User Guide of the ExcelAPI

Introduction

The ExcelAPI is a plugin for the Microsoft Excel. It enables Excel users to retrieve market prices and send or amend trading orders with much automation by implementing VBA (Visual Basic with Applications) code with Excel. The Excel users write VBA code (or use existing Excel Workbooks developed by Tradex which also contain VBA code) to call methods exposed by our ExcelAPI library. The ExcelAPI works with either MT (MultiTradex) or TMClient (Tradex Management System Client) to retrieve prices and send orders. This document will illustrate, with examples, things you can do with ExcelAPI and how you can do them.

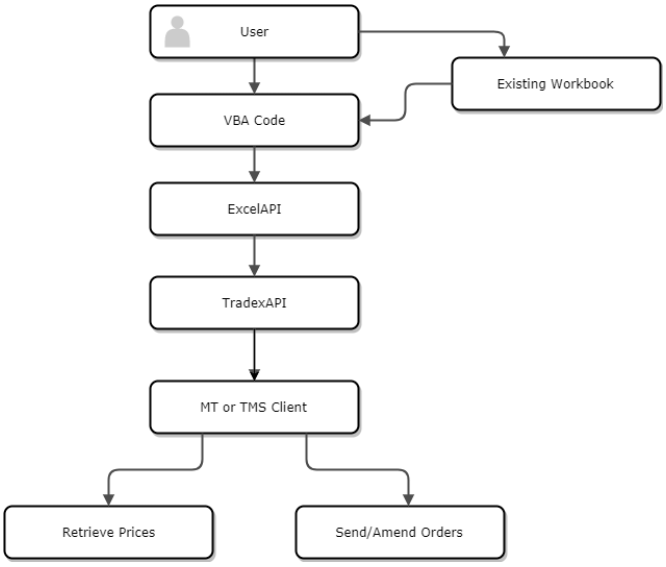


Figure 1: Working logic of the ExcelAPI. A Downward arrow means "calls, uses, or does"

It is also a comprehensive user manual which lists all features of ExcelAPI in List of library methods section.

Prerequisites

Skills

In order to use this API, you need to know how to work with Excel, and also how to program using VBA in Excel to implement simple interactive applications if you want to write your own pieces of VBA code instead of using existing ones.

MT or TMClient

Either MT or TMClient must be in working condition on your Windows PC. You also need to have a running session the software before trying out any of the examples below.

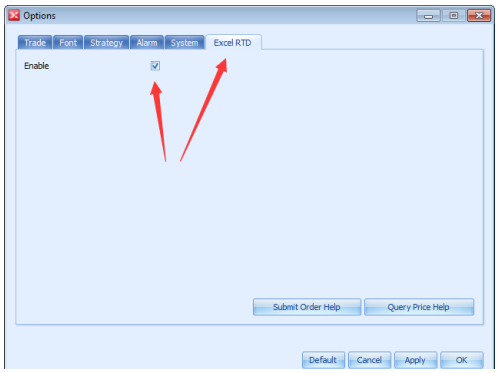


Figure 2

Setting

In MT or TMClient, go to Tools -> Options -> Excel RTD and check the Enable box (Figure 2). If Windows Firewall pops up a message concerning security, click Yes or OK. Note that you only need to set this once.

Microsoft Excel

Both 32 bit and 64 bit versions of Microsoft Office 2003, 2007, 2010 (32 bit and 64 bit), 2013 (32 bit and 64 bit), 2016 (32 bit and 64 bit) have been tested to work with our ExcelAPI. You can check your version of Excel by going to File -> Account -> About Excel (Figure 3).

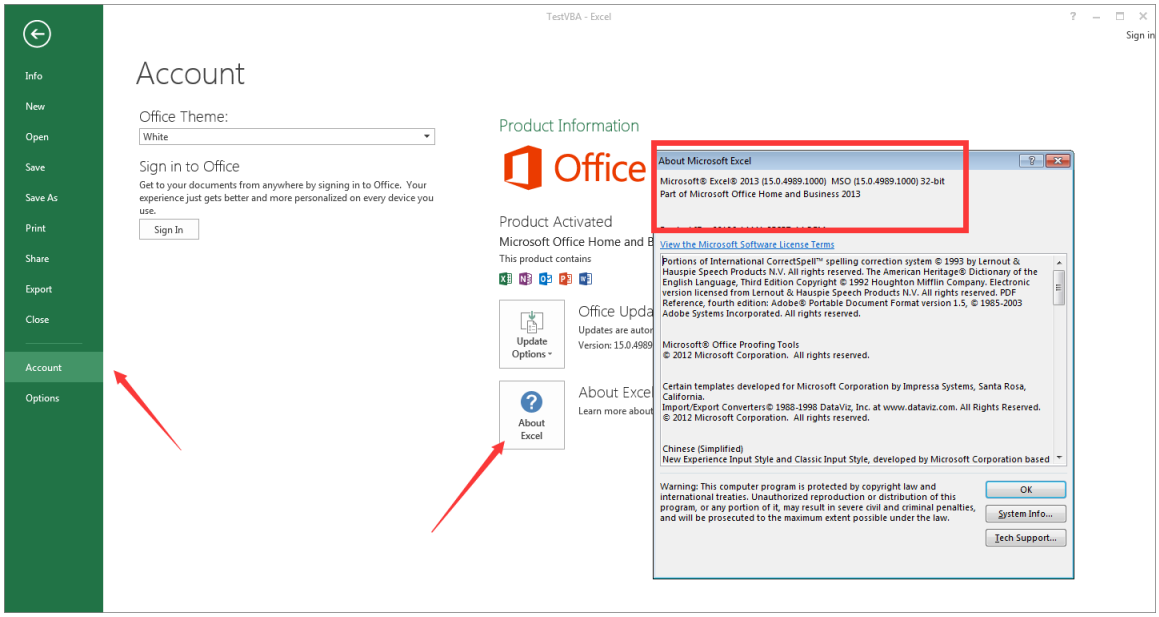


Figure 3

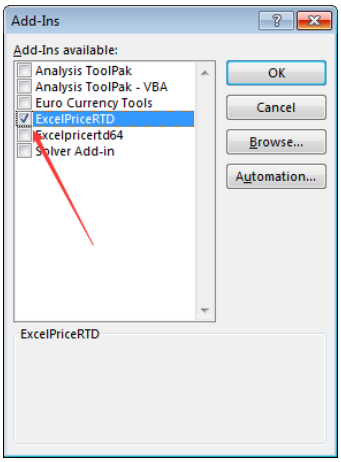


Figure 4

Setting:

In Excel go to File -> Options -> Add Ins -> Go -> Browse, and look for the file RTD/ExcelPriceRTD (or RTD/ExcelPriceRTD64 if your Excel is 64 bit) in the uncompressed ExcelAPI package. After adding the file, check the box before the just added file and press OK. Note that you only need to set this once, unless you are switching to a new version of ExcelAPI, in which case you need to input the file path to Excel again.

Test Example

Currenex\EURUSD\DEC07	30	1.16863	1.16883	204	1.16863	70	527
TOCOM\Gold\JUN07	4	2345	2352	6	2345	14	92
TOCOM\Gold\MAR08	18	2348	2353	3	2353	5	75

Figure 5: Part of an MT screen

In MT or TMClient, choose an active contract (TOCOM\Gold\MAR08 in this example). In an Excel session type this in a cell and hit Enter:

=tradex("TOCOM", "Gold", "MAR08", "bid")

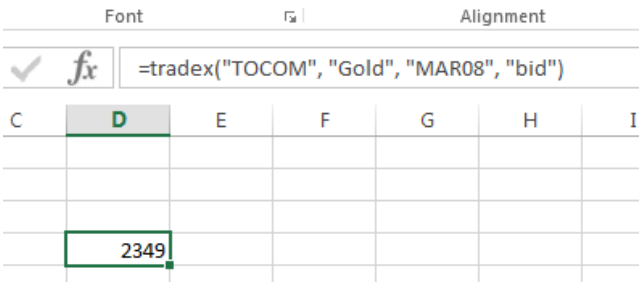


Figure 6

Note that the case of the letters must exactly match those in the contract. The double quotes are necessary. You should see the bidding price appears in the cell and it should match that seen in MT or TMClient (Figure 6). It should also updates itself when the bidding price changes in MT or TMClient.

If the above doesn't work, go to the Common Issues with Excel Settings: Section.

Price feed method tradex()

The above test is an example of grabbing a piece of market data in real time. The function tradex takes four arguments of type string:

object tradex(string exchangeName, string commodityName, string contractDate, string dataRequested)

The accepted input for dataRequested argument and the corresponding meaning is listed below

dataRequested =	meaning
"bid"	Bid price
"bidvol"	Bid volume
"offer"	Offer price
"offervol"	Offer volume
"last"	Last price
"lastvol"	Last volume

“stl”	Settlement price
“total”	Total volume

For instance, if you type in “last” for the forth argument of the tradex() method, it will try to retrieve the last price on the market of the relevant contract.

Price update interval

By default, Excel requests MT or TMClient for price update every 2 seconds and then refreshes any cell that receives a new value. You can check or set the update interval using VBA code in an Immediate Window (Alt + F11 -> View -> Immediate Window):

?Application.RTD.ThrottleInterval

Application.RTD.ThrottleInterval=1000

Note that ThrottleInvertal is the variable name representing the update interval in milliseconds. Its value will be preserved even after you close Excel. Also note that Excel won’t update prices more frequently than once per 0.2 seconds.

Place, amend or cancel orders

A worked example

Open the test Workbook TestVBA.xlsm in the RTD folder. This is an interactive program written using VBA calling ExcelAPI methods. Note that the 2nd and 3rd lines of Sheet1 maintains some price information by calling the method tradex(string, string, string, string) described in the previous session. Now click the button “Begin: Connect MT/TMClient” and dismiss the message box popped up. Select the cell D2 then click “Add Market Order”. For “Lots” input an integer, and for “Account” type in the account name after the backslash shown in MT or TMClient, in the case of Figure 7 it is DEMO1. Notice that orders will get reflected in MT or TMClient as well as in the Excel sheet. Now add a limit order in a similar way, entering a price that won’t be filled by the market. Notice that a working order is displayed in the sheet. Select that row of working order, and you can now try “Amend Order” and “Cancel Order” which are quite self-explanatory.

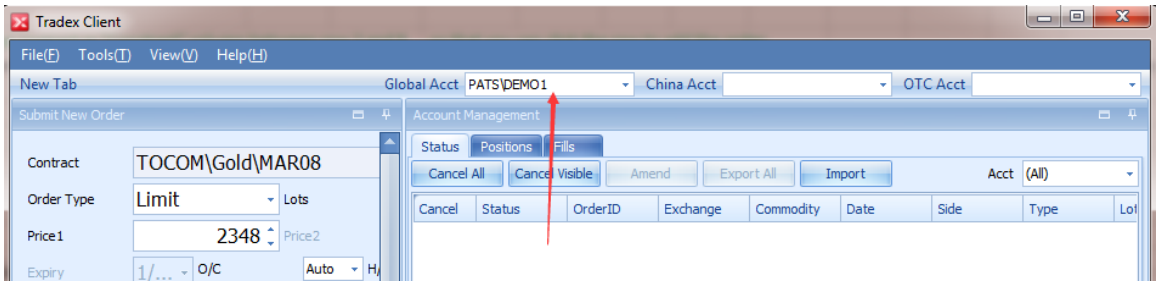


Figure 7

Each button triggers a VBA procedure which calls library methods in ExcelAPI to realize the functionality. You are encouraged to take a look at the VBA code underneath the buttons,

especially the lines calling ExcelAPI methods. As concerned by VBA these methods are external and need to be called using Application.Run() in this format:

MethodReturnType = Application.Run("MethodName", methodArg1, methodArg2, …)

As an example, to call the method ConnectMT():

Dim apiId As String

apiId = Application.Run("ConnectMT", “abcd” 1)

The method has the signature of string ConnectMT(string, int). Instead of typing apiid = ConnectMT(“abcd”, 1) as it should be in C#, you have to type apiId = Application.Run("ConnectMT", “abcd” 1) in VBA. The method name “ConnectMT” is passed as the first argument of Application.Run(), and the method arguments “abcd” and 1 is passed as subsequent argument of Application.Run() in the same order. The returned value of ConnectMT will be assigned to the string typed apiId variable.

List of library methods

Overview

You can examine the VBA code in RTD/TestVBA and RTD/TestVBAShort. The two are different in the way to register the callbacks.

Method tradex() retrieves market data in real time.

Method ConnectMT() needs to be called before any order submission/amending/cancellation. It connects the Excel to a running MT or TMClient session and returns a string called apiid which is the ID of this connecting session. Most other methods needs this apiid as an input. Method Disconnect() should be called when the connection is no longer in use.

Method AddOrder() is used to submit an order through MT or TMClient. Method AmendOrder() can change a working order’s lots and/or price, while CancelOrder will remove a working order completely. Read the detailed documentation for each of the methods’ detail.

To enable the VBA coder to perform certain tasks upon any change of status of an order, ExcelAPI provides two events for the VBA coder to subscribe. SetOrderCallBack() subscribes a function to an event which will fire when there’s a change in an order’s status, while SetFillCallBack() subscribes a function to an event which will fire when an order get filled. With these you can easily do things like popping up a message box showing some order information or fill Excel grids with order information whenever an order get submitted, fulfilled, or cancelled.

All these methods listed below can be called in VBA by using this syntax described in section: A worked example:

MethodReturnType = Application.Run("MethodName", methodArg1, methodArg2, …)

Each method can also be typed into a cell as a formula, as shown with tradex().

object tradex(string exchangeName, string commodityName, string contractDate, string dataRequested)

Obtains a specified piece of market information of a specified contract. This is detailed in Section: Price feed method tradex().

string ConnectMT(string clientOrderPrefix, int orderSeed)

Connects to the running session of MT or TMClient. This needs to be executed before submitting any orders.

- ### Parameters

  string clientOrderPrefix, int orderSeed: Used to predefine the clientOrderIDs that will be later generated during this connection session. For instance, if clientOrderPrefix = "somePrefix" and orderSeed = 1, the subsequent clientOrderID returned by later calls of AddOrder() will be "somePrefix1", somePrefix2", "somePrefix3"……

- ### Return Value

  string apiid: this is the identity of the connection between the Excel instance and the MT/TMClient instance. The value should be stored in your VBA program as it is needed for later calls of AddOrder, AmendOrder, SetOrderCallBack, SetFillCallBack and Disconnect.

void Disconnect(string apiid)

Closes down the connection between Excel and MT or TMClient

- ### Parameters

  string appid: the identity of the connection, and is generated at connection time by ConnectMT()

string AddOrder(string apiid, string traderAccount, string exchangeName, string contractName, string contractDate, string orderType, string side, int lots, string price1, string price2, string openClose, string HedgeFlag)

Submit an order through MT or TMClient to the market. Note that the strings are case-sensitive.

- ### Parameters

- string appid: the identity of the connection generated at connection time by ConnectMT()
- string exchangeName, string contractName, string contractDate, int lots: self-explanatory
- string traderAccount: the traderAccount used to submit the order. This should be the account name after the backslash shown in MT or TMClient, in the case of Figure 7 it is DEMO1. When traderAccount is left null or empty, it will be defaulted to the first trader account in MT or TMClient.
- string orderType: must be one of the following (case-sensitive): "Market", "Limit", "Stop", "StopLimit".
- string side: must either be "B" (for Buy) or "S" (for Sell).

- string price1, string price2: If orderType is StopLimit, price1 is the limit price and price2 is the stop price. Otherwise, only price1 is used.
- string openClose: must be one of the these (case-sensitive): “Open”, “Close”, “CloseToday”, “Auto”.
- string HedgeFlag: must be one of “S” (for Speculate), “H” (for Hedge) or “A” (for Arbitrage).

➢ Return Value

string: Returns the clientOrderID of the submitted order on success, “apiId error” when apiId specified is invalid, “Disconnected” when failing to send order to MT or TMClient. Invalid arguments can also cause these returned value (self-explanatory):
"OpenClose argument input wrong"
"OrderType argument is wrong"
"OrderSides argument is wrong"
"PositionType argument is wrong" (when HedgeFlag argument is invalid)

string AmendOrder(string apiied, string clientOrderID, int lots, string price1, string price2)

Amends a working order

➢ Parameters

- string appid: the identity of the connection generated at connection time by ConnectMT()
- string clientOrderID: identifies the order to amend
- int lots, string price1, string price2: the lots, price1 and price2 to be amended to

➢ Return Value

Returns “Sent” when successfully sent the amending request to MT or TMClient, “apiId error” when the apiid specified is invalid, “Disconnected” when failing to send amending request to MT or TMClient.

➢ Note

Not only working orders submitted via ExcelAPI, but also those submitted directly in MT or TMClient can be amended using AmendOrder. In the latter case, don’t use “OrderID” shown in MT or TMClient as the input clientOrderID for AmendOrder(). You should get the clientOrderID prior to calling AmendOrder() from OrderCallBack(). Read bool SetOrderCallBack(string apiid, int address).

string CancelOrder(string apiid, string clientOrderID)

Cancels a working order

➢ Parameters

- string apiid: the identity of the connection generated at connection time by ConnectMT()
- string clientOrderID: identifies the order to cancel

Return string

Returns “Sent” when successfully sent the cancellation request to MT or TMClient, “apiId error” when the apiid specified is invalid, “Disconnected” when failing to send cancellation request.

Note

Not only working orders submitted via ExcelAPI, but also those submitted directly in MT or TMClient can be cancelled using AmendOrder. In the latter case, don’t use “OrderID” shown in MT or TMClient as the input clientOrderID for CancelOrder(). You should get the clientOrderID prior to calling CancelOrder() from the OrderCallBack or FillCallBack. Read bool SetOrderCallBack(string apiid, int address) and bool SetFillCallBack(string apiid, int address).

bool SetOrderCallBack(string apiid, int address, string requestedOrderFields)

Registers a VBA function to an event so that it will execute whenever an order has changed its status (Sent, Working, AmendPending, CancelPending, Cancelled, Filled). It does not matter whether the order is submitted through ExcelAPI or through MT or TMClient directly.

Parameters

- string apiid: the identity of the connection generated at connection time by ConnectMT()
- int address, string requestedOrderFields: address is a pointer to a VBA function (the CallBack) which will be called when an order changes its status. You can obtain the address of a VBA function by the “AddressOf FunctionName” expression. The CallBack should be in this format:

```
Public Function CallBackFunctionName(ByVal order As String)
```

In the above the order parameter will receive a string containing the information of the order that just changed its status. You need to provide the requestedOrderFields containing the field names of an order delimited by comma when calling SetOrderCallBack, so that ExcelAPI will know which pieces of order data and in what sequence the data should be passed to the CallBack. The requested order fields in the requested sequence will be concatenated as a string delimited by Chr(31) and passed to your VBA CallBack. You can use “orderFields = Split(order, Chr(31))” to parse the order fields. As an example:

```
‘Set the CallBack to the OrderCallBack Function, requesting three fields from the each order

Application.Run(“SetOrderCallBack”, apiid, AddressOf OrderCallBack, “contractname, lots, status”)

…

‘the CallBack Function receives a string containing the requested three fields of order data delimited by the character ‘Chr(31)

Public Function OrderCallBack(ByVal order As String) As Boolean

Dim orderData() As String
```

```
'get the three fields by splitting the string by delimiter = Chr(31)

orderData = Split(order, Chr(31))

Dim contractName As String

Dim lots As String

Dim status As String

contractName = orderData(0)

lots = orderData(1)

status = orderData(2)

'return true to indicate success execution of the CallBack

OrderCallBack = True

End Function
```

Note that the field names are not case-sensitive and can contain spaces. Each two of them must be separated by a comma. Some of the fields have aliases so you can request them by different names. If you pass an invalid field name to requestedOrderFields, it will be ignored without an exception. All order field names that can be requested from ExcelAPI is listed below, each line contains one or a few field name aliases separated by a comma referring to the same field. Notice that any addition/removal of spaces or any changes of letter cases to a valid name is still valid.

Note: in certain 64 bit versions of Excel the AddressOf operator returns a double type instead of int32 type. This case is being handled correctly by ExcelAPI which can parse both int and double input as the second argument. Strictly-speaking the method signature of SetOrderCallBack is bool SetOrderCallBack(string apiid, object address, string requestedOrderFields). This note applies for all methods containing 'int address' in the argument list.

Status

Order ID, ID

Trader Account

ClientOrderId

Exchange Name, exchange

Contract name, contract

Contract date, expiry, contractExpiry

Side, BuyOrSell

Type, OrderType

Lots

price1, price

price2

Amountfilled, filledamount, filled

Averageprice, aveprice

Current Lots

Openclose

Hedge, hedgeflag

Note that if you are manually editing the Excel sheet when your VBA CallBack is trying to update a cell, VBA will generate an exception. You should handle this exception in the CallBack, and return false if an exception has occurred, true if there's no exception. ExcelAPI will try to execute the CallBack again one second later with the same order if it receives a returned false. Look at the code in RTD/TestVBAShort.xlms for reference.

- For the format of the order fields that the VBA CallBack will receive, RTD/TestVBA will display the strings in Sheet1 exactly as it is passed into the CallBack (Figure 9). Also notice that most of the strings can be found in their exact displaying format in the Account Management panel of MT or TMClient.

Status	OrderID	Account	Client OrderID	Exchange	Contract	Maturity	Side	Order Type	Lots	Price1	Price2	FillLots	Fill Avg	Remainin	Open/Clos	Hedge Flag
Cancelled	PATS\43647	PATS\DEMO1	SYS0	LME	CA	3M	Buy	Limit	5	444		0	0	5	Open	Speculate
Cancelled	PATS\43648	PATS\DEMO1	SYS1	TOCOM	Gold	8-Mar	Buy	Limit	1	2		0	0	1	Open	Speculate
Filled	PATS\43649	PATS\DEMO1	20180105:133218-1	LME	CA	3M	Sell	Market	1	0		1	6038.5	0	Open	Speculate
Working	PATS\43650	PATS\DEMO1	20180105:133218-2	LME	CA	3M	Buy	Limit	1	1	0	0	0	1	Open	Speculate
Cancelled	PATS\43651	PATS\DEMO1	20180105:133218-3	LME	CA	3M	Buy	Limit	5	2		0	0	5	Open	Speculate
Filled	PATS\43652	PATS\DEMO1	20180105:133218-4	SHFEx	cu	14-Sep	Buy	Market	1	0		1	44610	0	Open	Speculate
Working	PATS\43653	PATS\DEMO1	20180105:133218-5	SHFEx	cu	14-Sep	Buy	Limit	1	1	0	0	0	1	Open	Speculate

Figure 8: the strings representing the order details passed into the CallBack. Note that Status can be one of these string values: Sent, Working, Filled, Cancelled, CancelPending, AmendPending.

➢ Returns boolean

Indicating whether the registration of the function is successful

bool SetOrderCallBack(string apiid, int address)

This is an alternate way of setting the orderCallBack. This does not specify which fields from the order are requested, and ExcelAPI will pass all the available fields in a pre-defined sequence to your CallBack Function.

Registers a VBA function to an event so that it will execute whenever an order has changed its status (Sent, Working, AmendPending, CancelPending, Cancelled, Filled). It does not matter whether the order is submitted through ExcelAPI or through MT or TMClient directly.

➢ Parameters

- string apiid: the identity of the connection generated at connection time by ConnectMT()

- int address: the address that points to a VBA function (the CallBack) which will be called when an order changes its status. You can obtain the address of a VBA function by the "AddressOf FunctionName" expression. The CallBack should be in this format:

Public Function CallBackFunctionName(ByVal Status As String, ByVal OrderID As String, ByVal TraderAccount As_ String,_ ByVal clientOrderID As String, ByVal ExchangeName As String, ByVal ContractName As String, ByVal ContractDate As_ String, ByVal Side As String, ByVal OrderType As String, ByVal Lots As String, ByVal Price1 As String, ByVal Price2 As_ String, ByVal FillLots As String, ByVal FillAveragePrice As String, ByVal CurrentLots As String, ByVal OpenClose As_ String, ByVal HedgeFlags As String) As Boolean

So that ExcelAPI can pass the details of the order that just changed its status into the CallBack correctly. Note that if you are manually editing the Excel sheet when your VBA CallBack is trying to update a cell, VBA will generate an exception. You should handle this exception in the CallBack, and return false if an exception has occurred, true if there's no exception. ExcelAPI will try to execute the CallBack again one second later with the same order if it receives a returned false. Look at the code in RTD/TestVBA.xlms for reference.

- For the format of the order details that the VBA CallBack will receive, RTD/TestVBA will display the strings in Sheet1 exactly as it is passed into the CallBack and in the same order as it is passed into the CallBack (Figure 9). Also notice that the most of the strings can be found in their exact displaying format in the Account Management panel of MT or TMClient.

Status	OrderID	Account	Client OrderID	Exchange	Contract	Maturity	Side	Order Type	Lots	Price1	Price2	FillLots	Fill Avg	Remainin	Open/Clos	Hedge Flag
Cancelled	PATS\43647	PATS\DEMO1	SYS0	LME	CA	3M	Buy	Limit	5	444		0	0	5	Open	Speculate
Cancelled	PATS\43648	PATS\DEMO1	SYS1	TOCOM	Gold	8-Mar	Buy	Limit	1	2		0	0	1	Open	Speculate
Filled	PATS\43649	PATS\DEMO1	20180105:133218-1	LME	CA	3M	Sell	Market	1	0		1	6038.5	0	Open	Speculate
Working	PATS\43650	PATS\DEMO1	20180105:133218-2	LME	CA	3M	Buy	Limit	1	1	0	0	0	1	Open	Speculate
Cancelled	PATS\43651	PATS\DEMO1	20180105:133218-3	LME	CA	3M	Buy	Limit	5	2		0	0	5	Open	Speculate
Filled	PATS\43652	PATS\DEMO1	20180105:133218-4	SHFEx	cu	14-Sep	Buy	Market	1	0		1	44610	0	Open	Speculate
Working	PATS\43653	PATS\DEMO1	20180105:133218-5	SHFEx	cu	14-Sep	Buy	Limit	1	1	0	0	0	1	Open	Speculate

Figure 9: the strings representing the order details passed into the CallBack. Note that Status can be one of these string values: Sent, Working, Filled, Cancelled, CancelPending, AmendPending.

➢ Returns boolean

Indicating whether the registration of the function is successful

bool SetFillCallBack(string apiid, int address, string requestedFillFields)

Registers a VBA function to an event so that it will execute whenever an order has filled. It does not matter whether the order is submitted through ExcelAPI or through MT or TMClient directly.

➢ Parameters

- string apiid: the identity of the connection generated at connection time by ConnectMT()
- int address, string requestedFillFields: address is a pointer to a VBA function (the CallBack) which will be called when an order get filled. You can obtain the address of a VBA function by the "AddressOf FunctionName" expression. The CallBack should be in this format:

Public Function CallBackFunctionName(ByVal fill As String)

In the above the fill parameter will receive a string containing the information of the order that just filled. You need to provide the requestedFillFields containing the field names of a fill delimited by comma when calling SetFillCallBack, so that ExcelAPI will know which pieces of fill data and in what sequence the data should be passed to the CallBack. The requested fill fields in the requested sequence will be concatenated as a string delimited by Chr(31) and passed to your VBA CallBack. You can use "fillFields = Split(order, Chr(31))" to parse the fill fields. As an example:

```
'Set the CallBack to the FillCallBack Function, requesting three fields from the each fill

Application.Run("SetFillCallBack", apiid, AddressOf FillCallBack, "price, exchange, fill time")

…

'the CallBack Function receives a string containing the requested three fields of fill data delimited by the character
'Chr(31)

Public Function FillCallBack(ByVal fill As String) As Boolean

Dim fillData() As String

'get the three fields by splitting the string by delimiter = Chr(31)

fillData = Split(fill, Chr(31))

Dim price As String

Dim exchange As String

Dim fillTime As String

price = orderData(0)

exchange = orderData(1)

fillTime = orderData(2)

'return true to indicate success execution of the CallBack

FillCallBack = True

End Function
```

Note that the field names are not case-sensitive and can contain spaces. Each two of them must be separated by a comma. Some of the fields have aliases so you can request them by different names. If you pass an invalid field name to requestedFillFields, it will be ignored without an exception. All fill field names that can be requested from ExcelAPI is listed below, each line containing one or a few field name aliases separated by a comma referring to the same field. Notice that any addition/removal of spaces or any changes of letter cases to a valid name is still valid.

```
Fillid

Order ID

Trader Account
```

ClientOrderId

Exchange Name, exchange

Contract name, contract

Contract date, expiry, contractExpiry

Side, BuyOrSell

Lots

price1, price

Fill time, filled time

Openclose

Hedge, hedgeflag

Notice that if you are manually editing the Excel sheet when your VBA CallBack is trying to update a cell, VBA will generate an exception. You should handle this exception in the CallBack, and return false if an exception has occurred, true if there's no exception. ExcelAPI will try to execute the CallBack again one second later with the same order if it receives a returned false. Look at the code in RTD/TestVBAShort.xlms for reference.

- For the format of the fill details that the VBA CallBack will receive, RTD/TestVBA will display the strings in Sheet2 exactly as it is passed into the CallBack and in the same order as it is passed into the CallBack (Figure 11). Also notice that the most of the strings can be found in their exact displaying format in the Account Management panel of MT or TMClient.

FillID	OrderID	Client OrerID	Account	Exchange	Commodity	Maturity	Buy/Sell	Lots	Price	Filled Time	OpenClose	HedgeFlag
436477794 05/01/2018 13:32:33	PATS\43649	20180105:133218-1	PATS\DEMO1	LME	CA	3M	Sell	1	6038.5	1/5/2018 13:32	Open	Speculate
436488248 05/01/2018 13:53:46	PATS\43652	20180105:133218-4	PATS\DEMO1	SHFEx	cu	14-Sep	Buy	1	44610	1/5/2018 13:53	Open	Speculate
436492532 05/01/2018 14:12:42	PATS\43654	20180105:141218-1	PATS\DEMO1	LME	CA	3M	Buy	1	6041.5	1/5/2018 14:12	Open	Speculate

Figure 10

➢ Returns boolean

Indicating whether the registration of the function is successful

bool SetFillCallBack(string apiid, int address)

This is an alternate way of setting the fillCallBack. This does not specify which fields from the fill are requested, and ExcelAPI will pass all the available fields in a pre-defined sequence to your CallBack Function.

Registers a VBA function to an event so that it will execute whenever an order is filled. It does not matter whether the order is submitted th rough ExcelAPI or through MT or TMClient directly.

➢ Parameters

- string apiid: the identity of the connection generated at connection time by ConnectMT()
- int address: the address that points to a VBA function (the CallBack) which will be called when an order is filled. You can obtain the address of a VBA function by the "AddressOf FunctionName" expression. The CallBack should be in this format:

Public Function UpdateFillCall(ByVal FillID As String, ByVal OrderID As String, ByVal ClientOrderID As String, _
*ByVal TraderAccount As String, ByVal ExchangeName As String, ByVal ContractName As String, ByVal ContractDate_*
*As String, ByVal Side As String, ByVal Lots As String, ByVal Price As String, ByVal FillTime As String, ByVal OpenClose_*
As String, ByVal HedgeFlags As String) As Boolean

So that ExcelAPI can pass the details of the order that has just filled into the CallBack correctly. Notice that if you are manually editing the Excel sheet when your VBA CallBack is trying to update a cell, VBA will generate an exception. You should handle this exception in the CallBack, and return false if an exception has occurred, true if there's no exception. ExcelAPI will try to execute the CallBack again one second later with the same order if it receives a returned false. Look at the code in RTD/TestVBA.xlms for reference.

- For the format of the order details that the VBA CallBack will receive, RTD/TestVBA will display the strings in Sheet2 exactly as it is passed into the CallBack and in the same order as it is passed into the CallBack (Figure 11). Also notice that the most of the strings can be found in their exact displaying format in the Account Management panel of MT or TMClient.

	FillID	OrderID	Client OrerID	Account	Exchange	Commodity	Maturity	Buy/Sell	Lots	Price	Filled Time	OpenClose	HedgeFlag
2	436477794 05/01/2018 13:32:33	PATS\43649	20180105:133218-1	PATS\DEMO1	LME	CA	3M	Sell	1	6038.5	1/5/2018 13:32	Open	Speculate
3	436488248 05/01/2018 13:53:46	PATS\43652	20180105:133218-4	PATS\DEMO1	SHFEx	cu	14-Sep	Buy	1	44610	1/5/2018 13:53	Open	Speculate
4	436492532 05/01/2018 14:12:42	PATS\43654	20180105:141218-1	PATS\DEMO1	LME	CA	3M	Buy	1	6041.5	1/5/2018 14:12	Open	Speculate

Figure 11

➢ Returns boolean

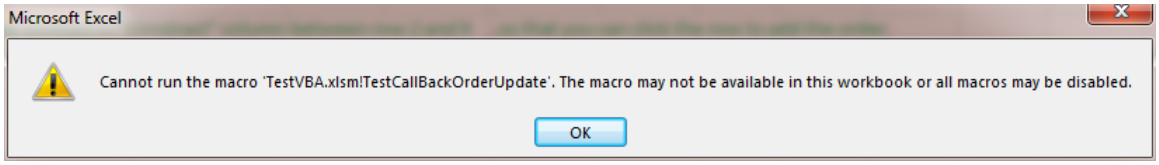
Indicating whether the registration of the function is successful

Troubleshoot

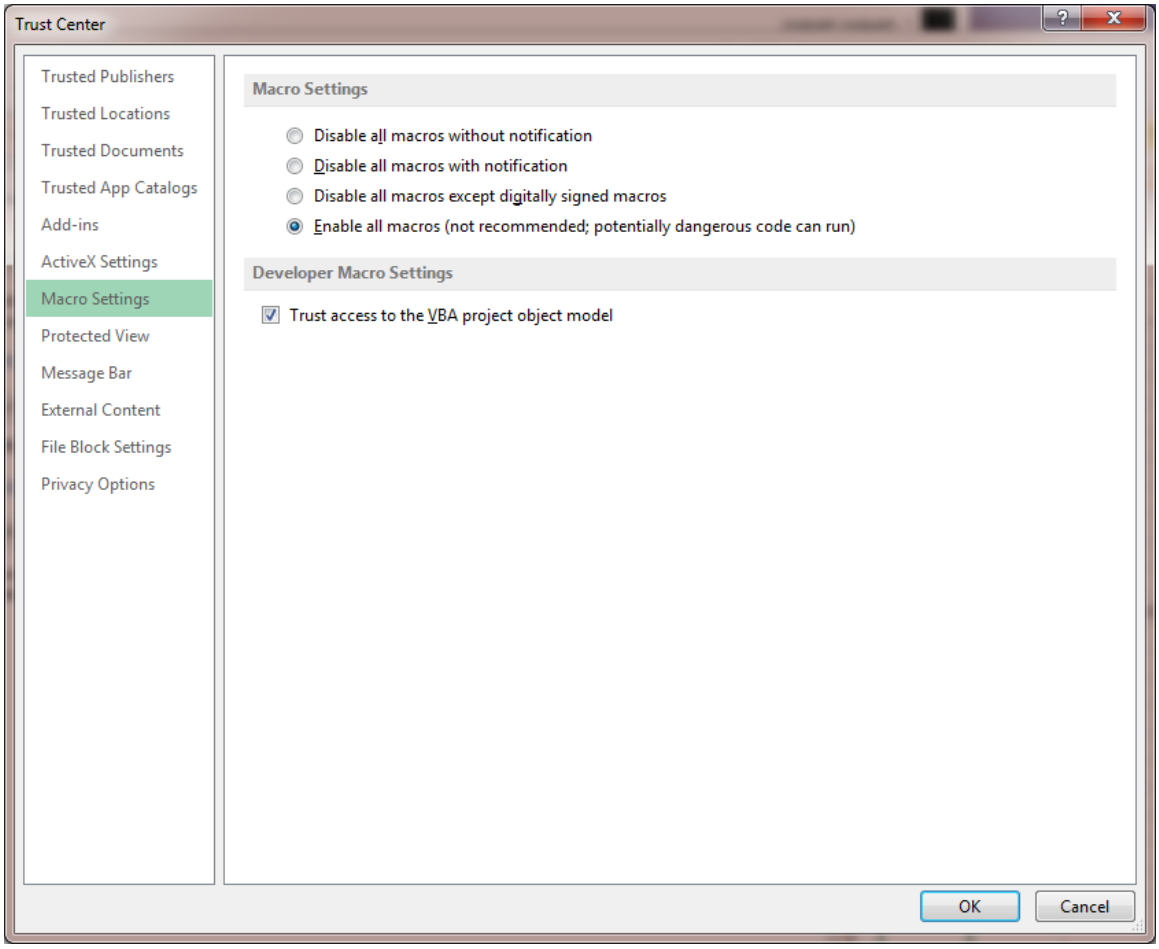
Common Issues with Excel Settings:

Here lists some screenshots of possible error messages generated by Excel. They can be easily solved by change Excel's settings. This list is not exclusive.

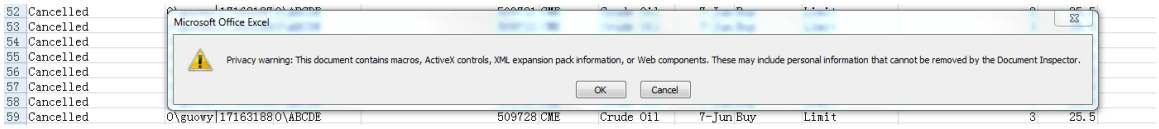
1,



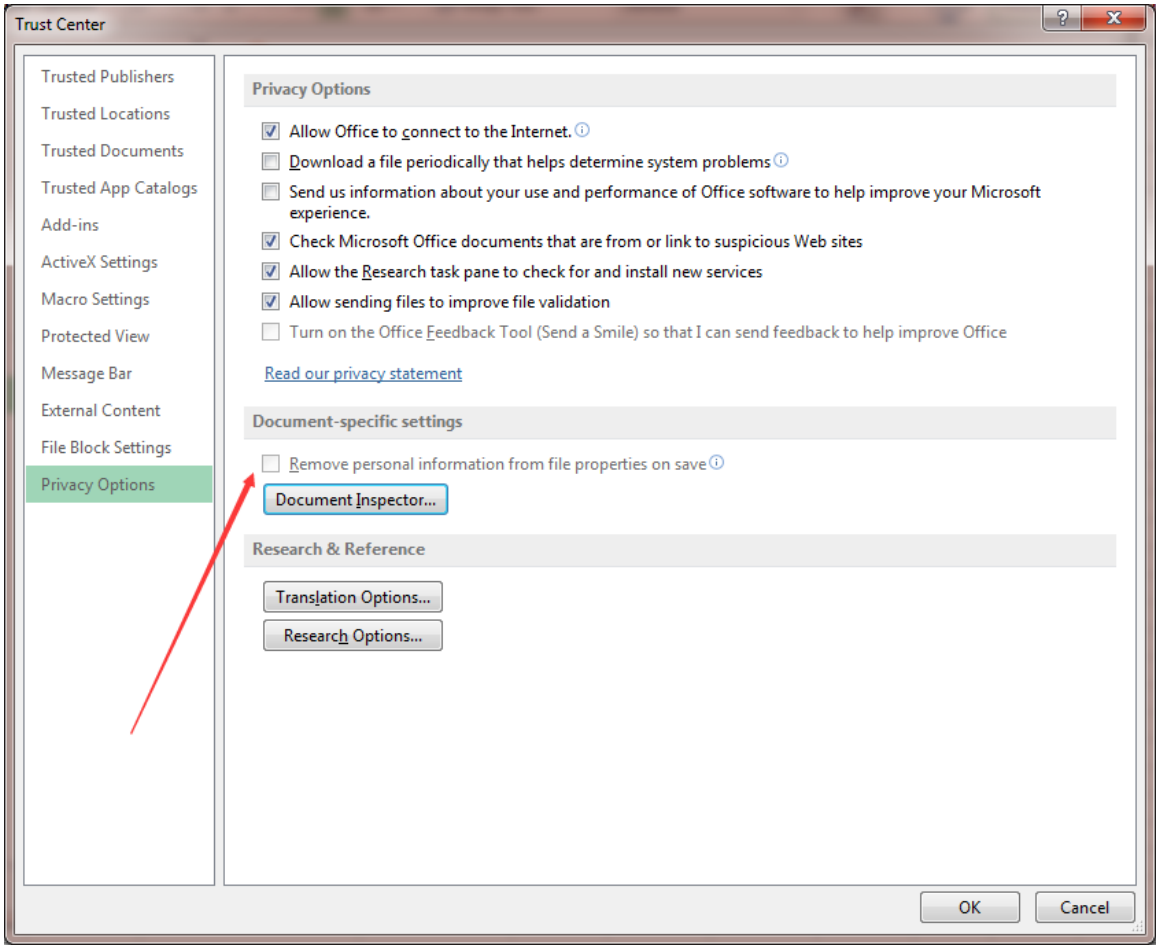
Solution: File -> Options -> Trust Center -> Trust Center Settings -> Macro Settings -> select “Enable all macros” and check “Trush access to VBA project object model” (below screenshot) then restart Excel.



2,



Solution: File -> Options -> Trust Center -> Trust Center Settings -> Privacy Options -> uncheck “Remove personal information from file properties on Save”.



Common Bugs from VBA

These information may help debugging your VBA code.

"api error"

Typically this is when you try to add/amend order when Excel has not connect to an MT/TM Client session. Call the ConnectMT method first.

"Run-time error '13': Type mismatch"

It's likely that you are trying to pass a variable that is of an incorrect type into an ExcelAPI method. Check the "List of library methods" section for method signatures.

More To Come…

Please contact Tradex Support support@tradex.com.sg should you experience difficulties in using ExcelAPI library.