStrategyList

|  |  |  |
| --- | --- | --- |
| 取消多筆智慧單委託。刪單欄位請參考智慧單被動回報回傳的內容。 | | |
| **宣告** | Long CancelStrategyList ([in] BSTR bstrLogInID, [in] struct CANCELSTRATEGYORDER\* pCancelOrder, [**out**] BSTR\* bstrMessage); | |
| **參數** | bstrLogInID | 登入ID。 |
| pCancelOrder | 智慧單刪單物件(參考[CANCELSTRATEGYORDER](#_智慧單多筆刪單)) |
| bstrMessage | **非同步刪單：**  參照OnAsyncOrder。  如果回傳值為 0表示刪單成功，訊息內容則為刪單之Thread ID。  回傳值非0表示刪單失敗，訊息內容為失敗原因。 |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。 | |
| **備註** | 使用非同步委託，委託結果請由[OnAsyncOrder](#_OnAsyncOrder_1)取得。  需簽署證券智慧單風險預告書及未簽署者可至金融網－同意書簽署專區申請。  刪單後，請透過智慧單被動回報確認狀態。 | |

|  |
| --- |
| Python範例程式碼\_TSStrategyOrder.py類別CancelForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖    虛線處請參考取消智慧單委託 |
| # buttonCancelStrategyList          self.buttonCancelStrategyList = tk.Button(self)          self.buttonCancelStrategyList["text"] = "多筆刪單送出(以逗號分隔，EX：682020,682021)"          self.buttonCancelStrategyList["command"] = self.buttonCancelStrategyList\_Click          self.buttonCancelStrategyList.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 buttonCancelStrategyList\_Click(self):            pCancelOrder = sk.CANCELSTRATEGYORDER()          pCancelOrder.bstrLogInID = comboBoxUserID.get()          pCancelOrder.bstrFullAccount = comboBoxAccount.get()          if (comboBoxnMarket.get() == "1：國內證"):              pCancelOrder.nMarket = 1          elif (comboBoxnMarket.get() == "2：國內期"):              pCancelOrder.nMarket = 2          elif (comboBoxnMarket.get() == "3：國外證"):              pCancelOrder.nMarket = 3          elif (comboBoxnMarket.get() == "4：國外期"):              pCancelOrder.nMarket = 4          pCancelOrder.bstrSmartKey = textBoxbstrSmartKey.get()          # 取消多筆智慧單委託。刪單欄位請參考智慧單被動回報回傳的內容。          bstrMessage,nCode= m\_pSKOrder.CancelStrategyList(comboBoxUserID.get(), pCancelOrder)          msg = "【CancelStrategyList】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### GetTSSmartStrategyReport

|  |  |  |
| --- | --- | --- |
| 查詢證券智慧單。 | | |
| **宣告** | Long GetTSSmartStrategyReport ([in] BSTR bstrLogInID, [in] BSTR bstrAccount, [in] BSTR bstrMarketType, [in] LONG nReportStatus, [in] BSTR bstrKind, [in] BSTR bstrDate); | |
| **參數** | bstrLogInID | 登入ID。 |
| bstrAccount | 委託帳號 (帳號) 。 |
| bstrMarketType | TS :證券市場 |
| nReportStatus | 0:全部的委託單。 |
| bstrKind | 智慧單類型  DayTrade:當沖  ClearOut:出清  **MIT：觸價單、MIT長效單**  **OCO：二擇一**  **MIOC：多次IOC**  **MST：移動停損、MST長效單**  **AB：看A下B單**  **CB：自組單** |
| bstrDate | 查詢日期(ex:20201001) |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。 | |
| **備註** | 結果由 [OnTSSmartStrategyReport](#_OnRealBalanceReport) 事件回傳。  需簽署證券智慧單風險預告書及未簽署者可至金融網－同意書簽署專區申請。 | |

|  |
| --- |
| Python範例程式碼\_TSStrategyOrder.py類別GetForm(tk.Frame)裡面 |
| UI程式碼(放置於def createWidgets(self): 的下一層)/UI示意圖 |
| # comboBoxbstrKind          tk.Label(self, text = "智慧單類型").grid(row=0, column=1)              #輸入框          self.comboBoxbstrKind = ttk.Combobox(self, state='readonly')          self.comboBoxbstrKind['values'] = Config.comboBoxbstrKind          self.comboBoxbstrKind.grid(row=0, column=2)          global comboBoxbstrKind          comboBoxbstrKind = self.comboBoxbstrKind          # textBoxbstrDate          tk.Label(self, text = "查詢日期(ex:20220601)").grid(row=1, column=1)          #輸入框          self.textBoxbstrDate = tk.Entry(self)          self.textBoxbstrDate.grid(row=1, column=2)          global textBoxbstrDate          textBoxbstrDate = self.textBoxbstrDate          # buttonGetTSSmartStrategyReport          self.buttonGetTSSmartStrategyReport = tk.Button(self)          self.buttonGetTSSmartStrategyReport["text"] = "查詢送出"          self.buttonGetTSSmartStrategyReport["command"] = self.buttonGetTSSmartStrategyReport\_Click          self.buttonGetTSSmartStrategyReport.grid(row=1, column=3) |

|  |
| --- |
| Config.py |
| #GetForm  comboBoxbstrKind =(      "DayTrade:當沖", "ClearOut:出清", "MIT：觸價單、MIT長效單","OCO：二擇一", "MIOC：多次IOC", "MST：移動停損、MST長效單","AB：看A下B單", "CB：自組單"  ) |
| 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 buttonGetTSSmartStrategyReport\_Click(self):          if (comboBoxbstrKind.get() == "DayTrade:當沖"):              bstrKind = "DayTrade"          elif (comboBoxbstrKind.get() == "ClearOut:出清"):              bstrKind = "ClearOut"          elif (comboBoxbstrKind.get() == "MIT：觸價單、MIT長效單"):              bstrKind = "MIT"          elif (comboBoxbstrKind.get() == "OCO：二擇一"):              bstrKind = "OCO"          elif (comboBoxbstrKind.get() == "MIOC：多次IOC"):              bstrKind = "MIOC"          elif (comboBoxbstrKind.get() == "MST：移動停損、MST長效單"):              bstrKind = "MST"          elif (comboBoxbstrKind.get() == "AB：看A下B單"):              bstrKind = "AB"          elif (comboBoxbstrKind.get() == "CB：自組單"):              bstrKind = "CB"          # 查詢證券智慧單          nCode= m\_pSKOrder.GetTSSmartStrategyReport(comboBoxUserID.get(), comboBoxAccount.get(), "TS", 0, bstrKind, textBoxbstrDate.get())          msg = "【GetTSSmartStrategyReport】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode)          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

## 事件

### OnTSSmartStrategyReport

<新版證券智慧單被動回報>依各種策略單別提供格式內容

#### MIOC

|  |  |  |
| --- | --- | --- |
| 證券智慧單被動查詢結果。透過呼叫[GetTSSmartStrategyReport](#_GetTSSmartStrategyReport)後，資訊由該事件回傳。 | | |
| **宣告** | void OnTSSmartStrategyReport ([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 | | 1 | 智慧單號 |  | | 2 | 市場別 | TS：國內證券 | | 3 | 交易日 | 20200707 | | 4 | 交易所代碼 | TSE上市  OTC上櫃 | | 5 | 分公司代碼 |  | | 6 | 帳號 |  | | 7 | 子帳 |  | | 8 | 策略別 | 29:MIOC | | 9 | 商品代碼 |  | | 10 | 買賣別 | B：買 S：賣 | | 11 | 委託種類別 | 0：現股 1：代資  2：代券  8：無券普賣 | | 12 | 委託別 | 0：一般 | | 13 | 委託價格類別 | 第一委託價格別  0:前日收盤價 (平盤價)  1:漲停價  2:跌停價  7:使用者輸入價 | | 14 | 委託價格別 | 1：市價  2：限價 | | 15 | 委託時效 | 3：立即成交IOC | | 16 | 下單時間 | 2020-07-07 12:32:10.000 | | 17 | 單次委託量 |  | | 18 | 委託總數量 |  | | 19 | 委託價格上限 |  | | 20 | 委託價格下限 |  | | 21 | 智慧單委託狀態-  代碼 | 32：中台收單成功  33：中台收單失敗  34：洗價中  36：洗價失敗  37：洗價觸發  38：觸發下單  39：下單失敗  40：使用者刪單  41：全部成交(多次IOC用) | | 22 | 目前成交數量 |  | | 23 | 訊息更新時間 | 2020-07-07 13:30:41 | | 24 | 訊息 | [智慧單號]  [商品代碼]  訊息:參考市場委託狀態  Ex: [537500][2801]已加入洗價  訊息中逗號皆為全形”，” | |  |  |  | |  |  |  | |
| **備註** | * 回傳【M003,無資料】表示目前查無被動回報 * 資料排序為：先回傳最後一筆委託, 後回傳最早之委託。 * 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。 | |

#### MST

|  |  |  |
| --- | --- | --- |
| 證券智慧單被動查詢結果。透過呼叫[GetTSSmartStrategyReport](#_GetTSSmartStrategyReport)後，資訊由該事件回傳。 | | |
| **宣告** | void OnTSSmartStrategyReport ([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 | | 1 | 智慧單號 |  | | 2 | 市場別 | TS：國內證券 | | 3 | 交易日 | 20200707 | | 4 | 交易所代碼 | TSE上市  OTC上櫃 | | 5 | 分公司代碼 |  | | 6 | 帳號 |  | | 7 | 子帳 |  | | 8 | 策略別 | 9:MST | | 9 | 商品代碼 |  | | 10 | 買賣別 | B：買 S：賣 | | 11 | 委託種類別 | 0：現股 **3：融資**  **4：融券**  8：無券普賣 | | 12 | 委託別 | 0：一般 | | 13 | 委託價格類別 | 第一委託價格別  0:前日收盤價 (平盤價)  1:漲停價  2:跌停價  7:使用者輸入價 | | 14 | 委託價格別 | 1：市價  2：限價 | | 15 | 委託時效 | 0:ROD;3：IOC;4:FOK | | 16 | 下單時間 | 2020-07-07 12:32:10.000 | | 17 | 委託價格 |  | | 18 | 移動點數 |  | | 19 | 委託數量 |  | | 20 | 觸發價 |  | | 21 | 觸發時間 |  | | 22 | 市價 |  | | 23 | 觸價基準 |  | | 24 | 智慧單委託狀態-  代碼 | 32：中台收單成功  33：中台收單失敗  34：洗價中  35：洗價中-觸發價更新(移動停損單)  36：洗價失敗  37：洗價觸發  38：觸發下單  39：下單失敗  40：使用者刪單  41：全部成交(多次IOC用) | | 25 | 市場委託狀熊 | 空白：無委託單  0：預約  2：全部成交  3：全部取消  4：部分成交，剩餘已取消  5：部分成交，剩餘可取消  6：委託失敗  7:委託成功  8:取消失敗  【交易所動態退單為F開頭編碼】  F1：動態退單-全部取消  F2：動態退單-部分成交，剩餘已取消  F3：動態退單-部分委託成功 | | 26 | 訊息更新時間 | 2020-07-07 13:30:41 | | 27 | 訊息含狀態 | [智慧單號]  [商品代碼]  訊息:參考市場委託狀態  Ex: [537500][2801]已加入洗價  訊息中逗號皆為全形”，” | | 28 | 前一個觸發價格 |  | | 29 | 是否自訂啟動價格 | 0:否,由市價啟動價格  1:是,由自訂啟動價格 | | 30 | [自訂]啟動價格 |  | | 31 | [自訂]啟動價格觸發方向 |  | |
| **備註** | * 回傳【M003,無資料】表示目前查無被動回報 * 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。 | |

#### 新版MIT

|  |  |  |
| --- | --- | --- |
| 證券智慧單被動查詢結果。透過呼叫[GetTSSmartStrategyReport](#_GetTSSmartStrategyReport)後，資訊由該事件回傳。 | | |
| **宣告** | void OnTSSmartStrategyReport ([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 | | 1 | 智慧單號 |  | | 2 | 市場別 | TS：國內證券 | | 3 | 交易日 | 20200707 | | 4 | 交易所代碼 | TSE上市  OTC上櫃 | | 5 | 分公司代碼 |  | | 6 | 帳號 |  | | 7 | 子帳 | 無子帳以0000000 | | 8 | 策略別 | 8:MIT | | 9 | 商品代碼 |  | | 10 | 買賣別 | B：買 S：賣 | | 11 | 委託種類別 | 0：現股 **3：融資**  **4：融券**  8：無券普賣 | | 12 | 委託別 | 0：一般 | | 13 | 委託價格類別 | 第一委託價格別  0:前日收盤價 (平盤價)  1:漲停價  2:跌停價  7:使用者輸入價 | | 14 | 委託價格別 | 1：市價  2：限價 | | 15 | 委託時效 | 0:ROD;3：IOC;4:FOK | | 16 | 下單時間 | 2020-07-07 12:32:10.000 | | 17 | 委託價格 |  | | 18 | 委託數量 |  | | 19 | 觸發價 |  | | 20 | 觸發時間 |  | | 21 | 智慧單委託狀態-  代碼 | 32：中台收單成功  33：中台收單失敗  34：洗價中  36：洗價失敗  37：洗價觸發  38：觸發下單  39：下單失敗  40：使用者刪單  41：全部成交(多次IOC用) | | 22 | 市場委託狀熊 | 空白：無委託單  0：預約  2：全部成交  3：全部取消  4：部分成交，剩餘已取消  5：部分成交，剩餘可取消  6：委託失敗  7:委託成功  8:取消失敗  【交易所動態退單為F開頭編碼】  F1：動態退單-全部取消  F2：動態退單-部分成交，剩餘已取消  F3：動態退單-部分委託成功 | | 23 | 訊息更新時間 | 2020-07-07 13:30:41 | | 24 | 訊息含狀態 | [智慧單號]  [商品代碼]  訊息:參考市場委託狀態  Ex: [537500][2801]已加入洗價  訊息中逗號皆為全形”，” | | 25 | 當下市價價格 |  | | 26 | 觸價方向 | 1:大於等於 ; 2: 小於等於 | | 27 | 成交價觸發價格 |  | | 28 | 萬用訊息 | 適用於智慧單狀態代碼為999情況 | | 29 | 預風控註記 | 0:無開啟 ; 1: 開啟預風控 | | 30 | 拆單註記 |  | | 31 | 是否觸發即停止 | 0：否  1：是 | | 32 | 是否為內外盤委託量 |  | | 33 | 內外盤委託量 | 1：大於等於  2：小於等於 | | 34 | 內外盤委託量觸發方向 | 1:條件觸發後，該筆長效單功能即失效 0:不管有無觸發，每日都送一次條件單，直到長效單結束日後才失效 | | 35 | 長效單註記 | 0：非長效單  1：長效單 | | 36 | 長效單序號 |  | | 37 | 長效單結束日期 | 20230829 | | 38 | 長效單類別 | 0：None  1：效期內觸發即失效  2：[停用]長效單結束日失效  3：效期內完全成交即失效 | | 39 | 長效單委託總量 |  | | 40 | 長效單成交總量 |  | |
| **備註** | * 回傳【M003,無資料】表示目前查無被動回報 * 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。 * 自SKCOM V2.13.30(含)版本起，提供MIT新版被動回報 | |

#### 當沖單- 母單

|  |  |  |
| --- | --- | --- |
| 證券智慧單被動查詢結果。透過呼叫[GetTSSmartStrategyReport](#_GetTSSmartStrategyReport)後，資訊由該事件回傳。 | | |
| **宣告** | void OnTSSmartStrategyReport ([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 | | 1 | 智慧單號 |  | | 2 | 市場別 | TS：國內證券 | | 3 | 交易日 | 20200707 | | 4 | 交易所代碼 | TSE上市  OTC上櫃 | | 5 | 分公司代碼 |  | | 6 | 帳號 |  | | 7 | 子帳 |  | | 8 | 策略別 | 11:DayTrade當沖 | | 9 | 商品代碼 |  | | 10 | 買賣別 | B：買 S：賣 | | 11 | 委託種類別 | 0：現股 **3：融資**  **4：融券**  8：無券普賣 | | 12 | 委託別 | 0：一般 | | 13 | 委託價格類別 | 第一委託價格別  0:前日收盤價 (平盤價)  1:漲停價  2:跌停價  7:使用者輸入價 | | 14 | 委託價格別 | 1：市價  2：限價 | | 15 | 委託時效 | 0:ROD;3：IOC;4:FOK | | 16 | 下單時間 | 2020-07-07 12:32:10.000 | | 17 | 委託價格 |  | | 18 | 委託數量 |  | | 19 | 智慧單委託狀態-  代碼 | 32：中台收單成功  33：中台收單失敗  34：洗價中  36：洗價失敗  37：洗價觸發  38：觸發下單  39：下單失敗  40：使用者刪單  41：全部成交(多次IOC用)  999:萬用狀態，請注意「萬用訊息」欄說明 | | 20 | 市場委託狀熊 | 空白：無委託單  0：預約  2：全部成交  3：全部取消  4：部分成交，剩餘已取消  5：部分成交，剩餘可取消  6：委託失敗  7:委託成功  8:取消失敗  【交易所動態退單為F開頭編碼】  F1：動態退單-全部取消  F2：動態退單-部分成交，剩餘已取消  F3：動態退單-部分委託成功 | | 21 | 訊息更新時間 | 2020-07-07 13:30:41 | | 22 | 訊息含狀態 | [智慧單號]  [商品代碼]  訊息:參考市場委託狀態  Ex: [537500][2801]已加入洗價  訊息中逗號皆為全形”，” | | 23 | 委託書號 |  | | 24 | 流水號(13碼) |  | | 25 | 原始流水號(13碼) |  | | 26 | 是否啟用MIT進場 | 0:否; 1:是 | | 27 | MIT進場觸發價 |  | | 28 | MIT進場觸發時間 |  | | 29 | 出清方式 | 17=只選時間出清條件  18=GTE(條件一)  19=LTE(條件二)  20=GTE\_LTE(條件一+條件二) | | 30 | 停利洗價模式 | 1=指定價格  2=指定漲跌幅(%) | | 31 | 停利指定漲跌幅 |  | | 32 | 停利觸發價 |  | | 33 | 停利委託方式 | 1=市價  2=限價 | | 34 | 停利委託價 |  | | 35 | 停利委託時效 | 0=ROD  3=IOC  4=FOK | | 36 | 停損洗價模式 | 1=指定價格  2=指定漲跌幅(%) | | 37 | 停損指定漲跌幅 |  | | 38 | 停損觸發價 |  | | 39 | 停損委託方式 | 1=市價  2=限價 | | 40 | 停損委託價 |  | | 41 | 停損委託時效 | 0=ROD  3=IOC  4=FOK | | 42 | 是否執行時間出清 |  | | 43 | 指定出清時間 |  | | 44 | 時間出清委託方式 | 1=市價  2=限價 | | 45 | 時間出清委託價 |  | | 46 | 時間出清委託時效 | 0=ROD  3=IOC  4=FOK | | 47 | 是否執行盤後定價 |  | | 48 | 盤後定價執行時間 |  | | 49 | 萬用訊息 | 適用於智慧單狀態代碼為999情況 | | 49 | 進場單資訊 | (預約單無此欄), 若設進場MIT才有此欄  細項參閱當沖-進場單格式  **若無此欄將以[in]代表無進場單資訊** | | 50 | 出場單資訊 | (預約單無此欄)  細項參閱當沖-出場單格式  **若無此欄將以[out]代表無出場單資訊** | |

#### 當沖單- 進場單

|  |  |  |
| --- | --- | --- |
| 證券智慧單被動查詢結果。透過呼叫[GetTSSmartStrategyReport](#_GetTSSmartStrategyReport)後，資訊由該事件回傳。 | | |
| **宣告** | void OnTSSmartStrategyReport ([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 | | 1 | 智慧單號 |  | | 2 | 市場別 | TS：國內證券 | | 3 | 交易日 | 20200707 | | 4 | 交易所代碼 | TSE上市  OTC上櫃 | | 5 | 分公司代碼 |  | | 6 | 帳號 |  | | 7 | 子帳 |  | | 8 | 策略別 | 11:DayTrade當沖 | | 9 | 商品代碼 |  | | 10 | 買賣別 | B：買 S：賣 | | 11 | 委託種類別 | 0：現股 **3：融資**  **4：融券**  8：無券普賣 | | 12 | 委託別 | 0：一般 | | 13 | 委託價格類別 | 第一委託價格別  0:前日收盤價 (平盤價)  1:漲停價  2:跌停價  7:使用者輸入價 | | 14 | 委託價格別 | 1：市價  2：限價 | | 15 | 委託時效 | 0:ROD;3：IOC;4:FOK | | 16 | 下單時間 | 2020-07-07 12:32:10.000 | | 17 | 委託價格 |  | | 18 | 委託數量 |  | | 19 | 智慧單委託狀態-  代碼 | 32：中台收單成功  33：中台收單失敗  34：洗價中  36：洗價失敗  37：洗價觸發  38：觸發下單  39：下單失敗  40：使用者刪單  41：全部成交(多次IOC用) | | 20 | 市場委託狀熊 | 空白：無委託單  0：預約  2：全部成交  3：全部取消  4：部分成交，剩餘已取消  5：部分成交，剩餘可取消  6：委託失敗  7:委託成功  8:取消失敗  【交易所動態退單為F開頭編碼】  F1：動態退單-全部取消  F2：動態退單-部分成交，剩餘已取消  F3：動態退單-部分委託成功 | | 21 | 訊息更新時間 | 2020-07-07 13:30:41 | | 22 | 訊息含狀態 | [智慧單號]  [商品代碼]  訊息:參考市場委託狀態  Ex: [537500][2801]已加入洗價  訊息中逗號皆為全形”，” | | 23 | 委託書號 |  | | 24 | 流水號(13碼) |  | | 25 | 原始流水號(13碼) |  | | 26 | 是否啟用MIT進場 | 0:否; 1:是 | | 27 | MIT進場觸發價 |  | | 28 | MIT進場觸發時間 |  | | 29 | 出清方式 | 17=只選時間出清條件  18=GTE(條件一)  19=LTE(條件二)  20=GTE\_LTE(條件一+條件二) | | 30 | 停利洗價模式 | 1=指定價格  2=指定漲跌幅(%) | | 31 | 停利指定漲跌幅 |  | | 32 | 停利觸發價 |  | | 33 | 停利委託方式 | 1=市價  2=限價 | | 34 | 停利委託價 |  | | 35 | 停利委託時效 | 0=ROD  3=IOC  4=FOK | | 36 | 停損洗價模式 | 1=指定價格  2=指定漲跌幅(%) | | 37 | 停損指定漲跌幅 |  | | 38 | 停損觸發價 |  | | 39 | 停損委託方式 | 1=市價  2=限價 | | 40 | 停損委託價 |  | | 41 | 停損委託時效 | 0=ROD  3=IOC  4=FOK | | 42 | 是否執行時間出清 |  | | 43 | 指定出清時間 |  | | 44 | 時間出清委託方式 | 1=市價  2=限價 | | 45 | 時間出清委託價 |  | | 46 | 時間出清委託時效 | 0=ROD  3=IOC  4=FOK | | 47 | 是否執行盤後定價 |  | | 48 | 盤後定價執行時間 |  | | 49 | 進場單資訊 | (預約單無此欄), 若設進場MIT才有此欄  細項參閱當沖-進場單格式  **若無此欄將以[in]代表無進場單資訊** | | 50 | 出場單資訊 | (預約單無此欄)  細項參閱當沖-出場單格式  **若無此欄將以[out]代表無出場單資訊** |   當沖-進場單格式, 若設進場MIT才有此欄   |  |  |  | | --- | --- | --- | | 49-1 | 流水號(13碼) |  | | 49-2 | 原始流水號(13碼) |  | | 49-3 | 委託書號 |  | | 49-4 | 委託狀態 | 32：中台收單成功  33：中台收單失敗  34：洗價中  36：洗價失敗  37：洗價觸發  38：觸發下單 39：下單失敗  40：使用者刪單 41：全部成交  42：進場單委託中(當沖)  43：進場單成交，執行出場條件(當沖) | | 49-5 | 委託數量 |  | | 49-6 | 觸價方向 | 1:大於等於  2:小於等於 | | 49-7 | 觸發價格 | MIT進場單觸發價 | | 49-8 | 觸發時間 | MIT進場單觸發時間 | | 49-9 | 訊息 | 智慧單回報：  [智慧單號][商品代碼]訊息  市場回報：參考市場委託狀態 | | 49-10 | 更新時間 | 智慧單異動時間 | | 49-11 | 市場回報狀態 | 空白：無委託單 0：預約 2：全部成交 3：全部取消 4：部分成交，剩餘已取消 5：部分成交，剩餘可取消 6：委託失敗 7:委託成功 8:取消失敗  【交易所動態退單為F開頭編碼】 | |

當沖單- 出場單

|  |  |  |
| --- | --- | --- |
| 證券智慧單被動查詢結果。透過呼叫[GetTSSmartStrategyReport](#_GetTSSmartStrategyReport)後，資訊由該事件回傳。 | | |
| **宣告** | void OnTSSmartStrategyReport ([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 | | 1 | 智慧單號 |  | | 2 | 市場別 | TS：國內證券 | | 3 | 交易日 | 20200707 | | 4 | 交易所代碼 | TSE上市  OTC上櫃 | | 5 | 分公司代碼 |  | | 6 | 帳號 |  | | 7 | 子帳 |  | | 8 | 策略別 | 11:DayTrade當沖 | | 9 | 商品代碼 |  | | 10 | 買賣別 | B：買 S：賣 | | 11 | 委託種類別 | 0：現股 **3：融資**  **4：融券**  8：無券普賣 | | 12 | 委託別 | 0：一般 | | 13 | 委託價格類別 | 第一委託價格別  0:前日收盤價 (平盤價)  1:漲停價  2:跌停價  7:使用者輸入價 | | 14 | 委託價格別 | 1：市價  2：限價 | | 15 | 委託時效 | 0:ROD;3：IOC;4:FOK | | 16 | 下單時間 | 2020-07-07 12:32:10.000 | | 17 | 委託價格 |  | | 18 | 委託數量 |  | | 19 | 智慧單委託狀態-  代碼 | 32：中台收單成功  33：中台收單失敗  34：洗價中  36：洗價失敗  37：洗價觸發  38：觸發下單  39：下單失敗  40：使用者刪單  41：全部成交  \*可刪單狀態: 32, 34, 37, 38 | | 20 | 市場委託狀熊 | 空白：無委託單  0：預約  2：全部成交  3：全部取消  4：部分成交，剩餘已取消  5：部分成交，剩餘可取消  6：委託失敗  7:委託成功  8:取消失敗  【交易所動態退單為F開頭編碼】  F1：動態退單-全部取消  F2：動態退單-部分成交，剩餘已取消  F3：動態退單-部分委託成功 | | 21 | 訊息更新時間 | 2020-07-07 13:30:41 | | 22 | 訊息含狀態 | [智慧單號]  [商品代碼]  訊息:參考市場委託狀態  Ex: [537500][2801]已加入洗價  訊息中逗號皆為全形”，” | | 23 | 委託書號 |  | | 24 | 流水號(13碼) |  | | 25 | 原始流水號(13碼) |  | | 26 | 是否啟用MIT進場 | 0:否; 1:是 | | 27 | MIT進場觸發價 |  | | 28 | MIT進場觸發時間 |  | | 29 | 出清方式 | 17=只選時間出清條件  18=GTE(條件一)  19=LTE(條件二)  20=GTE\_LTE(條件一+條件二) | | 30 | 停利洗價模式 | 1=指定價格  2=指定漲跌幅(%) | | 31 | 停利指定漲跌幅 |  | | 32 | 停利觸發價 |  | | 33 | 停利委託方式 | 1=市價  2=限價 | | 34 | 停利委託價 |  | | 35 | 停利委託時效 | 0=ROD  3=IOC  4=FOK | | 36 | 停損洗價模式 | 1=指定價格  2=指定漲跌幅(%) | | 37 | 停損指定漲跌幅 |  | | 38 | 停損觸發價 |  | | 39 | 停損委託方式 | 1=市價  2=限價 | | 40 | 停損委託價 |  | | 41 | 停損委託時效 | 0=ROD  3=IOC  4=FOK | | 42 | 是否執行時間出清 | 0:否;1:是 | | 43 | 指定出清時間 |  | | 44 | 時間出清委託方式 | 1=市價  2=限價 | | 45 | 時間出清委託價 |  | | 46 | 時間出清委託時效 | 0=ROD  3=IOC  4=FOK | | 47 | 是否執行盤後定價 | 0:否;1:是 | | 48 | 盤後定價執行時間 |  | | 49 | 進場單資訊 | (預約單無此欄), 若設進場MIT才有此欄  細項參閱當沖-進場單格式  **若無此欄將以[in]代表無進場單資訊** | | 50 | 出場單資訊 | (預約單無此欄)  細項參閱當沖-出場單格式  **若無此欄將以[out]代表無出場單資訊** | | 51 |  |  |   當沖-進場單格式, 若設進場MIT才有此欄   |  |  |  | | --- | --- | --- | | 49-1 | 流水號(13碼) |  | | 49-2 | 原始流水號(13碼) |  | | 49-3 | 委託書號 |  | | 49-4 | 委託狀態 | 32：中台收單成功  33：中台收單失敗  34：洗價中  36：洗價失敗  37：洗價觸發  38：觸發下單 39：下單失敗  40：使用者刪單 41：全部成交  42：進場單委託中(當沖)  43：進場單成交，執行出場條件(當沖)  \*可刪單狀態: 32, 34, 37, 38 | | 49-5 | 委託數量 |  | | 49-6 | 觸價方向 | 1:大於等於  2:小於等於 | | 49-7 | 觸發價格 | MIT進場單觸發價 | | 49-8 | 觸發時間 | MIT進場單觸發時間 | | 49-9 | 訊息 | 智慧單回報：  [智慧單號][商品代碼]訊息  市場回報：參考市場委託狀態 | | 49-10 | 更新時間 | 智慧單異動時間 | | 49-11 | 市場回報狀態 | 空白：無委託單 0：預約 2：全部成交 3：全部取消 4：部分成交，剩餘已取消 5：部分成交，剩餘可取消 6：委託失敗 7:委託成功 8:取消失敗  【交易所動態退單為F開頭編碼】 |   當沖-出場單格式   |  |  |  | | --- | --- | --- | | 50-1 | 出場單號 |  | | 50-2 | 流水號(13碼) |  | | 50-3 | 原始流水號(13碼) |  | | 50-4 | 委託書號 |  | | 50-5 | 委託狀態 | 32：中台收單成功 ;33：中台收單失敗 34：洗價中 ;36：洗價失敗  37：洗價觸發 ;38：觸發下單  39：下單失敗 ; 40：使用者刪單 41：全部成交 ; 42：進場單委託中(當沖)  43：進場單成交，執行出場條件(當沖)  \*可刪單狀態: 32, 34, 37, 38 | | 50-6 | 進場成交價 |  | | 50-7 | 停利觸發價 |  | | 50-8 | 停損觸發價 |  | | 50-9 | 觸發時間 | MIT進場單觸發時間 | | 50-10 | 觸發條件 | 條件一觸發，價格:xxx 條件二觸發，價格:xxx 時間出清  盤後定價 | | 50-11 | 訊息 | 智慧單回報：  [**智慧單出場單號**][商品代碼]訊息  市場回報：參考市場委託狀態 | | 50-12 | 市場回報狀態 | 空白：無委託單 0：預約 2：全部成交 3：全部取消 4：部分成交，剩餘已取消 5：部分成交，剩餘可取消 6：委託失敗 7:委託成功 8:取消失敗  【交易所動態退單為F開頭編碼】 | | 50-13 | 委託數量 | 32：中台收單成功 33：中台收單失敗 34：洗價中  35：洗價中-觸發價更新(移動停損單) 36：洗價失敗  37：洗價觸發  38：觸發下單 39：下單失敗  40：使用者刪單 41：全部成交  42：進場單委託中(當沖)  43：進場單成交，執行出場條件(當沖) | | 50-14 | 已成交量 |  | |

#### 出清單

|  |  |  |
| --- | --- | --- |
| 證券智慧單被動查詢結果。透過呼叫 GetTSSmartStrategyReport 後，資訊由該事件回傳。 | | |
| **宣告** | void OnTSSmartStrategyReport ([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 | | 1 | 智慧單號 |  | | 2 | 市場別 | TS：國內證券 | | 3 | 交易日 | 20200707 | | 4 | 交易所代碼 | TSE上市  OTC上櫃 | | 5 | 分公司代碼 |  | | 6 | 帳號 |  | | 7 | 子帳 |  | | 8 | 策略別 | 9:MST | | 9 | 商品代碼 |  | | 10 | 買賣別 | B：買 S：賣 | | 11 | 委託種類別 | 0：現股 **3：融資**  **4：融券**  8：無券普賣 | | 12 | 委託別 | 0：一般 | | 13 | 委託價格類別 | 第一委託價格別  0:前日收盤價 (平盤價)  1:漲停價  2:跌停價  7:使用者輸入價 | | 14 | 委託價格別 | 1：市價  2：限價 | | 15 | 委託時效 | 0:ROD;3：IOC;4:FOK | | 16 | 下單時間 | 2020-07-07 12:32:10.000 | | 17 | 委託價格 |  | | 18 | 委託數量 |  | | 19 | 觸發價 |  | | 20 | 觸發時間 |  | | 21 | 智慧單委託狀態-  代碼 | 32：中台收單成功  33：中台收單失敗  34：洗價中  36：洗價失敗  37：洗價觸發  38：觸發下單  39：下單失敗  40：使用者刪單  41：全部成交(多次IOC用) | | 22 | 市場委託狀熊 | 空白：無委託單  0：預約  2：全部成交  3：全部取消  4：部分成交，剩餘已取消  5：部分成交，剩餘可取消  6：委託失敗  7:委託成功  8:取消失敗  【交易所動態退單為F開頭編碼】  F1：動態退單-全部取消  F2：動態退單-部分成交，剩餘已取消  F3：動態退單-部分委託成功 | | 23 | 訊息更新時間 | 2020-07-07 13:30:41 | | 24 | 訊息含狀態 | [智慧單號]  [商品代碼]  訊息:參考市場委託狀態  Ex: [537500][2801]已加入洗價  訊息中逗號皆為全形”，” | | 25 | 委託書號 |  | | 26 | 流水號(13碼) |  | | 27 | 原始流水號(13碼) |  | | 28 | 出清條件類別 | 17=只勾選時間出清條件  18=GTE(條件一)大於  19=LTE(條件二)小於  20=GTE\_LTE(條件一+條件二) | | 29 | 條件一觸發價 |  | | 30 | 條件一委託方式 | 1=市價  2=限價 | | 31 | 條件一委託價 |  | | 32 | 條件一委託時效 | 0=ROD  3=IOC  4=FOK | | 33 | 條件二觸發價 |  | | 34 | 條件二委託方式 | 1=市價  2=限價 | | 35 | 條件二委託價 |  | | 36 | 條件二委託時效 | 0=ROD  3=IOC  4=FOK | | 37 | 是否執行時間出清 |  | | 38 | 指定出清時間 |  | | 39 | 時間出清委託方式 | 1=市價  2=限價 | | 40 | 時間出清委託價 |  | | 41 | 時間出清委託時效 | 0=ROD  3=IOC  4=FOK | | 42 | 是否執行盤後定價 |  | | 43 | 盤後定價執行時間 |  | | 44 | 條件一是否有觸發 | 0=否  1=是 | | 45 | 條件一觸發價格 |  | | 46 | 條件二是否有觸發 | 0=否  1=是 | | 47 | 條件二觸發價格 |  | | 48 | 時間出清是否有觸發 | 0=否  1=是 | | 49 | 盤後定價是否有觸發 | 0=否  1=是 | | 50 | 總委託量 |  | | 51 | 已成交量 |  | | 52 | 進場成交價 |  | | 53 | 子單 | (預約單無此欄)  細項參閱出清-子單格式  **若無此欄將以[out]代表無子單資訊** |   出清子單—**觸**發歷程：   |  |  |  | | --- | --- | --- | | 53-1 | 流水號(13碼) |  | | 53-2 | 委託書號 |  | | 53-3 | 觸發時間 |  | | 53-4 | 觸發條件 | 條件一觸發，價格:xxx 條件二觸發，價格:xxx 時間出清  盤後定價 | | 53-5 | 委託訊息 | [智慧單號][商品代碼]訊息 | | 53-6 | 市場委託狀態 | 空白：無委託單 0：預約 2：全部成交 3：全部取消 4：部分成交，剩餘已取消 5：部分成交，剩餘可取消 6：委託失敗 7:委託成功 8:取消失敗  【交易所動態退單為F開頭編碼】  F1：動態退單-全部取消 F2：動態退單-部分成交，剩餘已取消 F3：動態退單-部分委託成功 | | 53-7 | 成交張數 |  | |  |  |  | |
| **備註** | * 回傳【M003,無資料】表示目前查無被動回報 * 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。 | |

|  |  |  |
| --- | --- | --- |
| [OCO 二擇一單]證券智慧單被動查詢結果。透過呼叫 GetTSSmartStrategyReport後，資訊由該事件回傳。 | | |
| **宣告** | void OnTSSmartStrategyReport ([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 | | 1 | 智慧單號 |  | | 2 | 市場別 | TS：國內證券 | | 3 | 交易日 | 20200707 | | 4 | 交易所代碼 | TSE上市  OTC上櫃 | | 5 | 分公司代碼 |  | | 6 | 帳號 |  | | 7 | 子帳 |  | | 8 | 策略別 | 9:MST | | 9 | 商品代碼 |  | | 10 | 買賣別 | B：買 S：賣 | | 11 | 委託種類別 | 0：現股 **3：融資**  **4：融券**  8：無券普賣 | | 12 | 委託別 | 0：一般 | | 13 | 委託價格類別 | 第一委託價格別  0:前日收盤價 (平盤價)  1:漲停價  2:跌停價  7:使用者輸入價 | | 14 | 委託價格別 | 1：市價  2：限價 | | 15 | 委託時效 | 0:ROD;3：IOC;4:FOK | | 16 | 下單時間 | 2020-07-07 12:32:10.000 | | 17 | 委託價格 |  | | 18 | 委託數量 |  | | 19 | 觸發價 |  | | 20 | 觸發時間 |  | | 21 | 智慧單委託狀態-  代碼 | 32：中台收單成功  33：中台收單失敗  34：洗價中  36：洗價失敗  37：洗價觸發  38：觸發下單  39：下單失敗  40：使用者刪單  41：全部成交(多次IOC用) | | 22 | 市場委託狀熊 | 空白：無委託單  0：預約  2：全部成交  3：全部取消  4：部分成交，剩餘已取消  5：部分成交，剩餘可取消  6：委託失敗  7:委託成功  8:取消失敗  【交易所動態退單為F開頭編碼】  F1：動態退單-全部取消  F2：動態退單-部分成交，剩餘已取消  F3：動態退單-部分委託成功 | | 23 | 訊息更新時間 | 2020-07-07 13:30:41 | | 24 | 訊息含狀態 | [智慧單號]  [商品代碼]  訊息:參考市場委託狀態  Ex: [537500][2801]已加入洗價  訊息中逗號皆為全形”，” | | 25 | 委託書號 |  | | 26 | 流水號(13碼) |  | | 27 | 原始流水號(13碼) |  | | 28 | 第一隻腳觸發價 |  | | 29 | 第二隻腳觸發價 |  | | 30 | 第二隻腳買賣別 | B：買 S：賣 | | 31 | 第二隻腳委託價格別 | 0:前日收盤價 (平盤價)  1:漲停價  2:跌停價  7:使用者輸入價 | | 32 | 第二隻腳委託種類別 | 0：現股 3：自資  4：自券  8：無券普賣 | | 33 | 第二隻腳委託價格類別 | 1：市價  2：限價 | | 34 | 第二隻腳委託價 |  | | 35 | 第二隻腳委託時效 | 0：當日有效ROD  3：立即成交IOC  4：立即全部成交FOK | |
| **備註** | * 回傳【M003,無資料】表示目前查無被動回報 * 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。 | |

#### AB單

|  |  |  |
| --- | --- | --- |
| 證券智慧單被動查詢結果。透過呼叫[GetTSSmartStrategyReport](#_GetTSSmartStrategyReport)後，資訊由該事件回傳。 | | |
| **宣告** | void OnTSSmartStrategyReport ([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 | | 1 | 智慧單號 |  | | 2 | 市場別 | TS：國內證券 | | 3 | 交易日 | 20200707 | | 4 | 交易所代碼 | TSE上市  OTC上櫃 | | 5 | 分公司代碼 |  | | 6 | 帳號 |  | | 7 | 子帳 | 無子帳以0000000 | | 8 | 委託書號 |  | | 9 | 流水號13碼 |  | | 10 | 原始流水號 |  | | 11 | 策略別 | 10:AB | | 12 | 商品代碼 |  | | 13 | 下單市場別 |  | | 14 | 下單交易所別 |  | | 15 | 下單商品代碼 |  | | 16 | 商品契約月份 |  | | 17 | 履約價格 |  | | 18 | 下單商品代碼2 |  | | 19 | 商品契約月份2 |  | | 20 | 履約價格2 |  | | 21 | 買賣別 | B：買 S：賣 | | 22 | 委託種類別 | 0：現股 3：融資  4：融券  8：無券普賣 | | 23 | 委託別 | 0：一般 | | 24 | 委託價格類別 | 第一委託價格別  0:前日收盤價 (平盤價)  1:漲停價  2:跌停價  7:使用者輸入價 | | 25 | 委託價格別 | 1：市價  2：限價 | | 26 | 委託時效 | 0:ROD;3：IOC;4:FOK | | 27 | 委託價格 |  | | 28 | 委託數量 |  | | 29 | 觸發價 |  | | 30 | 觸發時間 |  | | 31 | 智慧單委託狀態-  代碼 | 32：中台收單成功  33：中台收單失敗  34：洗價中  36：洗價失敗  37：洗價觸發  38：觸發下單  39：下單失敗  40：使用者刪單  41：全部成交(多次IOC用) | | 32 | 下單時間 | 2020-07-07 12:32:10.000 | | 33 | 訊息含狀態 | [智慧單號]  [商品代碼]  訊息:參考市場委託狀態  Ex: [537500][2801]已加入洗價  訊息中逗號皆為全形”，” | | 34 | 訊息更新時間 | 2020-07-07 13:30:41 | | 35 | 市場委託狀熊 | 空白：無委託單  0：預約  2：全部成交  3：全部取消  4：部分成交，剩餘已取消  5：部分成交，剩餘可取消  6：委託失敗  7:委託成功  8:取消失敗  【交易所動態退單為F開頭編碼】  F1：動態退單-全部取消  F2：動態退單-部分成交，剩餘已取消  F3：動態退單-部分委託成功 | | 36 | 當下市價價格 |  | | 37 | 觸價方向 | 1：大於等於  2：小於等於 | | 38 | 是否為選擇權商品 | 0：否  1：是 | | 39 | 預約單註記 | 0：否  1：是 | | 40 | 預約單序號 |  | | 41 | 預約單交易日 |  | | 42 | 是否觸發即停止 | 0：否  1：是 | | 43 | 是否為內外盤委託量 | 0：否  1：是 | | 44 | 內外盤委託量 |  | | 45 | 內外盤委託量觸發方向 | 1：大於等於  2：小於等於 | | 46 | 成交價觸發價格 |  | | 47 | 成交量 |  | | 48 | 新平倉 | 0：新單  1：平倉  2：自動 | | 49 | 是否為當沖 | 0：否  1：是 | | 50 | 買賣權 | N：None  C：Call  P：Put | | 51 | 委託價分子 |  | | 52 | 委託價分母 |  | | 53 | 成交價分子 |  | | 54 | 成交價分母 |  | | 55 | 觸發價分子 |  | | 56 | 觸發價分母 |  | | 57 | 萬用訊息 |  | |
| **備註** | * 回傳【M003,無資料】表示目前查無被動回報 * 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。 * 自SKCOM V2.13.30(含)版本起，提供MIT新版被動回報 | |

#### CB單(自組單)

|  |  |  |
| --- | --- | --- |
| 證券智慧單被動查詢結果。透過呼叫[GetTSSmartStrategyReport](#_GetTSSmartStrategyReport)後，資訊由該事件回傳。 | | |
| **宣告** | void OnTSSmartStrategyReport ([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 | | 1 | 智慧單號 |  | | 2 | 市場別 | TS：國內證券 | | 3 | 交易日 | 20200707 | | 4 | 交易所代碼 | TSE上市  OTC上櫃 | | 5 | 分公司代碼 |  | | 6 | 帳號 |  | | 7 | 子帳 |  | | 8 | 策略別 | 27:CB | | 9 | 商品代碼 |  | | 10 | 買賣別 | B：買 S：賣 | | 11 | 委託種類別 | 0：現股 3：融資  4：融券  8：無券普賣 | | 12 | 委託別 | 0：一般 | | 13 | 委託價格類別 | 第一委託價格別  0:前日收盤價 (平盤價)  1:漲停價  2:跌停價  7:使用者輸入價 | | 14 | 委託價格別 | 1：市價  2：限價 | | 15 | 委託時效 | 0:ROD;3：IOC;4:FOK | | 16 | 下單時間 | 2020-07-07 12:32:10.000 | | 17 | 委託價格 |  | | 18 | 委託數量 |  | | 19 | 觸發價 |  | | 20 | 觸發時間 |  | | 21 | 智慧單委託狀態-  代碼 | 32：中台收單成功  33：中台收單失敗  34：洗價中  35：洗價中-觸發價更新(移動停損單)  36：洗價失敗  37：洗價觸發  38：觸發下單  39：下單失敗  40：使用者刪單  41：全部成交(多次IOC用) | | 22 | 市場委託狀態 | 空白：無委託單  0：預約  2：全部成交  3：全部取消  4：部分成交，剩餘已取消  5：部分成交，剩餘可取消  6：委託失敗  7:委託成功  8:取消失敗  【交易所動態退單為F開頭編碼】  F1：動態退單-全部取消  F2：動態退單-部分成交，剩餘已取消  F3：動態退單-部分委託成功 | | 23 | 訊息更新時間 | 2020-07-07 13:30:41 | | 24 | 訊息含狀態 | [智慧單號]  [商品代碼]  訊息:參考市場委託狀態  Ex: [537500][2801]已加入洗價  訊息中逗號皆為全形”，” | | 25 | 是否全部條件成立 | 0：任一條件成立  1：全部條件成立 | | 26 | 是否自訂啟動時間 | 0：否  1：是 | | 27 | 自訂啟動時間 | 08:00:00.000 | | 28 | 是否執行成交價 | 0：否  1：是 | | 29 | 成交價 |  | | 30 | 成交價觸價方向 | 0：None  1：GTE  2：LTE | | 31 | 成交價觸發紀錄 | 0：未觸發  1：觸發 | | 32 | 成交價觸發價格 |  | | 33 | 是否執行委買價 | 0：否  1：是 | | 34 | 委買價 |  | | 35 | 委買價觸價方向 | 0：None  1：GTE  2：LTE | | 36 | 委買價觸發紀錄 | 0：未觸發  1：觸發 | | 37 | 委買價觸發價格 |  | | 38 | 是否執行委賣價 | 0：否  1：是 | | 39 | 委賣價 |  | | 40 | 委賣價觸價方向 | 0：None  1：GTE  2：LTE | | 41 | 委賣價觸發記號 | 0：未觸發  1：觸發 | | 42 | 委賣價觸發價格 |  | | 43 | 是否執行漲跌tick | 0：否  1：是 | | 44 | Tick數 |  | | 45 | 漲跌tick觸價方向 | 0：None  1：GTE  2：LTE | | 46 | 漲跌tick觸發記號 | 0：未觸發  1：觸發 | | 47 | 漲跌tick觸發tick |  | | 48 | 是否執行漲跌幅(%) | 0：否  1：是 | | 49 | 漲跌幅(%) |  | | 50 | 漲跌幅的觸價方向 | 0：None  1：GTE  2：LTE | | 51 | 漲跌幅觸發記號 | 0：未觸發  1：觸發 | | 52 | 漲跌幅觸發幅度 |  | | 53 | 是否執行總量 | 0：否  1：是 | | 54 | 總量 |  | | 55 | 總量觸價方向 | 0：None  1：GTE  2：LTE | | 56 | 總量觸發記號 | 0：未觸發  1：觸發 | | 57 | 總量觸發數量 |  | | 58 | 是否執行單量 | 0：否  1：是 | | 59 | 單量 |  | | 60 | 單量觸價方向 | 0：None  1：GTE  2：LTE | | 61 | 單量觸發記號 | 0：未觸發  1：觸發 | | 62 | 單量觸發數量 |  | | 63 | 萬用訊息 | 適用於智慧單狀態代碼為999情況 | |
| 備註 | * 回傳【M003,無資料】表示目前查無被動回報 * 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\_TSStrategyOrder.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) |
| # 證券智慧單被動查詢結果。透過呼叫GetTSSmartStrategyReport後，資訊由該事件回傳。      def OnTSSmartStrategyReport(self, bstrData):          msg = "【OnTSSmartStrategyReport】" + bstrData          richTextBoxMessage.insert('end', msg + "\n")          richTextBoxMessage.see('end') |

### 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\_TSStrategyOrder.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') |

## Struct 結構物件

### STOCKSTRATEGYORDER(證券智慧單物件-當沖/CB)

#### 證券當沖

struct STOCKSTRATEGYORDER

{

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

//商品//

|  |
| --- |
| BSTR bstrStockNo; //委託股票代號  LONG nBuySell; //當沖進場 0:現股買;1:無券賣出  LONG nQty; //委託張數  LONG nOrderPriceCond; //委託時效(0:ROD,3:IOC,4:FOK)  BSTR bstrOrderPrice; //當沖單進場委託價  LONG nOrderPriceType; //委託價類別(1:市價; 2:限價) |

//當沖進場條件//

|  |
| --- |
| LONG nInnerOrderIsMIT; //進場是否MIT( 0: N; 1:Y)  LONG nMITDir; //進場MIT觸價方向(0:未啟用MIT,1: 向上觸發(大於等於) , 2:向下觸發(小於等於))  BSTR bstrMITTriggerPrice; //進場MIT觸發價(若未啟用MIT,請填0)  BSTR bstrMITDealPrice; //進場MIT當下市價(若未啟用MIT,請填0) |

//停損出場條件//

|  |
| --- |
| LONG nStopLossFlag; //出場停損條件0:否；1:是  LONG nRDOSLPercent; //停損類型0:觸發價；1:漲跌幅  BSTR bstrSLTrigger; //停損觸發價  BSTR bstrSLPercent; //停損百分比  BSTR bstrSLOrderPrice; //停損委託價  LONG nRDSLMarketPriceType; ///停損委託價方式1:市價; 2:限價  LONG nStopLossOrderCond; //停損出場時效 0:ROD;3:IOC;4:FOK |

//停利出場條件//

|  |
| --- |
| LONG nTakeProfitFlag; //出場停利條件 0:否；1:是  LONG nRDOTPPercent; //停利類型 0:觸發價；1:漲幅  BSTR bstrTPTrigger; //停利觸發價  BSTR bstrTPPercent; //停利百分比  BSTR bstrTPOrderPrice; //停利委託價  LONG nRDTPMarketPriceType; //停利委託價方式1:市價; 2:限價  LONG nTakeProfitOrderCond; //停利出場時效 0:ROD;3:IOC;4:FOK |

//時間出清條件//

|  |
| --- |
| LONG nClearAllFlag; //執行時間出清0:否；1:是  BSTR bstrClearCancelTime; //出清時間-時＋分（hhmm）{每日13:20截止}  BSTR bstrClearAllOrderPrice; //出清委託價  LONG 　　nClearAllPriceType; //出清委託價方式 1:市價; 2:限價  LONG nClearOrderCond; //出清出場時效 0:ROD;3:IOC;4:FOK |

//盤後定價交易出場條件//

|  |
| --- |
| LONG nFinalClearFlag; //盤後定盤交易 0:否；1:是 |

}

#### 證券自組單(CB)

struct STOCKSTRATEGYORDER

{

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

//商品//

|  |
| --- |
| BSTR bstrStockNo; //委託股票代號  LONG nOrderType;  //委託交易別(0:現股, 3:融資, 4:融券, 8:無券普賣)  LONG nOrderPriceCond; //委託時效(0:ROD,3:IOC,4:FOK)  LONG nBuySell; //0買;1賣 |

//組合//

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | LONG nInnerOrderIsMIT; //觸發條件是否為成交價(0:否, 1:是)  LONG nMITDir; //成交價觸價方向(0:None, 1:GTE大於等於, 2:LTE小於等於)  BSTR bstrMITTriggerPrice;//成交價 | | LONG nTakeProfitFlag;             //觸發條件是否為委買價(0:否, 1:是)  LONG  nTPDir; //委買價觸價方向 (0:None, 1:GTE大於等於, 2:LTE小於等於)  BSTR bstrTPOrderPrice;          //委買價 | | LONG nUpDownFlag; //是否為漲跌幅 (0:否, 1:是)  LONG nUpDownDir; //漲跌幅觸價方向(0:None, 1:GTE大於等於, 2:LTE小於等於)  BSTR bstrSLPercent; //漲跌幅數 | | LONG nPreQtyFlag; //是否執行單量 (0:否, 1:是)  LONG nPreQtyDir; //單量觸價方向(0:None, 1:GTE大於等於, 2:LTE小於等於)  BSTR bstrPreQty; //單量 | | LONG nStopLossFlag;  //是否為委賣價 (0:否, 1:是)  LONG nSLDir; //委賣價觸價方向(0:None, 1:GTE大於等於, 2:LTE小於等於)  BSTR bstrSLOrderPrice;           //委賣價 | | LONG nTickFlag; //是否為tick觸發 (0:否, 1:是)  LONG nTickDir; //tick觸發方向 (0:None, 1:GTE大於等於, 2:LTE小於等於)  BSTR bstrTick; //tick數 | | LONG nSumQtyFlag; //觸發條件是否為總量 (0:否, 1:是)  LONG nSumQtyDir; //總量觸價方向(0:None, 1:GTE大於等於, 2:LTE小於等於)  BSTR bstrSumQty; //總量 | |

//觸發時間//

|  |
| --- |
| LONG nClearAllFlag; //是否為自訂啟動時間 (0:立即, 1:自訂)  BSTR bstrClearCancelTime; //自訂啟動時間(14天內) 格式：hhmmss (時時分分秒秒) |

//組合條件關係//

|  |
| --- |
| LONG nFinalClearFlag; //條件是否為全部成立 (0:任一成立, 1:全部成立) |

//委託張數//

|  |
| --- |
| LONG nQty; //委託張數 |

//委託張數//

|  |
| --- |
| LONG nOrderPriceType; //委託價類別(1:市價; 2:限價)  BSTR bstrOrderPrice; //委託價 |

)

### STOCKSTRATEGYORDEROUT(證券智慧單物件-出清)

struct STOCKSTRATEGYORDEROUT

{

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

//商品//

|  |
| --- |
| BSTR bstrStockNo; //委託股票代號  LONG nBuySell; //0買;1賣  LONG nOrderType; //出清進場委託交易別(0:現股, 3:融資, 4:融券)  LONG nQty; //委託張數 |

//大於出場條件//

|  |
| --- |
| LONG nGTEFlag; //出場成交價大於指定價後觸發0:否；1:是  BSTR bstrGTETriggerPrice; //大於指定觸發價  BSTR bstrGTEOrderPrice; //大於出場指定委託價  LONG nGTEMarketPrice; //(大於條件)以1:市價; 2:限價 做為出場價格  LONG nGTEOrderCond; //大於條件出場時效0:ROD;3:IOC;4:FOK |

//小於出場條件//

|  |
| --- |
| **LONG** nLTEFlag; //出場成交價小於指定價後觸發0:否；1:是  BSTR bstrLTETriggerPrice; //小於指定觸發價  BSTR bstrLTEOrderPrice; //小於出場指定委託價  LONG nLTEMarketPrice; //(小於條件)以1:市價; 2:限價 做為出場價格  LONG nLTEOrderCond; //小於條件出場時效0:ROD;3:IOC;4:FOK |

//時間出清條件//

|  |
| --- |
| LONG nClearAllFlag; //執行時間出清0:否；1:是  BSTR bstrClearCancelTime; //出清時間-時＋分（hhmm）{每日13:20截止}  BSTR bstrClearAllOrderPrice; //出清委託價  LONG nClearAllPriceType; //出清委託價方式=1:市價; 2:限價  LONG nClearOrderCond; //出清出場時效0:ROD;3:IOC;4:FOK |

//盤後定價交易條件//

|  |
| --- |
| LONG nFinalClearFlag; //盤後定盤交易 0:否；1:是 |

}

### STOCKSTRATEGYORDERMIT(證券智慧單物件-MIT/MST/AB)

#### MIT單

**(新版本MIT， V2.13.30 含以上版本適用)**，並自V2.13.30 (含)以上版本起，不再支援舊版下單)

struct STOCKSTRATEGYORDERMIT

{

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

BSTR bstrDealPrice; //成交價 (當下市價)，洗價機留存用

//商品//

|  |
| --- |
| BSTR bstrStockNo; //委託股票代號  LONG nOrderType; //委託交易別(0:現股, 3:融資, 4:融券, 8:無券普賣)  LONG nOrderCond; //委託時效(0:ROD,3:IOC,4:FOK)  LONG nBuySell; //0買;1賣 |

//組合//

|  |
| --- |
| LONG nTriggerDir; //觸價方向(1:GTE大於等於,2:LTE小於等於)  BSTR bstrTriggerPrice; //觸發價  LONG nQty; //委託張數  BSTR bstrOrderPrice; //委託價 (若為市價單, 委託價請填0)  LONG nOrderPriceType; //委託價類別(1:市價; 2:限價)  LONG nLongActionFlag; //是否為長效單(0:否, 1:是)  BSTR bstrLongEndDate //長效單結束日期(YYYYMMDD共8碼, EX: 20220630)  LONG nLAType //觸發結束條件(1:效期內觸發即失效, 3:效期內完全成交即失效) |

//預風控//

|  |
| --- |
| LONG nPreRiskFlag; //預風控功能 0:關閉預風控 ;1:開啟預風控(不支援信用交易) |

};

比較表

|  |
| --- |
| 異動後V2.13.45 |
| **MIT下單新增長效單功能：**是否為長效單(0:否, 1:是)、長效單結束日期(YYYYMMDD共8碼)、觸發結束條件  **MIT下單新增預風控欄位** |

#### MST單

struct STOCKSTRATEGYORDERMIT

{

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

//商品//

|  |
| --- |
| BSTR bstrStockNo; //委託股票代號  LONG nPrime; //0:上市,1:上櫃  LONG nOrderType;  //委託交易別(0:現股, 3:融資, 4:融券, 8:無券普賣)  LONG nOrderCond; //委託時效(0:ROD,3:IOC,4:FOK)  LONG nBuySell; // 0:買; 1:賣 |

//組合//

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | LONG nTriggerMethod; //啟動條件(0：以市價立即啟動, 1：以自訂價格啟動)  BSTR bstrDealPrice; //[市價啟動適用]請填市價  BSTR bstrStartPrice; //[自訂價格啟動適用]觸發價  LONG nTriggerDir //觸價方向(1:GTE大於等於,2:LTE小於等於) | | //移動點數  BSTR bstrMovePoint; //移動點數 | | //委託張數  LONG nQty; //委託張數 | | //委託價格  BSTR bstrOrderPrice; //委託價  LONG nOrderPriceType; //委託價類別(1:市價; 2:限價) | | //長效單  LONG nLongActionFlag; //是否為長效單(0:否, 1:是)  BSTR bstrLongEndDate //長效單結束日期(YYYYMMDD共8碼, EX: 20220630) | | //觸發結束條件  LONG nLAType //觸發結束條件(0:非長效單, 1:效期內觸發即失效) | |

};

#### AB單

struct STOCKSTRATEGYORDERMIT

{

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

//看A//

|  |
| --- |
| BSTR bstrStockNo2; //A商品 委託股票代號  LONG nMarketNo; //A商品 市場編號(1:國內證, 2:國內期, 3:國外證, 4:國外期)  BSTR bstrExchangeNo; // A商品 交易所代碼(EX: TSE、TAIFEX、CME)  BSTR bstrStartPrice; // A商品 成交價  LONG nTriggerDir; // A商品 觸價方向(1:GTE大於等於,  2:LTE小於等於)  BSTR bstrTriggerPrice; // A商品 觸發價 |

//下B//

|  |  |  |
| --- | --- | --- |
| |  | | --- | | //委託條件  LONG nBuySell; // B商品 買賣別(0:買, 1:賣)  LONG nOrderType;  // B商品 委託交易別(0:現股, 3:融資, 4:融券, 8:無券普賣)  LONG nOrderCond; // B商品 委託時效(0:ROD,3:IOC,4:FOK) | | //委託商品  BSTR bstrStockNo; // B商品 委託股票代號  LONG nPrime; // B商品 0:上市,1:上櫃  LONG nQty; // B商品 委託張數  LONG nOrderPriceType; // B商品委託價類別(1:市價; 2:限價)  BSTR bstrOrderPrice; // B商品 委託價 | |

LONG nReserved; //是否為預約單(0:否, 1:是)，A商品為國內期選商品時可選預約單，其餘市場為非預約單

}

### STOCKSTRATEGYORDEROCO(證券智慧單物件-OCO)

struct STOCKSTRATEGYORDEROCO

{

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

//商品//

|  |
| --- |
| BSTR bstrStockNo; //委託股票代號 |

//組合//

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | //當市價>=  BSTR bstrTriggerUp; //第一腳 觸發價  LONG nBuySellUp; //第一腳 0:買;1:賣  LONG nOrderCondUp; //第一腳 委託時效(0:ROD,3:IOC,4:FOK)  LONG nOrderTypeUp; //第一腳 委託交易別(0:現股, 3:融資, 4:融券;8:無券普賣)  BSTR bstrOrderPrice; //第一腳 委託價  LONG nOrderPriceTypeUp; //第一腳 委託價類別(1:市價; 2:限價) | | //當市價<=  BSTR bstrTriggerDown; //第二腳 觸發價  LONG nBuySellDown; //第二腳 0:買;1:賣  LONG nOrderCondDown; //第二腳 委託時效(0:ROD,3:IOC,4:FOK)  LONG nOrderTypeDown; //第二腳 委託交易別(0:現股, 3:融資, 4:融券;8:無券普賣)  BSTR bstrOrderPrice2; //第二腳 委託價  LONG nOrderPriceTypeDown; //第二腳 委託價類別(1:市價; 2:限價) | | //委託張數  LONG nQty; //委託張數 | |

};

\*v2.13.48新增無券普賣委託別

### STOCKSTRATEGYORDERMIOC(證券智慧單物件-多次IOC)

struct STOCKSTRATEGYORDERMIOC

{

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

//商品//

|  |
| --- |
| BSTR bstrStockNo; //委託商品代號  LONG nPrime; //0:上市,1:上櫃  LONG nOrderType; //委託交易別(0:現;3: (自)融資; 4: (自)融券;8無券賣)  LONG nBuySell; // 0:買進 1:賣出 |

//組合//

|  |
| --- |
| LONG nOrderPriceType; //0:市價; 1: (買單)委賣價或(賣單)委買價;  //委買、賣價實際價格由中台決定  LONG nOneceQtyLimit; //單次交易張數上限  BSTR bstrOrderPriceUp; //委託價上限  BSTR bstrOrderPriceDown;//委託價下限  LONG nTotalQty; //總委張數 |

};

### CANCELSTRATEGYORDER(智慧單刪單物件)

#### 證券當沖

struct CANCELSTRATEGYORDER

{

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

LONG nMarket; //市場別1:國內證券

BSTR bstrParentSmartKey; //智慧母單號;

BSTR bstrSmartKey; //智慧單號; 出場單則給出場單號

LONG nTradeKind; //11:當沖;

LONG nDeleteType; //刪單類型 1: 全部; 2: 進場單 ; 3:出場單

BSTR bstrSeqNo; //委託序號(1~3三種類型均需提供, 預約單可忽略)

BSTR bstrOrderNo; //((刪進場/出場單提供))委託書號，（若觸發，需給書號）

BSTR bstrSmartKeyOut; //出場單號(**刪全部時提供**，刪進場單及出場單時不需提供出場單號) //委託量大於1時，會不止一筆出場單，出場單號以逗號「, 」區隔

}

＊若被動查詢回報結果，當無書號時，則不需提供；若無尚無出場單號，則可不填。

＊刪全部及刪進場單時，智慧單母單號與智慧單號相同。

#### 證券出清

struct CANCELSTRATEGYORDER

{

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

LONG nMarket; //市場別1:國內證券

BSTR bstrSmartKey; //智慧單號; 出場單則給出場單號

LONG nTradeKind; //17:出清

BSTR bstrSeqNo; //委託序號 (預約單可忽略)

BSTR bstrOrderNo; //委託書號（若觸發，需給書號）

}

#### 證券二擇一OCO、MIT、MST、AB、CB

struct CANCELSTRATEGYORDER

{

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

LONG nMarket; //市場別1:國內證券

BSTR bstrSmartKey; //智慧單號

LONG nTradeKind; //3:OCO、8:MIT、9:MST、10:AB、27：CB

BSTR bstrSeqNo; //委託序號 (預約單可忽略)

BSTR bstrOrderNo; //委託書號（若觸發，需給書號）

BSTR bstrLongActionKey; //長效單號(非長效單請輸入0)

}

比較表

|  |
| --- |
| 異動後V2.13.45 |
| **智慧單刪單物件新增長效單號、新增刪單別：**非長效單之刪單，長效單號可忽略 |
| 異動後V2.13.48 |
| **移除MBA、LLS刪單。**  **非長效單請於長效單號輸入0。** |

#### 智慧單多筆刪單

struct CANCELSTRATEGYORDER

{

BSTR bstrFullAccount; //帳號

LONG nMarket; //刪單市場別，1:國內證, 2:國內期, 3:國外證, 4:國外期 (AB單刪單請選擇A商品之市場別)

BSTR bstrSmartKey; //多筆智慧單號 (以逗號分隔，EX：682020,682021)

}

#### 證券智慧單—可刪單狀態參考

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 代碼 | 是否可刪單 | 當沖(進場單) | 當沖(出場單) | MIOC單 | 當沖(刪全) | 出清 |
| 32 | Y | Y | Y | Y | Y | Y |
| 34 | Y | Y | Y | Y | Y | Y |
| 35\* | Y |  |  |  |  |  |
| 37 | Y | Y | Y | Y | Y | Y |
| 38 | Y | Y | Y | Y | Y | Y |
| 42 |  | Y |  |  | Y |  |
| 43 |  | Y |  |  | Y |  |

##### 代碼參考表

|  |  |  |
| --- | --- | --- |
| 代碼 | 說明+前台顯示訊息 | Message |
| 32 | 中台收單成功 | [智慧單號][股票代號]收單 ex. [375][1101]收單 |
| 34 | 洗價中 | [智慧單號][股票代號] 已加入洗價 ex. [375][1101]已加入洗價 |
| 35 | 洗價中-觸發價更新(移動停損單) | [智慧單號][股票代號] 觸發價更新 |
| 37 | 洗價觸發 | [智慧單號][股票代號]觸發條件觸發 ex.[375][1101]成交價觸發 |
| 38 | 觸發下單 | [智慧單號][股票代號]Tandem回覆訊息  ex. [353][1101]委託資料傳送交易所中! |
| 42 | 進場單委託中(當沖) | MIT進場->[智慧單號][股票代號]進場單-MIT觸發，送出委託 |
| 43 | 進場單成交，執行出場條件(當沖) | [智慧單號][股票代號]進場單成交，執行出場條件 |