策略王COM元件使用說明

-下單-國內期選智慧單-

# 函式範例

## SKOrderLib(國內期選)

*函式*

智慧單

|  |  |  |
| --- | --- | --- |
| **功能** | **函式名稱** | **備註** |
| **停損STP委託** | |  |  | | --- | --- | | **期貨** | [SendFutureSTPOrder](#_SendFutureSTPOrderV1)V1 | | [SendFutureStopLossOrder](#_SendFutureStopLossOrder)(舊版:僅近月) | | **選擇權** | [SendOptionStopLossOrder](#_SendOptionStopLossOrder) | | 簽署期貨智慧單風險預告書  可委託近月或指定月份 |
| **期貨移動停損委託** | |  | | --- | | [SendFutureMSTOrder](#_SendFutureMSTOrderV1)V1 | | [SendMovingStopLossOrder](#_SendMovingStopLossOrder)(舊版:僅近月) | |  |
| **MIT委託** | |  |  | | --- | --- | | **期貨** | [SendFutureMITOrder](#_SendFutureMITOrderV1)V1 | | [SendFutureMITOrder](#_SendFutureMITOrder)(舊版:僅近月) | | **選擇權** | [SendOptionMITOrder](#_SendOptionMITOrder) | |  |
| **期貨二擇一委託** | |  | | --- | | [SendFutureOCOOrder](#_SendFutureOCOOrderV1)V1 | | [SendFutureOCOOrder](#_SendFutureOCOOrder)(舊版:僅近月) | |  |
| **期貨AB單委託** | [SendFutureABOrder](#_SendFutureABOrder) |  |

|  |  |  |
| --- | --- | --- |
| **新版)期貨刪單V1** | [CancelTFStrategyOrderV1](#_CancelTFStrategyOrderV1) | 簽署期貨智慧單風險預告書  智慧單號及13碼序號，若委託已產生書號亦需一併填入 |
| **取消多筆智慧單委託** | [CancelStrategyList](#_CancelStrategyList) |  |
| **期貨智慧單查詢(停損單)** | [GetStopLossReport](#_GetStopLossReport) |  |

*事件*

|  |  |  |
| --- | --- | --- |
| **功能** | **事件名稱** | **備註** |
| **非同步委託結果** | [OnAsyncOrder](#_OnAsyncOrder) |  |
| **停損單資料** | [OnStopLossReport](#_OnStopLossReport) |  |

##### SendFutureSTPOrderV1

|  |  |  |
| --- | --- | --- |
| (指定月份需填商品契約年月)新版—送出期貨停損委託。 | | |
| **宣告** | Long SendFutureSTPOrderV1([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **FUTUREORDER**\* 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[**FUTUREORDER**](#_期貨智慧單STP_▼)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、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](#_4-2-b_OnAsyncOrder)取得。  需簽署期貨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。  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\_TFStrategyOrder.py類別STPForm(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)          # textBoxbstrStockNo          tk.Label(self, text = "期權商品代號").grid(row=0, column=1)          #輸入框          self.textBoxbstrStockNo = tk.Entry(self)          self.textBoxbstrStockNo.grid(row=0, column=2)          global textBoxbstrStockNo          textBoxbstrStockNo = self.textBoxbstrStockNo          # comboBoxsNewClose          tk.Label(self, text = "新平倉，0:新倉 1:平倉 2:自動").grid(row=0, column=3)              #輸入框          self.comboBoxsNewClose = ttk.Combobox(self, state='readonly')          self.comboBoxsNewClose['values'] = Config.comboBoxsNewClose          self.comboBoxsNewClose.grid(row=0, column=4)          global comboBoxsNewClose          comboBoxsNewClose = self.comboBoxsNewClose          # comboBoxsBuySell          tk.Label(self, text = "0:買進 1:賣出").grid(row=0, column=5)              #輸入框          self.comboBoxsBuySell = ttk.Combobox(self, state='readonly')          self.comboBoxsBuySell['values'] = Config.comboBoxsBuySell          self.comboBoxsBuySell.grid(row=0, column=6)          global comboBoxsBuySell          comboBoxsBuySell = self.comboBoxsBuySell          # comboBoxsTradeType          tk.Label(self, text = "0:ROD 3:IOC 4:FOK").grid(row=0, column=7)              #輸入框          self.comboBoxsTradeType = ttk.Combobox(self, state='readonly')          self.comboBoxsTradeType['values'] = Config.comboBoxsTradeType          self.comboBoxsTradeType.grid(row=0, column=8)          global comboBoxsTradeType          comboBoxsTradeType = self.comboBoxsTradeType          # comboBoxsDayTrade          tk.Label(self, text = "當沖0:否 1:是，可當沖商品請參考交易所規定").grid(row=0, column=9)              #輸入框          self.comboBoxsDayTrade = ttk.Combobox(self, state='readonly')          self.comboBoxsDayTrade['values'] = Config.comboBoxsDayTrade          self.comboBoxsDayTrade.grid(row=0, column=10)          global comboBoxsDayTrade          comboBoxsDayTrade = self.comboBoxsDayTrade          # textBoxbstrSettlementMonth          tk.Label(self, text = "委託商品年月，YYYYMM共6碼(EX: 202206)").grid(row=1, column=1)          #輸入框          self.textBoxbstrSettlementMonth = tk.Entry(self)          self.textBoxbstrSettlementMonth.grid(row=1, column=2)          global textBoxbstrSettlementMonth          textBoxbstrSettlementMonth = self.textBoxbstrSettlementMonth          # textBoxbstrTrigger          tk.Label(self, text = "觸發價，觸發基準價").grid(row=2, column=1)          #輸入框          self.textBoxbstrTrigger = tk.Entry(self)          self.textBoxbstrTrigger.grid(row=2, column=2)          global textBoxbstrTrigger          textBoxbstrTrigger = self.textBoxbstrTrigger            # textBoxnQty          tk.Label(self, text = "交易口數").grid(row=3, column=1)          #輸入框          self.textBoxnQty = tk.Entry(self)          self.textBoxnQty.grid(row=3, column=2)          global textBoxnQty          textBoxnQty = self.textBoxnQty          # comboBoxnOrderPriceType          tk.Label(self, text = "2: 限價; 3:範圍市價 （不支援市價）").grid(row=3, column=3)              #輸入框          self.comboBoxnOrderPriceType = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderPriceType['values'] = Config.comboBoxnOrderPriceType          self.comboBoxnOrderPriceType.grid(row=3, column=4)          global comboBoxnOrderPriceType          comboBoxnOrderPriceType = self.comboBoxnOrderPriceType          # textBoxbstrPrice          tk.Label(self, text = "委託價格，(指定限價時，需填此欄) //請設nOrderPriceType代表特殊價格「P」範圍市價").grid(row=3, column=5)          #輸入框          self.textBoxbstrPrice = tk.Entry(self)          self.textBoxbstrPrice.grid(row=3, column=6)          global textBoxbstrPrice          textBoxbstrPrice = self.textBoxbstrPrice          # comboBoxsReserved          tk.Label(self, text = "盤別，0:盤中(T盤及T+1盤)；1:T盤預約").grid(row=4, column=1)              #輸入框          self.comboBoxsReserved = ttk.Combobox(self, state='readonly')          self.comboBoxsReserved['values'] = Config.comboBoxsReserved          self.comboBoxsReserved.grid(row=4, column=2)          global comboBoxsReserved          comboBoxsReserved = self.comboBoxsReserved          # comboBoxnLongActionFlag          tk.Label(self, text = "是否為長效單(0:否, 1:是)").grid(row=5, column=1)              #輸入框          self.comboBoxnLongActionFlag = ttk.Combobox(self, state='readonly')          self.comboBoxnLongActionFlag['values'] = Config.comboBoxnLongActionFlag          self.comboBoxnLongActionFlag.grid(row=5, column=2)          global comboBoxnLongActionFlag          comboBoxnLongActionFlag = self.comboBoxnLongActionFlag          # textBoxbstrLongEndDate          tk.Label(self, text = "長效單結束日期(YYYYMMDD共8碼, EX: 20240205)").grid(row=5, column=3)          #輸入框          self.textBoxbstrLongEndDate = tk.Entry(self)          self.textBoxbstrLongEndDate.grid(row=5, column=4)          global textBoxbstrLongEndDate          textBoxbstrLongEndDate = self.textBoxbstrLongEndDate          # comboBoxnLAType          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=6, column=1)              #輸入框          self.comboBoxnLAType = ttk.Combobox(self, state='readonly')          self.comboBoxnLAType['values'] = Config.comboBoxnLAType          self.comboBoxnLAType.grid(row=6, column=2)          global comboBoxnLAType          comboBoxnLAType = self.comboBoxnLAType          # buttonSendFutureSTPOrderV1          self.buttonSendFutureSTPOrderV1 = tk.Button(self)          self.buttonSendFutureSTPOrderV1["text"] = "期貨STP送出"          self.buttonSendFutureSTPOrderV1["command"] = self.buttonSendFutureSTPOrderV1\_Click          self.buttonSendFutureSTPOrderV1.grid(row=7, column=1)          # buttonSendOptionStopLossOrder          self.buttonSendOptionStopLossOrder = tk.Button(self)          self.buttonSendOptionStopLossOrder["text"] = "選擇權STP送出"          self.buttonSendOptionStopLossOrder["command"] = self.buttonSendOptionStopLossOrder\_Click          self.buttonSendOptionStopLossOrder.grid(row=7, column=2) |

|  |
| --- |
| Config.py |
| # STPForm  comboBoxsNewClose =(      "0:新倉", "1:平倉", "2:自動"  )  comboBoxsBuySell =(      "0:買進", "1:賣出"  )  comboBoxsTradeType =(      "ROD", "IOC", "FOK"  )  comboBoxsDayTrade =(      "0:否", "1:是"  )  comboBoxnOrderPriceType =(      "2: 限價", "3:範圍市價"  )  comboBoxsReserved =(      "0:盤中(T盤及T+1盤)", "1:T盤預約"  )  comboBoxnLongActionFlag =(      "0:否", "1:是"  )  comboBoxnLAType =(      "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 buttonSendFutureSTPOrderV1\_Click(self):            pOrder = sk.FUTUREORDER()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNo.get()          pOrder.bstrSettlementMonth = textBoxbstrSettlementMonth.get()          if (comboBoxsNewClose.get() == "0:新倉"):              pOrder.sNewClose = 0          elif (comboBoxsNewClose.get() == "1:平倉"):              pOrder.sNewClose = 1          else:              pOrder.sNewClose = 2          if (comboBoxsBuySell.get() == "0:買進"):              pOrder.sBuySell = 0          else:              pOrder.sBuySell = 1          if (comboBoxsTradeType.get() == "ROD"):              pOrder.sTradeType = 0          elif (comboBoxsTradeType.get() == "IOC"):              pOrder.sTradeType = 3          else:              pOrder.sTradeType = 4          if (comboBoxsDayTrade.get() == "0:否"):              pOrder.sDayTrade = 0          else:              pOrder.sDayTrade = 1          pOrder.bstrPrice = textBoxbstrPrice.get()          pOrder.nQty = int(textBoxnQty.get())          pOrder.bstrTrigger = textBoxbstrTrigger.get()          if (comboBoxsReserved.get() == "0:盤中(T盤及T+1盤)"):              pOrder.sReserved = 0          else:              pOrder.sReserved = 1            if (comboBoxnOrderPriceType.get() == "2: 限價"):              pOrder.nOrderPriceType = 2          else:              pOrder.nOrderPriceType = 3          if (comboBoxnLongActionFlag.get() == "0:否"):              pOrder.nLongActionFlag = 0          else:              pOrder.nLongActionFlag = 1          pOrder.bstrLongEndDate = textBoxbstrLongEndDate.get()          if (comboBoxnLAType.get() == "1:效期內觸發即失效"):              pOrder.nLAType = 1          else:              pOrder.nLAType = 3          # (指定月份需填商品契約年月)新版—送出期貨停損委託。          bstrMessage,nCode= m\_pSKOrder.SendFutureSTPOrderV1(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendFutureSTPOrderV1】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end')        def buttonSendOptionStopLossOrder\_Click(self):            pOrder = sk.FUTUREORDER()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNo.get()          pOrder.bstrSettlementMonth = textBoxbstrSettlementMonth.get()          if (comboBoxsNewClose.get() == "0:新倉"):              pOrder.sNewClose = 0          elif (comboBoxsNewClose.get() == "1:平倉"):              pOrder.sNewClose = 1          else:              pOrder.sNewClose = 2          if (comboBoxsBuySell.get() == "0:買進"):              pOrder.sBuySell = 0          else:              pOrder.sBuySell = 1          if (comboBoxsTradeType.get() == "ROD"):              pOrder.sTradeType = 0          elif (comboBoxsTradeType.get() == "IOC"):              pOrder.sTradeType = 3          else:              pOrder.sTradeType = 4          if (comboBoxsDayTrade.get() == "0:否"):              pOrder.sDayTrade = 0          else:              pOrder.sDayTrade = 1          pOrder.bstrPrice = textBoxbstrPrice.get()          pOrder.nQty = int(textBoxnQty.get())          pOrder.bstrTrigger = textBoxbstrTrigger.get()          if (comboBoxsReserved.get() == "0:盤中(T盤及T+1盤)"):              pOrder.sReserved = 0          else:              pOrder.sReserved = 1            if (comboBoxnOrderPriceType.get() == "2: 限價"):              pOrder.nOrderPriceType = 2          else:              pOrder.nOrderPriceType = 3          if (comboBoxnLongActionFlag.get() == "0:否"):              pOrder.nLongActionFlag = 0          else:              pOrder.nLongActionFlag = 1          pOrder.bstrLongEndDate = textBoxbstrLongEndDate.get()          if (comboBoxnLAType.get() == "1:效期內觸發即失效"):              pOrder.nLAType = 1          else:              pOrder.nLAType = 3          # 送出選擇權停損委託          bstrMessage,nCode= m\_pSKOrder.SendOptionStopLossOrder(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendOptionStopLossOrder】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

###### SendFutureStopLossOrder

|  |  |  |
| --- | --- | --- |
| ［限近月商品代碼］送出期貨停損委託。 | | |
| **宣告** | Long SendFutureStopLossOrder([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **FUTUREORDER**\* 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[**FUTUREORDER**](#_STP)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、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。**  **其他錯誤則由「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](#_4-2-b_OnAsyncOrder)取得。  需簽署期貨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。  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 | |

###### SendOptionStopLossOrder

|  |  |  |
| --- | --- | --- |
| 送出選擇權停損委託。 | | |
| **宣告** | Long SendOptionStopLossOrder([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **FUTUREORDER**\* 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[**FUTUREORDER**](#_期貨智慧單STP_▼)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、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](#_4-2-b_OnAsyncOrder)取得。  需簽署期貨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。 | |

範例程式碼可參考[SendFutureSTPOrderV1](#_SendFutureSTPOrderV1)做法

##### SendFutureMSTOrderV1

|  |  |  |
| --- | --- | --- |
| (指定月份需填商品契約年月)新版—送出移動停損委託。 | | |
| **宣告** | Long SendFutureMSTOrderV1([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **FUTUREORDER**\* 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[**FUTUREORDER**](#_期貨智慧單MST_▼)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，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](#_4-2-b_OnAsyncOrder)取得。  需簽署期貨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\_TFStrategyOrder.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 = "期權商品代號").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          # comboBoxsNewCloseMST          tk.Label(self, text = "新平倉，0:新倉 1:平倉 2:自動").grid(row=0, column=3)              #輸入框          self.comboBoxsNewCloseMST = ttk.Combobox(self, state='readonly')          self.comboBoxsNewCloseMST['values'] = Config.comboBoxsNewCloseMST          self.comboBoxsNewCloseMST.grid(row=0, column=4)          global comboBoxsNewCloseMST          comboBoxsNewCloseMST = self.comboBoxsNewCloseMST            # comboBoxsBuySellMST          tk.Label(self, text = "0:買進 1:賣出").grid(row=0, column=5)              #輸入框          self.comboBoxsBuySellMST = ttk.Combobox(self, state='readonly')          self.comboBoxsBuySellMST['values'] = Config.comboBoxsBuySellMST          self.comboBoxsBuySellMST.grid(row=0, column=6)          global comboBoxsBuySellMST          comboBoxsBuySellMST = self.comboBoxsBuySellMST          # comboBoxsTradeTypeMST          tk.Label(self, text = "3:IOC 4:FOK").grid(row=0, column=7)              #輸入框          self.comboBoxsTradeTypeMST = ttk.Combobox(self, state='readonly')          self.comboBoxsTradeTypeMST['values'] = Config.comboBoxsTradeTypeMST          self.comboBoxsTradeTypeMST.grid(row=0, column=8)          global comboBoxsTradeTypeMST          comboBoxsTradeTypeMST = self.comboBoxsTradeTypeMST          # comboBoxsDayTradeMST          tk.Label(self, text = "當沖0:否 1:是，可當沖商品請參考交易所規定").grid(row=0, column=9)              #輸入框          self.comboBoxsDayTradeMST = ttk.Combobox(self, state='readonly')          self.comboBoxsDayTradeMST['values'] = Config.comboBoxsDayTradeMST          self.comboBoxsDayTradeMST.grid(row=0, column=10)          global comboBoxsDayTradeMST          comboBoxsDayTradeMST = self.comboBoxsDayTradeMST          # textBoxbstrSettlementMonthMST          tk.Label(self, text = "委託商品年月，YYYYMM共6碼(EX: 202206)").grid(row=1, column=1)          #輸入框          self.textBoxbstrSettlementMonthMST = tk.Entry(self)          self.textBoxbstrSettlementMonthMST.grid(row=1, column=2)          global textBoxbstrSettlementMonthMST          textBoxbstrSettlementMonthMST = self.textBoxbstrSettlementMonthMST          # textBoxbstrMovingPoint          tk.Label(self, text = "移動點數").grid(row=2, column=1)          #輸入框          self.textBoxbstrMovingPoint = tk.Entry(self)          self.textBoxbstrMovingPoint.grid(row=2, column=2)          global textBoxbstrMovingPoint          textBoxbstrMovingPoint = self.textBoxbstrMovingPoint          # textBoxbstrTriggerMST          tk.Label(self, text = "觸價基準").grid(row=2, column=3)          #輸入框          self.textBoxbstrTriggerMST = tk.Entry(self)          self.textBoxbstrTriggerMST.grid(row=2, column=4)          global textBoxbstrTriggerMST          textBoxbstrTriggerMST = self.textBoxbstrTriggerMST          # textBoxnQtyMST          tk.Label(self, text = "交易口數").grid(row=3, column=1)          #輸入框          self.textBoxnQtyMST = tk.Entry(self)          self.textBoxnQtyMST.grid(row=3, column=2)          global textBoxnQtyMST          textBoxnQtyMST = self.textBoxnQtyMST          # comboBoxnOrderPriceTypeMST          tk.Label(self, text = "2: 限價; 3:範圍市價 （不支援市價）").grid(row=3, column=3)              #輸入框          self.comboBoxnOrderPriceTypeMST = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderPriceTypeMST['values'] = Config.comboBoxnOrderPriceTypeMST          self.comboBoxnOrderPriceTypeMST.grid(row=3, column=4)          global comboBoxnOrderPriceTypeMST          comboBoxnOrderPriceTypeMST = self.comboBoxnOrderPriceTypeMST          # comboBoxsReservedMST          tk.Label(self, text = "盤別，0:盤中(T盤及T+1盤)；1:T盤預約").grid(row=4, column=1)              #輸入框          self.comboBoxsReservedMST = ttk.Combobox(self, state='readonly')          self.comboBoxsReservedMST['values'] = Config.comboBoxsReservedMST          self.comboBoxsReservedMST.grid(row=4, column=2)          global comboBoxsReservedMST          comboBoxsReservedMST = self.comboBoxsReservedMST          # buttonSendFutureMSTOrderV1          self.buttonSendFutureMSTOrderV1 = tk.Button(self)          self.buttonSendFutureMSTOrderV1["text"] = "MST送出"          self.buttonSendFutureMSTOrderV1["command"] = self.buttonSendFutureMSTOrderV1\_Click          self.buttonSendFutureMSTOrderV1.grid(row=5, column=1) |

|  |
| --- |
| Config.py |
| # MSTForm  comboBoxsNewCloseMST =(      "0:新倉", "1:平倉", "2:自動"  )  comboBoxsBuySellMST =(      "0:買進", "1:賣出"  )  comboBoxsTradeTypeMST =(      "IOC", "FOK"  )  comboBoxsDayTradeMST =(      "0:否", "1:是"  )  comboBoxnOrderPriceTypeMST =(      "2: 限價", "3:範圍市價"  )  comboBoxsReservedMST =(      "0:盤中(T盤及T+1盤)", "1:T盤預約"  ) |
| 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 buttonSendFutureMSTOrderV1\_Click(self):            pOrder = sk.FUTUREORDER()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNoMST.get()          if (comboBoxsTradeTypeMST.get() == "IOC"):              pOrder.sTradeType = 3          else:              pOrder.sTradeType = 4          if (comboBoxsBuySellMST.get() == "0:買進"):              pOrder.sBuySell = 0          else:              pOrder.sBuySell = 1          if (comboBoxsDayTradeMST.get() == "0:否"):              pOrder.sDayTrade = 0          else:              pOrder.sDayTrade = 1          if (comboBoxsNewCloseMST.get() == "0:新倉"):              pOrder.sNewClose = 0          elif (comboBoxsNewCloseMST.get() == "1:平倉"):              pOrder.sNewClose = 1          else:              pOrder.sNewClose = 2          pOrder.nQty = int(textBoxnQtyMST.get())          pOrder.bstrTrigger = textBoxbstrTriggerMST.get()          if (comboBoxsReservedMST.get() == "0:盤中(T盤及T+1盤)"):              pOrder.sReserved = 0          else:              pOrder.sReserved = 1          pOrder.bstrSettlementMonth = textBoxbstrSettlementMonthMST.get()          if (comboBoxnOrderPriceTypeMST.get() == "2: 限價"):              pOrder.nOrderPriceType = 2          else:              pOrder.nOrderPriceType = 3          pOrder.bstrMovingPoint = textBoxbstrMovingPoint.get()          # (指定月份需填商品契約年月)新版—送出移動停損委託。          bstrMessage,nCode= m\_pSKOrder.SendFutureMSTOrderV1(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendFutureMSTOrderV1】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

###### SendMovingStopLossOrder

|  |  |  |
| --- | --- | --- |
| ［限近月商品代碼］送出移動停損委託。 | | |
| **宣告** | Long SendMovingStopLossOrder([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **FUTUREORDER**\* 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[**FUTUREORDER**](#_期貨智慧單MST_▼)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，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](#_4-2-b_OnAsyncOrder)取得。  需簽署期貨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。 | |

##### SendFutureMITOrderV1

|  |  |  |
| --- | --- | --- |
| (指定月份需填商品契約年月)新版—送出期貨MIT委託。 | | |
| **宣告** | Long SendFutureMITOrderV1([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **FUTUREORDER**\* 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[**FUTUREORDER**](#_期貨、選擇權智慧單MIT_▼)物件，將下單條件填入該物件後，再帶入此欄位中。  （此類型委託中的觸發價與成交價、觸價方向為必要欄位，請務必填入） |
| 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、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](#_4-2-b_OnAsyncOrder)取得。  需簽署期貨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\_TFStrategyOrder.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 = "期權商品代號").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          # comboBoxsNewCloseMIT          tk.Label(self, text = "新平倉，0:新倉 1:平倉 2:自動").grid(row=0, column=3)              #輸入框          self.comboBoxsNewCloseMIT = ttk.Combobox(self, state='readonly')          self.comboBoxsNewCloseMIT['values'] = Config.comboBoxsNewCloseMIT          self.comboBoxsNewCloseMIT.grid(row=0, column=4)          global comboBoxsNewCloseMIT          comboBoxsNewCloseMIT = self.comboBoxsNewCloseMIT            # comboBoxsBuySellMIT          tk.Label(self, text = "0:買進 1:賣出").grid(row=0, column=5)              #輸入框          self.comboBoxsBuySellMIT = ttk.Combobox(self, state='readonly')          self.comboBoxsBuySellMIT['values'] = Config.comboBoxsBuySellMIT          self.comboBoxsBuySellMIT.grid(row=0, column=6)          global comboBoxsBuySellMIT          comboBoxsBuySellMIT = self.comboBoxsBuySellMIT          # comboBoxsTradeTypeMIT          tk.Label(self, text = "0:ROD 3:IOC 4:FOK").grid(row=0, column=7)              #輸入框          self.comboBoxsTradeTypeMIT = ttk.Combobox(self, state='readonly')          self.comboBoxsTradeTypeMIT['values'] = Config.comboBoxsTradeTypeMIT          self.comboBoxsTradeTypeMIT.grid(row=0, column=8)          global comboBoxsTradeTypeMIT          comboBoxsTradeTypeMIT = self.comboBoxsTradeTypeMIT          # comboBoxsDayTradeMIT          tk.Label(self, text = "當沖0:否 1:是，可當沖商品請參考交易所規定").grid(row=0, column=9)              #輸入框          self.comboBoxsDayTradeMIT = ttk.Combobox(self, state='readonly')          self.comboBoxsDayTradeMIT['values'] = Config.comboBoxsDayTradeMIT          self.comboBoxsDayTradeMIT.grid(row=0, column=10)          global comboBoxsDayTradeMIT          comboBoxsDayTradeMIT = self.comboBoxsDayTradeMIT          # textBoxbstrSettlementMonthMIT          tk.Label(self, text = "委託商品年月，YYYYMM共6碼(EX: 202206)").grid(row=1, column=1)          #輸入框          self.textBoxbstrSettlementMonthMIT = tk.Entry(self)          self.textBoxbstrSettlementMonthMIT.grid(row=1, column=2)          global textBoxbstrSettlementMonthMIT          textBoxbstrSettlementMonthMIT = self.textBoxbstrSettlementMonthMIT          # comboBoxnTriggerDirectionMIT          tk.Label(self, text = "觸發方向1:GTE(大於), 2:LTE(小於)").grid(row=2, column=1)              #輸入框          self.comboBoxnTriggerDirectionMIT = ttk.Combobox(self, state='readonly')          self.comboBoxnTriggerDirectionMIT['values'] = Config.comboBoxnTriggerDirectionMIT          self.comboBoxnTriggerDirectionMIT.grid(row=2, column=2)          global comboBoxnTriggerDirectionMIT          comboBoxnTriggerDirectionMIT = self.comboBoxnTriggerDirectionMIT          # textBoxbstrTriggerMIT          tk.Label(self, text = "觸發價").grid(row=2, column=3)          #輸入框          self.textBoxbstrTriggerMIT = tk.Entry(self)          self.textBoxbstrTriggerMIT.grid(row=2, column=4)          global textBoxbstrTriggerMIT          textBoxbstrTriggerMIT = self.textBoxbstrTriggerMIT          # textBoxnQtyMIT          tk.Label(self, text = "交易口數").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 = "2: 限價; 3:範圍市價 （不支援市價）").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          # textBoxbstrPriceMIT          tk.Label(self, text = "委託價格，(指定限價時，需填此欄)").grid(row=2, column=9)          #輸入框          self.textBoxbstrPriceMIT = tk.Entry(self)          self.textBoxbstrPriceMIT.grid(row=2, column=10)          global textBoxbstrPriceMIT          textBoxbstrPriceMIT = self.textBoxbstrPriceMIT          # textBoxbstrDealPriceMIT          tk.Label(self, text = "成交價").grid(row=3, column=1)          #輸入框          self.textBoxbstrDealPriceMIT = tk.Entry(self)          self.textBoxbstrDealPriceMIT.grid(row=3, column=2)          global textBoxbstrDealPriceMIT          textBoxbstrDealPriceMIT = self.textBoxbstrDealPriceMIT          # buttonSendFutureMITOrderV1          self.buttonSendFutureMITOrderV1 = tk.Button(self)          self.buttonSendFutureMITOrderV1["text"] = "期貨MIT送出"          self.buttonSendFutureMITOrderV1["command"] = self.buttonSendFutureMITOrderV1\_Click          self.buttonSendFutureMITOrderV1.grid(row=4, column=1)          # buttonSendOptionMITOrder          self.buttonSendOptionMITOrder = tk.Button(self)          self.buttonSendOptionMITOrder["text"] = "選擇權MIT送出"          self.buttonSendOptionMITOrder["command"] = self.buttonSendOptionMITOrder\_Click          self.buttonSendOptionMITOrder.grid(row=4, column=2) |

|  |
| --- |
| Config.py |
| # MITForm  comboBoxsNewCloseMIT =(      "0:新倉", "1:平倉", "2:自動"  )  comboBoxsBuySellMIT =(      "0:買進", "1:賣出"  )  comboBoxsTradeTypeMIT =(      "ROD", "IOC", "FOK"  )  comboBoxsDayTradeMIT =(      "0:否", "1:是"  )  comboBoxnTriggerDirectionMIT =(      "1:GTE大於等於", "2:LTE小於等於"  )  comboBoxnOrderPriceTypeMIT =(      "2: 限價", "3:範圍市價"  ) |
| 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 buttonSendFutureMITOrderV1\_Click(self):            pOrder = sk.FUTUREORDER()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNoMIT.get()          pOrder.bstrPrice = textBoxbstrPriceMIT.get()          if (comboBoxsTradeTypeMIT.get() == "ROD"):              pOrder.sTradeType = 0          elif (comboBoxsTradeTypeMIT.get() == "IOC"):              pOrder.sTradeType = 3          else:              pOrder.sTradeType = 4          if (comboBoxsBuySellMIT.get() == "0:買進"):              pOrder.sBuySell = 0          else:              pOrder.sBuySell = 1          if (comboBoxsDayTradeMIT.get() == "0:否"):              pOrder.sDayTrade = 0          else:              pOrder.sDayTrade = 1          if (comboBoxsNewCloseMIT.get() == "0:新倉"):              pOrder.sNewClose = 0          elif (comboBoxsNewCloseMIT.get() == "1:平倉"):              pOrder.sNewClose = 1          else:              pOrder.sNewClose = 2          pOrder.nQty = int(textBoxnQtyMIT.get())          pOrder.bstrTrigger = textBoxbstrTriggerMIT.get()          pOrder.bstrSettlementMonth = textBoxbstrSettlementMonthMIT.get()            if (comboBoxnOrderPriceTypeMIT.get() == "2: 限價"):              pOrder.nOrderPriceType = 2          else:              pOrder.nOrderPriceType = 3          pOrder.bstrDealPrice = textBoxbstrDealPriceMIT.get()          if (comboBoxnTriggerDirectionMIT.get() == "1:GTE大於等於"):              pOrder.nTriggerDirection = 1          else:              pOrder.nTriggerDirection = 2          # (指定月份需填商品契約年月)新版—送出期貨MIT委託。          bstrMessage,nCode= m\_pSKOrder.SendFutureMITOrderV1(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendFutureMITOrderV1】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end')      def buttonSendOptionMITOrder\_Click(self):            pOrder = sk.FUTUREORDER()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNoMIT.get()          pOrder.bstrPrice = textBoxbstrPriceMIT.get()          if (comboBoxsTradeTypeMIT.get() == "ROD"):              pOrder.sTradeType = 0          elif (comboBoxsTradeTypeMIT.get() == "IOC"):              pOrder.sTradeType = 3          else:              pOrder.sTradeType = 4          if (comboBoxsBuySellMIT.get() == "0:買進"):              pOrder.sBuySell = 0          else:              pOrder.sBuySell = 1          if (comboBoxsDayTradeMIT.get() == "0:否"):              pOrder.sDayTrade = 0          else:              pOrder.sDayTrade = 1          if (comboBoxsNewCloseMIT.get() == "0:新倉"):              pOrder.sNewClose = 0          elif (comboBoxsNewCloseMIT.get() == "1:平倉"):              pOrder.sNewClose = 1          else:              pOrder.sNewClose = 2          pOrder.nQty = int(textBoxnQtyMIT.get())          pOrder.bstrTrigger = textBoxbstrTriggerMIT.get()          pOrder.bstrSettlementMonth = textBoxbstrSettlementMonthMIT.get()            if (comboBoxnOrderPriceTypeMIT.get() == "2: 限價"):              pOrder.nOrderPriceType = 2          else:              pOrder.nOrderPriceType = 3          pOrder.bstrDealPrice = textBoxbstrDealPriceMIT.get()          if (comboBoxnTriggerDirectionMIT.get() == "1:GTE大於等於"):              pOrder.nTriggerDirection = 1          else:              pOrder.nTriggerDirection = 2          # 送出選擇權MIT委託          bstrMessage,nCode= m\_pSKOrder.SendOptionMITOrder(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendOptionMITOrder】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

###### SendFutureMITOrder

|  |  |  |
| --- | --- | --- |
| ［限近月商品代碼］送出期貨MIT委託。 | | |
| **宣告** | Long SendFutureMITOrder([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **FUTUREORDER**\* 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[**FUTUREORDER**](#_期貨、選擇權智慧單MIT_▼)物件，將下單條件填入該物件後，再帶入此欄位中。  （此類型委託中的觸發價與成交價、觸價方向為必要欄位，請務必填入） |
| 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、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](#_4-2-b_OnAsyncOrder)取得。  需簽署期貨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。 | |

###### SendOptionMITOrder

|  |  |  |
| --- | --- | --- |
| 送出選擇權MIT委託。 | | |
| **宣告** | Long SendOptionMITOrder([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **FUTUREORDER**\* 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[**FUTUREORDER**](#_期貨、選擇權智慧單MIT_▼)物件，將下單條件填入該物件後，再帶入此欄位中。  （此類型委託中的觸發價與成交價、觸價方向為必要欄位，請務必填入） |
| 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、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](#_4-2-b_OnAsyncOrder)取得。  需簽署期貨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。 | |

範例程式碼可參考[SendFutureMITOrderV1](#_SendFutureMITOrderV1)做法

##### SendFutureOCOOrderV1

|  |  |  |
| --- | --- | --- |
| (指定月份需填商品契約年月)新版—送出期貨二擇一委託。 | | |
| **宣告** | Long SendFutureOCOOrderV1([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **FUTUREORDER**\* 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[**FUTUREORDER**](#_期貨智慧單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、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](#_4-2-b_OnAsyncOrder)取得。  需簽署期貨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\_TFStrategyOrder.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 = "期權商品代號").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          # comboBoxsNewCloseOCO          tk.Label(self, text = "新平倉，0:新倉 1:平倉 2:自動").grid(row=0, column=3)              #輸入框          self.comboBoxsNewCloseOCO = ttk.Combobox(self, state='readonly')          self.comboBoxsNewCloseOCO['values'] = Config.comboBoxsNewCloseOCO          self.comboBoxsNewCloseOCO.grid(row=0, column=4)          global comboBoxsNewCloseOCO          comboBoxsNewCloseOCO = self.comboBoxsNewCloseOCO          # textBoxbstrSettlementMonthOCO          tk.Label(self, text = "委託商品年月，YYYYMM共6碼(EX: 202206)").grid(row=1, column=1)          #輸入框          self.textBoxbstrSettlementMonthOCO = tk.Entry(self)          self.textBoxbstrSettlementMonthOCO.grid(row=1, column=2)          global textBoxbstrSettlementMonthOCO          textBoxbstrSettlementMonthOCO = self.textBoxbstrSettlementMonthOCO          # textBoxbstrTriggerOCO          tk.Label(self, text = "第一腳觸發價(當市價大於觸發價1時觸發)").grid(row=2, column=1)          #輸入框          self.textBoxbstrTriggerOCO = tk.Entry(self)          self.textBoxbstrTriggerOCO.grid(row=2, column=2)          global textBoxbstrTriggerOCO          textBoxbstrTriggerOCO = self.textBoxbstrTriggerOCO          # comboBoxsBuySellOCO          tk.Label(self, text = "0:買進 1:賣出").grid(row=2, column=3)              #輸入框          self.comboBoxsBuySellOCO = ttk.Combobox(self, state='readonly')          self.comboBoxsBuySellOCO['values'] = Config.comboBoxsBuySellOCO          self.comboBoxsBuySellOCO.grid(row=2, column=4)          global comboBoxsBuySellOCO          comboBoxsBuySellOCO = self.comboBoxsBuySellOCO          # textBoxbstrTrigger2          tk.Label(self, text = "第二腳觸發價(當市價小於觸發價2時觸發)").grid(row=3, column=1)          #輸入框          self.textBoxbstrTrigger2 = tk.Entry(self)          self.textBoxbstrTrigger2.grid(row=3, column=2)          global textBoxbstrTrigger2          textBoxbstrTrigger2 = self.textBoxbstrTrigger2          # comboBoxsBuySell2OCO          tk.Label(self, text = "0:買進 1:賣出").grid(row=3, column=3)              #輸入框          self.comboBoxsBuySell2OCO = ttk.Combobox(self, state='readonly')          self.comboBoxsBuySell2OCO['values'] = Config.comboBoxsBuySell2OCO          self.comboBoxsBuySell2OCO.grid(row=3, column=4)          global comboBoxsBuySell2OCO          comboBoxsBuySell2OCO = self.comboBoxsBuySell2OCO           # comboBoxsTradeTypeOCO          tk.Label(self, text = "0:ROD 3:IOC 4:FOK").grid(row=3, column=5)              #輸入框          self.comboBoxsTradeTypeOCO = ttk.Combobox(self, state='readonly')          self.comboBoxsTradeTypeOCO['values'] = Config.comboBoxsTradeTypeOCO          self.comboBoxsTradeTypeOCO.grid(row=3, column=6)          global comboBoxsTradeTypeOCO          comboBoxsTradeTypeOCO = self.comboBoxsTradeTypeOCO          # comboBoxnOrderPriceTypeOCO          tk.Label(self, text = "第一腳及第二腳委託價格類型。2限價;3範圍市價（不支援市價）").grid(row=3, column=7)              #輸入框          self.comboBoxnOrderPriceTypeOCO = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderPriceTypeOCO['values'] = Config.comboBoxnOrderPriceTypeOCO          self.comboBoxnOrderPriceTypeOCO.grid(row=3, column=8)          global comboBoxnOrderPriceTypeOCO          comboBoxnOrderPriceTypeOCO = self.comboBoxnOrderPriceTypeOCO          # textBoxbstrPriceOCO          tk.Label(self, text = "第一腳委託價格").grid(row=2, column=9)          #輸入框          self.textBoxbstrPriceOCO = tk.Entry(self)          self.textBoxbstrPriceOCO.grid(row=2, column=10)          global textBoxbstrPriceOCO          textBoxbstrPriceOCO = self.textBoxbstrPriceOCO            # textBoxbstrPrice2          tk.Label(self, text = "第二腳委託價格").grid(row=3, column=9)          #輸入框          self.textBoxbstrPrice2 = tk.Entry(self)          self.textBoxbstrPrice2.grid(row=3, column=10)          global textBoxbstrPrice2          textBoxbstrPrice2 = self.textBoxbstrPrice2            # textBoxnQtyOCO          tk.Label(self, text = "交易口數").grid(row=4, column=1)          #輸入框          self.textBoxnQtyOCO = tk.Entry(self)          self.textBoxnQtyOCO.grid(row=4, column=2)          global textBoxnQtyOCO          textBoxnQtyOCO = self.textBoxnQtyOCO          # comboBoxsDayTradeOCO          tk.Label(self, text = "當沖0:否 1:是，可當沖商品請參考交易所規定").grid(row=4, column=3)              #輸入框          self.comboBoxsDayTradeOCO = ttk.Combobox(self, state='readonly')          self.comboBoxsDayTradeOCO['values'] = Config.comboBoxsDayTradeOCO          self.comboBoxsDayTradeOCO.grid(row=4, column=4)          global comboBoxsDayTradeOCO          comboBoxsDayTradeOCO = self.comboBoxsDayTradeOCO          # comboBoxnLongActionFlagOCO          tk.Label(self, text = "是否為長效單(0:否, 1:是)").grid(row=5, column=1)              #輸入框          self.comboBoxnLongActionFlagOCO = ttk.Combobox(self, state='readonly')          self.comboBoxnLongActionFlagOCO['values'] = Config.comboBoxnLongActionFlagOCO          self.comboBoxnLongActionFlagOCO.grid(row=5, column=2)          global comboBoxnLongActionFlagOCO          comboBoxnLongActionFlagOCO = self.comboBoxnLongActionFlagOCO          # textBoxbstrLongEndDateOCO          tk.Label(self, text = "長效單結束日期(YYYYMMDD共8碼, EX: 20240205)").grid(row=5, column=3)          #輸入框          self.textBoxbstrLongEndDateOCO = tk.Entry(self)          self.textBoxbstrLongEndDateOCO.grid(row=5, column=4)          global textBoxbstrLongEndDateOCO          textBoxbstrLongEndDateOCO = self.textBoxbstrLongEndDateOCO          # comboBoxnLATypeOCO          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=6, column=1)              #輸入框          self.comboBoxnLATypeOCO = ttk.Combobox(self, state='readonly')          self.comboBoxnLATypeOCO['values'] = Config.comboBoxnLATypeOCO          self.comboBoxnLATypeOCO.grid(row=6, column=2)          global comboBoxnLATypeOCO          comboBoxnLATypeOCO = self.comboBoxnLATypeOCO          # comboBoxsReservedOCO          tk.Label(self, text = "盤別，0:盤中(T盤及T+1盤)；1:T盤預約/是否為預約單(0:否, 1:是)").grid(row=7, column=1)              #輸入框          self.comboBoxsReservedOCO = ttk.Combobox(self, state='readonly')          self.comboBoxsReservedOCO['values'] = Config.comboBoxsReservedOCO          self.comboBoxsReservedOCO.grid(row=7, column=2)          global comboBoxsReservedOCO          comboBoxsReservedOCO = self.comboBoxsReservedOCO          # buttonSendFutureOCOOrderV1          self.buttonSendFutureOCOOrderV1 = tk.Button(self)          self.buttonSendFutureOCOOrderV1["text"] = "OCO送出"          self.buttonSendFutureOCOOrderV1["command"] = self.buttonSendFutureOCOOrderV1\_Click          self.buttonSendFutureOCOOrderV1.grid(row=8, column=1) |

|  |
| --- |
| Config.py |
| # OCOForm  comboBoxsNewCloseOCO =(      "0:新倉", "1:平倉", "2:自動"  )  comboBoxsBuySellOCO =(      "0:買進", "1:賣出"  )  comboBoxsBuySell2OCO =(      "0:買進", "1:賣出"  )  comboBoxsTradeTypeOCO =(      "ROD", "IOC", "FOK"  )  comboBoxnOrderPriceTypeOCO =(      "2: 限價", "3:範圍市價"  )  comboBoxsDayTradeOCO =(      "0:否", "1:是"  )  comboBoxnLongActionFlagOCO =(      "0:否", "1:是"  )  comboBoxnLATypeOCO =(      "1:效期內觸發即失效", "3:效期內完全成交即失效"  )  comboBoxsReservedOCO =(      "0:盤中(T盤及T+1盤)", "1:T盤預約"  ) |
| 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 buttonSendFutureOCOOrderV1\_Click(self):            pOrder = sk.FUTUREORDER()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNoOCO.get()          if (comboBoxsTradeTypeOCO.get() == "ROD"):              pOrder.sTradeType = 0          elif (comboBoxsTradeTypeOCO.get() == "IOC"):              pOrder.sTradeType = 3          else:              pOrder.sTradeType = 4          if (comboBoxsBuySellOCO.get() == "0:買進"):              pOrder.sBuySell = 0          else:              pOrder.sBuySell = 1          if (comboBoxsBuySell2OCO.get() == "0:買進"):              pOrder.sBuySell2 = 0          else:              pOrder.sBuySell2 = 1          if (comboBoxsDayTradeOCO.get() == "0:否"):              pOrder.sDayTrade = 0          else:              pOrder.sDayTrade = 1          if (comboBoxsNewCloseOCO.get() == "0:新倉"):              pOrder.sNewClose = 0          elif (comboBoxsNewCloseOCO.get() == "1:平倉"):              pOrder.sNewClose = 1          else:              pOrder.sNewClose = 2          pOrder.nQty = int(textBoxnQtyOCO.get())          pOrder.bstrTrigger = textBoxbstrTriggerOCO.get()          if (comboBoxsReservedOCO.get() == "0:盤中(T盤及T+1盤)"):              pOrder.sReserved = 0          else:              pOrder.sReserved = 1          pOrder.bstrSettlementMonth = textBoxbstrSettlementMonthOCO.get()            if (comboBoxnOrderPriceTypeOCO.get() == "2: 限價"):              pOrder.nOrderPriceType = 2          else:              pOrder.nOrderPriceType = 3          if (comboBoxnLongActionFlagOCO.get() == "0:否"):              pOrder.nLongActionFlag = 0          else:              pOrder.nLongActionFlag = 1          pOrder.bstrLongEndDate = textBoxbstrLongEndDateOCO.get()          if (comboBoxnLATypeOCO.get() == "1:效期內觸發即失效"):              pOrder.nLAType = 1          else:              pOrder.nLAType = 3          pOrder.bstrPrice = textBoxbstrPriceOCO.get()          pOrder.bstrPrice2 = textBoxbstrPrice2.get()          pOrder.bstrTrigger2 = textBoxbstrTrigger2.get()          pOrder.nTimeFlag = 1          # (指定月份需填商品契約年月)新版—送出期貨二擇一委託。          bstrMessage,nCode= m\_pSKOrder.SendFutureOCOOrderV1(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendFutureOCOOrderV1】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

###### SendFutureOCOOrder

|  |  |  |
| --- | --- | --- |
| ［限近月商品代碼］送出期貨二擇一委託。 | | |
| **宣告** | Long SendFutureOCOOrder([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **FUTUREOCOORDER**\* 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[**FUTUREOCOORDER**](#_FUTUREOCOORDER_(二擇一OCO下單物件_))物件，將下單條件填入該物件後，再帶入此欄位中。 |
| bstrMessage | **同步委託：**如果回傳值為 0表示委託成功，訊息內容則為委託日期、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。**  **其他錯誤則由「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](#_4-2-b_OnAsyncOrder)取得。  需簽署期貨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。 | |

##### SendFutureABOrder

|  |  |  |
| --- | --- | --- |
| (指定月份需填商品契約年月)新版—送出期貨看A下B委託。 | | |
| **宣告** | Long SendFutureABOrder ([in] BSTR bstrLogInID, [in] VARIANT\_BOOL bAsyncOrder, [in] struct **FUTUREORDER**\* 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[**FUTUREORDER**](#_期貨智慧單AB_▼)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、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](#_4-2-b_OnAsyncOrder)取得。  需簽署期貨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。  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\_TFStrategyOrder.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          # textBoxbstrCIDTandem          tk.Label(self, text = "交易所代碼(EX: TSE、TAIFEX、CME)").grid(row=0, column=5)          #輸入框          self.textBoxbstrCIDTandem = tk.Entry(self)          self.textBoxbstrCIDTandem.grid(row=0, column=6)          global textBoxbstrCIDTandem          textBoxbstrCIDTandem = self.textBoxbstrCIDTandem          # comboBoxnTimeFlag          tk.Label(self, text = "0:非證券,1:上市, 2:上櫃").grid(row=0, column=7)              #輸入框          self.comboBoxnTimeFlag = ttk.Combobox(self, state='readonly')          self.comboBoxnTimeFlag['values'] = Config.comboBoxnTimeFlag          self.comboBoxnTimeFlag.grid(row=0, column=8)          global comboBoxnTimeFlag          comboBoxnTimeFlag = self.comboBoxnTimeFlag          # comboBoxnTriggerDirectionAB          tk.Label(self, text = "觸發方向1:GTE(大於), 2:LTE(小於)").grid(row=1, column=1)              #輸入框          self.comboBoxnTriggerDirectionAB = ttk.Combobox(self, state='readonly')          self.comboBoxnTriggerDirectionAB['values'] = Config.comboBoxnTriggerDirectionAB          self.comboBoxnTriggerDirectionAB.grid(row=1, column=2)          global comboBoxnTriggerDirectionAB          comboBoxnTriggerDirectionAB = self.comboBoxnTriggerDirectionAB            # textBoxbstrTriggerAB          tk.Label(self, text = "觸發價").grid(row=1, column=3)          #輸入框          self.textBoxbstrTriggerAB = tk.Entry(self)          self.textBoxbstrTriggerAB.grid(row=1, column=4)          global textBoxbstrTriggerAB          textBoxbstrTriggerAB = self.textBoxbstrTriggerAB            # textBoxbstrDealPriceAB          tk.Label(self, text = "成交價").grid(row=1, column=5)          #輸入框          self.textBoxbstrDealPriceAB = tk.Entry(self)          self.textBoxbstrDealPriceAB.grid(row=1, column=6)          global textBoxbstrDealPriceAB          textBoxbstrDealPriceAB = self.textBoxbstrDealPriceAB          # comboBoxsBuySellAB          tk.Label(self, text = "0:買進 1:賣出").grid(row=3, column=1)              #輸入框          self.comboBoxsBuySellAB = ttk.Combobox(self, state='readonly')          self.comboBoxsBuySellAB['values'] = Config.comboBoxsBuySellAB          self.comboBoxsBuySellAB.grid(row=3, column=2)          global comboBoxsBuySellAB          comboBoxsBuySellAB = self.comboBoxsBuySellAB          # comboBoxsDayTradeAB          tk.Label(self, text = "當沖0:否 1:是，可當沖商品請參考交易所規定").grid(row=3, column=3)              #輸入框          self.comboBoxsDayTradeAB = ttk.Combobox(self, state='readonly')          self.comboBoxsDayTradeAB['values'] = Config.comboBoxsDayTradeAB          self.comboBoxsDayTradeAB.grid(row=3, column=4)          global comboBoxsDayTradeAB          comboBoxsDayTradeAB = self.comboBoxsDayTradeAB            # comboBoxsNewCloseAB          tk.Label(self, text = "新平倉，0:新倉 1:平倉 2:自動").grid(row=3, column=5)              #輸入框          self.comboBoxsNewCloseAB = ttk.Combobox(self, state='readonly')          self.comboBoxsNewCloseAB['values'] = Config.comboBoxsNewCloseAB          self.comboBoxsNewCloseAB.grid(row=3, column=6)          global comboBoxsNewCloseAB          comboBoxsNewCloseAB = self.comboBoxsNewCloseAB          # comboBoxsTradeTypeAB          tk.Label(self, text = "0:ROD 3:IOC 4:FOK").grid(row=3, column=7)              #輸入框          self.comboBoxsTradeTypeAB = ttk.Combobox(self, state='readonly')          self.comboBoxsTradeTypeAB['values'] = Config.comboBoxsTradeTypeAB          self.comboBoxsTradeTypeAB.grid(row=3, column=8)          global comboBoxsTradeTypeAB          comboBoxsTradeTypeAB = self.comboBoxsTradeTypeAB          # comboBoxsBuySell2AB          tk.Label(self, text = "買賣別2(0:買進, 1:賣出)，非價差商品請填0").grid(row=4, column=1)              #輸入框          self.comboBoxsBuySell2AB = ttk.Combobox(self, state='readonly')          self.comboBoxsBuySell2AB['values'] = Config.comboBoxsBuySell2AB          self.comboBoxsBuySell2AB.grid(row=4, column=2)          global comboBoxsBuySell2AB          comboBoxsBuySell2AB = self.comboBoxsBuySell2AB          # textBoxbstrStockNoAB          tk.Label(self, text = "期權商品代號").grid(row=5, column=1)          #輸入框          self.textBoxbstrStockNoAB = tk.Entry(self)          self.textBoxbstrStockNoAB.grid(row=5, column=2)          global textBoxbstrStockNoAB          textBoxbstrStockNoAB = self.textBoxbstrStockNoAB          # comboBoxnCallPut          tk.Label(self, text = "是否為選擇權(0:否, 1:Call, 2:Put)").grid(row=5, column=3)              #輸入框          self.comboBoxnCallPut = ttk.Combobox(self, state='readonly')          self.comboBoxnCallPut['values'] = Config.comboBoxnCallPut          self.comboBoxnCallPut.grid(row=5, column=4)          global comboBoxnCallPut          comboBoxnCallPut = self.comboBoxnCallPut            # textBoxbstrStrikePrice          tk.Label(self, text = "履約價(非選擇權商品請填0)").grid(row=5, column=5)          #輸入框          self.textBoxbstrStrikePrice = tk.Entry(self)          self.textBoxbstrStrikePrice.grid(row=5, column=6)          global textBoxbstrStrikePrice          textBoxbstrStrikePrice = self.textBoxbstrStrikePrice          # comboBoxnFlag          tk.Label(self, text = "是否委託價差商品(0:否, 1:是)").grid(row=5, column=7)              #輸入框          self.comboBoxnFlag = ttk.Combobox(self, state='readonly')          self.comboBoxnFlag['values'] = Config.comboBoxnFlag          self.comboBoxnFlag.grid(row=5, column=8)          global comboBoxnFlag          comboBoxnFlag = self.comboBoxnFlag          # textBoxbstrSettlementMonthAB          tk.Label(self, text = "委託商品年月，YYYYMM共6碼(EX: 202206)").grid(row=6, column=1)          #輸入框          self.textBoxbstrSettlementMonthAB = tk.Entry(self)          self.textBoxbstrSettlementMonthAB.grid(row=6, column=2)          global textBoxbstrSettlementMonthAB          textBoxbstrSettlementMonthAB = self.textBoxbstrSettlementMonthAB            # textBoxbstrSettlementMonth2          tk.Label(self, text = "商品契約月份2(YYYYMM共6碼, EX: 202206)，非價差商品請填0").grid(row=6, column=3)          #輸入框          self.textBoxbstrSettlementMonth2 = tk.Entry(self)          self.textBoxbstrSettlementMonth2.grid(row=6, column=4)          global textBoxbstrSettlementMonth2          textBoxbstrSettlementMonth2 = self.textBoxbstrSettlementMonth2            # textBoxnQtyAB          tk.Label(self, text = "交易口數").grid(row=6, column=5)          #輸入框          self.textBoxnQtyAB = tk.Entry(self)          self.textBoxnQtyAB.grid(row=6, column=6)          global textBoxnQtyAB          textBoxnQtyAB = self.textBoxnQtyAB          # comboBoxnOrderPriceTypeAB          tk.Label(self, text = "2: 限價; 3:範圍市價 （不支援市價）").grid(row=6, column=7)              #輸入框          self.comboBoxnOrderPriceTypeAB = ttk.Combobox(self, state='readonly')          self.comboBoxnOrderPriceTypeAB['values'] = Config.comboBoxnOrderPriceTypeAB          self.comboBoxnOrderPriceTypeAB.grid(row=6, column=8)          global comboBoxnOrderPriceTypeAB          comboBoxnOrderPriceTypeAB = self.comboBoxnOrderPriceTypeAB          # textBoxbstrPriceAB          tk.Label(self, text = "委託價格").grid(row=6, column=9)          #輸入框          self.textBoxbstrPriceAB = tk.Entry(self)          self.textBoxbstrPriceAB.grid(row=6, column=10)          global textBoxbstrPriceAB          textBoxbstrPriceAB = self.textBoxbstrPriceAB            # comboBoxsReservedAB          tk.Label(self, text = "盤別，0:盤中(T盤及T+1盤)；1:T盤預約/是否為預約單(0:否, 1:是)").grid(row=7, column=1)              #輸入框          self.comboBoxsReservedAB = ttk.Combobox(self, state='readonly')          self.comboBoxsReservedAB['values'] = Config.comboBoxsReservedAB          self.comboBoxsReservedAB.grid(row=7, column=2)          global comboBoxsReservedAB          comboBoxsReservedAB = self.comboBoxsReservedAB          # buttonSendFutureABOrder          self.buttonSendFutureABOrder = tk.Button(self)          self.buttonSendFutureABOrder["text"] = "AB送出"          self.buttonSendFutureABOrder["command"] = self.buttonSendFutureABOrder\_Click          self.buttonSendFutureABOrder.grid(row=8, column=1) |

|  |
| --- |
| Config.py |
| # ABForm  comboBoxnMarketNo =(      "1:國內證", "2:國內期", "3:國外證", "4:國外期"  )  comboBoxnTimeFlag =(      "1:上市", "2:上櫃(不開放權證、興櫃商品)", "非證券商品請填0"  )  comboBoxnTriggerDirectionAB =(      "1:GTE大於等於", "2:LTE小於等於"  )  comboBoxsBuySellAB =(      "0:買進", "1:賣出"  )  comboBoxsDayTradeAB =(      "0:否", "1:是"  )  comboBoxsNewCloseAB =(      "0:新倉", "1:平倉", "2:自動"  )  comboBoxsTradeTypeAB =(      "ROD", "IOC", "FOK"  )  comboBoxsBuySell2AB =(      "0:買進", "1:賣出"  )  comboBoxnCallPut =(      "0:否", "1:Call", "2:Put"  )  comboBoxnFlag =(      "0:否", "1:是"  )  comboBoxnOrderPriceTypeAB =(      "2: 限價", "3:範圍市價"  )  comboBoxsReservedAB =(      "0:盤中(T盤及T+1盤)", "1:T盤預約"  ) |
| 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 buttonSendFutureABOrder\_Click(self):            pOrder = sk.FUTUREORDER()          pOrder.bstrFullAccount = comboBoxAccount.get()          pOrder.bstrStockNo = textBoxbstrStockNoAB.get()          pOrder.bstrStockNo2 = textBoxbstrStockNo2.get()          pOrder.bstrDealPrice = textBoxbstrDealPriceAB.get()          if (comboBoxnTriggerDirectionAB.get() == "1:GTE大於等於"):              pOrder.nTriggerDirection = 1          else:              pOrder.nTriggerDirection = 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          else:              pOrder.nMarketNo = 4          pOrder.bstrCIDTandem = textBoxbstrCIDTandem.get()          if (comboBoxnTimeFlag.get() == "1:上市"):              pOrder.nTimeFlag = 1          elif(comboBoxnTimeFlag.get() == "2:上櫃(不開放權證、興櫃商品)"):              pOrder.nTimeFlag = 2          else:              pOrder.nTimeFlag = 0          if (comboBoxnCallPut.get() == "0:否"):              pOrder.nCallPut = 0          elif (comboBoxnCallPut.get() == "1:Call"):              pOrder.nCallPut = 1          else:              pOrder.nCallPut = 2          pOrder.bstrStrikePrice = textBoxbstrStrikePrice.get()          if (comboBoxnFlag.get() == "0:否"):              pOrder.nFlag = 0          else:              pOrder.nFlag = 1          pOrder.bstrSettlementMonth2 = textBoxbstrSettlementMonth2.get()          if (comboBoxsTradeTypeAB.get() == "ROD"):              pOrder.sTradeType = 0          elif (comboBoxsTradeTypeAB.get() == "IOC"):              pOrder.sTradeType = 3          else:              pOrder.sTradeType = 4          if (comboBoxsBuySellAB.get() == "0:買進"):              pOrder.sBuySell = 0          else:              pOrder.sBuySell = 1          if (comboBoxsBuySell2AB.get() == "0:買進"):              pOrder.sBuySell2 = 0          else:              pOrder.sBuySell2 = 1          if (comboBoxsDayTradeAB.get() == "0:否"):              pOrder.sDayTrade = 0          else:              pOrder.sDayTrade = 1          if (comboBoxsNewCloseAB.get() == "0:新倉"):              pOrder.sNewClose = 0          elif(comboBoxsNewCloseAB.get() == "1:平倉"):              pOrder.sNewClose = 1          else:              pOrder.sNewClose = 2          pOrder.bstrPrice = textBoxbstrPriceAB.get()          pOrder.nQty = int(textBoxnQtyAB.get())          pOrder.bstrTrigger = textBoxbstrTriggerAB.get()          if (comboBoxsReservedAB.get() == "0:盤中(T盤及T+1盤)"):              pOrder.sReserved = 0          else:              pOrder.sReserved = 1          pOrder.bstrSettlementMonth = textBoxbstrSettlementMonthAB.get()            if (comboBoxnOrderPriceTypeAB.get() == "2: 限價"):              pOrder.nOrderPriceType = 2          else:              pOrder.nOrderPriceType = 3          # (指定月份需填商品契約年月)新版—送出期貨看A下B委託。          bstrMessage,nCode= m\_pSKOrder.SendFutureABOrder(comboBoxUserID.get(), bAsyncOrder, pOrder)          msg = "【SendFutureABOrder】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode) + bstrMessage          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

##### CancelTFStrategyOrderV1

|  |  |  |
| --- | --- | --- |
| 新版—取消期貨智慧單委託。  已產生書號之委託，請填入書號，否則可能影響解除保證金風控。 | | |
| **宣告** | Long CancelTFStrategyOrderV1 ( [in] struct CANCELSTRATEGYORDER\* pCancelOrder, [**in**] VARIANT\_BOOL bAsyncOrder, [**out**] BSTR\* bstrMessage); | |
|  | pCancelOrder | 智慧單刪單物件(參考[CANCELSTRATEGYORDER](#_CANCELSTRATEGYORDER(智慧單刪單物件_)))  (委託成功可取得完整刪單訊息：智慧單號 、13碼序號、 書號) |
| bAsyncOrder | 是否為非同步委託。 |
| 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](#_4-2-b_OnAsyncOrder)取得。  需簽署**期貨**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\_TFStrategyOrder.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          # comboBoxnTradeKind          tk.Label(self, text = "3:OCO、5:STP、8:MIT、9:MST、10：AB").grid(row=1, column=1)              #輸入框          self.comboBoxnTradeKind = ttk.Combobox(self, state='readonly')          self.comboBoxnTradeKind['values'] = Config.comboBoxnTradeKind          self.comboBoxnTradeKind.grid(row=1, column=2)          global comboBoxnTradeKind          comboBoxnTradeKind = self.comboBoxnTradeKind            # comboBoxnMarket          tk.Label(self, text = "市場別(AB單需選欲刪單之A商品市場)").grid(row=2, column=1)              #輸入框          self.comboBoxnMarket = ttk.Combobox(self, state='readonly')          self.comboBoxnMarket['values'] = Config.comboBoxnMarket          self.comboBoxnMarket.grid(row=2, column=2)          global comboBoxnMarket          comboBoxnMarket = self.comboBoxnMarket            # textBoxbstrSeqNo          tk.Label(self, text = "委託序號 (預約單可忽略)").grid(row=3, column=1)          #輸入框          self.textBoxbstrSeqNo = tk.Entry(self)          self.textBoxbstrSeqNo.grid(row=3, 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=4, column=1)          #輸入框          self.textBoxbstrOrderNo = tk.Entry(self)          self.textBoxbstrOrderNo.grid(row=4, 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=5, column=1)          #輸入框          self.textBoxbstrLongActionKey = tk.Entry(self)          self.textBoxbstrLongActionKey.grid(row=5, column=2)          global textBoxbstrLongActionKey          textBoxbstrLongActionKey = self.textBoxbstrLongActionKey          # buttonCancelTFStrategyOrderV1          self.buttonCancelTFStrategyOrderV1 = tk.Button(self)          self.buttonCancelTFStrategyOrderV1["text"] = "新版刪單送出"          self.buttonCancelTFStrategyOrderV1["command"] = self.buttonCancelTFStrategyOrderV1\_Click          self.buttonCancelTFStrategyOrderV1.grid(row=6, column=2) |

|  |
| --- |
| Config.py |
| # CancelForm  comboBoxnTradeKind =(      "3:OCO", "5:STP", "8:MIT", "9:MST", "10：AB"  )  comboBoxnMarket =(      "1：國內證", "2：國內期", "3：國外證", "4：國外期"  ) |
| 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 buttonCancelTFStrategyOrderV1\_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()          if (comboBoxnTradeKind.get() == "3:OCO"):              pCancelOrder.nTradeKind = 3          elif (comboBoxnTradeKind.get() == "5:STP"):              pCancelOrder.nTradeKind = 5          elif (comboBoxnTradeKind.get() == "8:MIT"):              pCancelOrder.nTradeKind = 8          elif (comboBoxnTradeKind.get() == "9:MST"):              pCancelOrder.nTradeKind = 9          else:              pCancelOrder.nTradeKind = 10          pCancelOrder.bstrSeqNo = textBoxbstrSeqNo.get()          pCancelOrder.bstrOrderNo = textBoxbstrOrderNo.get()          pCancelOrder.bstrLongActionKey = textBoxbstrLongActionKey.get()          # 新版—取消期貨智慧單委託。已產生書號之委託，請填入書號，否則可能影響解除保證金風控。          bstrMessage,nCode= m\_pSKOrder.CancelTFStrategyOrderV1(pCancelOrder, bAsyncOrder)          msg = "【CancelTFStrategyOrderV1】" + 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](#_4-2-b_OnAsyncOrder)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\_TFStrategyOrder.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') |

##### GetStopLossReport

|  |  |  |
| --- | --- | --- |
| 新版期貨停損委託單查詢。 | | |
| **宣告** | Long GetStopLossReport([in] BSTR bstrLogInID, [in] BSTR bstrAccount, [in] LONG nReportStatus, [in] BSTR bstrKind, [in] BSTR bstrDate); | |
| **參數** | bstrLogInID | 登入ID。 |
| bstrAccount | 委託帳號 (帳號) 。 |
| nReportStatus | 0:全部的委託單。 |
| bstrKind | 智慧單類型  STP:一般停損（含選擇權停損）(長效單)  MST:移動停損  OCO:二擇一(長效單)  MIT(含選擇權MIT)  AB：AB單 |
| bstrDate | 查詢日期(ex:20220601) |
| **回傳值** | 0表示成功，其餘非0數值都表示失敗。 | |
| **備註** | 結果由[OnStopLossReport](#_OnStopLossReport)事件回傳。  需簽署期貨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。  \*V2.13.38版本起配合智慧單平台異動至新版本 | |

|  |
| --- |
| Python範例程式碼\_TFStrategyOrder.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          # buttonGetOFSmartStrategyReport          self.buttonGetOFSmartStrategyReport = tk.Button(self)          self.buttonGetOFSmartStrategyReport["text"] = "查詢送出"          self.buttonGetOFSmartStrategyReport["command"] = self.buttonGetOFSmartStrategyReport\_Click          self.buttonGetOFSmartStrategyReport.grid(row=1, column=3) |

|  |
| --- |
| Config.py |
| # GetForm  comboBoxbstrKind =(      "STP:一般停損（含選擇權停損）(長效單)", "MST:移動停損", "OCO:二擇一(長效單)", "MIT(含選擇權MIT)", "AB：AB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)  # 是否為非同步委託  bAsyncOrder = False |
| Current Block(放置於def createWidgets(self): 的同一層) |
| def buttonGetOFSmartStrategyReport\_Click(self):          if (comboBoxbstrKind.get() == "STP:一般停損（含選擇權停損）(長效單)"):              bstrKind = "STP"          elif (comboBoxbstrKind.get() == "MST:移動停損"):              bstrKind = "MST"          elif (comboBoxbstrKind.get() == "OCO:二擇一(長效單)"):              bstrKind = "OCO"          elif (comboBoxbstrKind.get() == "MIT(含選擇權MIT)"):              bstrKind = "MIT"          else:              bstrKind = "AB"          # 新版期貨停損委託單查詢          nCode= m\_pSKOrder.GetStopLossReport(comboBoxUserID.get(), comboBoxAccount.get(), 0, bstrKind, textBoxbstrDate.get())          msg = "【GetStopLossReport】" + m\_pSKCenter.SKCenterLib\_GetReturnCodeMessage(nCode)          richTextBoxMethodMessage.insert('end',  msg + "\n")          richTextBoxMethodMessage.see('end') |

# 事件

## 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\_TFStrategyOrder.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') |

## OnStopLossReport

### STP

|  |  |  |
| --- | --- | --- |
| 新版期貨智慧單(包含停損單、移動停損、二擇一、觸價單)被動回報查詢。  透過呼叫[GetStopLossReport](#_GetStopLossReport)後，資訊由該事件回傳。 | | |
| **宣告** | | void OnStopLossReport([in] BSTR bstrData); |
| **參數** | | bstrData |
|  | | ＊注意：新版本回傳內容及欄位與舊版本相異。  如果回傳的第一筆資料是一位英文M接連著三碼數字及停損單總數，其中 "M000" 開頭，表示為成功回傳，其後提供欲查詢之停損單總數。  Ex:  (成功)M000,12  第二筆後面接的就是訊息內容。回傳的字串內容，以「，」分隔每一個欄位，欄位依序為： |
| STP:   |  |  | | --- | --- | | 名稱 | 備註 | | 智慧單號 |  | | 市場別 | TF：國內期選 | | 交易所代碼 | TAIFEX | | 分公司代碼 |  | | 帳號 |  | | 子帳 |  | | 委託書號 |  | | 序號 | (13碼)已觸發時，成交單含IOC、 FOK產生的取消單，可以以此欄比對市場上回報資料 | | 原始序號 | (13碼) | | 策略別 | 3：OCO二擇一  5：STP停損單  8：MIT觸價單  9：MST移動停損 | | 智慧單委託狀態 | 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 | | 交易日 |  | | 下單時間 | 建立智慧單時間 | | 商品代碼 | 第一腳商品代碼 | | 商品契約月份 | 第一腳商品年月 | | 履約價格 | 若無則為0 | | 買賣別 | B：買 ;S：賣 | | 委託價格別 | 第一委託價格別  7:使用者輸入價 | | 委託價格類別 | 2：限價 ;3：範圍市價 | | 委託時效 | 0： ROD當日有效 ;3： IOC立即成交; 4： FOK立即全部成交 | | 委託價格 |  | | 委託數量 |  | | 觸發價格 |  | | 觸發時間 |  | | 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  訊息中逗號皆為全形”，” | | 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  F:【交易所動態退單相關代碼】 | | 智慧單異動時間 | 智慧單異動時間 | | 預約單註記 | (MIT不提供)  0:盤中單  1:預約單 | | 預約單序號 | (MIT不提供)  當預約單註記為時1 | | 交易日 |  | | 第一腳下單商品代碼 | 後台下單代碼 EX：FITX(MST 、OCO 不提供) | | 第二腳下單商品代碼 | 後台下單代碼(MST 、OCO 不提供) | | 第二腳商品契約月份 |  | | 第二腳履約價格 |  | | 新平倉 | 倉位 0：新倉; 1：平倉 ; 2：自動 | | 是否為當沖 | 空:非當沖； Y:當沖 | | 買賣權 | 買賣權 N：None ;C：Call ; P：Put | | 是否觸發即停止 |  | | 長效單註記 | 0：非長效單  1：長效單 | | 長效單序號 | 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 | | 長效單結束日期 | 20230829 | | 長效單結束條件 | 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 | | 長效單委託總量 |  | | 長效單成交總量 |  | | | |
| **備註** | 當全部資料已經全部回傳完畢，將回傳一筆以「##」開頭的內容，表示查詢結束。 \*V2.13.38版本起配合智慧單平台異動至新版格式 | |

### MST

|  |  |  |
| --- | --- | --- |
| 新版期貨智慧單(包含停損單、移動停損、二擇一、觸價單)查詢。透過呼叫 [GetStopLossReport](#_GetStopLossReport)後，資訊由該事件回傳。 | | |
| **宣告** | | void OnStopLossReport( [in] BSTR bstrData ); |
| **參數** | | bstrData |
|  | | ＊注意：新版本回傳內容及欄位與舊版本相異。  如果回傳的第一筆資料是一位英文M接連著三碼數字及停損單總數，其中 "M000" 開頭，表示為成功回傳，其後提供欲查詢之停損單總數。  Ex:  (成功)M000,12  第二筆後面接的就是訊息內容。回傳的字串內容，以「,」分隔每一個欄位，欄位依序為： |
| 回傳的內容以「,」分隔每一個欄位，欄位依序為：  MST:   |  |  | | --- | --- | | 名稱 | 備註 | | 智慧單號 |  | | 市場別 | TF：國內期選 | | 交易所代碼 | TAIFEX | | 分公司代碼 |  | | 帳號 |  | | 子帳 |  | | 委託書號 |  | | 序號 | (13碼)已觸發時，成交單含IOC、 FOK產生的取消單，可以以此欄比對市場上回報資料 | | 原始序號 | (13碼) | | 策略別 | 3：OCO單  5：停損單  8：MIT單  9：MST停損 | | 智慧單委託狀態 | 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 | | 交易日 |  | | 下單時間 | 建立智慧單時間 | | 商品代碼 | 第一腳商品代碼 | | 商品契約月份 | 第一腳商品年月 | | 履約價格 | 若無則為0 | | 買賣別 | B：買 ;S：賣 | | 委託價格別 | 第一委託價格別  7:使用者輸入價 | | 委託價格類別 | 2：限價 ;3：範圍市價 | | 委託時效 | 0： ROD當日有效 ;3： IOC立即成交; 4： FOK立即全部成交 | | 委託價格 |  | | 委託數量 |  | | 觸發價格 |  | | 觸發時間 |  | | 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  訊息中逗號皆為全形”，” | | 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  F:【交易所動態退單相關代碼】 | | 智慧單異動時間 | 智慧單異動時間 | | 預約單註記 | 0:盤中單;1:預約單 | | 預約單序號 | 當預約單註記為時1 | | 交易日 |  | | 第一腳下單商品代碼 | (MST 、OCO 不提供)後台下單代碼 EX：FITX | | 第二腳下單商品代碼 | (MST 、OCO 不提供)後台下單代碼 | | 第二腳商品契約月份 | (MST 、OCO 不提供) | | 倉位 | 0：新倉; 1：平倉 ; 2：自動 | | 是否為當沖 | 空:非當沖； Y:當沖 | | ▼接續為移動停損單專屬欄位 | | | 移動點數 |  | | 觸發基準價 |  | | 當前市價 | 當下前端帶進來的市價 | | 前一個觸發價格 |  | |  |  | | | |
| **備註** | 當全部資料已經全部回傳完畢，將回傳一筆以「##」開頭的內容，表示查詢結束。  \*V2.13.38版本起配合智慧單平台異動至新版格式 | |

### MIT

|  |  |  |
| --- | --- | --- |
| 新版期貨智慧單(包含停損單、移動停損、二擇一、觸價單)查詢。透過呼叫 [GetStopLossReport](#_GetStopLossReport)後，資訊由該事件回傳。 | | |
| **宣告** | | void OnStopLossReport( [in] BSTR bstrData ); |
| **參數** | | bstrData |
|  | | ＊注意：新版本回傳內容及欄位與舊版本相異。  如果回傳的第一筆資料是一位英文M接連著三碼數字及停損單總數，其中 "M000" 開頭，表示為成功回傳，其後提供欲查詢之停損單總數。  Ex:  (成功)M000,12  第二筆後面接的就是訊息內容。回傳的字串內容，以「,」分隔每一個欄位，欄位依序為： |
| 回傳的內容以「,」分隔每一個欄位，欄位依序為：  MIT:   |  |  | | --- | --- | | 名稱 | 備註 | | 智慧單號 |  | | 市場別 | TF：國內期選 | | 交易所代碼 | TAIFEX | | 分公司代碼 |  | | 帳號 |  | | 子帳 |  | | 委託書號 |  | | 序號 | (13碼)已觸發時，成交單含IOC、 FOK產生的取消單，可以以此欄比對市場上回報資料 | | 原始序號 | (13碼) | | 策略別 | 3：OCO單  5：停損單  8：MIT單  9：MST停損 | | 智慧單委託狀態 | 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 | | 交易日 |  | | 下單時間 | 建立智慧單時間 | | 商品代碼 | 第一腳商品代碼 | | 商品契約月份 | 第一腳商品年月 | | 履約價格 | 若無則為0 | | 買賣別 | B：買 ;S：賣 | | 委託價格別 | 第一委託價格別  7:使用者輸入價 | | 委託價格類別 | 2：限價 ;3：範圍市價 | | 委託時效 | 0： ROD當日有效 ;3： IOC立即成交; 4： FOK立即全部成交 | | 委託價格 |  | | 委託數量 |  | | 觸發價格 |  | | 觸發時間 |  | | 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  訊息中逗號皆為全形”，” | | 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  F:【交易所動態退單相關代碼】 | | 智慧單異動時間 | 智慧單異動時間 | | 預約單註記 | 0:盤中單  1:預約單 | | 預約單序號 | 當預約單註記為時1 | | 交易日 |  | | 第一腳下單商品代碼 | 後台下單代碼 EX：FITX | | 第二腳下單商品代碼 | 後台下單代碼 | | 第二腳商品契約月份 |  | | 第二腳履約價格 |  | | 第一腳倉位 | 倉位 0：新倉; 1：平倉 ; 2：自動 | | 是否為當沖 | 空:非當沖； Y:當沖 | | 買賣權 |  | | ▼接續為MIT單專屬欄位 | | | 當下市價價格 |  | | 觸價方向 | 0：None ;1：GTE ;2：LTE | | 成交價觸發價格 |  | | | |
| **備註** | 當全部資料已經全部回傳完畢，將回傳一筆以「##」開頭的內容，表示查詢結束。  \*V2.13.38版本起配合智慧單平台異動至新版格式 | |

### OCO

|  |  |  |
| --- | --- | --- |
| 新版期貨智慧單(包含停損單、移動停損、二擇一、觸價單)查詢。透過呼叫 [GetStopLossReport](#_GetStopLossReport)後，資訊由該事件回傳。 | | |
| **宣告** | | void OnStopLossReport( [in] BSTR bstrData ); |
| **參數** | | bstrData |
|  | | ＊注意：新版本回傳內容及欄位與舊版本相異。  如果回傳的第一筆資料是一位英文M接連著三碼數字及停損單總數，其中 "M000" 開頭，表示為成功回傳，其後提供欲查詢之停損單總數。  Ex:  (成功)M000,12  第二筆後面接的就是訊息內容。回傳的字串內容，以「,」分隔每一個欄位，欄位依序為： |
| 回傳的內容以「,」分隔每一個欄位，欄位依序為：  OCO：   |  |  | | --- | --- | | 名稱 | 備註 | | 智慧單號 |  | | 市場別 | TF：國內期選 | | 交易所代碼 | TAIFEX | | 分公司代碼 |  | | 帳號 |  | | 子帳 |  | | 委託書號 |  | | 序號 | (13碼)已觸發時，成交單含IOC、 FOK產生的取消單，可以以此欄比對市場上回報資料 | | 原始序號 | (13碼) | | 策略別 | 3：OCO單  5：STP單  8：MIT單  9：MST單 | | 智慧單委託狀態 | 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 | | 交易日 |  | | 下單時間 | 建立智慧單時間 | | 商品代碼 | 第一腳商品代碼 | | 商品契約月份 | 第一腳商品年月 | | 履約價格 | 若無則為0 | | 買賣別 | B：買 ;S：賣 | | 委託價格別 | 第一委託價格別  7:使用者輸入價 | | 委託價格類別 | 2：限價 ;3：範圍市價 | | 委託時效 | 0： ROD當日有效 ;3： IOC立即成交; 4： FOK立即全部成交 | | 委託價格 |  | | 委託數量 |  | | 觸發價格 |  | | 觸發時間 |  | | 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  訊息中逗號皆為全形”，” | | 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  F:【交易所動態退單相關代碼】 | | 智慧單異動時間 | 智慧單異動時間 | | 預約單註記 | 0:盤中單  1:預約單 | | 預約單序號 | 當預約單註記為時1 | | 交易日 |  | | 第一腳下單商品代碼 | (MST 、OCO 不提供)後台下單代碼 EX：FITX | | 第二腳下單商品代碼 | (MST 、OCO 不提供)後台下單代碼 | | 第二腳商品契約月份 |  | | 第一腳倉位 | 倉位 0：新倉; 1：平倉 ; 2：自動 | | 是否為當沖 | 空:非當沖； Y:當沖 | | ▼接續為二擇一停損單專屬欄位 | | | 第一腳觸發價 |  | | 第二腳觸發價 |  | | 第二腳買賣別 | B:買；S:賣 | | 第二腳委託價格別 | 7:使用者輸入價 | | 第二腳委託價格類別 | 2：限價;3：範圍市價 | | 第二腳委託價 |  | | 第二腳委託時效 | 0： ROD當日有效 ;3： IOC立即成交; 4： FOK立即全部成交 | | 第二腳倉位 | 倉位 0：新倉; 1：平倉 ; 2：自動 | | 萬用訊息 | 適用於智慧單狀態代碼為999情況 | | ▼接續為因應長效單新增欄位 | | | 是否觸發即停止 | 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 | | 長效單註記 | 0：非長效單  1：長效單 | | 長效單序號 |  | | 長效單結束日期 | 20230829 | | 長效單結束條件 | 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 | | 長效單委託總量 |  | | 長效單成交總量 |  | | | |
| **備註** | 當全部資料已經全部回傳完畢，將回傳一筆以「##」開頭的內容，表示查詢結束。  \*V2.13.38版本起配合智慧單平台異動至新版格式 | |

### AB

|  |  |  |
| --- | --- | --- |
| 新版期貨智慧單(包含停損單、移動停損、二擇一、觸價單)查詢。透過呼叫 [GetStopLossReport](#_GetStopLossReport)後，資訊由該事件回傳。 | | |
| **宣告** | | void OnStopLossReport( [in] BSTR bstrData ); |
| **參數** | | bstrData |
|  | | ＊注意：新版本回傳內容及欄位與舊版本相異。  如果回傳的第一筆資料是一位英文M接連著三碼數字及停損單總數，其中 "M000" 開頭，表示為成功回傳，其後提供欲查詢之停損單總數。  Ex:  (成功)M000,12  第二筆後面接的就是訊息內容。回傳的字串內容，以「,」分隔每一個欄位，欄位依序為： |
| 回傳的內容以「,」分隔每一個欄位，欄位依序為：  AB：   |  |  | | --- | --- | | 名稱 | 備註 | | 智慧單號 |  | | 市場別 | TF：國內期選 | | 交易日 | 20230911 | | 交易所代碼 | TAIFEX | | 分公司代碼 |  | | 帳號 |  | | 子帳 | 無子帳以0000000 | | 委託書號 |  | | 流水號13碼 | (13碼)已觸發時，成交單含IOC、 FOK產生的取消單，可以以此欄比對市場上回報資料 | | 原始流水號 | (13碼) | | 策略別 | 3：OCO單  5：STP單  8：MIT單  9：MST單  10：AB單 | | 商品代碼 |  | | 下單市場別 |  | | 下單交易所別 |  | | 下單商品代碼 |  | | 商品契約月份 |  | | 履約價格 |  | | 下單商品代碼2 |  | | 商品契約月份2 |  | | 履約價格2 |  | | 買賣別 | B：買 S：賣 | | 委託價格別 | 第一委託價格別  7:使用者輸入價 | | 委託價格類別 | 2：限價 ;3：範圍市價 | | 委託時效 | 0： ROD當日有效 ;3： IOC立即成交; 4： FOK立即全部成交 | | 委託價格 |  | | 委託數量 |  | | 觸發價格 |  | | 觸發時間 |  | | 下單時間 | 2020-07-07 12:32:10.000 | | 智慧單委託狀態 | 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 | | 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  訊息中逗號皆為全形”，” | | 智慧單異動時間 | 智慧單異動時間 | | 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  F:【交易所動態退單相關代碼】 | | 當下市價價格 |  | | 觸價方向 | 1：大於等於  2：小於等於 | | 是否為選擇權商品 | 0：否  1：是 | | 預約單註記 | 0：否  1：是 | | 預約單序號 |  | | 預約單交易日 |  | | 是否為內外盤委託量 | 0：否  1：是 | | 內外盤委託量 |  | | 內外盤委託量觸發方向 | 1：大於等於  2：小於等於 | | 成交價觸發價格 |  | | 成交量 |  | | 新平倉 | 0：新單  1：平倉  2：自動 | | 是否為當沖 | 0：否  1：是 | | 買賣權 | N：None  C：Call  P：Put | | 委託價分子 |  | | 委託價分母 |  | | 成交價分子 |  | | 成交價分母 |  | | 觸發價分子 |  | | 觸發價分母 |  | | 萬用訊息 |  | | | |
| **備註** | 當全部資料已經全部回傳完畢，將回傳一筆以「##」開頭的內容，表示查詢結束。  \*V2.13.38版本起配合智慧單平台異動至新版格式 | |

|  |
| --- |
| Python範例程式碼\_TFStrategyOrder.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) |
| # 新版期貨智慧單(包含停損單、移動停損、二擇一、觸價單)被動回報查詢。透過呼叫GetStopLossReport後，資訊由該事件回傳。      def OnStopLossReport(self, bstrData):          msg = "【OnStopLossReport】" + bstrData          richTextBoxMessage.insert('end', msg + "\n")          richTextBoxMessage.see('end') |

# Struct 結構物件

## FUTUREORDER (期選智慧單下單物件)▼

struct FUTUREORDER

{

BSTR bstrFullAccount; //期貨帳號，Broker id (例如: F020000)＋帳號7碼

BSTR bstrStockNo; //委託期權商品代號

SHORT sTradeType; //0:ROD 3:IOC 4:FOK

//sTradeType為ROD時，nOrderPriceType僅可指定限價

SHORT sBuySell; //0:買進 1:賣出

SHORT sDayTrade; //當沖0:否 1:是，可當沖商品請參考交易所規定。

SHORT sNewClose; //新平倉，0:新倉 1:平倉 2:自動

BSTR bstrPrice; //委託價格，(指定限價時，需填此欄)、[OCO]停利委託價。

//請設nOrderPriceType代表範圍市價，不可用特殊價代碼「P」代表範圍市價

//{移動停損**不須填**委託價格}

　　BSTR bstrPrice2; //[OCO]停損委託價。

LONG nQty; //交易口數

BSTR bstrTrigger; //觸發價，觸發基準價、[OCO]停利觸發價。

//{期貨停損、移動停損、選擇權停損、MIT、OCO下單使用：不可0、不可給特殊價代碼P}

BSTR bstrTrigger2; //[OCO]停損觸發價。

BSTR bstrMovingPoint; //{僅移動停損下單使用}移動點數。

SHORT sReserved; //盤別，0:盤中(T盤及T+1盤)；1:T盤預約{MIT 單不須填盤別}

BSTR bstrDealPrice;

//成交價 {限MIT下單使用：不可0、不可給特殊價代碼, 需自行取得委託當下成交價}

BSTR bstrSettlementMonth; // 委託商品年月，YYYYMM共6碼(EX: 202206)

LONG nOrderPriceType; // 委託價類別 2: 限價; 3:範圍市價 （不支援市價）

//sTradeType為ROD時，nOrderPriceType僅可指定限價

LONG nTriggerDirection; //{限MIT下單使用} 觸發方向1:GTE, 2:LTE

};

註: 若委託期貨商品代號為TX00、 MTX00等近月商品，則可忽略bstrSettlementMonth商品年月

### 期貨智慧單STP ▼

struct FUTUREORDER

{

BSTR bstrFullAccount; //期貨帳號，Broker id (例如: F020000)＋帳號7碼

|  |
| --- |
| //商品  BSTR bstrStockNo; //期權商品代號  BSTR bstrSettlementMonth; // 委託商品年月，YYYYMM共6碼(EX: 202206)  SHORT sNewClose; //新平倉，0:新倉 1:平倉 2:自動  SHORT sBuySell; //0:買進 1:賣出  SHORT sTradeType; //0:ROD 3:IOC 4:FOK  SHORT sDayTrade; //當沖0:否 1:是，可當沖商品請參考交易所規定 |

|  |
| --- |
| //條件  BSTR bstrTrigger; //觸發價，觸發基準價。{期貨停損、選擇權停損、不可0、不可給特殊價代碼}  LONG nQty; //交易口數  LONG nOrderPriceType; // 2: 限價; 3:範圍市價（不支援市價）  BSTR bstrPrice; //委託價格，(指定限價時，需填此欄) //請設nOrderPriceType代表特殊價格「P」範圍市價 |

SHORT sReserved; //盤別，0:盤中(T盤及T+1盤)；1:T盤預約

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:短期內完全成交即失效)

};

比較表

|  |
| --- |
| 異動後V2.13.45 |
| **期貨STP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 |

### 期貨智慧單MST ▼

struct FUTUREORDER

{

BSTR bstrFullAccount; //期貨帳號，Broker id (例如: F020000)＋帳號7碼

|  |
| --- |
| //商品  BSTR bstrStockNo; //委託期權商品代號  BSTR bstrSettlementMonth; // 委託商品年月，YYYYMM共6碼(EX: 202206)// bstrStockNo指定月份，需填商品契約年月  SHORT sNewClose; //新平倉，0:新倉 1:平倉 2:自動  SHORT sBuySell; //0:買進 1:賣出  SHORT sTradeType; // 3:IOC ; 4:FOK  SHORT sDayTrade; //當沖0:否 1:是，可當沖商品請參考交易所規定。 |

|  |
| --- |
| //條件  BSTR bstrMovingPoint; //移動點數。{僅移動停損下單使用}  BSTR bstrTrigger; //觸價基準。  LONG nQty; //交易口數  LONG nOrderPriceType; // 2: 限價; 3:範圍市價 （不支援市價） |

SHORT sReserved; //盤別，0:盤中(T盤及T+1盤)；1:T盤預約

};

### 期貨、選擇權智慧單MIT ▼

struct FUTUREORDER

{

BSTR bstrFullAccount; //期貨帳號，Broker id (例如: F020000)＋帳號7碼

|  |
| --- |
| //商品  BSTR bstrStockNo; //委託期權代號  BSTR bstrSettlementMonth; // 委託商品年月，YYYYMM共6碼(EX: 202206)//若 bstrStockNo指定月份，需填商品契約年月  BSTR bstrDealPrice; //成交價 {限MIT下單使用：不可0、不可給特殊價代碼}  SHORT sNewClose; //新平倉，0:新倉 1:平倉 2:自動  SHORT sBuySell; //0:買進 1:賣出  SHORT sTradeType; //0:ROD 3:IOC 4:FOK  SHORT sDayTrade; //當沖0:否 1:是，可當沖商品請參考交易所規定。 |

|  |
| --- |
| //組合  LONG nTriggerDirection; //觸發方向1:GTE(大於), 2:LTE(小於)  BSTR bstrTrigger; //觸發價。//{ MIT下單使用：不可0、不可給特殊價代碼}  LONG nQty; //交易口數  LONG nOrderPriceType; // 委託價類別 2: 限價; 3:範圍市價  BSTR bstrPrice; //委託價格 (指定限價時，需填此欄) |

};

### 期貨智慧單OCO ▼

struct FUTUREORDER

{

BSTR bstrFullAccount; //期貨帳號，Broker id (例如: F020000)＋帳號7碼

|  |
| --- |
| //商品  BSTR bstrStockNo; //委託期權代號  BSTR bstrSettlementMonth; // 指定月份商品契約年月，共6碼EX:202212// bstrStockNo指定月份，需填商品契約年月  SHORT sNewClose; //新平倉，0:新倉 1:平倉 2:自動 |

|  |
| --- |
| //第一隻腳  BSTR bstrTrigger; //第一腳觸發價，當市價大於觸發價1時觸發。  SHORT sBuySell; //0:買進 1:賣出  SHORT sTradeType; //0:ROD 3:IOC 4:FOK  LONG nOrderPriceType; // 第一腳及第二腳委託價格類型。2限價;3範圍市價（不支援市價）  BSTR bstrPrice; //第一腳委託價格 |
| //第二隻腳  BSTR bstrTrigger2; //第二腳觸發價，當市價小於觸發價2時觸發。  SHORT sBuySell2; //0:買進 1:賣出  BSTR bstrPrice2; //第二腳委託價格 |

|  |
| --- |
| //條件  LONG nQty; //交易口數  SHORT sDayTrade; //當沖0:否 1:是，可當沖商品請參考交易所規定。 |

SHORT sReserved; //盤別 0:盤中; 1:預約

SHORT nTimeFlag; //預約單盤別，預設為盤中1:T

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:短期內完全成交即失效)

};

比較表

|  |
| --- |
| 異動後V2.13.45 |
| **期貨OCO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 |

### 期貨智慧單AB ▼

struct FUTUREORDER

{

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

|  |
| --- |
| //看A  BSTR bstrStockNo2; //商品代號  LONG nMarketNo; //市場編號(1:國內證, 2:國內期, 3:國外證, 4:國外期) ，請輸入A商品市場別  BSTR bstrCIDTandem; //交易所代碼(EX: TSE、TAIFEX、CME)  LONG nTimeFlag; //1:上市, 2:上櫃(不開放權證、興櫃商品)，非證券商品請填0  BSTR bstrDealPrice; //成交價  LONG nTriggerDirection; //觸價方向(1:GTE大於等於,2:LTE小於等於)  BSTR bstrTrigger //觸發價 |

|  |
| --- |
| //下B  SHORT sBuySell; //買賣別(0:買進, 1:賣出)  SHORT sBuySell2; //買賣別2(0:買進, 1:賣出)，非價差商品請填0  SHORT sDayTrade; //是否為當沖(0:否, 1:是)  SHORT sNewClose; //新平倉(0:新倉, 1:平倉, 2:自動)  LONG sTradeType; //委託時效(0:ROC, 3:IOC, 4:FOK, 委託範圍市價無ROD)  BSTR bstrStockNo; //委託股票代號  BSTR bstrSettlementMonth; //商品契約月份(YYYYMM共6碼, EX: 202206)  BSTR bstrSettlementMonth2; //商品契約月份2(YYYYMM共6碼, EX: 202206)，非價差商品請填0  LONG nCallPut; //是否為選擇權(0:否, 1:Call, 2:Put)  BSTR bstrStrikePrice //履約價(非選擇權商品請填0)  LONG nFlag; //是否委託價差商品(0:否, 1:是)  LONG nQty; //委託口數  BSTR bstrPrice; //委託價格  LONG nOrderPriceType; //委託價類別(2:限價, 3:範圍市價) |

SHORT sReserved; //是否為預約單(0:否, 1:是)

};

## FUTUREOCOORDER (二擇一OCO下單物件)

struct FUTUREOCOORDER

{

BSTR bstrFullAccount; //期貨帳號，Broker id (例如: F020000)＋帳號7碼

BSTR bstrStockNo; //委託期權代號

BSTR bstrPrice; //第一腳委託價格

BSTR bstrTrigger; //第一腳觸發價，當市價大於觸發價1時觸發。

BSTR bstrPrice2; //第二腳委託價格

BSTR bstrTrigger2; //第二腳觸發價，當市價小於觸發價2時觸發。

SHORT sTradeType; //3:IOC 4:FOK

SHORT sBuySell; //0:買進 1:賣出

SHORT sDayTrade; //當沖0:否 1:是，可當沖商品請參考交易所規定。

SHORT sNewClose; //新平倉，0:新倉 1:平倉 2:自動

LONG nQty; //交易口數

LONG sOrderPriceType1; // 第一腳及第二腳委託價格類型。2限價;3範圍市價（不支援市價）

SHORT sBuySell2; //0:買進 1:賣出

SHORT sReserved; //盤別 0:盤中; 1:預約

};

## CANCELSTRATEGYORDER(智慧單刪單物件)

### 期貨智慧單刪單STP、MST、MIT、OCO、AB

struct CANCELSTRATEGYORDER

{

BSTR bstrLogInID; //登入ID

BSTR bstrFullAccount; //期貨帳號

LONG nMarket; //市場別2:國內期貨選擇權

BSTR bstrSmartKey; //智慧單號

LONG nTradeKind; //3:OCO、5:STP、8:MIT、9:MST、10：AB

BSTR bstrSeqNo; //委託序號 (預約單可忽略)

BSTR bstrOrderNo; //委託書號（若欲刪除之委託已產生書號則需填入；預約單可忽略）

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

}

\*請留意，未填書號將影響解除保證金等風控

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 |

### 智慧單多筆刪單

struct CANCELSTRATEGYORDER

{

BSTR bstrLogInID; //登入ID

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)

}