

## API Reference Guide

of alternatives for connectivity directly with Interactive Application Program Interface (API) on a Broker Workstation and does not require a dedicated FIX server. This solution is called Basic. In addition, we offer an option already supporting a FIX connection. Both solutions can be used simultaneously. See the sub-level tabs for connection types.

over the Internet. It is available for IB and IBX. The connection uses the same port and need to log in as 'IBUsers' and the password.



**API Reference Guide**  
**June 2009**  
**Updated through API Release 9.62**

© 2009 Interactive Brokers LLC. All rights reserved.

Sun, Sun Microsystems, the Sun Logo and Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Excel, Windows and Visual Basic (VB) are trademarks or registered trademarks of the Microsoft Corporation in the United States and/or in other countries.

Any symbols displayed within these pages are for illustrative purposes only, and are not intended to portray any recommendation.

# Contents

<b>1 Overview .....</b>	<b>13</b>
About the APIs .....	14
Configure the Application to Support API Components .....	15
Recommendations .....	17
API Logging .....	18
Example Log Entry .....	19
API Request/Server Response Message Identifiers.....	19
Historical Data Limitations .....	20
Valid Duration and Bar Size Settings for Historical Data Requests .....	21
API Orders and TWS Precautionary Settings .....	22
API Order IDs.....	23
New Order Example.....	23
Modified Order Example .....	23
Requests for Quotes (RFQs) .....	24
Submitting RFQs using the API.....	24
Delta-Neutral RFQs.....	24
RFQ Samples .....	24
Requesting Real-Time Index Premium Data .....	25
Uninstalling and Re-installing the TWS API Software on Windows .....	26
<b>2 DDE for Excel.....</b>	<b>27</b>
Getting Started with the DDE for Excel API.....	28
Download the API Components and Spreadsheet.....	28
Configure the Application to Support API Components .....	29
Open the Sample Spreadsheet.....	31
Using the DDE for Excel Sample Spreadsheet .....	32
Tickers Page .....	33
Using the Tickers Page .....	33
Tickers Page Toolbar Buttons.....	36
Basic Orders Page.....	37
Placing Orders.....	38
Placing a Combination Order .....	39

Supported Order Types.....	41
Order Attributes .....	42
Basic Orders Page Toolbar Buttons.....	43
Extended Order Attributes Page .....	44
Manually Program Extended Order Attributes.....	45
Apply Extended Order Attributes to Individual Orders and Groups of Orders.....	45
Extended Order Attributes.....	46
Conditional Orders Page .....	50
Setting Up Conditional Orders.....	50
Conditional Order Examples .....	51
If-Filled order .....	51
Price-change order .....	52
Conditional Orders Page Toolbar Buttons.....	53
Advanced Orders Page .....	54
Placing a Bracket Order .....	55
Placing a Volatility Order.....	56
Placing a Trailing Stop Limit Order .....	57
Placing a Scale Order .....	58
Placing a Relative Order.....	59
Advanced Orders Page Toolbar Buttons.....	60
Open Orders Page.....	61
Viewing Open Orders.....	62
Open Orders Tab Toolbar .....	62
Executions Page .....	63
Viewing Executions .....	63
Executions Page Toolbar Buttons .....	64
Executions Reporting Page .....	65
Running Execution Reports .....	66
Account Page .....	67
Using the Account Page .....	68
Account Page Values .....	69
Account Page Toolbar Buttons .....	73
.....	73
Portfolio Page.....	74
Viewing Your Portfolio .....	74
Portfolio Page Toolbar Buttons .....	75

Historical Data Page .....	76
Viewing Historical Data .....	77
Historical Data Page Query Specification Fields .....	79
Historical Data Page Toolbar Buttons .....	81
Market Scanner Page .....	82
Starting a Market Scanner Subscription .....	83
Market Scanner Parameters .....	83
Available Market Scanners .....	85
Market Scanner Page Toolbar Buttons.....	89
Contract Details Page .....	90
Requesting Contract Details .....	90
Contract Details Page Toolbar Buttons .....	91
Bond Contract Details Page .....	92
Requesting Bond Contract Details .....	92
Bond Contract Details Page Toolbar Buttons .....	93
Market Depth Page .....	94
Using the Market Depth Page .....	95
Market Depth Page Toolbar Buttons.....	96
Advisors Page .....	97
Allocating Shares to a Single Account .....	98
Placing an Order using an FA Account Group and Method .....	99
Placing an Order using an Allocation Profile .....	100
Advisors Page Toolbar Buttons.....	101
DDE for Excel API Reference .....	102
Viewing the Code.....	102
Modules.....	103
Named Ranges .....	103
Macros .....	104
DDE Syntax for Excel .....	105
<b>3 ActiveX .....</b>	<b>111</b>
Linking to the Application using ActiveX .....	112
Registering Third-Party ActiveX Controls .....	113
Running the ActiveX API on 64-bit Windows XP Systems .....	113
Using the Visual Basic Sample Program .....	114
ActiveX Methods .....	115

## Contents

connect()	116
disconnect()	116
reqMktDataEx()	116
reqMktData()	117
reqMktData2()	118
cancelMktData()	119
placeOrderEx()	119
placeOrder()	120
placeOrder2()	122
cancelOrder()	123
reqOpenOrders()	124
reqAllOpenOrders()	124
reqAutoOpenOrders()	124
reqExecutionsEx()	124
reqExecutions()	124
reqIds()	125
reqContractDetailsEx()	125
reqContractDetails()	126
reqContractDetails2()	127
reqMktDepthEx()	127
reqMktDepth()	128
reqMktDepth2()	129
cancelMktDepth()	129
addComboLeg()	130
clearComboLegs()	131
reqNewsBulletins()	131
cancelNewsBulletins()	131
setServerLogLevel()	131
reqManagedAccts()	132
reqAccountUpdates()	132
requestFA()	132
replaceFA()	133
reqHistoricalDataEx()	134
reqHistoricalData()	137
exerciseOptionsEx()	139
exerciseOptions()	140

reqScannerParameters()	140
reqScannerSubscriptionEx()	141
reqScannerSubscription()	141
cancelHistoricalData()	143
cancelScannerSubscription()	143
reqRealTimeBarsEx()	143
reqRealTimeBars()	144
cancelRealTimeBars()	145
reqCurrentTime()	145
createComboLegList()	145
createContract()	146
createExecutionFilter()	146
createOrder()	146
createScannerSubscription()	146
createTagValueList	146
createUnderComp()	147
reqFundamentalData()	147
cancelFundamentalData()	147
ActiveX Events	148
tickPrice()	149
tickSize()	149
tickOptionComputation()	150
tickGeneric()	150
tickString()	150
tickEFP()	151
tickSnapshotEnd()	151
orderStatus()	152
errMsg()	153
connectionClosed()	153
openOrderEx()	154
openOrder1()	154
openOrder2()	155
openOrder3()	156
openOrder4()	158
updateAccountValue()	163
updatePortfolioEx()	164

## Contents

updatePortfolio()	165
updateAccountTime()	166
nextValidId()	166
permId()	166
contractDetailsEx()	167
contractDetails()	167
contractDetailsEnd()	168
execDetailsEx()	168
execDetails()	169
execDetailsEnd()	170
updateMktDepth()	170
updateMktDepthL2()	171
updateNewsBulletin()	171
managedAccounts()	172
receiveFA()	172
historicalData()	172
bondContractDetails()	174
scannerParameters()	175
scannerDataEx()	175
scannerData()	175
scannerDataEnd()	176
realtimeBar()	176
currentTime()	177
fundamentalData()	177
ActiveX COM Objects	178
IExecution	179
IExecutionFilter	179
IContract	181
IContractDetails	183
IComboLeg	184
IComboLegList	185
IOrder	185
IOrderState	191
IScannerSubscription	192
ITagValueList	193
ITagValue	193

IUnderComp .....	193
ActiveX Properties.....	194
Placing a Combination Order .....	199
Example.....	199
<b>4 C++ .....</b>	<b>203</b>
Linking to TWS using the TwsSocketClient.dll .....	204
Using the C++ TestSocketClient Sample Program .....	209
To run the pre-built sample application.....	209
To run the TestSocketClient program from Microsoft Visual Studio 2008 .....	209
In case of errors: .....	209
Class EClientSocket Functions .....	211
EClientSocket() .....	211
eConnect() .....	212
eDisconnect().....	212
isConnected().....	212
reqMktData() .....	213
cancelMktData() .....	213
placeOrder() .....	213
cancelOrder().....	214
checkMessages() .....	214
reqOpenOrders() .....	214
reqAccountUpdates().....	214
reqExecutions() .....	214
reqIDs() .....	215
reqContractDetails().....	215
reqMktDepth() .....	215
cancelMktDepth() .....	215
reqNewsBulletins().....	216
cancelNewsBulletins() .....	216
setLogLevel() .....	216
reqAllOpenOrders() .....	216
reqAutoOpenOrders() .....	217
reqManagedAccts() .....	217
requestFA() .....	217
replaceFA() .....	218

reqHistoricalData()	218
exerciseOptions()	220
reqScannerParameters()	220
reqScannerSubscription()	220
cancelHistoricalData()	221
cancelScannerSubscription()	221
reqRealTimeBars()	221
cancelRealTimeBars()	222
reqCurrentTime()	222
serverVersion()	222
TwsConnectionTime()	222
reqFundamentalData()	222
cancelFundamentalData()	223
Class EWrapper Functions	224
tickPrice()	225
tickSize()	225
tickOptionComputation()	226
tickGeneric()	226
tickString()	227
tickEFP()	227
tickSnapshotEnd()	228
orderStatus()	229
error()	230
winError()	230
connectionClosed()	231
managedAccounts()	231
openOrder()	231
updateAccountValue()	232
updatePortfolio()	233
updateAccountTime()	233
nextValidId()	234
contractDetails()	234
contractDetailsEnd()	234
execDetails()	234
execDetailsEnd()	235
updateMktDepth()	235

updateMktDepthL2()	235
updateNewsBulletin()	236
receiveFA()	237
bondContractDetails()	237
historicalData()	237
scannerParameters()	238
scannerData()	238
scannerDataEnd()	238
realtimeBar()	239
currentTime()	239
fundamentalData()	239
SocketClient Properties	240
Execution	241
ExecutionFilter	241
Contract	243
ContractDetails	245
ComboLeg	246
Order	247
OrderState	251
ScannerSubscription	252
UnderComp	253
Placing a Combination Order	254
Example	254
<b>5 Java</b>	<b>257</b>
Linking to TWS using the Java API	258
Running the Java Test Client Sample Program	261
Java Test Client Overview	265
Package	265
TestJavaClient Classes	265
Java API Overview	267
Java EClientSocket Methods	268
EClientSocket()	269
eConnect()	269
eDisconnect()	269
isConnected()	269

reqMktData()	270
cancelMktData()	270
placeOrder()	270
cancelOrder()	270
reqOpenOrders()	271
reqAccountUpdates()	271
reqExecutions()	271
reqContractDetails()	271
reqMktDepth()	272
cancelMktDepth()	272
reqNewsBulletins()	272
cancelNewsBulletins()	272
setServerLogLevel()	272
reqAllOpenOrders	273
reqAutoOpenOrders()	273
reqManagedAccts()	273
requestFA()	273
replaceFA()	274
reqScannerParameters()	274
reqScannerSubscription()	275
cancelScannerSubscription()	275
reqHistoricalData()	276
cancelHistoricalData()	277
reqRealTimeBars()	278
cancelRealTimeBars()	278
exerciseOptions()	279
reqCurrentTime()	279
serverVersion()	279
TwsConnectionTime()	279
reqFundamentalData()	279
cancelFundamentalData()	280
Java EWrapper Methods	281
tickPrice()	282
tickSize()	283
tickOptionComputation()	283
tickGeneric()	284

tickString()	284
tickEFP()	284
tickSnapshotEnd()	285
orderStatus()	286
error()	287
connectionClosed()	288
managedAccounts()	288
openOrder()	288
updateAccountValue()	289
updatePortfolio()	290
updateAccountTime()	290
nextValidId()	290
contractDetails()	291
contractDetailsEnd()	291
bondContractDetails()	291
execDetails()	291
execDetailsEnd()	292
updateMktDepth()	292
updateMktDepthL2()	293
updateNewsBulletin()	293
receiveFA()	294
historicalData()	294
scannerParameters()	295
scannerData()	295
scannerDataEnd()	295
realtimeBar()	296
currentTime()	296
fundamentalData()	296
Java SocketClient Properties	297
Execution	298
ExecutionFilter	298
Contract	300
ContractDetails	302
ComboLeg	303
Order	304
OrderState	309

ScannerSubscription .....	310
UnderComp.....	311
Placing a Combination Order .....	312
Example.....	312
<b>6 Advisors .....</b>	<b>315</b>
Financial Advisor Orders and Account Configuration.....	316
Excel DDE Support.....	316
Support by Other API Technologies .....	316
Improved Financial Advisor Execution Reporting .....	317
Allocation Methods for Account Groups .....	318
EqualQuantity Method .....	318
NetLiq Method .....	318
AvailableEquity Method .....	318
PctChange Method.....	318
<b>7 ActiveX for Excel .....</b>	<b>321</b>
<i>Getting Started with the ActiveX for Excel API</i> .....	322
Download the API Components and Spreadsheet.....	322
Running the ActiveX for Excel API on 64-bit Windows XP Systems .....	323
Open the Sample Spreadsheet.....	323
Using the ActiveX for Excel Sample Spreadsheet .....	324
General Page .....	325
General Page Toolbar Buttons.....	327
Tickers Page .....	328
Using the Tickers Page .....	328
Tickers Page Toolbar Buttons.....	330
Bulletins Page .....	331
Bulletins Page Toolbar Buttons .....	331
Market Depth Page .....	332
Using the Market Depth Page .....	333
Market Depth Page Toolbar Buttons.....	333
Basic Orders Page.....	334
Placing Orders.....	335
Placing a Combination Order .....	336
Supported Order Types.....	338

Basic Orders Page Toolbar Buttons .....	338
Advanced Orders Page .....	339
Placing a Bracket Order .....	341
Placing a Volatility Order.....	342
Placing a Trailing Stop Limit Order .....	343
Placing a Scale Order .....	344
Placing a Relative Order.....	345
Advanced Orders Page Toolbar Buttons.....	345
Extended Order Attributes Page .....	346
Manually Program Extended Order Attributes.....	347
Apply Extended Order Attributes to Individual Orders and Groups of Orders.....	347
Open Orders Page.....	348
Viewing Open Orders.....	349
Open Orders Tab Toolbar .....	349
Account Page .....	350
Using the Account Page .....	350
Account Page Toolbar Buttons .....	352
Portfolio Page.....	353
Viewing Your Portfolio .....	353
Exercising Options .....	353
Portfolio Page Toolbar Buttons .....	354
Executions Page .....	355
Viewing Executions .....	356
Executions Page Toolbar Buttons .....	356
Historical Data Page .....	357
Viewing Historical Data .....	358
Historical Data Page Query Specification Fields .....	360
Historical Data Page Toolbar Buttons .....	362
Contract Details Page .....	363
Requesting Contract Details .....	363
Contract Details Page Toolbar Buttons .....	364
Bond Contract Details Page .....	365
Requesting Bond Contract Details .....	365
Bond Contract Details Page Toolbar Buttons .....	366
Real Time Bars Page .....	367
Real Time Bars Page Toolbar Buttons .....	368

Market Scanner Page .....	369
Starting a Market Scanner Subscription .....	370
Market Scanner Parameters .....	370
Available Market Scanners .....	372
Market Scanner Page Toolbar Buttons.....	376
Fundamentals Page.....	377
Fundamentals Page Toolbar Buttons.....	378
Advisors Page .....	379
Allocating Shares to a Single Account .....	380
Placing an Order using an FA Account Group and Method .....	381
Placing an Order using an Allocation Profile .....	382
Advisors Page Toolbar Buttons.....	383
Log Page .....	384
<b>8    POSIX.....</b>	<b>385</b>
<b>9    Reference Tables .....</b>	<b>387</b>
API Message Codes .....	388
Error Codes .....	388
System Message Codes .....	397
Warning Message Codes.....	397
Tick Types .....	398
Generic Tick Types.....	400
Using the SHORTABLE Tick.....	401
TAG Values for FUNDAMENTAL_RATIOS tickType .....	402
Extended Order Attributes .....	408
Available Market Scanners .....	412
IBAlgo Parameters .....	416

# Overview

This chapter provides an overview of the APIs (Application Programming Interfaces) available, including the following topics:

- [\*\*About the APIs\*\*](#)
- [\*\*Configure the Application to Support API Components\*\*](#)
- [\*\*Recommendations\*\*](#)
- [\*\*API Logging\*\*](#)
- [\*\*Historical Data Limitations\*\*](#)
- [\*\*API Orders and TWS Precautionary Settings\*\*](#)
- [\*\*API Order IDs\*\*](#)
- [\*\*Requests for Quotes \(RFQs\)\*\*](#)
- [\*\*Requesting Real Time Premium Data\*\*](#)
- [\*\*Uninstalling the API Software\*\*](#)

## About the APIs

We provide several APIs via which you can write custom applications that link to our system. Using an API connection you can:

- Connect to one or multiple workstations
- View market data
- Submit, modify, and cancel orders
- Download open orders when you start up your application, and
- View updated account and portfolio statuses.

To view syntax for specific functionality, see the [DDE for Excel](#), [ActiveX](#), [C++](#) or [Java](#) topics in this guide. Customers with no programming expertise should begin with the DDE for Excel section, which uses an everyday Excel® spreadsheet to link to TWS via the API.

You can connect using:

- The DDE component to link through Excel (for Windows platforms only).
- The ActiveX control to link through a Visual Basic and .NET application (on Windows platforms only).
- The Windows C++ socket client component to link through a C++ application (for Windows platforms).
- The Java API to link from a Java application (for all platforms).

**Note:** API topics are written for experienced programmers and provide little guidance for non-technical users.

### To use the API components and view sample source code and spreadsheets

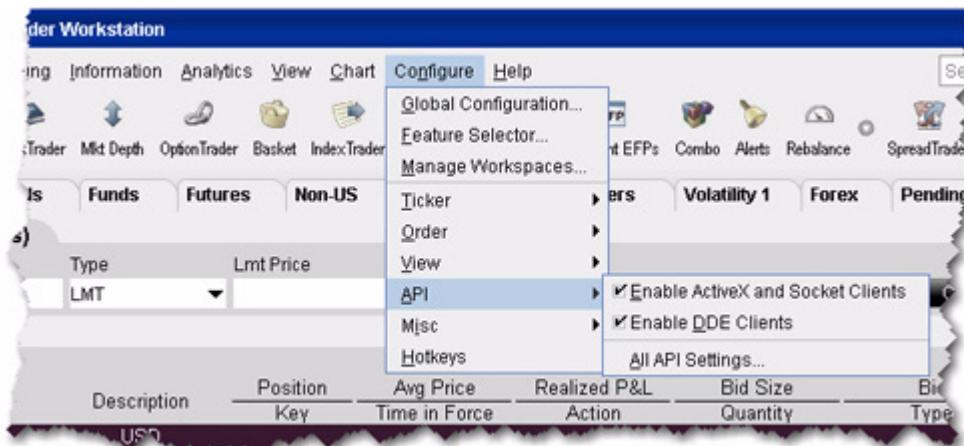
- 1 Install or upgrade the latest API and sample files (InstallIAx.exe) from the IB website. On the **Software** menu, select *Application Programming* and then select the **Proprietary API** tab. Click **Download latest version** under the appropriate OS column and install the program on your computer.
- 2 [Configure the application](#) to support the API.
- 3 Use the sample application to learn how to request market data, submit orders, etc.
- 4 Customize the sample applications to meet your needs, or create your own application using described syntax and functionality.

## Configure the Application to Support API Components

You must have your system running to use any of the API components.

### To configure the application to support accessing its functionality via the API

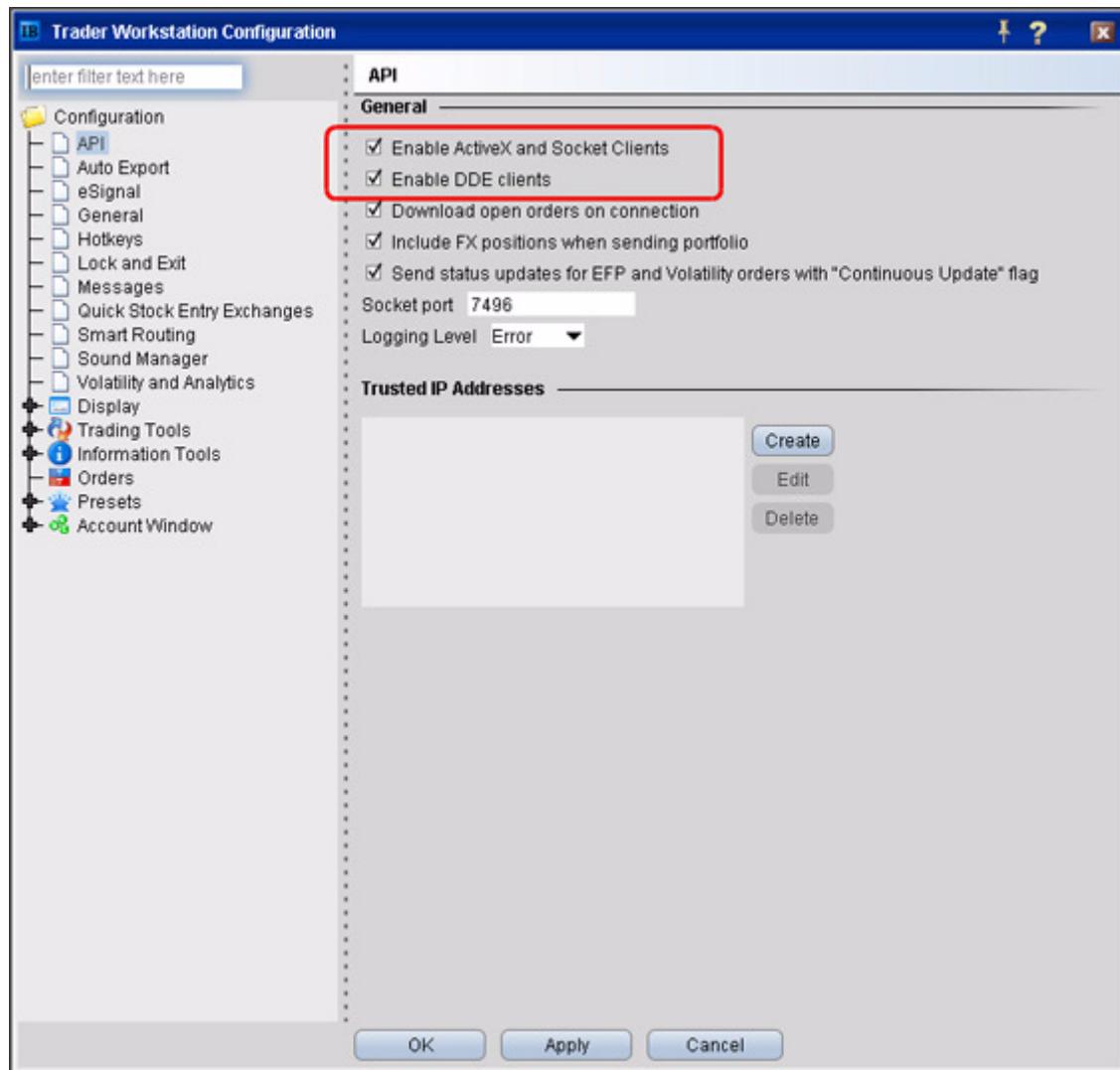
- 1 Log into TWS.
- 2 On the **Configure** menu, select **API** then select the check box for either *Enable ActiveX and Socket Clients* (ActiveX, C++ and Java API connections), or *Enable DDE Clients* (for DDE for Excel API connections only) to configure TWS for the appropriate API connection.



- You can also select *Global Configuration* from the **Configure** menu to display the TWS API Configuration window, then make your selections and click **OK**.

## Overview

Configure the Application to Support API Components



**Note:** Note that not more than one API application can simultaneously access a single instance. With the exception of DDE, the API application does not need to be running on the same computer on which the application is running.

## Recommendations

Before you use our TWS API to create your own customized trading application, you should consider the following important recommendations:

- Order IDs - Each order you place must have a unique Order ID. We recommend that you increment your own Order IDs to avoid conflicts between orders placed from your API application.
- Please test your API application with an IB Paper Trading account to catch and avoid any errors. You can request a Paper Trading account from Account Management.
- While the API supports up to eight simultaneous API connections using the same login to a single running TWS, we recommend that you avoid this scenario. If possible, use a single API connection for your application to avoid performance overhead.

## API Logging

As client requests are processed (both system and API clients) it logs certain information to its 'log.txt' log file, which is located in the installation directory. The purpose of this file is to help resolve problems by providing some insight into the state of the program before the problem occurred.

API clients can specify how detailed they want these log entries to be by setting the log level. Log levels are:

- 1 = SYSTEM (least detailed)
- 2 = ERROR (default, if no level is specified)
- 3 = WARNING
- 4 = INFORMATION
- 5 = DETAIL (most detailed)

**Note:** Setting the log level to 5 will have a performance overhead, and should only be used when trying to resolve an issue.

The log entries for API requests have the format:

[ClientID:ClientVersion:ServerVersion:ClientType:Request:Response:Version:LogEntryType]

where:

- ClientID is the clientId used when connecting.
- ClientVersion identifies the client's request stream (for internal use).
- ServerVersion identifies the server's response stream (for internal use).
- ClientType is the type of API connection: DDE = 0, Socket = 1.
- Request: If greater than 0, indicates that the log entry is the result of an API client request. The number shown is the request identifier as listed in the "Outgoing Request Identifiers" section below.
- Response: If greater than 0, indicates that the log entry is the result of a server response to the API. The number shown is the response identifier as listed in the "Incoming Response Identifiers" section below.
- Version identifies the version of the request or response message. The version changes when the message format changes.
- LogEntryLevel identifies the type of log entry (i.e. the log level as listed above)

### Example Log Entry

```
[0:9:9:1:1:0:3:DET] Socket request -
[3;52;IBM;STK;null;0.0;2;SMART;null;null]
```

From this example, we can tell that a socket client with clientId=0 connected and made a request for market data. The version of the market data request, which was 3, implies what data should have been sent.

### API Request/Server Response Message Identifiers

Outgoing Request Identifiers	Incoming Response Identifiers
1 = Request Market Data	1 = Ticker Price
2 = Cancel Market Data	2 = Ticker Size
3 = Place Order	3 = Order Status
4 = Cancel Order	4 = Error Message
5 = Request Open Orders	5 = Open Order
6 = Request Account Data	6 = Account Value
7 = Request Execution Reports	7 = Portfolio Value
8 = Request Next Order Id	8 = Account Update Time
9 = Request Contract Details	9 = Next Valid Order Id
10 = Request Market Depth	10 = Contract Details
11 = Cancel Market Depth	11 = Execution Report Details
12 = Request News Bulletins	12 = NYSE Open Book Row Entry
13 = Cancel News Bulletins	13 = Level II Quotes Row Entry
14 = Set Server Log Level	14 = News Bulletin

**Note:** This information, along with the various request/response message versions, can be found in the EClientSocket implementation file supplied with the API installation.

## Historical Data Limitations

Historical data requests are subject to the following limitations:

- Historical data requests can go back one full calendar year.
- Each request is restricted to duration and bar size values that return no more than 2000 bars (2000 bars per request).

All of the API technologies support historical data requests. However, requesting the same historical data in a short period of time can cause extra load on the backend and subsequently cause pacing violations. The error code and message that indicates a pacing violation is:

162 - Historical Market Data Service error message: Historical data request pacing violation

The following conditions can cause a pacing violation:

- Making identical historical data requests within 15 seconds;
- Making six or more historical data requests for the same Contract, Exchange and Tick Type within two seconds.

Also, observe the following limitation when requesting historical data:

- Do not make more than 60 historical data requests in any ten-minute period.

**Note:** For more information about historical data requests, see [Viewing Historical Data](#) in the DDE for Excel chapter, [reqHistoricalDataEx\(\)](#) in the ActiveX chapter, [reqHistoricalData\(\)](#) in the C++ chapter, and [reqHistoricalData\(\)](#) in the Java chapter.

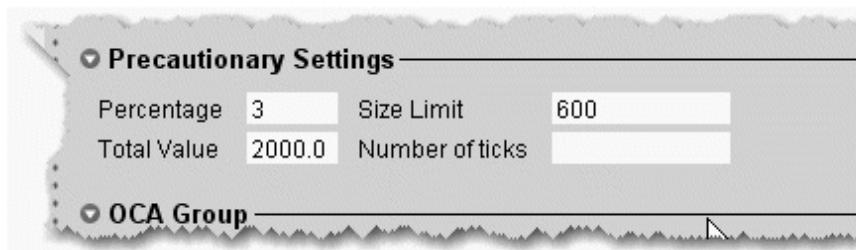
### Valid Duration and Bar Size Settings for Historical Data Requests

The following table lists valid duration and bar size settings for API historical data requests. Please note that these are only guidelines.

Duration	Bar Size
1 Y	1 day
6 M	1 day
3 M	1 day
1 M	1 day, 1 hour
1 W	1 day, 1 hour, 30 mins, 15 mins
2 D	1 hour, 30 mins, 15 mins, 3 mins, 2 mins, 1 min
1 D	1 hour, 30 mins, 15 mins, 5 mins 3 mins, 2 mins, 1 min, 30 secs
14400 S (4 hrs)	1 hour, 30 mins, 15 mins, 5 mins 3 mins, 2 mins, 1 min, 30 secs, 15 secs
7200 S (2 hrs)	1 hour, 30 mins, 15 mins, 5mins 3 mins, 2 mins, 1 min, 30 secs, 15 secs, 5 secs
3600 S (1 hr)	15 mins, 5 mins 3 mins, 2 mins, 1 min, 30 secs, 15 secs, 5 secs,
1800 S (30 mins)	15 mins, 5 mins 3 mins, 2 mins, 1 min, 30 secs, 15 secs, 5 secs, 1 secs
960 S (15 mins.)	5 mins 3 mins, 2 mins, 1 min, 30 secs, 15 secs 5 secs 1 secs
300 S (5 mins)	3 mins, 2 mins, 1 min, 30 secs, 15 secs, 5 secs, 1 secs
60 S ( 1 min)	30 secs, 15 secs, 5 secs, 1 secs

## API Orders and TWS Precautionary Settings

By default, Trader Workstation includes precautionary settings as part of its Order Presets on the TWS Configuration page. Precautionary settings are safety checks and include percentage, size limit, total value and number of ticks. They can be modified in TWS for most instrument types (stocks, options, and so on) or for specific tickers.



If your API order violates these settings, you will receive an error message. For example, the default precautionary setting for order size is 500. If you place an order for 1000 shares of stock, you will receive an error message indicating that the size specified violates the constraints specified in the default order settings. TWS precautionary settings apply to API orders placed from ALL API technologies.

You can override the precautionary settings by doing one of the following:

- In TWS, on the Configure menu, select *API* then *All API Settings*. Select the *Bypass Order Precautions for API Orders* check box, then click **OK**. All of your API orders will ignore the precautionary settings in TWS.
- In TWS, enter higher precautionary setting limits for the desired instrument types and or tickers. On the Configure menu, select *Order* then select *Order Presets*. Select the instrument type or ticker on the left, enter the desired limits in the Precautionary Settings section of the page, then click **OK**.

## API Order IDs

When you place a new order using the API, the order id number must be greater than the previously used numbers. For example, if you place an order with an Order ID of 11, the next order you place should have an Order ID of at least 12. So when you place a new order, the order id must be greater than the previously used order id number.

### New Order Example

In this example, a user is going to place two orders for IBM stock. The first order is a BUY order for 200 shares and set the limit price to \$85.25. The second order will be an order to sell 100 shares with the limit price set to \$84.25.

In this example, the first order is tagged with an Order ID of 1:

```
.placeOrder(1, IBM, BUY, $85.25, 200...)
```

Now, user can place a second order. This order is assigned an Order ID of 2:

```
.placeOrder(2, IBM, SELL, $84.25, 100...)
```

### Modified Order Example

To modify an order using the API, resubmit the order you want to modify using the same order id, but with the price or quantity modified as required. Only certain fields such as price or quantity can be altered using this method. If you want to change the order type or action, you will have to cancel the order and submit a new order.

In this example, a user initially decides to buy 100 shares and sets the limit price to \$85.25. Then, customer wants to modify the same order and change the limit price to \$86.25. Note that the first order is assigned an Order ID of 3:

```
.placeOrder(3, IBM, BUY, $85.25, 100...)
```

You can now modify the limit price for this order by calling the same .placeOrder method and using the same Order ID of 3, with the limit price modified to \$86.25

```
.placeOrder(3, IBM, BUY, $86.25, 100...)
```

## Requests for Quotes (RFQs)

RFQs from the IB Options Trading Desk allow you to get quotes for large orders from IB affiliate Timber Hill. Quotes are available for US equity and index options, and major European and Asian index options and combinations. For a complete list, please contact the [IB Options Trading Desk](#).

RFQs from the IB Options Trading Desk are available only to users who have access to these specific areas. Please contact the [IB Options Trading Desk](#) if you are interested in participating.

### Submitting RFQs using the API

Submit an RFQ by submitting an order with an order type of QUOTE. In the response, tickPrice()/tickSize() are called with the tickerId matching the orderId of the RFQ. Use orderId's with a relatively high number to avoid clashes. Additional space is required for non-RFQ tickerIds.

Market data for an RFQ is received until the user cancels the RFQ or the RFQ is canceled by the server. The server normally cancels an RFQ when it expires (approximately 1 minute) or if the RFQ request is invalid and/or for an unsupported product.

### Delta-Neutral RFQs

Submit Delta-Neutral RFQs by creating a combo order, even if a single contract must be hedged, and filling up and attaching an UnderComp structure to a contract underComp field.

In the UnderComp structure, you must specify the conId of the hedge contract. The price and delta fields can be left empty (0).

Upon accepting a Delta-Neutral DN RFQ, the server sends a deltaNeutralValidation() message with the UnderComp structure. If the delta and price fields are empty in the original request, the confirmation will contain the current values from the server. These values are locked when the RFQ is processed and remain locked until the RFQ is canceled.

### RFQ Samples

To learn more about submitting RFQs with the TWS API, look at the RFQ samples included in the 9.6 release of the API software. The samples are located in the *samples/rfq* folder in your API software installation folder. The *SampleRFQ.java* sample implements a small-state machine and shows how to submit RFQ's for:

- EU Stocks
- US Futures
- US Stock Options
- EU Stock Options
- Calendar Spread for Index Option (Delta-Neutral)
- US Stock Option (Delta-Neutral)
- US Index Option (Delta-Neutral)
- EU Index Option (Delta-Neutral)

## Requesting Real-Time Index Premium Data

You can request real-time Index Premium market data using the following APIs and API sample applications:

- ActiveX (including the ActiveX API sample application)
- C++ (including the C++ API sample application)
- Java (including the Java API sample application)
- ActiveX for Excel

To request real-time Index Premium data, you must do the following:

- Specify the Symbol, Security Type and Exchange.  
For example, INDU, IND and NYSE would get you Index Premium data for the Dow Jones Industrial Average.
- The exchange must match the index for which you want data.
- You must use the generic tick type 162 (for Index Future Premium).

## Uninstalling and Re-installing the TWS API Software on Windows

If you encounter problems running the TWS API software on the Windows platform, you can uninstall and re-install the API software.

**Note:** This procedure is usually only necessary when troubleshooting the most extreme API problems.

### To uninstall and re-install the TWS API software on Windows

- 1** Open the Windows Control Panel, then open *Add or Remove Programs*.
- 2** Select *TWS Interoperability Components* from the list of installed programs, then click **Change/Remove**.
- 3** Select *Automatic*, then click **Next** to uninstall the TWS API software.
- 4** In the Windows Explorer, delete the file *TwsSocketClient.dll* from the *Windows\system32* folder.
- 5** Reboot your computer.
- 6** Re-install the TWS API software.

# DDE for Excel

This chapter describes the DDE for Excel API, including the following topics:

- [Getting Started with the DDE for Excel API](#)
- [Using the DDE for Excel Sample Spreadsheet](#)
- [Viewing the Code](#)
- [DDE for Excel API Reference](#)

DDE is an acronym for Dynamic Data Exchange, a Microsoft-created communication method that allows multiple applications that are running simultaneously to exchange data and commands. We use this protocol to link Excel with your running version of TWS, allowing you to view real-time market data (including market depth) manage orders and monitor your executions and account information using an Excel spreadsheet.

The following figure shows the Tickers page in the Excel DDE API sample spreadsheet.

Symbol	Type	Expiry	Strike	RQ	Multiplier	Exchange	Primary Exchange	Currency	Contract Desc	Chg	Bid Imp Vol	Bid Delta	Bid Size	Bid Price	Ask Price	Ask Size
MSFT	STK					SMART	SMART	JPY		0		96	38.62	38.65	388	
YHOO	STK					SMART	SMART	JPY		0		164	20.12	20.13	133	
GE	STK					SMART	SMART	JPY		0		107	38.17	38.18	16	
0000	STK					SMART	SMART	JPY		0		1	42.47	42.76	8	
0000	STK					SMART	SMART	JPY		0		704	45.21	45.22	565	
Emi	STK					SMART	SMART	JPY		0		1	127.29	127.37	1	
MOT	STK					SMART	SMART	JPY		0		1	7.41	7.42	104	
Options	OPT	200812	229	C	100	SMART	SMART	JPY		0	0.3755722	0.990132	40	215.8	217.3	66
0000	OPT	200812	229	P	100	SMART	SMART	JPY		0	0.4339953	4.0774134	59	1.66	1.24	10
Emi	OPT	200810	135	P	100	PSE	PSE	JPY		0	0.2355923	4.6998109	181	10.3	10.5	196
Emi	OPT	200810	79	C	100	SMART	SMART	JPY		0	-	0	447	58.9	58.3	454
MSFT	OPT	200810	29	C	100	SMART	SMART	JPY		0	0.2533875	0.8391957	5306	6.65	6.25	4745
MSFT	OPT	200810	29	P	100	SMART	SMART	JPY		0	0.3999492	4.059912	9447	0.15	0.15	5294
Z	FUT	200812				LIFFE	LIFFE	JPY								
SPX	FUT	200812				CME	CME	JPY								
SPX	FUT	200812				GLOBEX	GLOBEX	JPY								
Interest Rate Futures																
ZB	FUT	200812				CBOT	CBOT	JPY								
ZB	FUT	200812				ECBOT	ECBOT	JPY								
Energy Futures																

## Getting Started with the DDE for Excel API

We have created a sample DDE-linked Excel spreadsheet, TwsDde.xls, that you can use with your TWS to create a custom Excel application. It's easy to get started with the DDE for Excel API:

- [Download the API components](#) and sample Excel spreadsheet.
- Ensure that the application server is running and that it is [configured to support DDE](#).
- [Open the spreadsheet](#) and start using the DDE for Excel API.

The sample spreadsheet currently comprises several pages complete with sample data and action buttons that make it easy for you to get market data, send orders and view your activity.

### Download the API Components and Spreadsheet

We recommend using the sample Excel spreadsheet that we provide as a starting point toward creating your own DDE for Excel API. Follow the steps below to download the sample spreadsheet.

#### To install the sample DDE Spreadsheet

- 1 From the [IB homepage](#), select *Application Programming* from the **Software** menu.
- 2 Click the *Proprietary API* tab, then In the column appropriate to your operating system, click *Download latest version*.

**Note:** Windows users can download the beta test version of the API by using the **Windows Beta** column, or revert to the previous production version by selecting *Downgrade to Previous Version*.
- 3 Save the installation program to your computer, and if desired, select a different directory. Click **Save**. Note that the API installation file is named for the API version; for example, *InstallAX\_960*.
- 4 Close any versions of TWS and Excel that you have running.
- 5 Locate the API installation program you just saved to your computer, then double-click the file to begin the API installation.
- 6 Follow the instructions in the installation wizard. By default, the sample DDE spreadsheet is saved to C:\IB\_API\_X\_XX\Excel\TwsDde.xls, where X\_XX is the API version number.

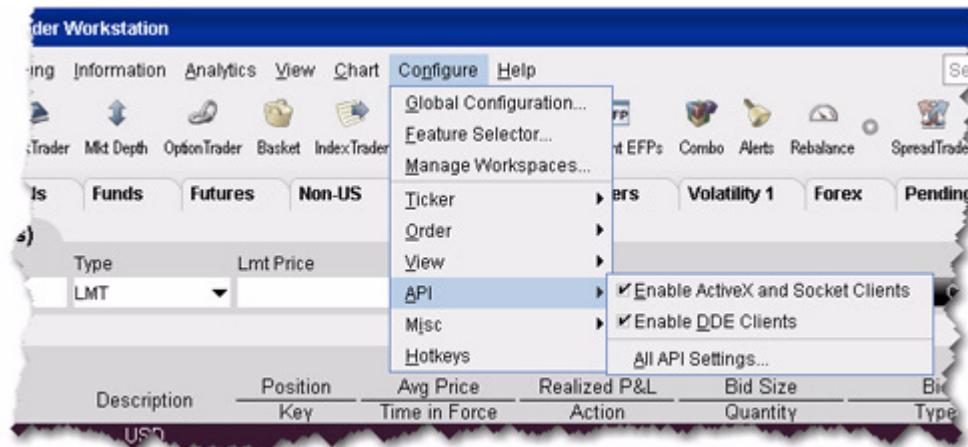
Before you can [use the spreadsheet](#), you must have TWS running and configured to support the DDE API.

## Configure the Application to Support API Components

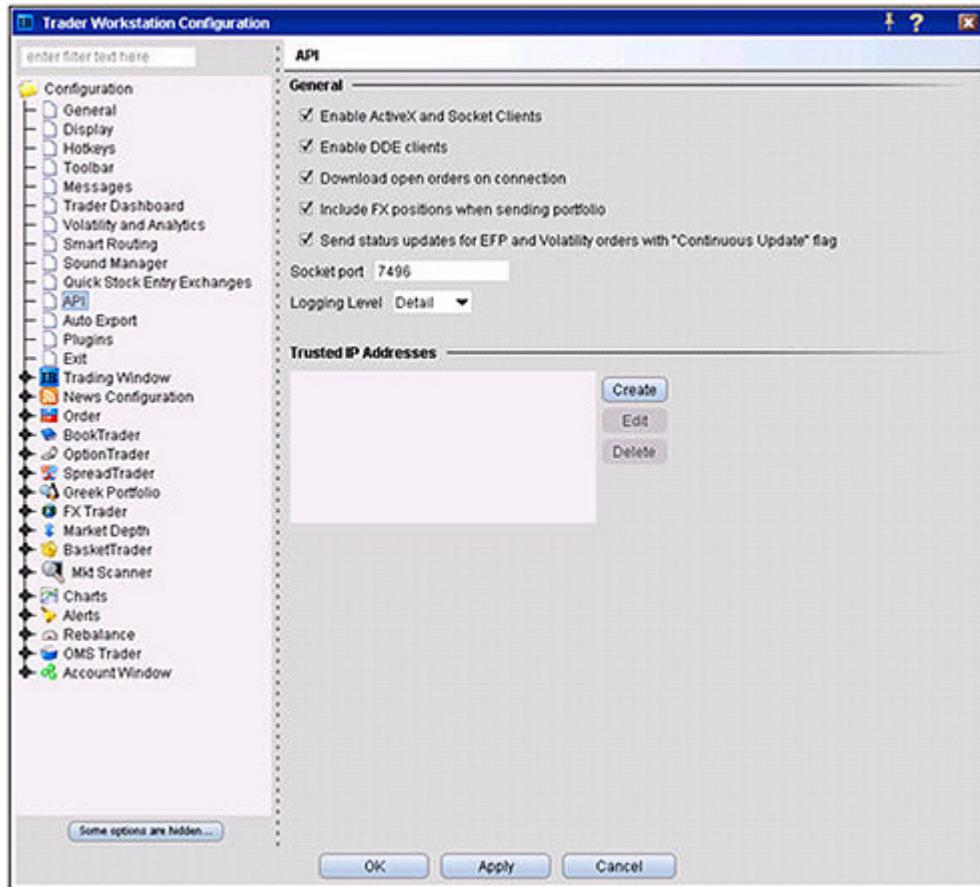
You must have your system running to use any of the API components.

### To configure the application to support accessing its functionality via the API

- 1 Run the application.
- 2 On the **Configure** menu, select *API* then select the check box for *Enable DDE Clients* (for DDE for Excel API connections only) to configure TWS.



- You can also select *Global Configuration* from the **Configure** menu to display the TWS API Configuration window, then make your selections and click **OK**.



**Note:** Note that not more than one API application can simultaneously access a single instance. With the exception of DDE, the API application does not need to be running on the same computer on which the application is running.

## Open the Sample Spreadsheet

After you have downloaded the sample spreadsheet and configured the application to allow the DDE for Excel API to link to it, open the spreadsheet and save it as your personal file.

### To open the sample spreadsheet

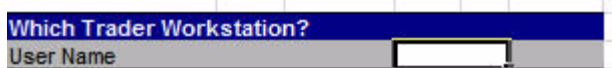
- 1 Go to the API installation folder in which the Excel API sample spreadsheet was installed (typically C:\IB\_API\_X\_XX\Excel, where X\_XX is the API version number) and double-click **TwsDde.xls**.
- 2 In the macro warning message box, click **Enable Macros**. If you receive a message asking if you want to link to information in another worksheet, click **Yes**.

**Note:** To use the spreadsheet macros, your Excel macro security must be set to Medium or Low. If you cannot open the spreadsheet or if the macros don't work, you need to modify your macro security level.

In Microsoft Excel 2007, click the Microsoft Office Button, click **Excel Options**, and then click **Trust Center** in the Excel Options window. In the Trust Center, click *Macro Settings*, then change your settings as required.

In previous versions of Excel, select *Macro* from the **Tools** menu, and then select *Security*. Set security to Medium or Low.

- 3 In the *User Name* field in the *Which Trader Workstation?* area, type your account user name. Note that you must type your User Name on each page of the worksheet to properly connect.



We recommend using this spreadsheet as the starting point for your API application. This means that when new features are added, you will need to cut and paste your information from your Excel spreadsheet to the newly released sample spreadsheet, and resave the application as your own.

## Using the DDE for Excel Sample Spreadsheet

The DDE for Excel API sample spreadsheet, TwsDde.xls, includes the following pages (tabs):

Page	Description
<a href="#"><u>Tickers</u></a>	Lets you set up your ticker lines and request market data. You can view market data for all asset types including EFPs and combination orders.
<a href="#"><u>Basic Orders</u></a>	Lets you send and modify orders, set up combination orders and EFPs, and request open orders.
<a href="#"><u>Extended Order Attributes</u></a>	Used in conjunction with the Basic Orders, Advanced Orders, Conditional Orders and Advisors pages, this page lets you change the time in force, create Hidden or Iceberg orders and apply many other order attributes.
<a href="#"><u>Conditional Orders</u></a>	Lets you create an order whose submission is contingent on other conditions being met, for example an order based on a prior fill.
<a href="#"><u>Open Orders</u></a>	Shows you all transmitted orders, including those that have been accepted by the IB system, and those that are working at an exchange.
<a href="#"><u>Advanced Orders</u></a>	Lets you send and modify advanced orders types that require the use of extended order attributes, such as Bracket, Scale and Trailing Stop Limit orders.
<a href="#"><u>Executions</u></a>	Lets you view all execution reports, and includes a filtering box so you can limit your results.
<a href="#"><u>Executions Reporting</u></a>	Linked to the Executions page, this page lets you run four different types of execution reports.
<a href="#"><u>Account</u></a>	Provides up to date account information and displays your portfolio.
<a href="#"><u>Portfolio</u></a>	Displays all your current positions.
<a href="#"><u>Historical Data</u></a>	Request historical data for an instrument based on data you enter in a query.
<a href="#"><u>Market Scanner</u></a>	Subscribe to TWS market scanners.
<a href="#"><u>Contract Details</u></a>	Lets you collect contract-specific information you will need for other actions, including the conid and supported order types for a contract
<a href="#"><u>Bond Contract Details</u></a>	Lets you collect bond contract-specific information you will need for other actions, including bond coupon and maturity date.
<a href="#"><u>Market Depth</u></a>	Lets you view market depth for selected quotes.
<a href="#"><u>Advisors</u></a>	Lets Financial Advisors send and modify FA orders.

**Note:** Two additional pages, Old Style Executions and Old Style Account-Portfolio, represent functionality that has been replaced by other pages in the spreadsheet (Executions, Account and Portfolio pages). While these older pages are still included in the TswDde.xls sample spreadsheet, they are no longer documented in this API Users' Guide and you should not use them.

## Tickers Page

Use the Tickers page to:

- Create market data (ticker) lines.
- Request market data.
- Create a combination order for options.
- Create market data line for Exchange for Physical (EFP) combination orders.

Symbol	Type	Expiry	Strike	PC	Multiplier	Exchange	Primary Exchange	Currency	Contract Desc	Chn	Bid Avg Vol	Bid Delta	Bid Size	Bid Price	Ask Price	Ask Size	Avg
MSFT	STK					SMART	ISLAND	USD		0			96	26.02	26.05	295	
YHOO	STK					SMART	ISLAND	USD		0			164	20.12	20.13	133	
GE	STK					SMART	ARCA	USD		0			107	28.17	28.18	10	
GOOG	STK					SMART	ISLAND	USD		0			1	482.47	482.76	9	
GOOG	STK					SMART	ARCA	USD		0			704	45.21	45.22	565	
BAA	STK					SMART	NYSE	USD		0			1	121.29	127.37	1	
MOT	STK					SMART		USD		0			1	7.41	7.42	164	
Options																	
GOOG	OPT	200812	279	C	100	SMART		USD		0	0.3755722	0.9993132	42	215.9	217.3	66	
GOOG	OPT	200812	279	P	100	SMART		USD		0	0.4399953	-0.0174134	58	1.05	1.24	10	
BAA	OPT	200810	135	P	100	PBE		USD		0	0.2359828	-0.0399139	101	10.3	10.6	196	
BAA	OPT	200810	79	C	100	SMART		USD		0	0.2535151	0	447	58.9	58.2	454	
MSFT	OPT	200810	23	C	100	SMART		USD		0	0.2539875	0.9991957	586	6.05	6.25	4745	
MSFT	OPT	200810	23	P	100	SMART		USD		0	0.3389402	-0.0589112	9447	0.13	0.15	5294	
Index Futures																	
Z	FUT	200812				LIFFE		GBP									
SPX	FUT	200812			250	CME		USD									
SPX	FUT	200812			250	GLOBEX		USD									
Interest Rate Futures																	
ZB	FUT	200812			1000	CBOT		USD									
ZB	FUT	200812			1000	EGBOT		USD									
Energy Futures																	

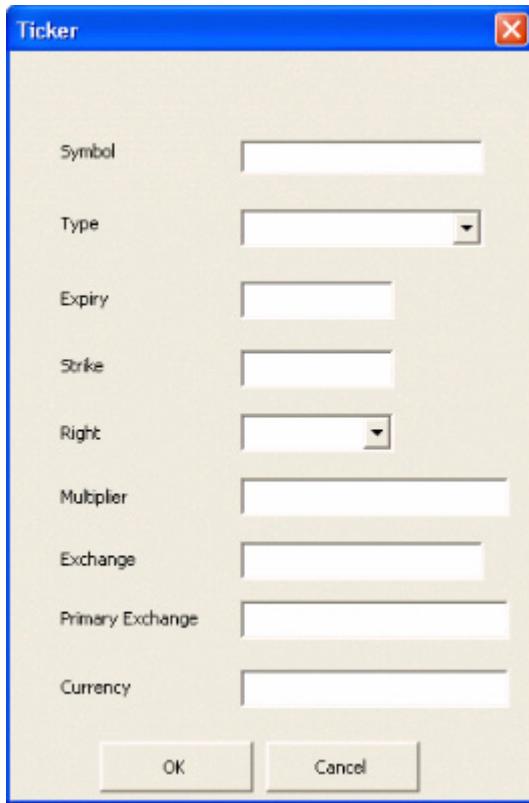
## Using the Tickers Page

**Note:** Ensure that TWS is running, and that you have entered your user name in the *User Name* field in the *Which Trader Workstation?* section of all pages in the Excel spreadsheet to properly connect to TWS.

### To create a ticker using the Create Ticker button

- 1 Click the **Tickers** tab at the bottom of the spreadsheet.
- 2 Click the line number to the left of a blank row to select the row. You must have a blank row selected to create a ticker line.
- 3 Click the **Create Ticker** button on the toolbar and enter information in the Tickers box.
- 4 Click **OK**.

For stocks, you only need to specify the *Symbol*, *Type*, *Exchange* (usually SMART), and *Currency*.



### To create a ticker on the spreadsheet

- 1 Select a blank cell in the *Symbol* column and enter a symbol.
- 2 Tab through the all contract description fields and enter data where necessary, for example if you are entering a stock ticker, you don't need values in the Expiry, Strike, P/C and Multiplier fields.

**Note:** The *Exchange* field accepts the following values: SMART (for smart order routing), and any valid exchange acronym.

### To request market data for a ticker

- 1 Select the ticker row for which you want to request market data by clicking the row number.
- 2 Press **Ctrl+R**, or click **Request Market Data** on the toolbar.

To get market data for a group of tickers, select multiple ticker rows while holding down the **Shift** key, then click **Request Market Data** multiple times until all rows are showing data.

## To set the refresh rate

The refresh rate determines how often the DDE link to TWS is refreshed.

- Type the refresh rate value (in milliseconds) in the *Refresh Rate* field, then click **Set Refresh Rate** on the toolbar.

TWS market data updates every 300 milliseconds by default, so setting the refresh rate to 250 will get every tick to the spreadsheet.

## To set the processing rate

The server processing rate affects the speed at which the DDE handles requests between TWS and the spreadsheet.

- Type the processing rate value (in milliseconds) in the *Processing Rate* field, then click **Set Processing Rate** on the toolbar.

The allowed range is 100 ms- 200 ms, inclusive.

## To set the level of detail for logging of API client requests

- 1 In the *Log Level* field in the *Which Trader Workstation?* area, enter the desired log level value (1 =SYSTEM, 2=ERROR, 3=WARNING, 4=INFORMATION, 5=DETAIL).
- 2 Move your cursor out of the *Log Level* field, then click the **Set Log Level** button.

## To remove all DDE links to TWS

The **Clear All Links** button on the Tickers page lets you remove all DDE links from the TwsDde.xls spreadsheet to TWS that the Visual Basic for Applications (VBA) code provided with the spreadsheet could create. You typically use this button when you are preparing to save the spreadsheet.

Clicking this button cancels all market data, historical data, market scanner subscriptions, and other data requests. If you add your own links to existing or new pages, update the *clearAllLinks* macro to clear those links as well. Each page in the spreadsheet contains its own *clearLinks* macro; these are all called by the *clearAllLinks* macro.

**Note:** Clearing all links does NOT cancel orders.

## Tickers Page Toolbar Buttons

The toolbar on the Tickers page includes the buttons described below.

Button	Description
<b>Create Ticker</b>	Opens the Ticker box. Enter information to create a market data line.
<b>Combo Legs</b>	Opens the Combination Legs box. Enter contract details to create legs of a combination order one by one.
<b>Request Market Data</b>	Select a line and click to get market data for the selected contract.
<b>Set Refresh Rate</b>	<p>The Refresh Rate value is in milliseconds, and determines how often the DDE link to TWS is refreshed. The default refresh rate is 1000 (updates every 1 second), and the allowed range is 100ms to 2000ms, inclusive.</p> <p>Note that the TWS market data updates every 300 milliseconds. This means the default "every 1 second" rate will only show 30% of the ticks. A Refresh Rate of 250 will get every tick to the spreadsheet.</p>
<b>Set Processing Rate</b>	Set the TWS/DDE server message processing rate (also in milliseconds) to affect the speed at which DDE will handle requests between the spreadsheet and TWS. The allowed range is 100ms to 2000ms, inclusive.
<b>Set Log Level</b>	This specifies the level of log entry detail used when processing API requests. Valid values include: 1 = SYSTEM 2 = ERROR 3 = WARNING 4 = INFORMATION 5 = DETAIL
<b>Show Errors</b>	Jumps to the Error Code field and shows the error code.
<b>Show Bulletins</b>	Opens the News Bulletins message. If you subscribe to bulletins, news will appear in the RED box in the upper right corner of the spreadsheet.
<b>Clear All Links</b>	Clears all DDE links to the TWS.

## Basic Orders Page

Use the Basic Orders page to:

- Create an order.
- Create a "basket" of orders.
- Modify and cancel orders.
- Create combination orders.

Symbol	Type	Expiry	Strike	P/O	Maturity	Exchange	Primary Exchange	Currency	Comb Legs	Leave this open	Action	Order Description		
												Quantity	Order Type	Limit Price
MSFT	STK					SMART	ISLAND	USD						
YHOO	STK					SMART	ISLAND	USD						
GE	STK					SMART	ARCA	USD						
GOOG	STK					SMART	ISLAND	USD						
GOOG	STK					SMART	ARCA	USD						
BID	STK					HYSE		USD						
<b>Options</b>														
GOOG	OPT	200812	270	C	100	SMART		USD						
GOOG	OPT	200812	270	P	100	SMART		USD						
BID	OPT	200812	120	P	100	PSE		USD						
BID	OPT	200812	120	C	100	SMART		USD						
MSFT	OPT	200812	20	C	100	SMART		USD						
MSFT	OPT	200812	20	P	100	SMART		USD						
<b>Index Futures</b>														
Z	FUT	200812				LIFFE		GBP						
SPX	FUT	200812				250 CME		USD						
SPX	FUT	200812				250 GLOBEX		USD						
<b>Interest Rate Futures</b>														
ZB	FUT	200812				1000 CBOT		USD						
ZB	FUT	200812				1000 ECBOT		USD						
<b>Energy Futures</b>														
OM	FUT	200812				NYMEX		USD						

## Placing Orders

This topic describes how to place the following types of orders on the Orders page:

- Simple orders
- Basket orders
- Modified orders

**Note:** Ensure that TWS is running, and that you have entered your user name in the *User Name* field in the *Which Trader Workstation?* section of all pages in the Excel spreadsheet to properly connect to TWS.

### To place an order

- 1 Click the **Basic Orders** tab at the bottom of the spreadsheet.
- 2 Define a contract by typing a symbol in a blank *Symbol* field, then entering information in the relevant contract description fields.
- 3 Select a contract and set up the order using the *Order Description* fields.  
You must define the *Action* (Buy, Sell or Short Sell), *Quantity*, *Order Type*, *Limit Price* (unless it's a market order) and if necessary, the *Aux. Price* for order types that require it.
- 4 If desired, apply extended order attributes by clicking the **Apply Extended Template** button on the toolbar. This applies all attributes you have defined on the [Extended Order Attributes](#) page.
- 5 Click the **Place/Modify Order** button in the *Toolbar* section of the page.

### To place a "basket" of orders

- 1 Click the **Basic Orders** tab at the bottom of the spreadsheet.
- 2 Define a contract by typing a symbol in a blank *Symbol* field, then entering information in the relevant contract description fields.
- 3 Select a contract and set up the order using fields in the *Order Description* section.
- 4 Repeat Steps 1 and 2 for additional orders.
- 5 Select a group of orders.
  - To select a group of contiguous orders, highlight the first order, hold down the **Shift** key, then highlight the last order of the group.
  - To select a group of non-contiguous orders, hold the **Ctrl** key down as you select each order.
- 6 Click the **Place/Modify Order** button.

### To modify an order (or group of orders)

- 1** On the Basic Orders page, change any necessary parameters in an order or group of orders.
- 2** Select the order or a group of orders.
  - To select a group of contiguous orders, highlight the first order, hold down the **Shift** key, then highlight the last order of the group.
  - To select a group of non-contiguous orders, hold the **Ctrl** key down as you select each order.
- 3** Click the **Place/Modify Order** button.

### Placing a Combination Order

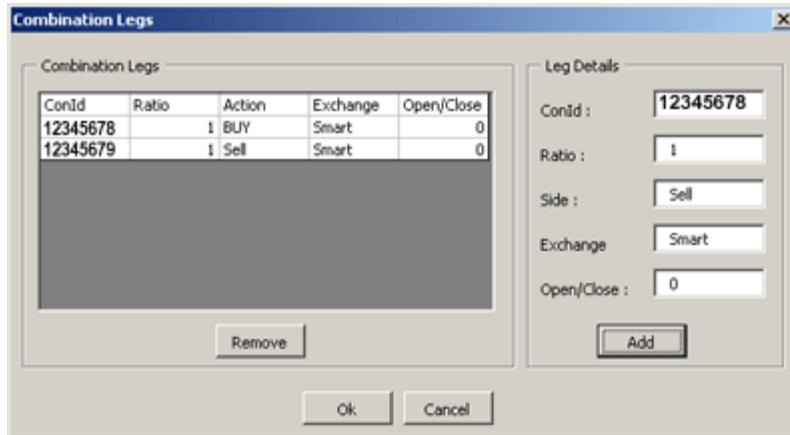
A combination order is a special type of order that is constructed of many separate legs but executed as a single transaction.

To buy a calendar spread, you would:

- Buy 1 OPT JUL03 17.5 CALL (100)
- Sell 1 OPT AUG03 17.5 CALL (100)

### To create a calendar spread order

The following example walks you through the process of placing a hypothetical calendar spread order for XYZ on ISE.



- 1** Use the [Contract Details](#) page to get the contract id for both of the leg definitions.
  - The conid for XYZ option JUL08 17.5 CALL on ISE is "12345678".
  - The conid for XYZ option AUG08 17.5 CALL on ISE is "12345679".
- 2** Click the **Basic Orders** tab to build the combo leg definitions. Click the **Combo Legs** button on the Basic Orders page toolbar and enter leg information. Your leg information is translated into the format:

[CMBLGS]\_[NumOfLegs]\_[Combo Leg Definitions]\_[CMBLGS]

where:

- [CMBLGS] is the delimiter used to identify the start and end of the leg definitions
- [NumOfLegs] is the number of leg definitions
- [Combo Leg Definitions] defines N leg definitions, and each leg definition consists of [conid]\_[ratio]\_[action]\_[exchange]\_[openClose], so the resulting combo substring looks as follows:

CMBLGS\_2\_17496957\_1\_BUY\_EMPTY\_0\_15910089\_1\_SELL\_EMPTY\_0\_CMBL  
GS

- 3** The combination leg definitions must occur before the extended order attributes. The full place order DDE request string will look like this:

```
=acctName|ord!id12345?place?BUY_1_XYZ_BAG_ISE_LMT_1_CMBLGS_2_1234  
5678_1_BUY_EMPTY_0_12345679_1_SELL_EMPTY_0_CMBLGS_DAY_EMPTY_0_O_0  
_EMPTY_0_EMPTY_0_0_0EMPTY_0_0
```

If the order legs do not constitute a valid combination, one of the following errors will be returned:

- 312 = The combo details are invalid.
- 313 = The combo details for '<leg number>' are invalid.
- 314 = Security type 'BAG' requires combo leg details.
- 315 = Stock combo legs are restricted to SMART exchange.

- Notes:**
1. The exchange for the leg definition must match that of the combination order. The exception is for a STK leg definition, which must specify the SMART exchange.
  2. The openClose leg definition value is always 'SAME' (i.e.0) for retail accounts. For institutional accounts, the value may be any of the following: (SAME, OPEN, CLOSE).

## Supported Order Types

The order types currently supported through the DDE for Excel API are:

- Limit (LMT)
- Market (MKT)
- Limit if Touched (LIT)
- Market if Touched (MIT)
- Market on Close (MOC)
- Limit on Close (LOC)
- Pegged to Market (PEGMKT)
- Relative (REL)
- Stop (STP)
- Stop Limit (STPLMT)
- Trailing Stop (TRAIL)
- Trailing Stop Limit (TRAILLIMIT)
- Volume-Weighted Average Price (VWAP)
- Volatility orders (VOL)

## Order Attributes

Field	Valid Values
side	Buy Sell
quantity	number value (1, 2, 3, etc)
symbol	any valid underlying symbol
secType	STK OPT FUT
exp	use the format: YYYYMM
strike	number value (120, 65, etc.)
right	P C for put or call
exchange	Smart any valid exchange/ECN symbol
orderType	LMT LMTCLS MKT MKTCLS PEGMKT STP STPLMT TRAIL VWAP REL
lmtPrice	number value
auxPrice	number value

## Basic Orders Page Toolbar Buttons

The toolbar on the Basic Orders page includes the following buttons:

Button	Description
<b>Combo Legs</b>	Opens the Combination Legs box. Enter contract details to create legs of a combination order one by one.
<b>Place/Modify Orders</b>	After you have completed the Order Description fields, and defined any extended attributes, click to create an order for the selected contract.
<b>Cancel Order</b>	This button cancels the order(s) you have highlighted.
<b>Apply Extended Template</b>	Applies the current values on the Extended Order Attributes page to the highlighted order row.
<b>Show Errors</b>	Jumps to the Error Code field and shows the error code.

## Extended Order Attributes Page

The Extended Order Attributes page includes all of the optional attributes you can use when you send an order, such as setting a display size to create an iceberg order, adding orders to an OCA group, and setting the transmit date for a Good After Time order. Once you define the attributes on this page, you can apply them to a single order or selected group of orders using the **Apply Extended Template** button, which occurs on both the Orders page and the Conditional Orders page. The attributes populate the extended order attributes fields that follow the *Order Status* fields to the far right of the page.

Attributes	Value	Note
TimeInForce(DAY, GTC, etc.)	DAY	The length of time over which an order will continue working before it is canceled.
OCA Group		One Cancels All Group name.
Account (Institutional only)		FINANCIAL ADVISORS ONLY - The code of a single account to which all shares of an order are allocated.
OpenClose(0=Open, C=Close) (Institutional only)	0	Sets the default to be opening a new position or closing a position.
Origin(0=Customer, 1=Firm) (Institutional only)	0	The origin of your order in relation to the broker.
Order Ref		An unique identifier you create to track your order. Can be a number or a name.
Transact(0=Fast, 1=true)	1	When set to 1, fast, all placed orders are transacted immediately. When set to 0 (false), orders are not transacted.
Parent Order Id		The ID of the parent or primary order. Use for bracket orders and auto trailing stop orders.
Block Order (0=false, 1=true)	0	Identifies a high-volume limit order as a block order.
Sweep To Fill (0=false, 1=true)	0	Identifies an order as a Sweep To Fill order.
Display Size	0	Publicly displayed order size.
Trigger Method	0	0=Default, 1=Double_Bid_Ask, 2=Last, 3=Double_Last, 4=Bid_Ask, 7=Last_or_Bid_Ask, 8=Mid-point.
Hidden (0=false, 1=true)	0	Identifies an order as hidden.
Decorative Amount (SMART Routing)		Use with a limit order to give the order a greater price range over which to execute.
Good After Time		FORMAT: 20060505 0800 00 [time zone]
Good Till Date		FORMAT: 20060505 0800 00 [time zone]
FA Group (Financial Advisors only)		FINANCIAL ADVISORS ONLY - The name of the account group to use for an order.
FA Method (Financial Advisors only)		FINANCIAL ADVISORS ONLY - The name of the default share allocation method for the account group (EqualQuantity, NetEq, Available).
FA Percentage (Financial Advisors only)		FINANCIAL ADVISORS ONLY - The percentage used by the PctChange allocation method for an order. Applies only to FA groups.
FA Profile (Financial Advisors only)		FINANCIAL ADVISORS ONLY - The name of the allocation profile to use for an order.
Short Sale Start (Institutional only)		1 If you hold the shares, 2 If they will be delivered from elsewhere. Only for Action=>SHORT.
Short Sale Location (Institutional only)		Shares will be delivered from elsewhere, specify where from in comma-delimited list, no spaces.
OCA Type		
Rule 88A		Individual = T, Agency = W, AgentOtherEnterer = W, IndividualPTA = T, AgencyPTA = U, AgentOtherMemberPTA = M, IndividualForInstiationalOnly.
Setting Firm (Institutional only)		Identifies an order as an Alter Home order.
All Or None (0=false, 1=true)		Identifies an order as a Uniform Quantity order.
Minimum Qty		Reserve orders only.
Percent Offset	0	0=Fixed, 1=From SMART Routing.
Electronic Trade Only	0	0=Fixed, 1=From SMART Routing.
FinnQuote Only		Maximum SMART order distance for the NBBO.
NBBQ Price Cap (SMART Routing)		BOX Exchange: 1 = normal, 2 = improved, 3 = transparent.
Auction Strategy		BOX Exchange.
Starting Price		BOX Exchange.
Stock Ref Price		BOX Exchange.
Data		Reserve to Stock or VOL orders.
Underlying Range (Low)		Reserve to Stock or VOL orders.
Underlying Range High		VOL orders. Note that this should be set as a percentage (e.g. 17.12 instead of 1712 as the market data is displayed).
Volatile		VOL orders.
Volatile Type (1=Daily, 2=Annual)		VOL orders.
Reference Price Type (1=Average, 2=BidOrAsk)		VOL orders. NOHIE if no cells needed is desired. Typical possibilities include MKT, LMT, REL. Send 1 for MKT or 0 for NONE to TWS.
Hedge Delta Order Type		VOL orders.
Continuous Update (0=false, 1=true)		VOL orders. specify only if cells hedge order type needs it.
Hedge Delta Aux Price		VOL orders. For sell orders, Trail Stop Price = Limit Price - Trailing Amount - Limit Offset. For buy orders, Trail Stop Price = Limit Price + Trailing Amount.
Trail Stop Price		This ATTRIBUTE IS NO LONGER SUPPORTED.
Scale Num Components		SCALE orders.
Scale Component Size		SCALE orders.
Scale Price Increment		SCALE orders.
Outside RTH (0=false, 1=true)		Allows an order to be opened or last outside of regular trading hours.

## Manually Program Extended Order Attributes

Observe the following guidelines when you manually assign an attribute:

- When appended to orderDescription, the number and order of attributes cannot be changed.
  - For any attribute that is not defined, use the value 'EMPTY' or {}. Since a string length is limited to 255 characters, we recommend using the open/close curly braces {}.
  - A place order message for a simple stock limit day order looks as follows, with the primary exchange "Supersoes" separating the extended attributes:

## Apply Extended Order Attributes to Individual Orders and Groups of Orders

Normally, values that you enter on the Extended Order Attributes page apply to all subsequent orders. However, you also can apply selected attributes to an individual order or a group of orders on the Orders page.

**Note:** You can also use this procedure to apply extended order attributes to orders on the Conditional Orders page.

#### To apply extended order attributes to individual orders or a group of orders

- 1 Enter the value or values on the Extended Order Attributes page that you want to apply to an individual order or group of orders.
  - 2 On the Orders page, select the order or group of orders.
  - 3 Click the **Apply Extended Template** button.

The extended order attributes are applied to the order(s) and the values you entered on the Extended Order Attributes page are added to the corresponding fields in the *Extended Order Attributes* section of the Orders page.

When you place the order or group of orders, the extended order attribute values you entered are applied to the order.

For example, you might want to assign a unique Order Ref number to a group or basket of orders. To do this, you would enter the number for the Order Ref attribute on the Extended Order Attributes page, then select all the orders in the group on the Orders page and click Apply Extended Template.

- 4** Delete the value of the extended order attributes you used for the order from the Extended Order Attributes page. These values will still apply to all subsequent orders that you place from the DDE for Excel API spreadsheet unless you remove the value.

## Extended Order Attributes

Attribute	Valid Values
timeInForce	DAY GTC OPG IOC
ocaGroup	String
account	String (for institutions)
open/close	O, C (for institutions)
origin	0, 1 (for institutions)
orderRef	String
transmit	0 ( <i>don't transmit</i> ) 1 (transmit)
parentId	String (the order ID used for the parent order, use for bracket and auto trailing stop orders)
blockOrder	0 (not a block order) 1 (this is a block order)
sweepToFill	0 (not a sweep-to-fill order) 1 (this is a sweep-to-fill order)
displaySize	String (publicly disclosed order size)

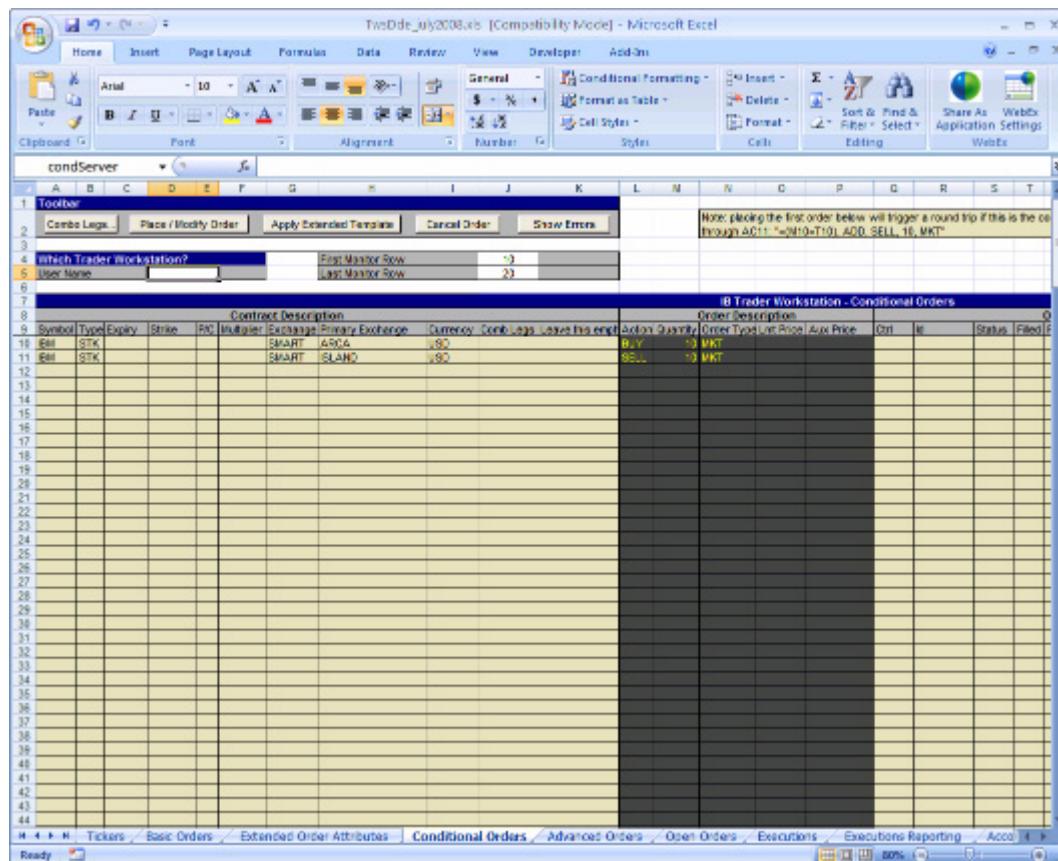
Attribute	Valid Values
triggerMethod	<p>Specifies how simulated Stop, Stop-Limit, and Trailing Stop orders are triggered.</p> <ul style="list-style-type: none"> <li>• <b>0</b> - the default value. The "double bid/ask" method will be used for orders for OTC stocks and US options. All other orders will used the "last" method.</li> <li>• <b>1</b> - use "double bid/ask" method, where stop orders are triggered based on two consecutive bid or ask prices.</li> <li>• <b>2</b> - "last" method, where stop orders are triggered based on the last price.</li> <li>• <b>3</b> - "double-last" method, where stop orders are triggered based on last two prices.</li> <li>• <b>4</b> - "bid-ask" method. For a buy order, a single occurrence of the bid price must be at or above the trigger price. For a sell order, a single occurrence of the ask price must be at or below the trigger price.</li> <li>• <b>7</b> - "last-or-bid-ask" method. For a buy order, a single bid price or the last price must be at or above the trigger price. For a sell order, a single ask price or the last price must be at or below the trigger price.</li> <li>• <b>8</b> - "mid-point" method, where the midpoint must be at or above (for a buy) or at or below (for a sell) the trigger price, and the spread between the bid and ask must be less than 0.1% of the midpoint.</li> </ul>
hidden	<p>Only valid for orders routed to Island.</p> <ul style="list-style-type: none"> <li>• 0</li> <li>• 1 (order not visible when viewing market depth)</li> </ul>
Discretionary Amount (SMART Routing)	Used in conjunction with a limit order to give the order a greater price range over which to execute.
Good After Time	Enter the date and time after which the order will become active. Use the format YYYYMMDD hh:mm:ss
Good 'Till Date	The order continues working until the close of market on the date you enter. Use the format YYYYMMDD. To specify a time of day to close the order, enter the time using the format HH:MM:SS. Specify the time zone using a valid three-letter acronym.
FA Group	For Advisor accounts only. The name of the Account Group.

Attribute	Valid Values
FA Method	For Advisor accounts only. The share allocation method. <ul style="list-style-type: none"><li>• EqualQuantity</li><li>• NetLiq</li><li>• AvailableEquity</li><li>• PctChange</li></ul>
FA Percentage	For Advisor accounts only. The share allocation percentage.
FA Profile	For Advisor accounts only. The name of the Share Allocation profile.
Short Sale Slot	For institutions only. Valid values are: <ul style="list-style-type: none"><li>• 1</li><li>• 2</li></ul>
Short Sale Location	Institutional accounts only.
OCA Type	1 = Cancel on Fill with Block 2 = Reduce on Fill with Block 3 = Reduce on Fill without Block
Rule 80A	Individual = 'I' Agency = 'A', AgentOtherMember = 'W' IndividualPTIA = 'J' AgencyPTIA = 'U' AgentOtherMemberPTIA = 'M' IndividualPT = 'K' AgencyPT = 'Y' AgentOtherMemberPT = 'N'
Settling Firm	Institutions only
All or None	0 = false 1 = true
Minimum Qty	Identifies the order as a minimum quantity order.
Percent Offset	The percent offset for relative orders.
Electronic Trade Only	0 = false 1 = true
Firm Quote Only	0 = false 1 = true
NBBO Price Cap	Maximum SMART order distance from the NBBO.
Auction Strategy	match = 1 improvement = 2 transparent = 3 For BOX exchange only.
Starting Price	The starting price. For BOX orders only.

Attribute	Valid Values
Stock Ref Price	Used for VOL orders to compute the limit price sent to an exchange (whether or not Continuous Update is used), and for price range monitoring. Also used for price improvement option orders.
Delta	The stock delta. For BOX orders only.
Underlying Range (Low)	The lower value for the acceptable underlying stock price range. For price improvement option orders on BOX and VOL orders with dynamic management.
Underlying Range (High)	The upper value for the acceptable underlying stock price range. For price improvement option orders on BOX and VOL orders with dynamic management.
Volatility	The option price in volatility, as calculated by TWS' Option Analytics. This value is expressed as a percent and is used to calculate the limit price sent to the exchange.
Volatility Type	1 = daily 2 = annual
Reference Price Type	1 = average 2 = BidOrAsk
Hedge Delta Order Type	Prior to TWS Release 859, use "1" to send a market order, "0" for no order. After TWS 859, enter an accepted order type such as: MKT, LMT, REL or MTL.
Continuous Update	0 = false 1 = true
Hedge Delta Aux Price	Enter the Aux Price for Hedge Delta order types that require one.
Trail Stop Price	Used for Trailing Stop Limit orders only. This is the stop trigger price for TRAILLMT orders.
Scale Component Size	Used for Scale orders only, this value defines the order size of the each order component.
Scale Price Increment	Used for Scale orders only, this value is used to calculate the per-unit price of each component in the order. This cannot be a negative number.
Outside RTH	0 = false 1 = true

## Conditional Orders Page

Use the Conditional Orders page to create an order whose submission is contingent on other conditions being met, for example, an order based on a prior fill or a change in the bid or ask price. **To see the Conditional Statement fields (in blue), use the scroll bar on the bottom of the page to scroll to the right.**



## Setting Up Conditional Orders

**Note:** Ensure that TWS is running, and that you have entered your user name in the *User Name* field in the *Which Trader Workstation?* section of all pages in the Excel spreadsheet to properly connect to TWS.

### To set up a conditional order

- 1 On the **Conditional Orders** page, first create the order you want transmitted when a condition is met by defining the contract in the *Contract Description* fields, and then using the *Order Description* area to set up the order parameters.
- 2 In the blue *Condition Statements* area, use the *Statement* field to set the criteria which must be met to trigger the order. When the Statement = TRUE, your order will be submitted.

The sample spreadsheet includes a pair of orders, with the second orders transmission depending on the first order being completely filled. In this case, the Statement field

trigger is that the value in cell T10 (the *Filled* field) must be equal to the value in M10 (the order *Quantity* field).

- 3 Type **ADD** in the *ADD/MOD* field because you are creating a one-time order.
- 4 Define the remaining order parameters just as you did in the *Order Description* area.

IB Trader Workstation - Conditional Orders													
Option	Order Description								Order				
	Primary Exchange	Currency	Comb Legs	Leave this empty	Action	Quantity	Order Type	Lmt Price	Aux Price	Ctr Id	St Filled	Remaining	
ARCA	USD				BUY	10	MKT			0	id1Fil	10	0
ISLAND	USD				SELL	10	MKT			0	id1Fil	10	0
	usd				buy	200	lmt	81.00					

- 5 Complete the necessary fields on the **Conditional Orders** page according to the syntax in the following table.

Field	Description
<i>Statement</i>	An Excel function which returns a true or false. When true, the order will be submitted; when false, nothing happens.
<i>ADD/MOD</i>	Use ADD for a one-time order. Use MOD to continue checking and modifying the order until it is completely filled. This is the field that activates a conditional order, and orders will be activated only with the "ADD" or "MOD" tags.
<i>Action</i>	BUY SELL
<i>Quantity</i>	Enter the quantity of the order.
<i>Order Type</i>	Refer to <a href="#">list of supported order types</a> .
<i>Lmt Price</i>	The limit price for Limit and Stop Limit order types.
<i>Aux. Price</i>	The stop-election price for Stop and Stop Limit order types, or the offset for relative orders.

All of the fields described above may be variables that depend on other cells, so any type of conditional order may be created.

## Conditional Order Examples

### If-Filled order

An if-filled order is an order that executes if a prior order executes. To create an if-filled order with the condition "If a Buy order fully executes, enter a sell limit order at a price of \$50.00":

Field	Value
<i>Statement</i>	Filled cell = 100
<i>ADD/MOD</i>	ADD
<i>Action</i>	SELL
<i>Quantity</i>	100

Field	Value
<i>Order Type</i>	LMT
<i>Lmt Price</i>	50
<i>Aux. Price</i>	empty

### Price-change order

A price-change order will be triggered if a specific bid or ask price is greater than, less than or equal to a specific price. To create a price change order with the condition "If the bid price drops below 81.20, submit a buy limit order for 200 shares with a limit price of \$81.10:

Field	Value
<i>Statement</i>	On the <b>Tickers</b> page, put your cursor in the bid price field you want to use, then copy the value that appears in the formula bar ("=" entry field) at the top of the spreadsheet. This value looks something like this:  =username tik!id4?bid where "4" identifies the bid price for a specific contract.  Paste this in the formula bar ("=" entry field) for the Statement, and add your qualifier, "=" ">" or "<" followed by the price. In this example, the formula would be:  =username tik!id4?bid<81.20
<i>ADD/MOD</i>	ADD
<i>Action</i>	BUY
<i>Quantity</i>	200
<i>Order Type</i>	LMT
<i>Lmt Price</i>	81.10
<i>Aux. Price</i>	Not used in this example.

### To modify an order (or basket of orders)

- 1** Select the order or a group of orders.
  - To select a group of contiguous orders, highlight the first order, hold down the **Shift** key, then highlight the last order of the group.
  - To select a group of non-contiguous orders, hold the **Ctrl** key down as you select each order.
- 2** Click the **Place/Modify Order** button.
- 3** Change any necessary parameters, then click the **Place/Modify Order** button.

## Conditional Orders Page Toolbar Buttons

The toolbar on the Conditional Orders page includes the following buttons:

Button	Description
<b>Combo Legs</b>	Opens the Combination Legs box. Enter contract details to create legs of a <a href="#">combination order</a> one by one.
<b>Place/Modify Order</b>	After you have completed the Order Description fields, and defined any extended attributes, click to create an order for the selected contract.
<b>Apply Extended Template</b>	Applies all attributes on the <a href="#">Extended Order Attributes page</a> to the selected order(s).
<b>Cancel Order</b>	This button cancels the order(s) you have highlighted.
<b>Show Errors</b>	Jumps to the Error Code field and shows the error code.

## Advanced Orders Page

Use the Advanced Orders page to:

- Create complex orders that require the use of extended order attributes, including:
  - Bracket orders
  - VOL orders
  - Trailing Stop Limit Orders
  - Scale Orders
  - Relative Orders

Symbol	Type	Expiry	Strike	Pct Multiplier	Exchange	Primary Exchange	Currency	Comb Legs	Leave this empty	Order Description				
										Action	Quantity	Order Type	Limit Price	
MOT	STK				SMART		USD			BUY	100	LMT		
MOT	STK				SMART		USD			SELL	100	STP		
MOT	STK				SMART		USD			SELL	100	LMT		
<b>VOL Orders</b>														
GOOG	OPT	200812	270	C	100	SMART	USD			BUY	1	VOL		
MSFT	OPT	200812	27.5	C	100	SMART	PHLX	USD		BUY	1	VOL		
<b>Trailing Stop Limit Order - SELL Order Example</b>														
DHYS	STK				SMART		USD			SELL		100	TRAILLIMIT	
<b>Scale Order - BUY LIMIT Order Example</b>														
GOOG	STK				SMART		USD			BUY	10000	LMT		
<b>Relative Order - BUY Order Example</b>														
GOOG	STK				SMART		USD			BUY	100	REL		

For more information about using extended order attributes for individual orders or groups of orders, see [Apply Extended Order Attributes to Individual Orders and Groups of Orders](#)

## Placing a Bracket Order

Bracket orders in the DDE for Excel sample spreadsheet require the use of the extended order attributes *Transmit* and *Parent Order Id*. You must turn *Transmit* off until the order is completely set up, and you must identify the first order in the bracket as the Parent Order.

### To place a Buy-Limit bracket order

- 1** Click the **Advanced Orders** tab at the bottom of the spreadsheet.
- 2** Enter the contract descriptions and order descriptions for all three orders on three contiguous rows:
  - The first order should be a BUY LMT order.
  - The second order should be a SELL STP order.
  - The third order should be a SELL LMT order.
- 3** Click the **Extended Order Attributes** tab. Change the value for *Transmit* to **0** (row 13 on the Extended Order Attributes page).

This ensures that your orders are not transmitted until you have completed the order setup.
- 4** Click the **Advanced Orders** tab, highlight the first order in the bracket order, then click the **Place/Modify Order** button.

The order is not executed, but the system generates an Order ID.
- 5** Copy the Order ID for the first order, omitting the "id" prefix, then click the **Extended Order Attributes** tab and paste the Order ID into the *Value* field for *Parent Order Id* (row 14). This value will be applied to all subsequent orders until you remove it from the Extended Order Attributes page.

The first order of the bracket order is now the primary order.
- 6** Click the **Advanced Orders** tab, highlight the second order, then click the **Place/Modify Order** button.

The order is not executed but is now associated with the primary order by means of the Parent Order Id extended order attribute.
- 7** Click the **Extended Order Attributes** tab and change the value for *Transmit* back to **1** (row 13).
- 8** Click the **Advanced Orders** tab, highlight the third order in the bracket order, then click the **Place/Modify Order** button. The entire bracket order is transmitted.
- 9** When you are done placing your bracket order, go to the **Extended Order Attributes** page and delete the *Parent Order Id* value you entered. If you do not, this value will be applied to all subsequent orders that you place in the spreadsheet.

## Placing a Volatility Order

In the DDE for Excel sample spreadsheet, you place VOL orders by entering values for the following extended order attributes:

- *Volatility*
- *Volatility Type*
- *Reference Price Type*
- *Continuous Update*
- *Underlying Range (Low)* - optional
- *Underlying Range (High)* - optional
- *Hedge Delta Order Type* - optional
- *Hedge Delta Aux Price* - optional

### To place a VOL order

- 1 Click the **Advanced Orders** tab at the bottom of the spreadsheet.
- 2 Define a contract by typing a symbol in a blank *Symbol* field, then entering information in the relevant contract description fields.
- 3 Select a contract and set up the order using the *Order Description* fields.
  - Enter **VOL** in the *Order Type* field.
- 4 Click the **Extended Order Attributes** tab. Enter values in the *Value* field for the following extended order attributes:
  - *Volatility* - This value represents the volatility to use in calculating a limit price for the option. Enter this value as a percentage, not as the market data is displayed. For example, enter 17.12 instead of .1712.
  - *Volatility Type* - Enter **1** for daily volatility or **2** for annual volatility.
  - *Reference Price Type* - This value is used to compute the limit price sent to an exchange and for stock range price monitoring. Enter **1** to use the average of the best bid and ask; or **2** to use NBB (bid) when buying a call or selling a put, or the NBO (ask) when selling a call or buying a put.
  - *Continuous Update* - Enter **1** to automatically update the option price as the underlying stock price (or futures price, for index options) moves. Enter **0** if you do not want to use this feature.
- 5 On the **Extended Order Attributes** page, enter values in the *Value* field for the following optional extended order attributes:
  - *Underlying Range (Low)* - Enter a low-end acceptable stock price relative to the selected option order. If the price of the underlying instrument falls below the lower stock range price, the option order will be canceled.

- *Underlying Range (High)* - Enter a high-end acceptable stock price relative to the selected option order. If the price of the underlying instrument rises above the higher stock range price, the option order will be canceled.
- *Hedge Delta Order Type* - Enter **LMT**, **MKT** or **REL**. Enter **NONE** if you do not want to use delta hedging.
- *Hedge Delta Aux Price* - If you have entered LMT or REL as the *Hedge Delta Order Type*, enter the price as the value for this attribute.

- 6 Click the **Advanced Orders** tab, then highlight the order row.
- 7 Click the **Apply Extended Template** button. The values you entered for the extended order attributes are applied to the order row and displayed in the *Extended Order Attributes* section of the page.
- 8 With the order row highlighted, click the **Place/Modify Order** button.
- 9 When you are done placing VOL orders, go to the **Extended Order Attributes** page and delete the VOL order values you entered. If you do not, these values will be applied to all subsequent orders that you place in the spreadsheet.

## Placing a Trailing Stop Limit Order

In TWS, there are four values that make up a trailing stop limit order:

- trailing amount
- stop price
- limit price
- limit offset

In the DDE for Excel API spreadsheet, you enter the trailing amount, stop price and limit price. There is no field or extended order attribute for the limit offset value. You must include the limit offset in the stop price (the *Trail Stop Price* extended order attribute).

### To create a Trailing Stop Limit Order

- 1 Click the **Advanced Orders** tab at the bottom of the spreadsheet.
- 2 Define a contract by typing a symbol in a blank *Symbol* field, then entering information in the relevant contract description fields.
- 3 Select a contract and set up the order using the *Order Description* fields.
  - Enter **BUY** or **SELL** in the *Action* field.
  - Enter the limit price in the *Lmt Price* field.
  - Enter **TRAILLIMIT** in the *Order Type* field.
  - Enter the trailing amount in the *Aux Price* field.

- 4** Click the **Extended Order Attributes** tab. Specify the trailing stop price as an extended order attribute. Type this value in the *Trail Stop Price Value* field.

- The Trail Stop Price value must include the limit offset.  
For a sell order:

**Trail Stop Price = Limit Price - Trailing Amount - Limit Offset**

For a buy order:

**Trail Stop Price = Limit Price + Trailing Amount + Limit Offset**

- 5** On the **Advanced Orders** page, select the order row and click the **Apply Extended Template** button. The *Trail Stop Price* value is applied to the selected order and displayed in the *Trail Stop Price* field in the *Extended Order Attributes* section of the page.
- 6** Click the **Place/Modify Order** button.
- 7** When you are done placing your order, go to the **Extended Order Attributes** page and delete the *Trail Stop Price* value you entered. If you do not, this value will be applied to all subsequent orders that you place in the spreadsheet.

## Placing a Scale Order

In the DDE for Excel sample spreadsheet, you place scale orders by entering values for the following extended order attributes:

- Scale Component Size*
- Scale Price Increment*

### To place a scale order

- 1** Click the **Advanced Orders** tab at the bottom of the spreadsheet.
- 2** Define a contract by typing a symbol in a blank *Symbol* field, then entering information in the relevant contract description fields.
- 3** Select a contract and set up the order using the *Order Description* fields. The order type should be LMT or REL.
- 4** Click the **Extended Order Attributes** tab. Enter values in the *Value* field for the following extended order attributes:
- Scale Component Size* - Enter the size of the first, or initial, order component. For example, if you submit a 10,000-share order with a Scale Component Size value of 1000, the first component will be fore 1000 shares.
  - Scale Price Increment* - Enter the amount used to calculate the per-unit price of each component in the scale ladder. This cannot be a negative number.

**Note:** As of API Release 9.41, the *Scale Num Components* not supported.

- 5 On the **Advanced Orders** page, select the order row and click the **Apply Extended Template** button. The scale order values are applied to the selected order and displayed in the *Extended Order Attributes* section of the page.
- 6 Click the **Place/Modify Order** button.
- 7 When you are done placing your order, go to the **Extended Order Attributes** page and delete the scale order values you entered. If you do not, these values will be applied to all subsequent orders that you place in the spreadsheet.

## Placing a Relative Order

In the DDE for Excel sample spreadsheet, you place relative orders by entering a value for the *Percent Offset* extended order attribute.

### To place a relative order

- 1 Click the **Advanced Orders** tab at the bottom of the spreadsheet.
- 2 Define a contract by typing a symbol in a blank *Symbol* field, then entering information in the relevant contract description fields.
- 3 Select a contract and set up the order using the *Order Description* fields.
  - Enter **REL** as the order type.
  - Enter the price cap in the *Lmt Price* cell.
- 4 Click the **Extended Order Attributes** tab. Enter a percentage in decimal form in the *Value* field for the *Percent Offset* extended order attribute.
- 5 On the **Advanced Orders** page, select the order row and click the **Apply Extended Template** button. The percent offset value is applied to the selected order and displayed in the *Extended Order Attributes* section of the page.
- 6 Click the **Place/Modify Order** button.
- 7 When you are done placing your order, go to the **Extended Order Attributes** page and delete the *Percent Offset* value you entered. If you do not, this value will be applied to all subsequent orders that you place in the spreadsheet.

## Advanced Orders Page Toolbar Buttons

The toolbar on the Advanced Orders page includes the following buttons:

Button	Description
<b>Combo Legs</b>	Opens the Combination Legs box. Enter contract details to create legs of a combination order one by one.
<b>Place/Modify Orders</b>	After you have completed the Order Description fields, and defined any extended attributes, click to create an order for the selected contract.
<b>Cancel Order</b>	This button cancels the order(s) you have highlighted.
<b>Apply Extended Template</b>	Applies the current values on the Extended Order Attributes page to the highlighted order row.
<b>Show Errors</b>	Jumps to the Error Code field and shows the error code.

## Open Orders Page

The Open Orders page shows you all transmitted orders, including those that have been accepted by the IB system, and those that are working at an exchange. Once you have subscribed, the page is updated each time you submit a new order, either through the API or in TWS.

Once an order executes, it remains on the Open Orders page for 30 seconds, with the Status value changed to FILLED. Then the filled order is cleared and you can see it on the Executions page if you subscribed to real-time executions.

## Viewing Open Orders

**Note:** Ensure that TWS is running, and that you have entered your user name in the *User Name* field in the *Which Trader Workstation?* section of all pages in the Excel spreadsheet to properly connect to TWS.

### To view open orders:

- 1** Click the **Open Orders** tab at the bottom of the spreadsheet.
- 2** Click **Subscribe to Open Orders** on the toolbar.

All of your open orders are displayed on the page, including orders you enter in the Excel API spreadsheet and in TWS.

Orders that fill remain on the page for 30 seconds with a value of *Fill* in the *Status* field.

### To remove open orders

- 1** Click the **Cancel Open Orders Subscription** button on the toolbar.
- 2** Click the **Clear Open Orders** button.

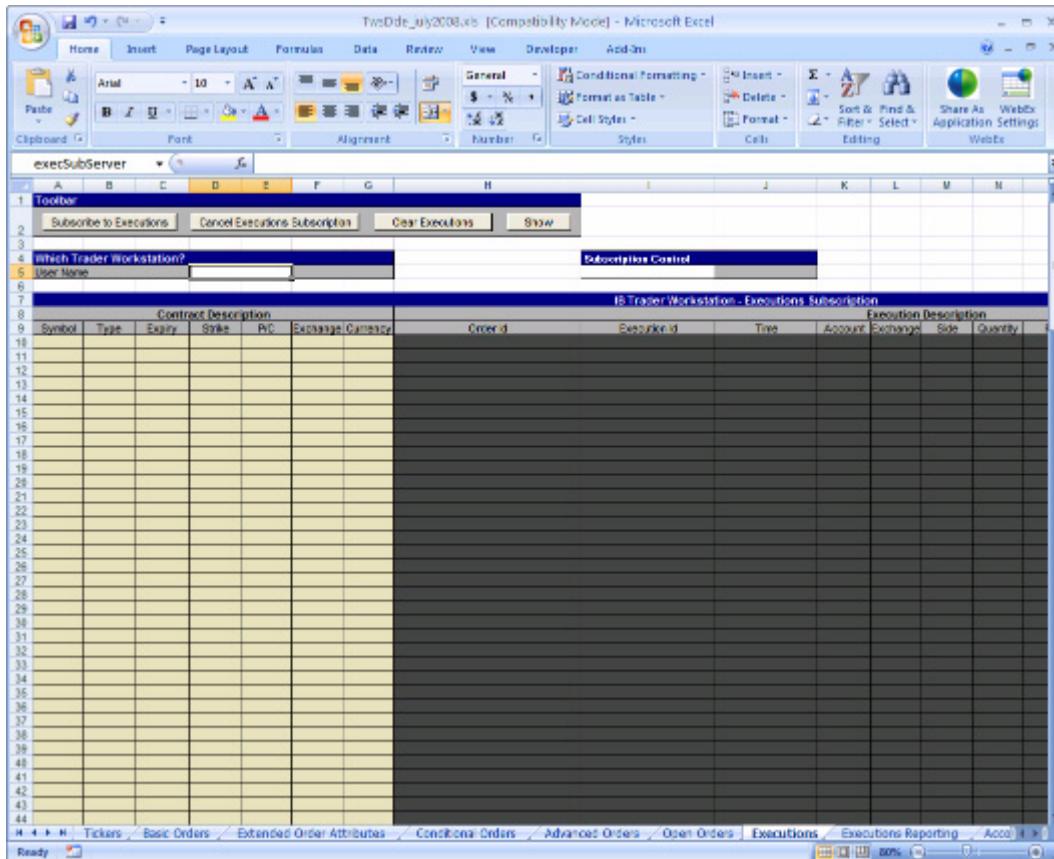
## Open Orders Tab Toolbar

The toolbar on the Open Orders page includes the following buttons:

Button	Description
<b>Subscribe to Open Orders</b>	Once you enter a valid user name, clicking this button queries TWS and returns all open orders. Once you subscribe to open orders, this page updates each time there is a new open order.
<b>Cancel Open Orders Subscription</b>	Cancels the open orders subscription. The page will no longer show your open orders.
<b>Clear Open Orders</b>	Removes all open orders from the page.
<b>Show Errors</b>	Jumps to the Error Code field and shows the error code.

## Executions Page

When you subscribe to executions, the Executions page displays information about all completed trades (also called "execution reports").



## Viewing Executions

**Note:** Ensure that TWS is running, and that you have entered your user name in the *User Name* field in the *Which Trader Workstation?* section of all pages in the Excel spreadsheet to properly connect to TWS.

### To view executions

- 1 Click the Executions tab at the bottom of the spreadsheet.
- 2 Click the **Subscribe to Executions** button in the toolbar.

### To remove execution reports

- 1 Click the **Cancel Executions Subscription** button on the toolbar.
- 2 Click the **Clear Executions** button.

## Executions Page Toolbar Buttons

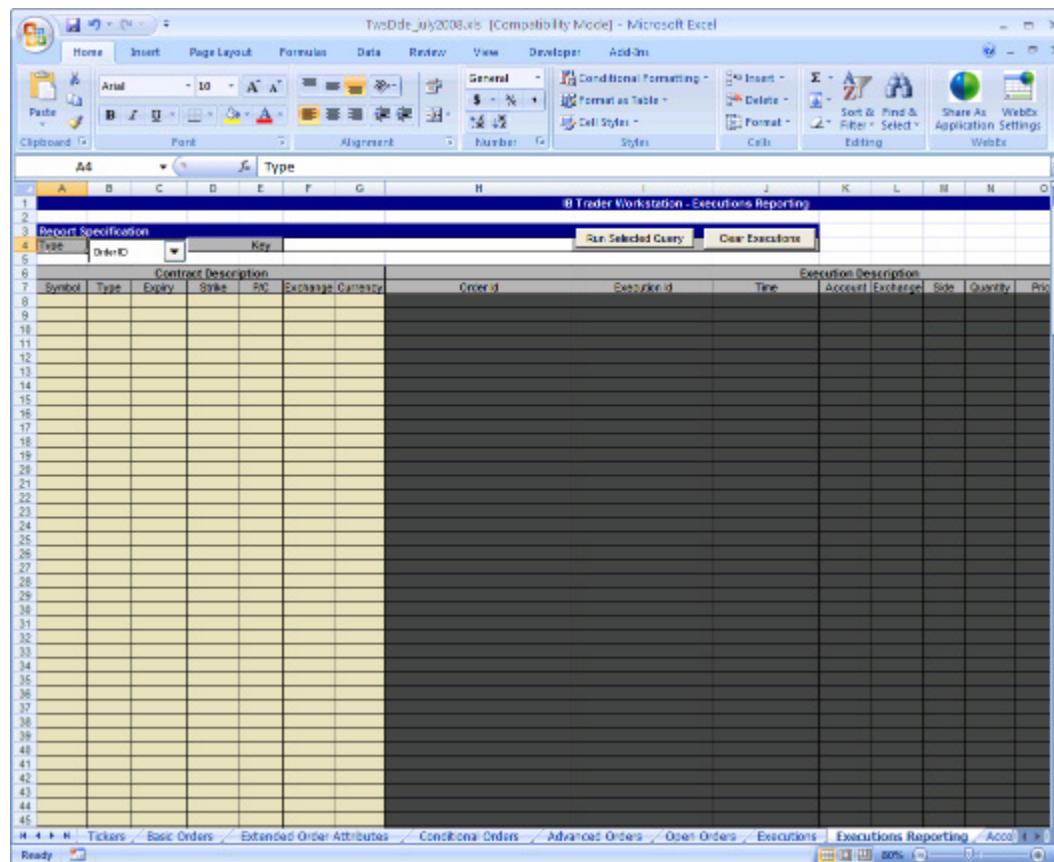
The toolbar on the Executions page includes the following buttons:

Button	Description
<b>Subscribe to Executions</b>	After you have entered a valid user name, this button queries TWS and returns information about all valid executions. After you subscribe to executions, this page updates each time an order executes.
<b>Cancel Executions Subscription</b>	Click to cancel the execution subscription.
<b>Clear Executions</b>	Removes all execution reports from the page.
<b>Show Errors</b>	Jumps to the Error Code field and shows the error code.

## **Executions Reporting Page**

Once you have subscribed to executions on the Executions page, you can use the Executions Reporting page to run reports based on an Order ID, Order Reference number, VOL order key, or strategy

From a programming point of view, the Executions Reporting page is a practical example of how you can extract array subscription data from the [named ranges](#) into which the data is put when it is received, and how such data can be used in your own custom DDE for Excel API applications.



## Running Execution Reports

**Note:** Ensure that TWS is running, and that you have entered your user name in the *User Name* field in the *Which Trader Workstation?* section of all pages in the Excel spreadsheet to properly connect to TWS.

### To run execution reports

- 1** On the Executions page, click the **Subscribe to Executions** button on the toolbar.
- 2** Click the **Executions Reporting** tab at the bottom of the worksheet.
- 3** In the *Type* field select from:
  - *Order ID* - finds all executions resulting from orders with a specified PermID.
  - *Order Ref* - finds all executions resulting from orders with a given order reference; for example executions from a specific basket order.
  - *VOL order* - finds all executions resulting from specific volatility order, including any hedge delta executions.
  - *Strategy* - in the *Key* field, enter a value to define the Type you selected. For example, if you selected *Order ID* as the type, enter a specific order ID in the *Key* field.

## Account Page

Use the Account page to:

- View account details including your current Equity with Loan Value and Available funds.
- View list of advisor-managed account codes.
- View your current portfolio.

The screenshot shows a Microsoft Excel window with the title "TwsDde\_July2008.xls [Compatibility Mode] - Microsoft Excel". The ribbon menu includes Home, Insert, Page Layout, Formulas, Data, Review, View, Developer, and Add-in. The developer tab is selected, showing various tools like Insert, Conditional Formatting, Format as Table, Cell Styles, Format, Cells, Sort & Find & Filter, and Application Settings.

The spreadsheet contains several tabs at the bottom: Extended Order Attributes, Conditional Orders, Advanced Orders, Open Orders, Executions, Executions Reporting, Account, Portfolio, and Help. The "Account" tab is currently active.

The visible data includes:

- Subscription Control:** Shows "SUBSCRIBED" in a table.
- FA Managed Accounts:** A table with columns "Ctrl" and "Accounts".
- IB Trader Workstation - Account Values:** A large table with columns "Symbol", "Value", "Currency", and "Account". It lists numerous entries such as AccountCode, AccountReady, AccountType, AccountCash, AccountCash-C, AvailableFunds, AvailableFunds-C, BuyingPower, CashBalance, Currency, and Position.
- Equity/LoanValue:** Rows 44 and 45 show "Equity/LoanValue" with values 629096.2 and 6245.95 respectively.

## Using the Account Page

**Note:** Ensure that TWS is running, and that you have entered your user name in the *User Name* field in the *Which Trader Workstation?* section of all pages in the Excel spreadsheet to properly connect to TWS.

### To view account information

- 1** Click the **Account** tab at the bottom of the spreadsheet.
- 2** Click the **Subscribe to Account Updates** button on the toolbar.

### To remove account information

- 1** Click the **Cancel Account Subscription** button.
- 2** Click the **Clear Account Data** button.

### To request the list of Financial Advisor (FA) managed account codes

- 1** Click the **Account** tab at the bottom of the spreadsheet.
- 2** Click the **Request Managed Accounts** button.

### To request details of a Financial Advisor (FA) managed account

- 1** Click the **Account** tab at the bottom of the spreadsheet.
- 2** In the *Account Code* field in the *Which Trader Workstation?* area, type the account code for which you want details.
- 3** Click the **Request Managed Accounts** button.

## Account Page Values

Field	Description	Notes
<i>Account Code</i>	The account number.	
<i>Account Ready</i>	For internal use only.	
<i>Account Type</i>	Identifies the IB account type.	
<i>Accrued Cash</i>	Reflects the current month's accrued debit and credit interest to date, updated daily.	At the beginning of each month, the past month's accrual is added to the cash balance and this field is zeroed out.
<i>Available Funds</i>	For securities: <ul style="list-style-type: none"> <li>• Equity with Loan Value - Initial margin</li> </ul> For commodities: <ul style="list-style-type: none"> <li>• Net Liquidation Value - Initial margin</li> </ul>	
<i>Buying Power</i>	Cash Account: <ul style="list-style-type: none"> <li>• (Minimum (Equity with Loan Value, Previous Day Equity with Loan Value)-Initial Margin)</li> </ul> Standard Margin Account: <ul style="list-style-type: none"> <li>• Available Funds*4</li> </ul>	
<i>Cash Balance</i>	For securities: <ul style="list-style-type: none"> <li>• Settled cash + sales at the time of trade</li> </ul> For commodities: <ul style="list-style-type: none"> <li>• Settled cash + sales at the time of trade + futures PNL</li> </ul>	
<i>Currency</i>	Shows the currency types that are listed in the Market Value area.	
<i>Cushion</i>	Shows your current margin cushion.	
<i>Day Trades Remaining</i>	Number of day trades left for pattern day trader period.	
<i>Day Trades Remaining T+1, T+2, T+3, T+4</i>	The number of day trades you have left for a 4-day pattern day-trader.	

Field	Description	Notes
<i>Equity With Loan Value</i>	<p>For Securities:</p> <ul style="list-style-type: none"> <li>• Cash Account: Settled Cash Margin Account:</li> </ul> <ul style="list-style-type: none"> <li>• Total cash value + stock value + bond value + (non-U.S. &amp; Canada securities options value)</li> </ul> <p>For Commodities:</p> <ul style="list-style-type: none"> <li>• Cash Account: Total cash value + commodities option value - futures maintenance margin requirement + minimum (0, futures PNL)</li> </ul> <p>Margin Account:</p> <ul style="list-style-type: none"> <li>• Total cash value + commodities option value - futures maintenance margin requirement</li> </ul>	
<i>Excess Liquidity</i>	Equity with Loan Value - Maintenance margin	
<i>Exchange Rate</i>	The exchange rate of the currency to your base currency.	
<i>Full Available Funds</i>	<p>For securities:</p> <ul style="list-style-type: none"> <li>• Equity with Loan Value - Initial margin</li> </ul> <p>For commodities:</p> <ul style="list-style-type: none"> <li>• Net Liquidation Value - Initial margin</li> </ul>	
<i>Full Excess Liquidity</i>	Equity with Loan Value - Maintenance margin	
<i>Full Init Margin Req</i>	Overnight initial margin requirement in the base currency of the account.	
<i>Full Maint Margin Req</i>	Maintenance margin requirement as of next period's margin change in the base currency of the account.	
<i>Future Option Value</i>	Real-time mark-to-market value of futures options.	
<i>Futures PNL</i>	Real-time change in futures value since last settlement.	
<i>Gross Position Value</i>	Long Stock Value + Short Stock Value + Long Option Value + Short Option Value.	
<i>Init Margin Req</i>	Initial margin requirement in the base currency of the account.	

Field	Description	Notes
<i>Leverage</i>	<p>For Securities:</p> <ul style="list-style-type: none"> <li>• Gross Position value / Net Liquidation value</li> </ul> <p>For Commodities:</p> <ul style="list-style-type: none"> <li>• Net Liquidation value - Initial margin</li> </ul>	
<i>Look Ahead Available Funds</i>	<p>For Securities:</p> <ul style="list-style-type: none"> <li>• Equity with loan value - look ahead initial margin.</li> </ul> <p>For Commodities:</p> <ul style="list-style-type: none"> <li>• Net Liquidation value - look ahead initial margin.</li> </ul>	
<i>Look Ahead Excess Liquidity</i>	Equity with loan value - look ahead maintenance margin.	
<i>Look Ahead Init Margin Req</i>	Initial margin requirement as of next period's margin change in the base currency of the account.	
<i>Look Ahead Maint Margin Req</i>	Maintenance margin requirement as of next period's margin change in the base currency of the account.	
<i>Maint Margin Req</i>	Maintenance margin requirement in the base currency of the account.	
<i>Net Liquidation</i>	<p>For Securities:</p> <ul style="list-style-type: none"> <li>• Total cash value + stock value + securities options value + bond value</li> </ul> <p>For Commodities:</p> <ul style="list-style-type: none"> <li>• Total cash value + commodities options value</li> </ul>	
<i>Net Liquidation by Currency</i>	Same as above for individual currencies.	
<i>Option Market Value</i>	Real-time mark-to-market value of securities options.	
<i>PNL</i>	The difference between the current market value of your open positions and the average cost, or Value - Average Cost.	
<i>Previous Day Equity with Loan Value</i>	Marginable Equity with Loan Value as of 16:00 ET the previous day, only applicable to securities.	

Field	Description	Notes
<i>Realized PnL</i>	Shows your profit on closed positions, which is the difference between your entry execution cost and exit execution cost, or (execution price + commissions to open the positions) - (execution price + commissions to close the position).	
<i>Reg T Equity</i>	Initial margin requirements calculated under US Regulation T rules.	
<i>Reg T Margin</i>	<p>For Securities:</p> <ul style="list-style-type: none"> <li>• Cash Account : Settled Cash</li> <li>• Margin Account : Total cash value + stock value + bond value + (non-U.S. &amp; Canada securities options value)</li> </ul> <p>For Commodities:</p> <ul style="list-style-type: none"> <li>• Cash Account : Total cash value + commodities option value - futures maintenance margin requirement + minimum (0, futures PNL)</li> <li>• Margin Account : Total cash value - futures maintenance margin requirement</li> </ul>	
<i>SMA</i>	Max ((EWL - US initial margin requirements)*, (Prior Day SMA +/- change in day's cash +/- US initial margin requirements**) for trades made during the day.)) *calculated end of day under US Stock rules, regardless of country of trading. **at the time of the trade	Only applicable for securities.
<i>Stock Market Value</i>	Real-time mark-to-market value of stock	
<i>Total Cash Balance</i>	Cash recognized at the time of trade + futures PNL	
<i>Total Cash Value</i>	Total cash value of stock, commodities and securities	

## Account Page Toolbar Buttons

The toolbar on the Account page includes the following buttons.

<b>Button</b>	<b>Description</b>
<b>Subscribe to Account Updates</b>	Each click gives you data for a specific account value. All blank lines that precede the Account Portfolio section will hold data. Continue to click until all lines are populated.
<b>Cancel Account Subscription</b>	Click this button one time for each position you hold. When you get a line of "0's" you know you have downloaded all current positions. These values continue to update in real-time.
<b>Clear Account Data</b>	Clears all information from the page. You must first cancel your subscription before you can clear the data.
<b>Request Managed Accounts</b>	For advisor accounts, click this button one time for each account.
<b>Show Errors</b>	Jumps to the Error Code field and shows the error code.

## Portfolio Page

The Portfolio page displays all of your current positions. This page communicates with TWS and updates the values every three minutes, which you can see in the *Last Update Time* field in the *Which Trader Workstation?* area of the page.

Symbol	SecType	Expiry	Strike	Right	Currency	LocalSymbol	Position	Market Price	Market Value	Avg Cost	Unrealized PNL	Realized PNL
AA	STK			R	\$0	AA	210	31.4156615	7618.2	38.24190475	-1222.6	
AIG	STK			R	\$0	AIG	110	25.3860066	2788.6	32.6981818	-887.2	
AMZN	STK			R	\$0	AMZN	100	77.8445656	7784.5	72.18	566.5	
AXP	STK			R	\$0	AXP	100	36.9660028	4655.6	42.8918182	-652.5	
BAC	STK			R	\$0	BAC	110	63.1600023	6941	73.3918182	-1132.1	
C	STK			R	\$0	C	100	18.2880047	2816.8	26.2418182	-215.8	
DELL	STK			R	\$0	DELL	100	24.6900015	2499	23.47	62	
DRYS	STK			R	\$0	DRYS	100	73.9700012	7397	74.59	-92	
ES	FUT	20080619		R	\$0	ESM8	15	125.25	1882.5	1757.4	-8887.5	
EUR	CASH			R	\$0	EURUSD	10000	1.55840005	28451.2	1.5917389	-488.1	
GE	STK			R	\$0	GE	200	28.25	5650	32.78	-906	
GOFQ	OPT	20081219	270 CALL	R	\$0	GOFQ	4	215.8446658	86260	23301	-6944	
GOFQ	STK			R	\$0	GOFQ	150	482.4446629	72345.75	524.0633334	-4382.75	
IBM	OPT	20081017	135 PUT	R	\$0	IBMG	-28	10.1000044	-38280	1386.3	8388.4	
IBM	STK			R	\$0	IBM	2500	127.7600022	334868.41	124.377772	8758.98	
KEY	STK			R	\$0	KEY	9802	11.3100044	116277.22	23.88911925	-121538.45	
MMV	STK			R	\$0	MMV	200	70.38500425	14079	68.67	-2055	
MOT	STK			R	\$0	MOT	400	7.46500015	2960	8.9475	-593	
MSFT	OPT	20081017	20 CALL	R	\$0	MSFT.O	3	6.17500402	1652.5	865.1666667	-863	
MSFT	OPT	20081017	20 PUT	R	\$0	MSFT.P	-15	0.14	-210	8.53333335	-92	
MSFT	STK			R	\$0	MSFT	400	26.0445693	10438	27.68098675	-634.39	
QOQ	STK			R	\$0	QOQ	100	45.38666865	4537	49.08	-371	
SOL	STK			R	\$0	SOL	105	15.6850044	267444.98	17.69948715	-34349.01	
UNH	STK			R	\$0	UNH	100	27.5245666	2752.5	36.42	-888.5	
USD	CASH			R	\$0	CAD	25000	1.02454965	25613.75	1.00955	478.25	
USD	CASH			R	\$0	CHF	25000	1.045225	26134.63	1.00955	881.88	
USD	CASH			R	\$0	PY	75000	108.125	8109375	103.4883334	347750	
XOM	STK			R	\$0	XOM	100	80.44666665	8045	83.99	-1354	
YHOO	STK			R	\$0	YHOO	300	20.12666615	6039	23.6205	-1647.75	
Z	FUT	20081219	95P	R	\$0	Z	4	53.79754105	215190	58882.95	-17141.8	

## Viewing Your Portfolio

**Note:** Ensure that TWS is running, and that you have entered your user name in the *User Name* field in the *Which Trader Workstation?* section of all pages in the Excel spreadsheet to properly connect to TWS.

### To view your portfolio

- Click the **Portfolio** tab at the bottom of the worksheet.
- Click the **Subscribe to Portfolio Updates** button.

### To remove portfolio information

- Click the **Cancel Portfolio Subscription** button.
- Click the **Clear Portfolio Data** button.

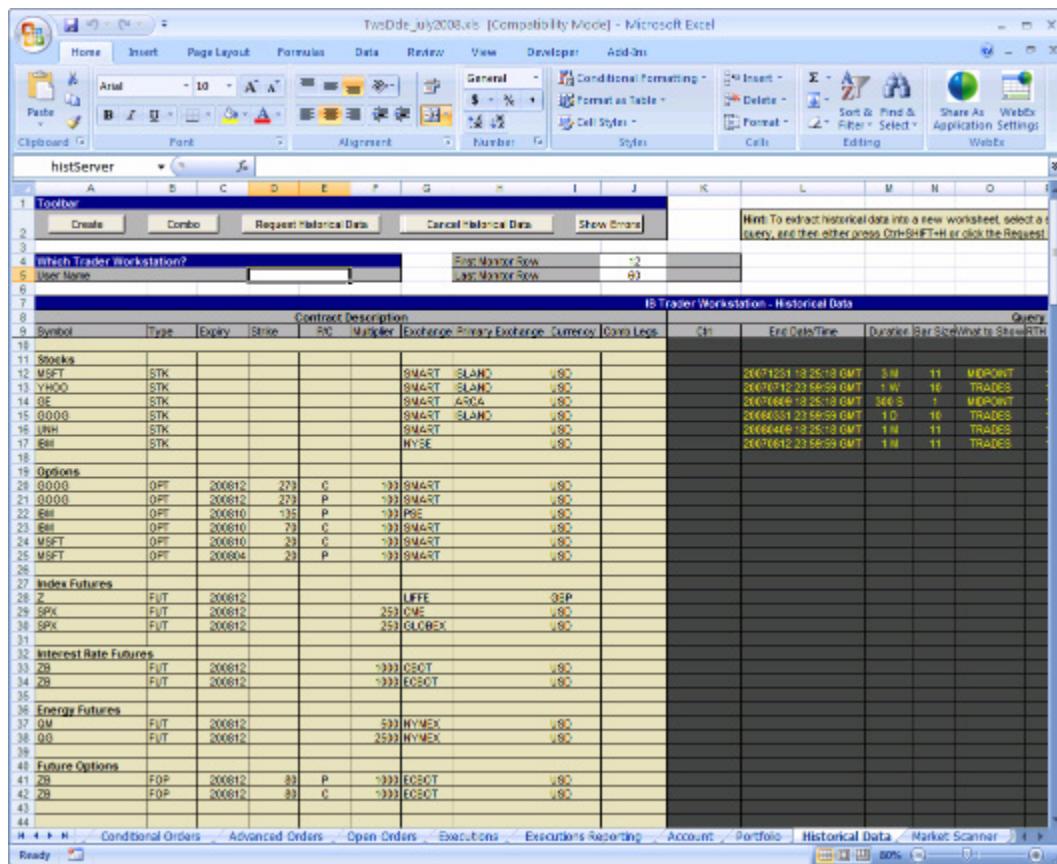
## Portfolio Page Toolbar Buttons

The toolbar on the Portfolio page includes the following buttons.

Button	Description
<b>Subscribe to Portfolio Updates</b>	Click to view all current portfolio data.
<b>Cancel Portfolio Subscription</b>	Cancels the connection to TWS that updates your portfolio information.
<b>Clear Portfolio Data</b>	Removes all data from the page. You must cancel your subscription before you can clear all data.
<b>Show Errors</b>	Jumps to the Error Code field and shows the <a href="#">error code</a> .

## Historical Data Page

Use the Historical Data page to request historical data for an instrument based on data you enter in query fields. The query results display on a separate worksheet page and creates a new page for the results if the page doesn't currently exist. Note that since the query returns in a named range of cells, you can write VBA macros to perform computations on it, and you can chart and sort the data in Excel.



**Note:** For a information about historical data request limitations, see [Historical Data Limitations](#).

## Viewing Historical Data

**Note:** Ensure that TWS is running, and that you have entered your user name in the *User Name* field in the *Which Trader Workstation?* section of all pages in the Excel spreadsheet to properly connect to TWS.

### To request historical data

- 1 Click the **Historical Data** tab at the bottom of the spreadsheet.
- 2 Create a ticker by filling in the fields in the *Contract Description* section of the page, or by clicking the **Create Ticker** button on the toolbar and [entering the required information in the Ticker box](#).
- 3 Enter the parameters of your query in the *Query Specification* fields. For complete descriptions of the query fields, [Historical Data Page Query Specification Fields](#).
- 4 Select the line, then click the **Request Historical Data** button. When the *Ctrl* field displays "Finished," the results are displayed on the specified page.

### To request historical data for expired contracts

- 1 On the Historical Data page, create a ticker by filling in the fields in the *Contract Description* section of the page, or by clicking the **Create Ticker** button on the toolbar and entering the required information in the Ticker box.
- 2 Enter the parameters of your query in the *Query Specification* fields.
- 3 In the *Expired* field in the *Query Specification* section, enter **TRUE**.
- 4 Select the line, then click the **Request Historical Data** button. When the *Ctrl* field displays "Finished," the results are displayed on the specified page.

**Note:** Historical data queries on expired contracts are limited to the last year of the life of the contract.

The following figure shows a typical historical data results page.

A	B	C	D	E	F	G	H	I	J	K	L	M
1												
2	DATE/TIME:	20060609	OPEN	HIGH	LOW	CLOSE	VOLUME	WAP	HAS GAPS			
3	20060609	14:20:18	77.89	77.89	77.89	77.89	-1	-1	FALSE			
4	20060609	14:20:19	77.89	77.89	77.89	77.89	-1	-1	FALSE			
5	20060609	14:20:20	77.89	77.89	77.89	77.89	-1	-1	FALSE			
6	20060609	14:20:21	77.89	77.89	77.89	77.89	-1	-1	FALSE			
7	20060609	14:20:22	77.89	77.89	77.89	77.89	-1	-1	FALSE			
8	20060609	14:20:23	77.89	77.89	77.89	77.89	-1	-1	FALSE			
9	20060609	14:20:24	77.89	77.89	77.89	77.89	-1	-1	FALSE			
10	20060609	14:20:25	77.89	77.89	77.89	77.89	-1	-1	FALSE			
11	20060609	14:20:26	77.89	77.89	77.89	77.89	-1	-1	FALSE			
12	20060609	14:20:27	77.89	77.89	77.89	77.89	-1	-1	FALSE			
13	20060609	14:20:28	77.89	77.89	77.89	77.89	-1	-1	FALSE			
14	20060609	14:20:29	77.89	77.89	77.89	77.89	-1	-1	FALSE			
15	20060609	14:20:30	77.89	77.89	77.89	77.89	-1	-1	FALSE			
16	20060609	14:20:31	77.89	77.89	77.89	77.89	-1	-1	FALSE			
17	20060609	14:20:32	77.89	77.89	77.89	77.89	-1	-1	FALSE			
18	20060609	14:20:33	77.89	77.89	77.89	77.89	-1	-1	FALSE			
19	20060609	14:20:34	77.89	77.89	77.89	77.89	-1	-1	FALSE			
20	20060609	14:20:35	77.89	77.89	77.89	77.89	-1	-1	FALSE			
21	20060609	14:20:36	77.89	77.89	77.89	77.89	-1	-1	FALSE			
22	20060609	14:20:37	77.89	77.89	77.89	77.89	-1	-1	FALSE			
23	20060609	14:20:38	77.89	77.89	77.89	77.89	-1	-1	FALSE			
24	20060609	14:20:39	77.89	77.89	77.89	77.89	-1	-1	FALSE			
25	20060609	14:20:40	77.89	77.89	77.89	77.89	-1	-1	FALSE			
26	20060609	14:20:41	77.89	77.89	77.89	77.89	-1	-1	FALSE			
27	20060609	14:20:42	77.89	77.89	77.89	77.89	-1	-1	FALSE			
28	20060609	14:20:43	77.89	77.89	77.89	77.89	-1	-1	FALSE			
29	20060609	14:20:44	77.89	77.89	77.89	77.89	-1	-1	FALSE			
30	20060609	14:20:45	77.89	77.89	77.89	77.89	-1	-1	FALSE			
31	20060609	14:20:46	77.89	77.89	77.89	77.89	-1	-1	FALSE			
32	20060609	14:20:47	77.89	77.89	77.89	77.89	-1	-1	FALSE			
33	20060609	14:20:48	77.89	77.89	77.89	77.89	-1	-1	FALSE			

◀ ▶ ⌂ / Conditional Orders / Executions / Account / Historical Data / Market Scanner / Contract Details / Bond Contract Details / Market Depth / HIST\_IBM /

## Historical Data Page Query Specification Fields

Parameter	Description																																		
End Date/Time	Use the format <code>yyyymmdd {space}hh:mm:ss{space}tmz</code> where the time zone is allowed (optionally) after a space at the end.																																		
Duration	<p>This is the time span the request will cover, and is specified using the format <code>integer {space} unit</code>, where valid units are:</p> <ul style="list-style-type: none"> <li>• S (seconds)</li> <li>• D (days)</li> <li>• W (weeks)</li> <li>• Y (years)</li> </ul> <p>This unit is currently limited to one. If no unit is specified, seconds are used.</p>																																		
Bar Size	<p>Specifies the size of the bars that will be returned. The following bar sizes may be used, and are specified using the parametric value:</p> <table style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; width: 40%;">Bar Size String</th> <th style="text-align: left; width: 60%;">Integer Value</th> </tr> </thead> <tbody> <tr><td>1 second</td><td style="text-align: center;">1</td></tr> <tr><td>5 seconds</td><td style="text-align: center;">2</td></tr> <tr><td>15 seconds</td><td style="text-align: center;">3</td></tr> <tr><td>30 seconds</td><td style="text-align: center;">4</td></tr> <tr><td>1 minutes</td><td style="text-align: center;">5</td></tr> <tr><td>2 minutes</td><td style="text-align: center;">6</td></tr> <tr><td>3 minutes</td><td style="text-align: center;">16</td></tr> <tr><td>5 minutes</td><td style="text-align: center;">7</td></tr> <tr><td>15 minutes</td><td style="text-align: center;">8</td></tr> <tr><td>30 minutes</td><td style="text-align: center;">9</td></tr> <tr><td>1 hour</td><td style="text-align: center;">10</td></tr> <tr><td>1 day</td><td style="text-align: center;">11</td></tr> <tr><td>1 week</td><td style="text-align: center;">12</td></tr> <tr><td>1 month</td><td style="text-align: center;">13</td></tr> <tr><td>3 months</td><td style="text-align: center;">14</td></tr> <tr><td>1 year</td><td style="text-align: center;">15</td></tr> </tbody> </table> <p>On the query return page, each "bar" is represented by a line in the spreadsheet. If you specify a duration of 300 seconds, and a bar size of "1" (one second) your return will include 300 lines, and the value in each line is equal to one second, or is a one-second bar. Note that you can use either the Integer value of the Bar Size String or the Integer Value to define the bar sizes.</p>	Bar Size String	Integer Value	1 second	1	5 seconds	2	15 seconds	3	30 seconds	4	1 minutes	5	2 minutes	6	3 minutes	16	5 minutes	7	15 minutes	8	30 minutes	9	1 hour	10	1 day	11	1 week	12	1 month	13	3 months	14	1 year	15
Bar Size String	Integer Value																																		
1 second	1																																		
5 seconds	2																																		
15 seconds	3																																		
30 seconds	4																																		
1 minutes	5																																		
2 minutes	6																																		
3 minutes	16																																		
5 minutes	7																																		
15 minutes	8																																		
30 minutes	9																																		
1 hour	10																																		
1 day	11																																		
1 week	12																																		
1 month	13																																		
3 months	14																																		
1 year	15																																		

Parameter	Description
What to Show	<p>Determines the nature of the data extracted. Valid values include:</p> <ul style="list-style-type: none"> <li>• Trades</li> <li>• Midpoint</li> <li>• Bid</li> <li>• Ask</li> <li>• Bid/Ask</li> </ul> <p>All but the Bid/Ask data contain the <i>start time, open, high, low, close, volume</i> and <i>weighted average price</i> during the time slice queried.</p> <p>For the Bid/Ask query, the <i>open</i> and <i>close</i> values are the time-weighted average bid and the time-weighted average offer, respectively. These bars are identical to the TWS charts' candlestick bars.</p>
RTH Only	<p>Regular Trading Hours only. Valid values include:</p> <ul style="list-style-type: none"> <li>• 0 - all data available during the time span requested is returned, including time intervals when the market in question was outside of regular trading hours.</li> <li>• 1 - only data within the regular trading hours for the product requested is returned, even if the time span falls partially or completely outside.</li> </ul>
Date Format Style	<p>Valid values include:</p> <ul style="list-style-type: none"> <li>• 1 - dates that apply to bars are returned in the format yyyyymmdd{space}{space}hh:mm:ss (the same format used when reporting executions).</li> <li>• 2 - the dates are returned as an integer specifying the number of seconds since 1/1/1970 GMT.</li> </ul>
Page Name	The name of the results page. This appears in the tab for the results page at the bottom of the worksheet.
Expired	<p>Valid values: TRUE, FALSE</p> <p>If TRUE, the data query can be done on an expired futures contract, limited to the last year of a contract's life.</p>

For a request with a duration of 300 seconds and a bar of one second, the query return looks like this (the scroll bar on the right side of the page allows you to scroll down and see all 300 bars).

Note that the new page is added to the right of the existing tabs on the worksheet.

## Historical Data Page Toolbar Buttons

The toolbar on the Historical Data page includes the following buttons.

Button	Description
<b>Create Ticker</b>	Opens the <a href="#">Ticker box</a> . Enter information to create a market data line.
<b>Combo Legs</b>	Opens the Combination Legs box. Enter contract details to create legs of a combination order one by one.
<b>Request Historical Data</b>	Submits your historical data query to TWS and displays the results on a separate worksheet page.
<b>Cancel Historical Data</b>	Cancels the historical data request.
<b>Show Errors</b>	Jumps to the Error Code field and shows the error code.

## Market Scanner Page

Use the Market Scanner page to subscribe to TWS market scanners. These scanners allow you to define criteria and set filters that return the top x number of underlyings which meet all scan criteria. The scan is continually updated in real time.

Row#	Crt	Page Name	Active Page	Scan Code	Instrument	Location Code	Stock Type	Number of Rows	Price Above	Price Below
19	HIGH_OPT_MP_VOLAT_OVER_HST	TRUE	HIGH_OPT_MP_VOLAT_OVER_HST	RTK	RTKJUB	Stock	20	\$		
11	LOW_OPT_MP_VOLAT_OVER_HST	TRUE	LOW_OPT_MP_VOLAT_OVER_HST	RTK	RTKJUB	Stock	20	\$		
12	HIGH_OPT_MP_VOLAT	TRUE	HIGH_OPT_MP_VOLAT	RTK	RTKJUB	Stock	20	\$		
13	LOW_OPT_MP_VOLAT	TRUE	LOW_OPT_MP_VOLAT	RTK	RTKJUB	Stock	20	\$		
14	HOT_OPT_MP_VOLAT_GAIN	TRUE	HOT_OPT_MP_VOLAT_GAIN	RTK	RTKJUB	Stock	20	\$		
15	TOP_OPT_MP_VOLAT_LOSS	TRUE	TOP_OPT_MP_VOLAT_LOSS	RTK	RTKJUB	Stock	20	\$		
16	HIGH_OPT_VOLUME_PUT_CALL_RATIO	FALSE	HIGH_OPT_VOLUME_PUT_CALL_RATIO	RTK	RTKJUB	Stock	20	\$		
17	LOW_OPT_VOLUME_PUT_CALL_RATIO	FALSE	LOW_OPT_VOLUME_PUT_CALL_RATIO	RTK	RTKJUB	Stock	20	\$		
18	OPT_VOLUME_MOST_ACTIVE	FALSE	OPT_VOLUME_MOST_ACTIVE	RTK	RTKJUB	Stock	20	\$		
19	HOT_BY_OPT_VOLUME	FALSE	HOT_BY_OPT_VOLUME	RTK	RTKJUB	Stock	20	\$		
20	HIGH_OPT_OPEN_WT_PUT_CALL_RATIO	FALSE	HIGH_OPT_OPEN_WT_PUT_CALL_RATIO	RTK	RTKJUB	Stock	20	\$		
21	LOW_OPT_OPEN_WT_PUT_CALL_RATIO	FALSE	LOW_OPT_OPEN_WT_PUT_CALL_RATIO	RTK	RTKJUB	Stock	20	\$		
22	TOP_GAIN	FALSE	TOP_GAIN	RTK	RTKJUB	Stock	20	\$		
23	MOST_ACTIVE_LIST	FALSE	MOST_ACTIVE_LIST	RTK	RTKJUB	Stock	20	\$		
24	MOST_ACTIVE_UP	FALSE	MOST_ACTIVE_UP	RTK	RTKJUB	Stock	20	\$		
25	EBO_CLOSE	FALSE	EBO_CLOSE	RTCKEJ	RTCKEJ	Stock	20	\$		
26	HOT_VOLUME	FALSE	HOT_BY_VOLUME	RTK	RTKJUB	Stock	20	\$		
27	HIGH_MP_VOL	FALSE	HIGH_OPT_MP_VOLAT	RTK	RTKJUB	Stock	20	\$		100
28	TOP_NO_GAIN	FALSE	TOP_GAIN	RTK	RTKJUB	Stock	40	\$		
29	TOP_FUT_GAIN	FALSE	TOP_FUT_GAIN	FUTUS	FUTUS	FUT	20	\$		
30	HOT_BY_PRICE	FALSE	HOT_BY_PRICE	RTK	RTKJUB	Stock	20	\$		
31	TOP_TRADE_COUNT	FALSE	TOP_TRADE_COUNT	RTK	RTKJUB	Stock	20	\$		
32	TOP_TRADE_RATE	FALSE	TOP_TRADE_RATE	RTK	RTKJUB	Stock	20	\$		
33	TOP_PRICE_RANGE	FALSE	TOP_PRICE_RANGE	RTK	RTKJUB	Stock	20	\$		
34	HOT_BY_PRICE_RANGE	FALSE	HOT_BY_PRICE_RANGE	RTK	RTKJUB	Stock	20	\$		
35	TOP_VOLUME_RATE	FALSE	TOP_VOLUME_RATE	RTK	RTKJUB	Stock	20	\$		
36	HOT_OPEN	FALSE	HOT_OPEN	RTK	RTKJUB	Stock	20	\$		
37	HALTED	FALSE	HALTED	RTK	RTKJUB	Stock	20	\$		
38	TOP_OPEN_PERC_GAIN	FALSE	TOP_OPEN_PERCENT_GAIN	RTK	RTKJUB	Stock	20	\$		
39	TOP_OPEN_PERC_LOSS	FALSE	TOP_OPEN_PERCENT_LOSS	RTK	RTKJUB	Stock	20	\$		
40	HIGH_OPEN_GAIN	FALSE	HIGH_OPEN_GAIN	RTK	RTKJUB	Stock	20	\$		
41	LOW_OPEN_GAIN	FALSE	LOW_OPEN_GAIN	RTK	RTKJUB	Stock	20	\$		
42	HIGH_OPT_MP_VOLAT_GAIN	FALSE	HIGH_OPT_MP_VOLAT_GAIN	RTK	RTKJUB	Stock	20	\$		
43	TOP_OPT_MP_VOLAT_LOSS	FALSE	TOP_OPT_MP_VOLAT_LOSS	RTK	RTKJUB	Stock	20	\$		

## Starting a Market Scanner Subscription

**Note:** Ensure that TWS is running, and that you have entered your user name in the *User Name* field in the *Which Trader Workstation?* section of all pages in the Excel spreadsheet to properly connect to TWS.

### To start a scanner subscription

- 1** Click the **Market Scanner** tab at the bottom of the spreadsheet.
- 2** Highlight an existing scanner row, or enter information for a different market scanner:
  - a** Type the name of the scan results page in the *Page Name* field.
  - b** Type **TRUE** or **FALSE** in the *Activate Page* field.

Setting these field to TRUE forces the scan results page to pop to the front of your application every time it updates. To stop this behavior, set the value of this field to FALSE.
  - c** Type values for the rest of the scan parameters in the lightly shaded section of the page.
- 3** Click the **Start Scanner Subscription** button in the toolbar. A new page for the scanner is created and is displayed after the subscription is processed.

## Market Scanner Parameters

The following table describes the market scanner parameters that make up a scanner subscription.

Parameter	Description
<i>Page name</i>	The name that will be given to the new page that is created with the scanner data.
<i>Activate Page?</i>	If set to true, the new scanner page will display on top of the worksheet every time the scan results update. This could be as often as every minute.
<i>Scan Code</i>	The type of scan.
<i>Instrument</i>	The instrument type used in the scan.
<i>Location code</i>	The market used (i.e. US Stocks) for the scan.
<i>Stock type filter</i>	Allows you to specify just stocks, just ETFs, or both.
<i>Number of rows</i>	The number of rows of data to return in the results.
<i>Price Above</i>	Filters out returns with prices below the named price. Can be left empty.
<i>Price Below</i>	Filters out returns with prices above the named price. Can be left empty.
<i>Average volume above</i>	Filters out returns with an average volume below the named price. Can be left empty.

Parameter	Description
<i>Average Option Volume Above</i>	Filters out returns with an average option volume below the named price. Can be left empty.
<i>Market Cap Above</i>	Filters out returns with a market capitalization value below the named price. Can be left empty.
<i>Market Cap Below</i>	Filters out returns with a market capitalization value above the named price. Can be left empty.
<i>Moody Rating Above</i>	Filters out returns with a Moody rating below the named price. Can be left empty.
<i>Moody Rating Below</i>	Filters out returns with a Moody rating above the named price. Can be left empty.
<i>S &amp; P Rating Above</i>	Filters out returns with an S&P rating below the named price. Can be left empty.
<i>S &amp; P Rating Below</i>	Filters out returns with an S&P rating above the named price. Can be left empty.
<i>Maturity Date Above</i>	Filters out returns with a maturity date below the named price. Can be left empty.
<i>Maturity Date Below</i>	Filters out returns with a maturity date above the named price. Can be left empty.
<i>Coupon Rate Above</i>	Filters out returns with a coupon rate below the named price. Can be left empty.
<i>Coupon Rate Below</i>	Filters out returns with a coupon rate above the named price. Can be left empty.
<i>Exclude Convertible</i>	Filters out convertible bonds. Can be left empty.
<i>Scanner Settings Pairs</i>	For example "Annual/True" used on the Top Option Implied Vol% Gainers would instruct the scan to return annualized volatilities. Delimit setting pairs by slashes.

## Available Market Scanners

The following table shows the available market scanners in the DDE for Excel API spreadsheet.

**Note:** To get more detailed market scan results than are available in the DDE for Excel API spreadsheet, run the Market Scanners in TWS.

Market Scanner (Scan Code)	Description
Low Opt Volume P/C Ratio (LOW_OPT_VOL_PUT_CALL_RATIO)*	Put option volumes are divided by call option volumes and the top underlying symbols with the lowest ratios are displayed.
High Option Imp Vol Over Historical (HIGH_OPT_IMP_VOLAT_OVER_HIST)*	Shows the top underlying contracts (stocks or indices) with the largest divergence between implied and historical volatilities.
Low Option Imp Vol Over Historical (LOW_OPT_IMP_VOLAT_OVER_HIST)*	Shows the top underlying contracts (stocks or indices) with the smallest divergence between implied and historical volatilities.
Highest Option Imp Vol (HIGH_OPT_IMP_VOLAT)*	Shows the top underlying contracts (stocks or indices) with the highest vega-weighted implied volatility of near-the-money options with an expiration date in the next two months.
Top Option Imp Vol % Gainers (TOP_OPT_IMP_VOLAT_GAIN)*	Shows the top underlying contracts (stocks or indices) with the largest percent gain between current implied volatility and yesterday's closing value of the 15 minute average of implied volatility.
Top Option Imp Vol % Losers (TOP_OPT_IMP_VOLAT_LOSE)*	Shows the top underlying contracts (stocks or indices) with the largest percent loss between current implied volatility and yesterday's closing value of the 15 minute average of implied volatility.
High Opt Volume P/C Ratio (HIGH_OPT_VOLUME_PUT_CALL_RATIO)	Put option volumes are divided by call option volumes and the top underlying symbols with the highest ratios are displayed.
Low Opt Volume P/C Ratio (LOW_OPT_VOLUME_PUT_CALL_RATIO)	Put option volumes are divided by call option volumes and the top underlying symbols with the lowest ratios are displayed.
Most Active by Opt Volume (OPT_VOLUME_MOST_ACTIVE)	Displays the most active contracts sorted descending by options volume.
Hot by Option Volume (HOT_BY_OPT_VOLUME)	Shows the top underlying contracts for highest options volume over a 10-day average.
High Option Open Interest P/C Ratio (HIGH_OPT_OPEN_INTEREST_PUT_CALL_RATIO)	Returns the top 50 contracts with the <b>highest</b> put/call ratio of outstanding option contracts.
Low Option Open Interest P/C Ratio (LOW_OPT_OPEN_INTEREST_PUT_CALL_RATIO)	Returns the top 50 contracts with the <b>lowest</b> put/call ratio of outstanding option contracts.

Market Scanner (Scan Code)	Description
Top % Gainers (TOP_PERC_GAIN)	Contracts whose last trade price shows the highest percent increase from the previous night's closing price.
Most Active (MOST_ACTIVE)	Contracts with the highest trading volume today, based on units used by TWS (lots for US stocks; contract for derivatives and non-US stocks).  The sample spreadsheet includes two Most Active scans: Most Active List, which displays the most active contracts in the NASDAQ, NYSE and AMEX markets, and Most Active US, which displays the most active stocks in the United States.
Top % Losers (TOP_PERC_LOSE)	Contracts whose last trade price shows the lowest percent increase from the previous night's closing price.
Hot Contracts by Volume (HOT_BY_VOLUME)	Contracts where: <ul style="list-style-type: none"> <li>today's Volume/avgDailyVolume is highest.</li> <li>avgDailyVolume is a 30-day exponential moving average of the contract's daily volume.</li> </ul>
Top % Futures Gainers (TOP_PERC_GAIN)	Futures whose last trade price shows the highest percent increase from the previous night's closing price.
Hot Contracts by Price (HOT_BY_PRICE)	Contracts where: <ul style="list-style-type: none"> <li>(lastTradePrice-prevClose)/avgDailyChange is highest in absolute value (positive or negative).</li> <li>The avgDailyChange is defined as an exponential moving average of the contract's (dailyClose-dailyOpen)</li> </ul>
Top Trade Count (TOP_TRADE_COUNT)	The top trade count during the day.
Top Trade Rate (TOP_TRADE_RATE)	Contracts with the highest number of trades in the past 60 seconds (regardless of the sizes of those trades).
Top Price Range (TOP_PRICE_RANGE)	The largest difference between today's high and low, or yesterday's close if outside of today's range.
Hot by Price Range (HOT_BY_PRICE_RANGE)	The largest price range (from Top Price Range calculation) over the volatility.
Top Volume Rate (TOP_VOLUME_RATE)	The top volume rate per minute.

Market Scanner (Scan Code)	Description
Lowest Option Imp Vol (LOW_OPT_IMP_VOLAT)	Shows the top underlying contracts (stocks or indices) with the lowest vega-weighted implied volatility of near-the-money options with an expiration date in the next two months.
Most Active by Opt Open Interest (OPT_OPEN_INTEREST_MOST_ACTIVE)	Returns the top 50 underlying contracts with the (highest number of outstanding call contracts) + (highest number of outstanding put contracts)
Not Open (NOT_OPEN)	Contracts that have not traded today.
Halted (HALTED)	Contracts for which trading has been halted.
Top % Gainers Since Open (TOP_OPEN_PERC_GAIN)	Shows contracts with the highest percent price INCREASE between the last trade and opening prices.
Top % Losers Since Open (TOP_OPEN_PERC_LOSE)	Shows contracts with the highest percent price DECREASE between the last trade and opening prices.
Top Close-to-Open % Gainers (HIGH_OPEN_GAP)	Shows contracts with the highest percent price INCREASE between the previous close and today's opening prices.
Top Close-to-Open % Losers (LOW_OPEN_GAP)	Shows contracts with the highest percent price DECREASE between the previous close and today's opening prices.
Lowest Option Imp Vol (LOW_OPT_IMP_VOLAT)*	Shows the top underlying contracts (stocks or indices) with the lowest vega-weighted implied volatility of near-the-money options with an expiration date in the next two months.
Top Option Imp Vol % Gainers (TOP_OPT_IMP_VOLAT_GAIN)*	Shows the top underlying contracts (stocks or indices) with the largest percent gain between current implied volatility and yesterday's closing value of the 15 minute average of implied volatility.
Top Option Imp Vol % Losers (TOP_OPT_IMP_VOLAT_LOSE)*	Shows the top underlying contracts (stocks or indices) with the largest percent loss between current implied volatility and yesterday's closing value of the 15 minute average of implied volatility.
13-Week High (HIGH_VS_13W_HL)	The highest price for the past 13 weeks.
13-Week Low (LOW_VS_13W_HL)	The lowest price for the past 13 weeks.
26-Week High (HIGH_VS_26W_HL)	The highest price for the past 26 weeks.
26-Week Low (LOW_VS_26W_HL)	The lowest price for the past 26 weeks.

Market Scanner (Scan Code)	Description
52-Week High (HIGH_VS_52W_HL)	The highest price for the past 52 weeks.
52-Week Low (LOW_VS_52W_HL)	The lowest price for the past 52 weeks.
<b>EFP</b> - High Synth Bid Rev Yield (HIGH_SYNTH_BID_REV_NAT_YIELD)	Highlights the highest synthetic EFP interest rates available. These rates are computed by taking the price differential between the SSF and the underlying stock and netting dividends to calculate an annualized synthetic implied interest rate over the period of the SSF. The High rates may present an investment opportunity.
<b>EFP</b> - Low Synth Bid Rev Yield (LOW_SYNTH_BID_REV_NAT_YIELD)	Highlights the lowest synthetic EFP interest rates available. These rates are computed by taking the price differential between the SSF and the underlying stock and netting dividends to calculate an annualized synthetic implied interest rate over the period of the SSF. The Low rates may present a borrowing opportunity.

\*30-day (V30) Implied Volatilities:

Implied volatility is calculated using a 100-step binary tree for American style options, and a Black-Scholes model for European style options. Interest rates are calculated using the settlement prices from the day's Eurodollar futures contracts, and dividends are based on historical payouts.

The IB 30-day volatility is the at-market volatility estimated for a maturity thirty calendar days forward of the current trading day. It is based on option prices from two consecutive expiration months. The first expiration month is that which has at least eight calendar days to run. The implied volatility is estimated for the eight options on the four closest to market strikes in each expiry. The implied volatilities are fit to a parabola as a function of the strike price for each expiry. The at-the-market implied volatility for an expiry is then taken to be the value of the fit parabola at the expected future price for the expiry. A linear interpolation (or extrapolation, as required) of the 30-day variance based on the squares of the at-market volatilities is performed. V30 is then the square root of the estimated variance. If there is no first expiration month with less than sixty calendar days to run, we do not calculate a V30.

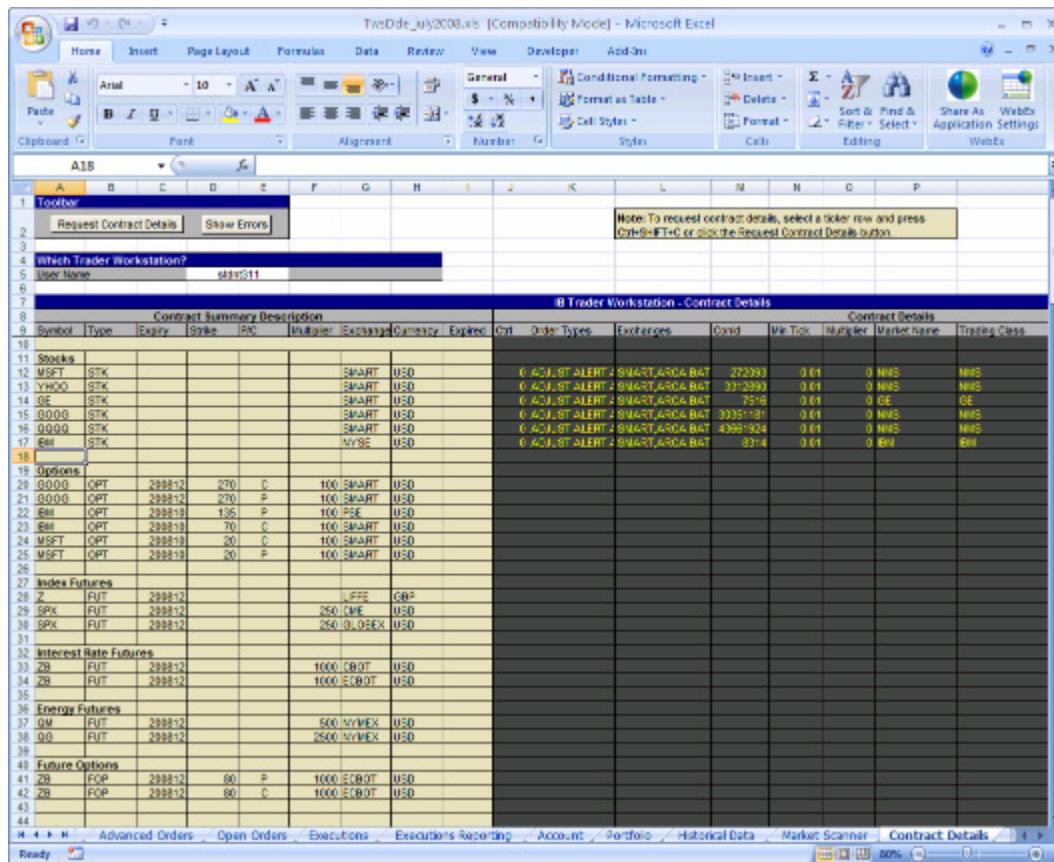
## Market Scanner Page Toolbar Buttons

The toolbar on the Market Scanner page includes the following buttons.

Button	Description
<b>Start Scanner Subscription</b>	Creates and displays a new page for results of the selected market scanner.
<b>Cancel Scanner Subscription</b>	Cancels the market scanner.
<b>Show Errors</b>	Jumps to the Error Code field and shows the error code.

## Contract Details Page

Use the Contract Details page to request contract-specific information such as supported order types, valid exchanges, the contract ID, and so on.



### Requesting Contract Details

**Note:** Ensure that TWS is running, and that you have entered your user name in the *User Name* field in the *Which Trader Workstation?* section of all pages in the Excel spreadsheet to properly connect to TWS.

#### To request details for a contract

- 1 Click the **Contract Details** tab at the bottom of the spreadsheet to open the Contract Details page.
- 2 Select or enter the ticker symbol for which you want to request contract details.
- 3 To request contract details for an expired contract, type **TRUE** in the *Expired* field.
- 4 Click the **Request Contract Details** button on the toolbar.

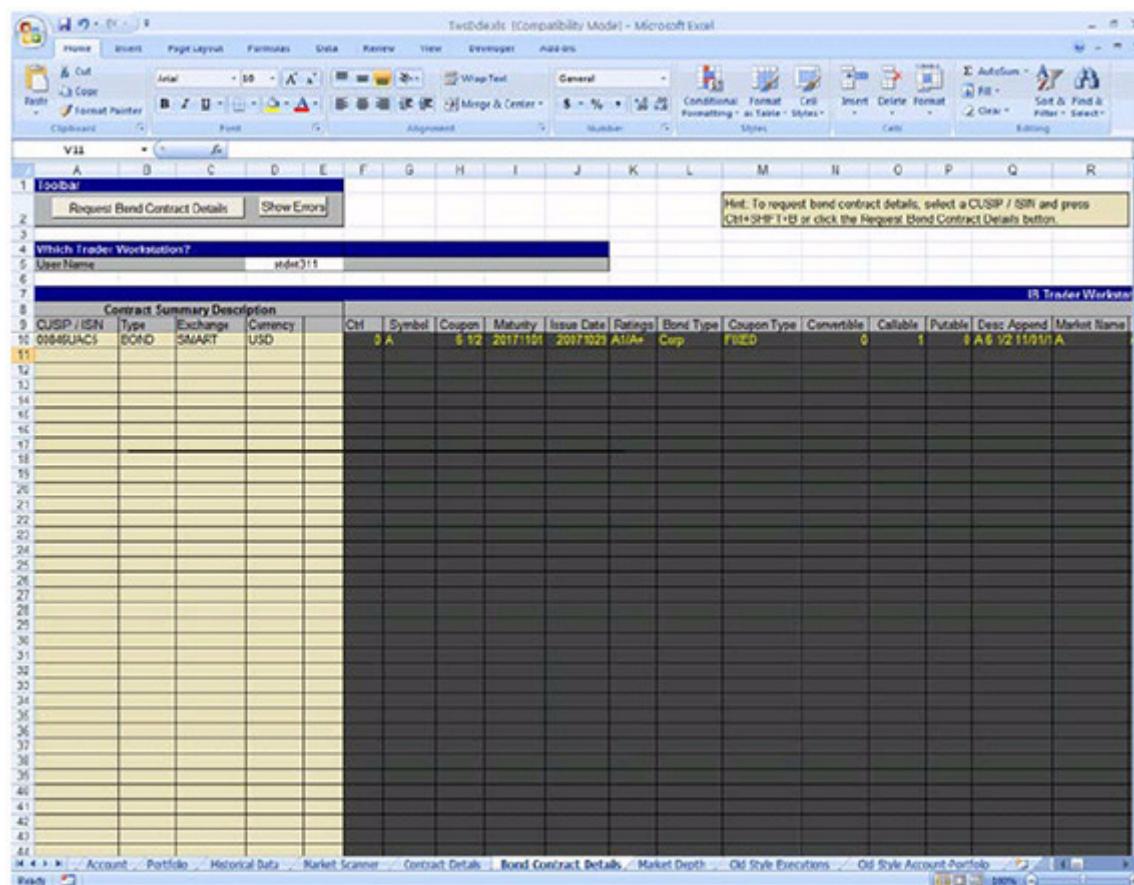
## Contract Details Page Toolbar Buttons

The toolbar on the Contract Details page includes the following buttons:

Button	Description
<b>Request Contract Details</b>	Returns information on the selected contract.
<b>Show Errors</b>	Jumps to the Error Code field and shows the error code.

## Bond Contract Details Page

Use the Bond Contract Details page to request contract-specific information for bonds, including the coupon, ratings, bond type, maturity date, and so on.



### Requesting Bond Contract Details

**Note:** Ensure that TWS is running, and that you have entered your user name in the *User Name* field in the *Which Trader Workstation?* section of all pages in the Excel spreadsheet to properly connect to TWS.

#### To request details for a bond contract

- 1 Click the **Bond Contract Details** tab at the bottom of the spreadsheet.
- 2 Enter the ticker symbol for which you want to request contract details.
- 3 Click the **Request Bond Contract Details** button on the toolbar.

## Bond Contract Details Page Toolbar Buttons

The toolbar on the Bond Contract Details page includes the following buttons:

Button	Description
<b>Request Bond Contract Details</b>	Gets bond information data for the selected contract.
<b>Show Errors</b>	Jumps to the Error Code field and shows the error code.

## Market Depth Page

Use the Market Depth page to view market depth for selected contracts. You can also view market depth for NYSE-listed products through the Open Book Market Data Subscription, and NASDAQ-listed products through the TotalView Market Data Subscriptions, if you have signed up for those subscriptions.

The screenshot shows a Microsoft Excel spreadsheet titled "TrisDde\_05/2008.xls [Compatibility Mode] - Microsoft Excel". The spreadsheet has a toolbar at the top with various icons for file operations, cell selection, and data manipulation. The menu bar includes Home, Insert, Page Layout, Formulas, Data, Review, View, Developer, and Add-Ins. The ribbon tabs at the bottom include Executions, Reporting, Account, Portfolio, Historical Data, Market Scanner, Contract Details, Bond Contract Details, Market Depth, and Advisors.

The main content area displays a table titled "Contract Summary Description (Enter SUPERBBOES exchange for Level II quotes)". The table has two sections: "Bid" and "Market Depth" and "Ask". The columns for the Bid section are: Symbol, Type, Expiry, Price, Multiplier, Exchange, Domestic, MM, Price, Size, Avg Size, Avg Price, MM, Price, Size, Avg Size, Avg Price, and Cmt. The columns for the Market Depth section are: Price, Size, Avg Price, and Cmt. The columns for the Ask section are: Price, Size, Avg Price, and Cmt. The table contains several rows of data, primarily for the symbol "H500".

Contract Summary Description (Enter SUPERBBOES exchange for Level II quotes)																			
10	Symbol	Type	Expiry	Price	RC	Multiplier	Exchange	Domestic	MM	Price	Size	Avg Size	Avg Price	MM	Price	Size	Avg Size	Avg Price	Cmt
11	C500	STK					SLAMC	190	H500	22.47	100	100	22.47	H500	22.44	339	339	22.44	
12									H500	22.42	335	77	22.4238	H500	22.45	241	480	22.44921	
13									H500	22.41	275	99	22.4191	H500	22.45	20	500	22.44822	
14									H500	22.4	157	153	22.4165	H500	22.45	103	600	22.44807	
15									H500	22.39	195	194	22.4127	H500	22.46	216	816	22.449718	
16										3	3	3	3		3	3	3	3	
17										3	3	3	3		6	3	0	0	
18										3	3	3	3		6	3	0	0	
19										3	3	3	3		6	3	0	0	
20										3	3	3	3		6	0	0	0	
21																			
22																			
23																			
24																			
25																			
26																			
27	Bid	STK					SMART	190											
28																			
29																			
30																			
31																			
32																			
33																			
34																			
35																			
36																			
37																			
38																			
39																			
40																			
41																			
42																			
43																			
44																			

## Using the Market Depth Page

**Note:** Ensure that TWS is running, and that you have entered your user name in the *User Name* field in the *Which Trader Workstation?* section of all pages in the Excel spreadsheet to properly connect to TWS.

### To request market depth for a contract

- 1** Click the **Market Depth** tab at the bottom of the spreadsheet to open the Market Depth page.
- 2** Select the ticker symbol for which you want to request the market depth, or enter a new ticker on a blank line.
- 3** Click the **Request Market Depth** button on the toolbar.

### To reset the market data refresh rate for tickers and market depth

- 1** Click the **Tickers** or **Market Depth** tab at the bottom of the spreadsheet.
- 2** Type the desired market data refresh rate in milliseconds in the *Refresh Rate* field in the *Which Trader Workstation?* area.
- 3** Move your cursor out of the *Refresh Rate* field.
- 4** Click the **Set Refresh Rate** button on the toolbar.

### To display more lines of market depth

You can edit the Request Market Depth macro to show more than the default 10 lines.

- 1** From the Market Depth page, press **Alt+F11**.  
The Visual Basic Editor (VBE) opens and displays the code for the Market Depth page.
- 2** In the Declarations section at the top of the code window for the page, change the number value in **numDisplayRows = 10** to a higher/lower value, then click the **Save** button on the VBE toolbar.
- 3** Close the Visual Basic Editor.

## Market Depth Page Toolbar Buttons

The toolbar on the Market Depth page includes the following buttons:

Button	Description
<b>Request Market Depth</b>	View bid/ask depth prices for the selected contract.
<b>Cancel Market Depth</b>	Cancel market depth for the selected contract.
<b>Set Refresh Rate</b>	Resets the refresh rate (in milliseconds) for market data.
<b>Show Errors</b>	Jumps to the Error Code field and shows the error code.

## Advisors Page

If you are a Financial Advisor and manage multiple accounts, use the Advisors page to create FA orders that:

- allocate shares to a single managed account
- use FA account groups and methods
- use allocation profiles

**Note:** You must set up your managed accounts, account groups, methods and allocation profiles in TWS before you can place FA orders in the DDE for Excel API sample spreadsheet.

## Allocating Shares to a Single Account

You can use the **Advisors** page to set up an order and allocate all shares in the order to a single account.

### To allocate shares to a single account:

- 1** Create an account group in TWS.
- 2** Click the **Advisors** tab at the bottom of the spreadsheet.
- 3** Enter the contract information in the *Contract Description* cells, then enter the order information in the *Order Description* cells.
- 4** Click the **Extended Order Attributes** tab. Enter the account code in the *Value* cell for the *Account (Institutional only)* extended order attribute.
- 5** Click the **Advisors** tab.
- 6** Highlight the order row, then click the **Apply Extended** button to apply the *Account* order attribute value to the order. The *Account* value is applied to the selected order and displayed in the *Extended Order Attributes* section of the page.
- 7** Click the **Place/Modify Order** button.
- 8** When you are done allocating shares to the account, delete the *Account* value from the **Extended Order Attributes** page. If you do not delete this value, it will be applied to all subsequent orders placed from the DDE for Excel API spreadsheet.

## Placing an Order using an FA Account Group and Method

You can also use the **Advisors** page to set up an order using an FA account group and FA method.

### To place an order using an FA account group and FA method:

- 1** Create the FA account group(s) and FA method(s) in TWS.
- 2** Click the **Advisors** tab at the bottom of the spreadsheet.
- 3** Enter the contract information in the *Contract Description* cells, then enter the order information in the *Order Description* cells.
- 4** Click the **Extended Order Attributes** tab. Enter values for the following extended order attributes:
  - *FA Group* - Enter the name of the account group.
  - *FA Method* - Enter the name of the allocation method to use for this order.
  - *FA Percentage* - Enter the percentage used by the PctChange allocation method to use for this order. This attribute applies only to FA groups that use this method.
- 5** Click the **Advisors** tab.
- 6** Highlight the order row, then click the **Apply Extended** button to apply the extended order attribute values to the order. The values for *FA Group*, *FA Method* and *FA Percentage* are applied to the selected order and displayed in the *Extended Order Attributes* section of the page.
- 7** Click the **Place/Modify Order** button.
- 8** When you are done allocating shares to the account, delete the values you entered on the **Extended Order Attributes** page. If you do not delete these values, they will be applied to all subsequent orders placed from the DDE for Excel API spreadsheet.

## Placing an Order using an Allocation Profile

You can also use the **Advisors** page to set up an order using an FA allocation profile.

### To place an order using an FA allocation profile:

- 1** Create the FA allocation profile in TWS.
- 2** Click the **Advisors** tab at the bottom of the spreadsheet.
- 3** Enter the contract information in the *Contract Description* cells, then enter the order information in the *Order Description* cells.
- 4** Click the **Extended Order Attributes** tab. Enter the name of the allocation profile in the *Value* field for the *FA Profile* extended order attribute.
- 5** Click the **Advisors** tab.
- 6** Highlight the order row, then click the **Apply Extended** button to apply the extended order attribute value to the order. The value for *FA Profile* is applied to the selected order and displayed in the *Extended Order Attributes* section of the page.
- 7** Click the **Place/Modify Order** button.
- 8** When you are done allocating shares to the account, delete the *FA Profile* value you entered on the **Extended Order Attributes** page. If you do not delete this value, it will be applied to all subsequent orders placed from the DDE for Excel API spreadsheet.

## Advisors Page Toolbar Buttons

The toolbar on the Basic Orders page includes the following buttons:

<b>Button</b>	<b>Description</b>
<b>Combo Legs</b>	Opens the Combination Legs box. Enter contract details to create legs of a combination order one by one.
<b>Place/Modify Orders</b>	After you have completed the Order Description fields, and defined any extended attributes, click to create an order for the selected contract.
<b>Cancel Order</b>	This button cancels the order(s) you have highlighted.
<b>Apply Extended</b>	Applies the current values on the Extended Order Attributes page to the highlighted order row.
<b>Show Errors</b>	Jumps to the Error Code field and shows the error code.

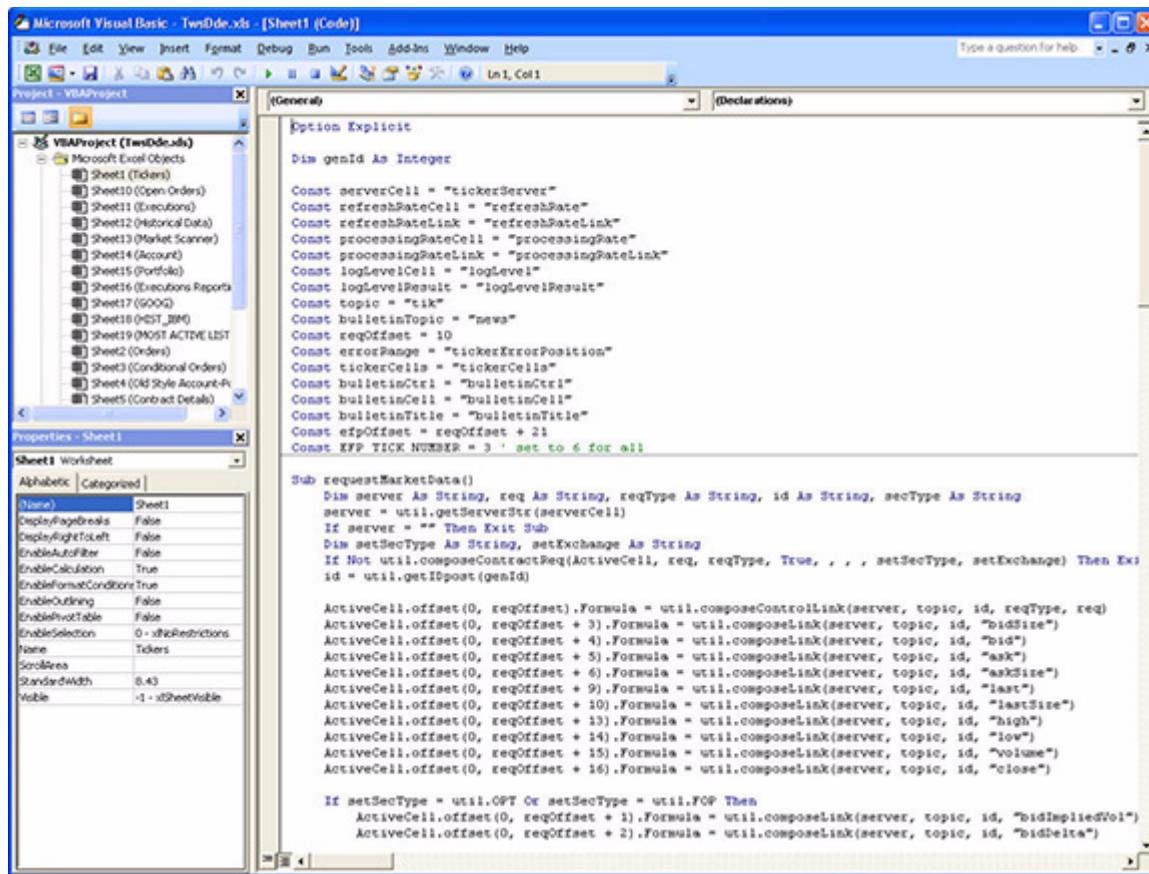
## DDE for Excel API Reference

This section provides a variety of reference information about the DDE for Excel API, including the following topics:

- Viewing the Code
- Modules
- Named Ranges
- Macros
- DDE Syntax for Excel

### Viewing the Code

To view the Visual Basic code behind the DDE for Excel API spreadsheet, press **Alt+F11** from any page. The Visual Basic Editor opens:



The Visual Basic Editor contains three main components:

- Project Explorer
- Properties Window
- Code Window

The Project Explorer contains a list of objects used in the spreadsheet. These object correspond to the pages in the spreadsheet; to view the code for a particular page, double-click the page's corresponding object in the Project Explorer.

## Modules

The Visual Basic code includes the following modules (visible in the VBE Project Explorer):

- ArrayQueries
- ErrorDisplay
- Orders
- util

The util module contains many pre-defined constants that you can use when you program your own DDE API application. Using these constants instead of hard-coded values will make your application more robust and easier to maintain. Specifically, the following util functions are particularly useful in building new Visual Basic functionality:

- composeLink – put together a link that receives data, such as a market data bid size, option model volatility, or execution id.
- composeControlLink – put together a link that causes operations to occur, such as subscribing to market data, placing orders, or subscribing to the market scanner.
- composeContractReq – Read a contract description out of a page like Tickers or Orders, and build the DDE string representing it.

## Named Ranges

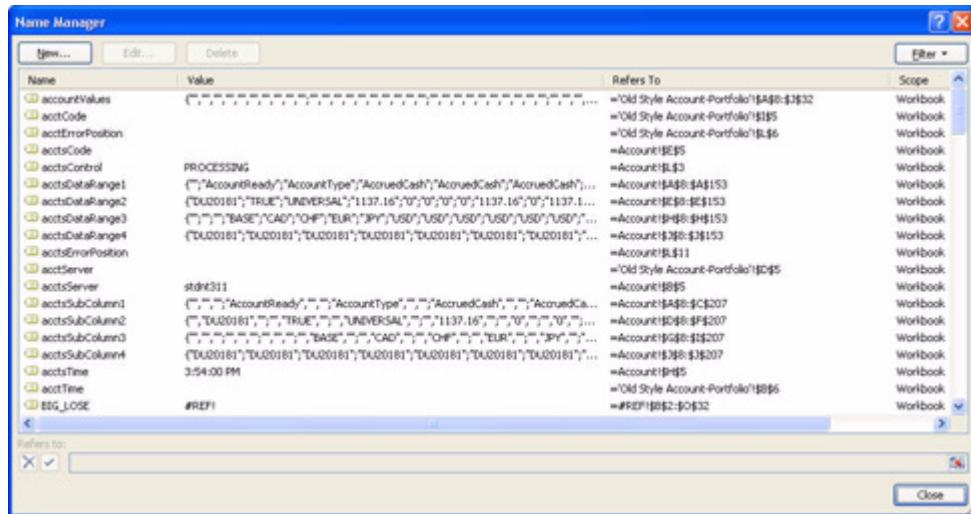
Named ranges are a Microsoft Excel feature that lets you assign meaningful names to a single cell or a range of cells in Microsoft Excel. The TwsDde.xls sample spreadsheet used named ranges extensively.

Named ranges help you to move data around on worksheets without breaking existing logic. It also makes important data available to your own custom macros and worksheets.

You can view all the named ranges used in the spreadsheet by doing the following, depending on which version of Excel you are using:

- In Excel 2007, you can see a complete list of all named ranges used in the spreadsheet by clicking Formulas then clicking Name Manager. The Name Manager displays every named range used in the spreadsheet, the value of the range, and the page and range of cells covered by the range.
- In earlier versions of Excel, you can view the named ranges by selecting **Name > Define** from the Tools menu. You can also download a free Name Manager from Microsoft that has additional functionality for these earlier versions of Excel.

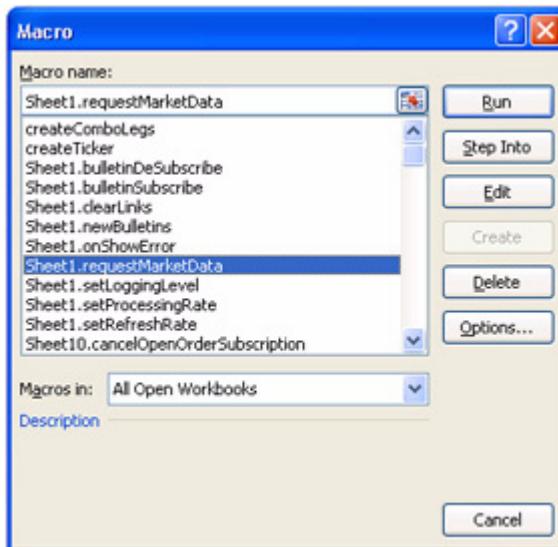
The following screen shows the Name Manager for the DDE for Excel API sample spreadsheet:



## Macros

The DDE API sample spreadsheet uses Microsoft Excel macros extensively. Each toolbar button on every page in the spreadsheet runs a macro when you click it. For example, when you click the Request Market Data button on the Tickers page, you are actually running a macro called `requestMarketData`.

You can see all the macros used in the sample spreadsheet by opening the Excel Macro dialog. From there you can choose to edit the macro, which opens macro code in the Visual Basic Editor.



**Note:** You must enable macros when you open Excel or none of the macros in the spreadsheet will function.

For information about recording, editing and viewing macros, refer to your Microsoft Excel documentation.

## DDE Syntax for Excel

The table below defines possible cell values for DDE-supported functionality. The basic syntax, which appears in the Excel formula bar (the "=" enterable field at the top of the spreadsheet) when you put your cursor in a cell, is:

=server|topic!id?reqType?field2

or

=server|error!error (for an optional tag that will display errors)

where:

- `server` = the username
- `topic` = the function category, such as orders (`ord`) or tickers (`tik`)
- `id` = `idn`, where *n* is some integer. For orders, *n* must always increase, even from session to session.
- `reqType` = the request type, such as place or cancel
- `field2` = additional information as noted below

Description	Server	Topic	id	reqType	field2
Place an order	server	ord	<code>idn</code>	place	<code>orderDescription</code>
Modify an order	server	ord	<code>idn</code>	modify	<code>orderModification</code>
Cancel an order	server	ord	<code>idn</code>	cancel	
Check order status	server	ord	<code>idn</code>	status	
Request open orders	server	ord	<code>idn</code>	open	
Request executions	server	ord	<code>idn</code>	executed	
Check shares filled in order	server	ord	<code>idn</code>	sharesFilled	
Check shares remaining in order	server	ord	<code>idn</code>	sharesRemaining	
Execution (average) price	server	ord	<code>idn</code>	price	
Underlying	server	ord	<code>idn</code>	symbol	
Security type	server	ord	<code>idn</code>	secType	Refer to Note 6
Expiry	server	ord	<code>idn</code>	expiry	Refer to Note 7
Strike	server	ord	<code>idn</code>	strike	Refer to Note 8
Right	server	ord	<code>idn</code>	right	Refer to Note 8
Specify contract multiplier for options and futures	server	ord	<code>idn</code>	multiplier	Refer to Note 8
Order destination	server	ord	<code>idn</code>	exchange	
Currency	server	ord	<code>idn</code>	currency	
Order side	server	ord	<code>idn</code>	side	
Order quantity	server	ord	<code>idn</code>	size	
Order type	server	ord	<code>idn</code>	orderType	

Description	Server	Topic	id	reqType	field2
Limit price	server	ord	idn	limitPrice	
Auxiliary price	server	ord	idn	auxPrice	
Local symbol	server	ord	idn	localSymbol	
Last fill price	server	ord	idn	lastFillPrice	
Create ticker	server	tik	idn	req	
Bid implied volatility	server	tik	idn	bidImpliedVol	
Bid delta	server	tik	idn	bidDelta	
Request bid size	server	tik	idn	bidSize	
Request bid price	server	tik	idn	bid	
Request ask price	server	tik	idn	ask	
Request ask size	server	tik	idn	askSize	
Ask implied volatility	server	tik	idn	askImpliedVol	
Ask delta	server	tik	idn	askDelta	
Request last price	server	tik	idn	last	
Request last size	server	tik	idn	lastSize	
Last implied volatility	server	tik	idn	lastImpliedVol	
Last delta	server	tik	idn	lastDelta	
Request today's high price	server	tik	idn	high	
Request today's low price	server	tik	idn	low	
Request today's volume size	server	tik	idn	volume	
Request last close price	server	tik	idn	close	
Request implied volatility calculated by the TWS option modeler	server	tik	idn	modelVolatility	
Request option delta calculated by the TWS option modeler	server	tik	idn	modelDelta	
Request the model price	server	tik	idn	modelPrice	
Request present value of dividends expected on the options underlier	server	tik	idn	pvDividend	
Request number of hold days until the expiry of the EFP	server	tik	idn	holdDays	
Request expiration date of the single stock future	server	tik	idn	futureExpiry	

<b>Description</b>	<b>Server</b>	<b>Topic</b>	<b>id</b>	<b>reqType</b>	<b>field2</b>
Request dividends expected until the expiration of the single stock future	server	tik	idn	dividendsToExpiry	
Request annualized basis points	server	tik	idn	basisPoints	
Request annualized basis points in percentage form	server	tik	idn	formattedBasis Points	
Request implied futures price	server	tik	idn	impliedFuture	
Request the dividend impact on the annualized basis points interest rate	server	tik	idn	dividendImpact	
Account statement control key	server	acct	idn	acctv	Account code (for Advisor-managed accounts only)
Request one account value string	server	acct	idn	key	
Account value	server	acct	idn	value	
Account currency	server	acct	idn	keyCurrency	
Account portfolio control key	server	acct	idn	acctp	Account code (for Advisor-managed accounts only)
Account portfolio underlying symbol	server	acct	idn	symbol	
Account portfolio security type	server	acct	idn	secType	
Account portfolio expiry	server	acct	idn	expiry	
Account portfolio strike price	server	acct	idn	strike	
Account portfolio right	server	acct	idn	right	
Account portfolio currency	server	acct	idn	currency	
Account portfolio local symbol	server	acct	idn	localSymbol	
Account portfolio market price	server	acct	idn	marketPrice	
Account portfolio market value	server	acct	idn	marketValue	
Account portfolio average cost	server	acct	idn	avgCost	
Account portfolio realized PNL	server	acct	idn	realizedPNL	

Description	Server	Topic	id	reqType	field2
Account portfolio unrealized PNL	server	acct	idn	unrealizedPNL	
Request contract details	server	contract	idn	req	contractDescription
Valid order types	server	contract	idn	orderTypes	
Valid exchanges	server	contract	idn	validExchanges	
Contract identifier	server	contract	idn	conid	
Minimum tick	server	contract	idn	minTick	
Order multiplier	server	contract	idn	multiplier	
Market name	server	contract	idn	marketName	
Trading class	server	contract	idn	tradingClass	
Execution order id	server	exec	idn	orderId	
Underlying	server	exec	idn	symbol	
Security type	server	exec	idn	secType	
Expiry	server	exec	idn	expiry	
Strike	server	exec	idn	strike	
Right	server	exec	idn	right	
Order destination	server	exec	idn	exchange	
Currency	server	exec	idn	currency	
Local symbol	server	exec	idn	localSymbol	
Execution id	server	exec	idn	execId	
Execution time	server	exec	idn	time	
Account number	server	exec	idn	acctnNumber	
Exchange where executed	server	exec	idn	eExchange	
Side	server	exec	idn	side	
Number of shares filled in order	server	exec	idn	shares	
Execution (average) price	server	exec	idn	price	
Order ID	server	exec	idn	permId	
Identifies position as one to be liquidated last	server	exec	idn	liquidation	
Request execution details	server	exec	idn	Req	executionFilter
Request list of Advisor-managed accounts	server	FAaccts	idn	Req	
List of Advisor-managed accounts	server	FAaccts	idn	Value	

Description	Server	Topic	id	reqType	field2
Request market depth	server	mktDepth	idn	req	contractDescription? num_display_rows Refer to <a href="#">note (1)</a> below.
Market maker	server	mktDepth	idn	mktMaker	rowId_side Refer to <a href="#">note (2)</a> below.
Order price	server	mktDepth	idn	price	rowId_side Refer to <a href="#">note (2)</a> below.
Order size	server	mktDepth	idn	size	rowId_side Refer to <a href="#">note (2)</a> below.
Market data refresh rate	server	refreshRate	idn	millisec	Number of milliseconds
Subscribe to news bulletins	server	news	sub	0	Refer to <a href="#">note 3</a> below
News bulletin message ID	server	news	newsID		
News bulletin message type	server	news	newsType		Refer to <a href="#">note 4</a> below
News bulletin message text	server	news	msg		
Exchange from which news bulletins originated	server	news	exchange		
Set the server log level	server	logLevel	<log_level>		Refer to <a href="#">note 5</a> below.

Where:

- orderDescription =  
 side\_quantity\_symbol\_secType\_exp\_strike\_right\_exchange\_orderType\_lmtPrice\_auxPrice  
 {\_timeInForce\_ocaGroup\_account\_open/close\_origin\_orderRef\_transmit\_parentId\_blockOr  
 der  
 \_sweepToFill\_displaySize\_triggerMethod\_ignoreRth\_hidden\_clientId\_accountCode\_goodAft  
 erTime\_  
 goodTillDate\_faGroup\_faMethod\_faPercentage\_faProfile\_shortSaleSlot\_  
 PRIMARYEXCHANGE  
 \_shortSaleLocation\_ocaType\_rthOnly\_rule80A\_settlingFirm\_allOrNone\_minimumQty\_perce  
 ntOffset\_  
 electronicTradeOnly\_firmQuoteOnly\_nbboPriceCap\_auctionStrategy\_startingPrice\_stockRef  
 Price\_  
 delta\_stockRangeLower\_stockRangeUpper\_volatility\_volatilityType\_referencePriceType\_hed  
 geDelta\_  
 continuousUpdate}

**Note:** Attributes in brackets are extended order attributes and are described in the topic [Extended Order Attributes](#).

- contractDescription - symbol\_secType\_exp\_strike\_right\_exchange
- tickerDescription = *symbol\_secType\_exp\_strike\_right\_exchange*
- ExecutionFilter = clientId\_accountCode\_date\_time\_symbol\_secType\_exchange\_side

**Note 1:** When requesting market depth you can specify the number of rows to display. If not supplied, the default number of rows is five (5) rows. This parameter can be used to optimize performance, as a low number of display rows requires less CPU overhead. For example: =edemo|mktDepth!id0?req?MSFT\_STK\_SMART?10 will request 10 rows of market depth orders.

**Note 2:** Field 2 of the market depth 'price' and 'size' reqTypes contain additional information to specify the order row and side that the data applies to. Market depth orders are divided into two sides, BID and ASK, and order entries start from the offset 0.

For example:

=edemo|mktDepth!id0?price?0ASK will request the ASK price for the order entry 0.

=edemo|mktDepth!id0?size?2BID will request the BID size for the order entry 2.

**Note 3:** When subscribing to news bulletins, the request should look like this:

=edemo|news!sub?0

where the request type is a zero. To unsubscribe simply clear the subscription cell.

**Note 4:** The valid news bulletin types are:

- 1 = Regular news bulletin
- 2 = Exchange no longer available for trading
- 3 = Exchange available for trading

**Note 5:** The valid log levels are:

- 1 = SYSTEM (least detailed)
- 2 = ERROR (default, if no level is specified)
- 3 = WARNING
- 4 = INFORMATION
- 5 = DETAIL (most detailed)

**Note 6:** If the order is a combo, combo legs are inserted after the aux price. Here is a three-legged IBM combo description:

CMBLGS\_3\_8314\_100\_BUY\_SMART\_0\_36930759\_1\_BUY\_SMART\_0\_36930816\_1\_SELL\_SMA  
RT\_0\_CMBLGS.

Each leg contains : contract ID, number of contracts, side, exchange, and open/close (0 or 1). Retail can always specify 0, institutional need to specify a 1 if the order being executed would close a position.

**Note 7:** If the order is an option or a future, the expiry is inserted after the sec type.

**Note 8:** If the order is an option, the strike and right (put or call) are inserted after the expiry. If the multiplier is specified, it follows the strike and right.

# ActiveX

This chapter describes the ActiveX API, including the following topics:

- [Linking to the Application using ActiveX](#)
- [Registering Third-Party ActiveX Controls](#)
- [Running the ActiveX API on 64-bit Windows XP Systems](#)
- [Using the Visual Basic Sample Program](#)
- [ActiveX Methods](#)
- [ActiveX Events](#)
- [ActiveX COM Objects](#)
- [ActiveX Properties](#)
- [Placing a Combination Order](#)

**Note:** The API software also includes an ActiveX for Excel sample spreadsheet, which duplicates most of the functionality of the DDE for Excel sample spreadsheet but is based on the ActiveX control, Tws.ocx. See [ActiveX for Excel](#) for details.

## Linking to the Application using ActiveX

Before you can use third-party ActiveX controls, you must [register them with Visual Basic](#).

### To link using the ActiveX control and VB, VBA or C++

- 1** Drop the **application** control onto a form or dialog box.
- 2** Call the following methods:
  - Call the connect() method to connect to the running application.
  - Call the methods you need to perform whatever operations you require, such as the reqMktData() method to request market data.
- 3** Call the placeOrder() method to place an order. Orders using extended attributes require that ActiveX properties representing them be set first.
- 4** Handle the following events:
  - Handle the nextValidId() event to receive the next available valid order ID. Increment the ID by one for successive orders.
  - Handle the tickPrice() and tickSize() events to receive the market data.
  - Handle the orderStatus() event to receive status information about orders.
  - Handle the error() event to receive error information.
  - Handle the connectionClosed() event to be notified in case the application stops communicating with the ActiveX control.

## Registering Third-Party ActiveX Controls

To use a third-party ActiveX control in Visual Basic it must be registered first.

**To register the ActiveX control with Visual Basic, follow these instructions:**

- 1** From the **Components** menu in your VB project, select the TWS ActiveX control (Tws.ocx).
- 2** Click **Apply**.
- 3** Verify that the TWS control appears in the toolbar with all standard controls.

## Running the ActiveX API on 64-bit Windows XP Systems

To run the ActiveX API on 64-bit Windows XP systems, do the following:

- 1** Install Microsoft Visual C++ 2005 SP1 Redistributable Package (x86).
- 2** Install Microsoft Visual J# 2.0 Redistributable Package.
- 3** Download and install the API software.

## Using the Visual Basic Sample Program

You can access the server through the ActiveX interface using the Visual Basic sample application. To run the sample you must:

- Install the API sample programs
- Configure the application to support the API components
- Have MS Visual Studio (Visual Basic 6.0 or higher) installed on your PC.

The Visual Basic program is a sample program that demonstrates use of the TWS ActiveX control to connect to the server from a Visual Basic application.

### To open the VB\_API\_sample program:

- 1 From the MS Visual Basic file menu, select **Open Project**.
- 2 Navigate to the \TestActiveXClient\_VB directory in your API installation folder, and select *VB\_API\_sample.vbp*. X\_XX represents the API version number; for example, 9.60.  
If you are using Visual Studio 2008, you will have to upgrade your project before it can be opened. Visual Basic presents an Upgrade Wizard to walk you through this simple process.
- 3 On the **Projects** menu, select *Components*.
- 4 In the Components dialog box, select the *TWS ActiveXControl* module and click **OK**.
- 5 Press **Ctrl+F5** to compile and run the project.

**Note:** If you have trouble getting the sample program to run, try re-registering the TWS ActiveX component, *Tws.ocx*.

## ActiveX Methods

ActiveX methods allow your application to call functions and request information from TWS.

**Note:** Please note that methods in red have been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated methods. The new "Ex" methods have been grouped with their deprecated counterparts for easy identification.

Connection and Server	Account
<code>connect()</code>	<code>reqAccountUpdates()</code>
<code>disconnect()</code>	<b>News Bulletins</b>
<code>reqCurrentTime()</code>	<code>reqNewsBulletins()</code>
<code>setServerLogLevel()</code>	<code>cancelNewsBulletins()</code>
<b>Market Data</b>	<b>Financial Advisors</b>
<code>reqMktDataEx()</code>	<code>reqManagedAccts()</code>
<code>reqMktData()</code>	<code>requestFA()</code>
<code>reqMktData2()</code>	<code>replaceFA()</code>
<code>cancelMktData()</code>	<b>Historical Data</b>
<b>Orders</b>	<code>reqHistoricalDataEx()</code>
<code>placeOrderEx()</code>	<code>requestHistoricalData()</code>
<code>placeOrder()</code>	<code>cancelHistoricalData()</code>
<code>placeOrder2()</code>	<b>Market Scanners</b>
<code>cancelOrder()</code>	<code>reqScannerParameters()</code>
<code>reqOpenOrders()</code>	<code>reqScannerSubscriptionEx()</code>
<code>reqAllOpenOrders()</code>	<code>reqScannerSubscription()</code>
<code>reqAutoOpenOrders()</code>	<code>cancelScannerSubscription()</code>
<code>reqIds()</code>	<b>Real Time Bars</b>
<code>exerciseOptionsEx()</code>	<code>reqRealTimeBarsEx()</code>
<code>exerciseOptions()</code>	<code>reqRealTimeBars()</code>
<code>addComboLeg()</code>	<code>cancelRealTimeBars()</code>
<code>clearComboLegs()</code>	<b>Factory Methods</b>
<b>Executions</b>	<code>createComboLegList()</code>
<code>reqExecutionsEx()</code>	<code>createContract()</code>
<code>reqExecutions()</code>	<code>createExecutionFilter()</code>
<b>Contract Details</b>	<code>createOrder()</code>
<code>reqContractDetailsEx()</code>	<code>createScannerSubscription()</code>
<code>reqContractDetails()</code>	<code>createTagValueList()</code>
<code>reqContractDetails2()</code>	<code>createUnderComp()</code>
<b>Market Depth</b>	<b>Fundamental Data</b>
<code>reqMktDepthEx()</code>	<code>reqFundamentalData()</code>
<code>reqMktDepth()</code>	<code>cancelFundamentalData()</code>
<code>reqMktDepth2()</code>	
<code>cancelMktDepth()</code>	

## connect()

Call this method to connect to the host application.

```
Public Overridable Sub connect(ByVal host as String, ByVal port as Integer, ByVal clientID as Integer)
```

Parameter	Description
<b>host</b>	The host name or IP address of the machine where TWS is running. Leave blank to connect to the local host.
<b>port</b>	Must match the port specified in TWS on the Configure>API>Socket Port field.
<b>clientID</b>	A number used to identify this client connection. All orders placed/modified from this client will be associated with this client identifier.  Note: Each client MUST connect with a unique clientId.

## disconnect()

Call this method to terminate the connections the host application. Calling this method does not cancel orders that have already been sent.

```
Public Overridable Sub disconnect()
```

## reqMktDataEx()

Call this method to request market data. The market data will be returned by the [tickPrice\(\)](#), [tickSize\(\)](#), [tickOptionComputation\(\)](#), [tickGeneric\(\)](#), [tickString\(\)](#) and [tickEFP\(\)](#) events in dispinterface\_DTwsEvents.

```
Public Overridable Sub reqMktDataEx(ByVal tickerId As Integer, ByVal contract As TWSLib.IContract, ByVal genericTicks As String, ByVal snapshot As Integer)
```

Parameter	Description
<b>tickerId</b>	The ticker id. Must be a unique value. When the market data returns, it will be identified by this tag. This is also used when canceling the market data.
<b>contract</b>	This object contains a description of the contract for which market data is being requested.
<b>genericTicks</b>	A comma delimited list of generic tick types. For more information about tick types, see <a href="#">Generic Tick Types</a> .
<b>snapshot</b>	Check to return a single snapshot of market data and have the market data subscription cancel. Do not enter any genericTicklist values if you use snapshot.

## reqMktData()

Call this method to request market data. The market data will be returned by the [tickPrice\(\)](#) and [tickSize\(\)](#) events.

**Note:** This method has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated methods.

```
Public Overridable Sub reqMktData(ByVal id As Integer, ByVal symbol As String,
ByVal secType As String, ByVal expiry As String, ByVal strike As Double, ByVal
right As String, ByVal multiplier As String, ByVal exchange As String, ByVal
primaryExchange As String, ByVal currency As String, ByVal genericTicks As
String, ByVal snapshot As Integer)
```

Parameter	Description
<b>id</b>	The ticker id. Must be a unique value. When the market data returns, it will be identified by this tag. This is also used when canceling the market data.
<b>symbol</b>	The symbol of the underlying for which you are requesting market data.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"> <li>• STK</li> <li>• OPT</li> <li>• FUT</li> <li>• IND</li> <li>• FOP</li> <li>• CASH</li> </ul>
<b>expiry</b>	The expiration date. Use the format YYYYMM.
<b>strike</b>	The strike price.
<b>right</b>	Specifies a Put or Call. Valid values are: P, PUT, C, CALL.
<b>multiplier</b>	- allows you to specify a futures or options multiplier in cases where multiple possibilities exist.
<b>exchange</b>	The order destination, such as Smart.
<b>primaryExchange</b>	The primary exchange on which the product trades, used to definitively identify a product. To clarify any ambiguity for Smart-routed contracts, include the primary exchange, along with the Smart designation, for the destination.
<b>currency</b>	Specifies the currency. Ambiguities may require that this field be specified, for example, when SMART is the exchange and IBM is being requested (IBM can trade in GBP or USD). Given the existence of this kind of ambiguity, it is a good idea to always specify the currency.
<b>snapshot</b>	Check to return a single snapshot of market data and have the market data subscription cancel. Do not enter any genericTicklist values if you use snapshot.

## reqMktData2()

Call this method to request market data using the exchange symbol. The market data will be returned by the [tickPrice\(\)](#) and [tickSize\(\)](#) events.

**Note:** This method has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated methods.

```
Public Overridable Sub reqMktData2(ByVal id As Integer, ByVal localSymbol As String, ByVal secType As String, ByVal exchange As String, ByVal primaryExchange As String, ByVal currency As String, ByVal genericTicks As String, ByVal snapshot As Integer)
```

Parameter	Description
<b>id</b>	The ticker id. Must be a unique value. When the market data returns, it will be identified by this tag. This is also used when canceling the market data.
<b>localSymbol</b>	The local exchange symbol of the underlying for which you are requesting market data.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"><li>• STK</li><li>• OPT</li><li>• FUT</li><li>• IND</li><li>• FOP</li><li>• CASH</li></ul>
<b>exchange</b>	The order destination, such as SMART.
<b>primaryExchange</b>	The primary exchange on which the product trades, used to definitively identify a product.
<b>currency</b>	Specifies the currency. Ambiguities may require that this field be specified, for example, when SMART is the exchange and IBM is being requested (IBM can trade in GBP or USD). Given the existence of this kind of ambiguity, it is a good idea to always specify the currency.
<b>genericTicklist</b>	A comma delimited list of generic tick types. Tick types can be found in the <a href="#">Generic Tick Types</a> page.
<b>snapshot</b>	Check to return a single snapshot of market data and have the market data subscription cancel. Do not enter any genericTicklist values if you use snapshot.

## cancelMktData()

After calling this method, market data for the specified id will stop flowing.

```
Public Overridable Sub cancelMktData(ByVal id As Integer)
```

Parameter	Description
<b>id</b>	The ID that was specified in the call to reqMktData().

## placeOrderEx()

Call this method to place an order. The order status will be returned by the [orderStatus\(\)](#) event in dispinterface\_DTwsEvents.

```
Public Overridable Sub placeOrderEx(ByVal orderId As Integer, ByVal  
contract As TWSLib.IContract, ByVal order As TWSLib.IOrder)
```

Parameter	Description
<b>orderId</b>	The order Id. You must specify a unique value. When the order status returns, it will be identified by this tag. This tag is also used when canceling the order.
<b>contract</b>	This object contains attributes used to describe the contract.
<b>order</b>	This object contains the details of the order. Note: Each client MUST connect with a unique clientId.

## placeOrder()

Call this method to place an order. The order status will be returned by the [orderStatus\(\)](#) event. For more details see [Active X Properties](#).

**Note:** This method has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated methods.

```
Public Overridable Sub placeOrder(ByVal id As Integer, ByVal action As String, ByVal quantity As Integer, ByVal symbol As String, ByVal secType As String, ByVal expiry As String, ByVal strike As Double, ByVal right As String, ByVal multiplier As String, ByVal exchange As String, ByVal primaryExchange As String, ByVal currency As String, ByVal orderType As String, ByVal price As Double, ByVal auxPrice As Double, ByVal goodAfterTime As String, ByVal group As String, ByVal faMethod As String, ByVal faPercentage As String, ByVal faProfile As String, ByVal goodTillDate As String)
```

Parameter	Description
<b>id</b>	The order id. You must specify a unique value. When the order status returns, it will be identified by this tag. This tag is also used when canceling the order.
<b>action</b>	Identifies the side. Valid values are: <ul style="list-style-type: none"> <li>• BUY</li> <li>• SELL</li> <li>• SSHORT</li> </ul>
<b>quantity</b>	The order quantity.
<b>symbol</b>	The symbol of the underlying asset.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"> <li>• STK</li> <li>• OPT</li> <li>• FUT</li> <li>• FOP</li> <li>• CASH</li> </ul>
<b>expiry</b>	The expiration date. Use the format YYYYMM.
<b>strike</b>	The strike price
<b>right</b>	Specifies a Put or Call. Valid values are: P, PUT, C, CALL.
<b>multiplier</b>	Allows you to specify a futures or options multiplier in cases where multiple possibilities exist.
<b>exchange</b>	The order destination, such as Smart.
<b>primaryExchange</b>	The primary exchange on which the product trades, used to definitively identify a product.

Parameter	Description
<b>currency</b>	Specifies the currency. Ambiguities may require that this field be specified, for example, when SMART is the exchange and IBM is being requested (IBM can trade in GBP or USD). Given the existence of this kind of ambiguity, it is a good idea to always specify the currency.
<b>orderType</b>	Identifies the order type. Valid values are: <ul style="list-style-type: none"> <li>• MKT</li> <li>• MKTCLS</li> <li>• LMT</li> <li>• LMTCLS</li> <li>• PEGMKT</li> <li>• SCALE</li> <li>• STP</li> <li>• STPLMT</li> <li>• TRAIL</li> <li>• REL</li> <li>• VWAP</li> </ul>
<b>lmtPrice</b>	The LIMIT price, used for limit, stop-limit and relative orders. In all other cases specify zero. For relative orders with no limit price, also specify zero.
<b>lmtPrice</b>	Identifies the STOP price for stop-limit orders, and the offset amount for relative orders. In all other cases, specify zero.
<b>goodAfterTime</b>	Specifies that the order becomes active after the time set. Send an empty string if not applicable.
<b>group</b>	Specifies the order's "Financial Advisor Group." Send an empty string if not an FA.
<b>faMethod</b>	Specifies the order's "Financial Advisor Allocation Method." Send an empty string if not an FA.
<b>faPercentage</b>	Specifies the order's "Financial Advisor Percentage." Send an empty string if not an FA.
<b>faProfile</b>	Specifies the order's "Financial Advisor Profile." Send an empty string if not an FA.
<b>goodTillDate</b>	Specifies the order's "Good Till Date." Send an empty string if not applicable.

## placeOrder2()

Call this method to place an order using the exchange symbol. The order status will be returned by the [orderStatus\(\)](#) event.

**Note:** This method has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated methods.

```
Public Overridable Sub placeOrder2(ByVal id As Integer, ByVal action As String, ByVal quantity As Integer, ByVal localSymbol As String, ByVal secType As String, ByVal exchange As String, ByVal primaryExchange As String, ByVal currency As String, ByVal orderType As String, ByVal lmtPrice As Double, ByVal auxPrice As Double, ByVal goodAfterTime As String, ByVal group As String, ByVal faMethod As String, ByVal faPercentage As String, ByVal faProfile As String, ByVal goodTillDate As String)
```

Parameter	Description
<b>id</b>	The order id. You must specify a unique value. When the order status returns, it will be identified by this tag. This tag is also used when canceling the order.
<b>action</b>	Identifies the side. Valid values are: <ul style="list-style-type: none"><li>• BUY</li><li>• SELL</li><li>• SSHORT</li></ul>
<b>quantity</b>	The order quantity.
<b>localSymbol</b>	The local exchange symbol of the underlying asset.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"><li>• STK</li><li>• OPT</li><li>• FUT</li><li>• FOP</li><li>• CASH</li></ul>
<b>exchange</b>	The order destination, such as Smart.
<b>primaryExchange</b>	The primary exchange on which the product trades, used to definitively identify a product.
<b>currency</b>	Specifies the currency. Ambiguities may require that this field be specified, for example, when SMART is the exchange and IBM is being requested (IBM can trade in GBP or USD). Given the existence of this kind of ambiguity, it is a good idea to always specify the currency.

Parameter	Description
<b>orderType</b>	Identifies the order type. Valid values are: <ul style="list-style-type: none"><li>• MKT</li><li>• MKTCLS</li><li>• LMT</li><li>• LMTCLS</li><li>• PEGMKT</li><li>• SCALE</li><li>• STP</li><li>• STPLMT</li><li>• TRAIL</li><li>• REL</li><li>• VWAP</li></ul>
<b>lmtPrice</b>	The LIMIT price, used for limit, stop-limit and relative orders. In all other cases specify zero. For relative orders with no limit price, also specify zero.
<b>auxPrice</b>	Identifies the STOP price for stop-limit orders, and the offset amount for relative orders. In all other cases, specify zero.
<b>goodAfterTime</b>	Specifies that the order becomes active after the time set. Send an empty string if not applicable.
<b>group</b>	Specifies the order's "Financial Advisor Group." Send an empty string if not an FA.
<b>faMethod</b>	Specifies the order's "Financial Advisor Allocation Method." Send an empty string if not an FA.
<b>faPercentage</b>	Specifies the order's "Financial Advisor Percentage." Send an empty string if not an FA.
<b>faProfile</b>	Specifies the order's "Financial Advisor Profile." Send an empty string if not an FA.
<b>goodTillDate</b>	Specifies the order's "Good Till Date." Send an empty string if not applicable.

## cancelOrder()

Call this method to cancel an order.

```
Public Overridable Sub cancelOrder(ByVal id As Integer)
```

Parameter	Description
<b>id</b>	The order ID that was specified previously in the call to placeOrder()

## reqOpenOrders()

Call this method to request the open orders that were placed from this client. Each open order will be fed back through the [openOrder1](#), [openOrder2](#), [openOrder3](#), [openOrder4](#) or [openOrderEx\(\)](#) events.

**Note:** The client with a clientId of 0 will also receive the application-owned open orders. These orders will be associated with the client and a new orderId will be generated. This association will persist over multiple API and application sessions.

```
Public Overridable Sub reqOpenOrders()
```

## reqAllOpenOrders()

Call this method to request the open orders that were placed from all clients and also from the application. Each open order will be fed back through the [orderStatus\(\)](#) event.

**Note:** No association is made between the returned orders and the requesting client

```
Public Overridable Sub reqAllOpenOrders()
```

## reqAutoOpenOrders()

Call this method to request that newly created application orders be implicitly associated with the client. When a new application order is created, the order will be associated with the client, and fed back through the [orderStatus\(\)](#) event.

**Note:** This request can only be made from a client with a clientId of 0.

```
Public Overridable Sub reqAutoOpenOrders(ByVal bAutoBind As Integer)
```

Parameter	Description
<b>bAutoBind</b>	If set to TRUE, newly created application orders will be implicitly associated with the client. If set to FALSE, no association will be made.

## reqExecutionsEx()

When this method is called, the execution reports that meet the filter criteria are downloaded to the client via the [execDetailsEx\(\)](#) event in [dispInterface\\_DTwSEvents](#).

```
Public Overridable Sub reqExecutionsEx(ByVal reqId As Integer, ByVal filter As TWSLib.IExecutionFilter)
```

Parameter	Description
<b>filter</b>	The filter criteria used to determine which execution reports are returned.

## reqExecutions()

When this method is called, all execution reports are downloaded to the client via the [execDetails\(\)](#) event.

```
Public Overridable Sub reqExecutions()
```

## reqIds()

Call this function to request the next valid ID that can be used when placing an order. After calling this method, the [nextValidId\(\)](#) event will be triggered, and the id returned is that next valid ID. That ID will reflect any autobinding that has occurred (which generates new IDs and increments the next valid ID therein).

```
Public Overridable Sub reqIds(ByVal numIds As Integer)
```

Parameter	Description
<b>numIds</b>	Set to 1.

## reqContractDetailsEx()

Call this method to download all details for a particular contract. The contract details will be received via the [contractDetailsEx\(\)](#) callback in dispinterface\_DTwsevents.

```
Public Overridable Sub reqContractDetailsEx(ByVal reqId As Integer,  
ByVal contract As TWSLib.IContract)
```

Parameter	Description
<b>reqId</b>	The ID of the data request. Ensures that responses are matched to requests if several requests are in process.
<b>contract</b>	This object contains a description of the contract for which market data is being requested.

## reqContractDetails()

Call this method to request details for a particular underlying or security. The contract details data will be returned by the [contractDetails\(\)](#) event.

**Note:** This method has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated methods.

```
Public Overridable Sub reqContractDetails(ByVal symbol As String, ByVal secType As String, ByVal expiry As String, ByVal strike As Double, ByVal right As String, ByVal multiplier As String, ByVal exchange As String, ByVal currency As String, ByVal includeExpired As Integer)
```

Parameter	Description
<b>symbol</b>	The symbol of the underlying for which you are requesting market data.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"><li>• STK</li><li>• OPT</li><li>• FUT</li><li>• IND</li><li>• FOP</li><li>• CASH</li></ul>
<b>expiry</b>	The expiration date. Use the format YYYYMM.
<b>strike</b>	The strike price.
<b>right</b>	Specifies a Put or Call. Valid values are: P, PUT, C, CALL.
<b>multiplier</b>	Allows you to specify a futures or options multiplier in cases where multiple possibilities exist.
<b>exchange</b>	The order destination, such as SMART.
<b>currency</b>	Specifies the currency. Ambiguities may require that this field be specified, for example, when SMART is the exchange and IBM is being requested (IBM can trade in GBP or USD). Given the existence of this kind of ambiguity, it is a good idea to always specify the currency.

## reqContractDetails2()

Call this method to request details for a particular security using the exchange symbol. The contract details data will be returned by the [contractDetails\(\)](#) event.

```
Public Overridable Sub reqContractDetails2(ByVal localSymbol As String,  
    ByVal secType As String, ByVal exchange As String, ByVal currency As  
    String, ByVal includeExpired As Integer)
```

Parameter	Description
<b>localSymbol</b>	The local exchange symbol of the underlying for which you are requesting market data.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"> <li>• STK</li> <li>• OPT</li> <li>• FUT</li> <li>• IND</li> <li>• FOP</li> <li>• CASH</li> </ul>
<b>exchange</b>	The order destination, such as SMART.
<b>currency</b>	Specifies the currency. Ambiguities may require that this field be specified, for example, when SMART is the exchange and IBM is being requested (IBM can trade in GBP or USD). Given the existence of this kind of ambiguity, it is a good idea to always specify the currency.

## reqMktDepthEx()

Call this method to request market depth for a specific contract. The market depth will be returned by the [updateMktDepth\(\)](#) and [updateMktDepthL2\(\)](#) events.

```
Public Overridable Sub reqMktDepthEx(ByVal tickerId As Integer, ByVal  
    contract As TWSLib.IContract, ByVal numRows As Integer)
```

Parameter	Description
<b>tickerId</b>	The ticker Id. Must be a unique value. When the market depth data returns, it will be identified by this tag. This is also used when canceling the market depth.
<b>contract</b>	This object contains a description of the contract for which market data is being requested.
<b>numRows</b>	Specifies the number of market depth rows to return.

## reqMktDepth()

Call this method to request market depth details for a particular underlying or security. The market depth data will be returned by the [updateMktDepth\(\)](#) and [updateMktDepthL2\(\)](#) events.

**Note:** This method has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated methods.

```
Public Overridable Sub reqMktDepth(ByVal id As Integer, ByVal symbol As String, ByVal secType As String, ByVal expiry As String, ByVal strike As Double, ByVal right As String, ByVal multiplier As String, ByVal exchange As String, ByVal currency As String, ByVal numRows As Integer)
```

Parameter	Description
<b>id</b>	The ticker id. Must be a unique value. When the market data returns, it will be identified by this tag. This is also used when canceling the market data.
<b>symbol</b>	The symbol of the underlying for which you are requesting market data.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"><li>• STK</li><li>• OPT</li><li>• FUT</li><li>• IND</li><li>• FOP</li><li>• CASH</li></ul>
<b>expiry</b>	The expiration date. Use the format YYYYMM.
<b>strike</b>	The strike price.
<b>right</b>	Specifies a Put or Call. Valid values are: P, PUT, C, CALL.
<b>multiplier</b>	Allows you to specify a futures or options multiplier in cases where multiple possibilities exist.
<b>exchange</b>	The order destination, such as SMART.
<b>currency</b>	Specifies the currency. Ambiguities may require that this field be specified, for example, when SMART is the exchange and IBM is being requested (IBM can trade in GBP or USD). Given the existence of this kind of ambiguity, it is a good idea to always specify the currency.
<b>numRows</b>	Specifies the number of market depth rows you want to display.

## reqMktDepth2()

Call this method to request market depth details for a particular underlying or security. The market depth data will be returned by the [updateMktDepth\(\)](#) and [updateMktDepthL2\(\)](#) events.

**Note:** This method has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated methods.

```
Public Overridable Sub reqMktDepth2(ByVal id As Integer, ByVal localSymbol As String, ByVal secType As String, ByVal exchange As String, ByVal currency As String, ByVal numRows As Integer)
```

Parameter	Description
<b>id</b>	The ticker id. Must be a unique value. When the market data returns, it will be identified by this tag. This is also used when canceling the market data.
<b>localSymbol</b>	The local exchange symbol of the underlying for which you are requesting market data.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"> <li>• STK</li> <li>• OPT</li> <li>• FUT</li> <li>• IND</li> <li>• FOP</li> <li>• CASH</li> </ul>
<b>exchange</b>	The order destination, such as SMART.
<b>currency</b>	Specifies the currency. Ambiguities may require that this field be specified, for example, when SMART is the exchange and IBM is being requested (IBM can trade in GBP or USD). Given the existence of this kind of ambiguity, it is a good idea to always specify the currency.
<b>numRows</b>	Specifies the number of market depth rows you want to display.

## cancelMktDepth()

After calling this method, market depth for the specified id will stop flowing.

```
Public Overridable Sub cancelMktDepth(ByVal id As Integer)
```

Parameter	Description
<b>id</b>	The ID that was specified in the call to reqMktDepth() or reqMktDepth2().

## addComboLeg()

This method adds a leg definition to an order and must be called for each leg before the combination order can be placed. Combination orders must have a secType of 'BAG.'

**Note:** This method has been deprecated as of API release 9.4 (December 2007) and will soon be removed.

```
Public Overridable Sub addComboLeg(ByVal conId As Integer, ByVal action As String, ByVal ratio As Integer, ByVal exchange As String, ByVal openClose As Integer, ByVal shortSaleSlot As Integer, ByVal designatedLocation As String)
```

Parameter	Description
<b>conid</b>	The unique contract identifier specifying the security.
<b>action</b>	Select the side (buy or sell) for the leg you are constructing.
<b>ratio</b>	Select the relative number of contracts for the leg you are constructing. To help determine the ratio for a specific combination order, use Interactive Analytics (see Interactive Analytics in the User's Guide).
<b>exchange</b>	The exchange to which the complete combination order will be routed.
<b>openClose</b>	Specifies whether the order is an open or close order. Valid values are: <ul style="list-style-type: none"> <li>• Same - (0) same as the parent security. This is the only option for retail customers.</li> <li>• Open - (1) This option is for Institutional Customers only.</li> <li>• Close - (2) This option is for Institutional Customers only.</li> </ul>
<b>shortSaleSlot</b>	For institutional customers only. <ul style="list-style-type: none"> <li>• 0 - inapplicable (i.e. retail customer or not short leg)</li> <li>• 1 - clearing broker</li> <li>• 2 - third party. If this value is used, you must enter a designated location.</li> </ul>
<b>designatedLocation</b>	If shortSaleSlot == 2, the designatedLocation must be specified. <b>Otherwise leave blank or orders will be rejected.</b>

## clearComboLegs()

This method must be called to remove all leg definitions from a combination order before constructing a new combination order.

**Note:** This method has been deprecated as of API release 9.4 (December 2007) and will soon be removed.

```
Public Overridable Sub clearComboLegs()
```

## reqNewsBulletins()

Call this method to start receiving news bulletins. Each bulletin will be returned by the updateNewsBulletin() event.

```
Public Overridable Sub reqNewsBulletins(ByVal allDaysMsgs As Integer)
```

Parameter	Description
<b>allDaysMsgs</b>	If set to TRUE, returns all the existing bulletins for the current day and any new ones. If set to FALSE, will only return new bulletins.

## cancelNewsBulletins()

Call this method to stop receiving news bulletins.

```
Public Overridable Sub cancelNewsBulletins()
```

## setServerLogLevel()

The default level is ERROR. See [API Logging](#) for more details.

```
Public Overridable Sub setServerLogLevel(ByVal logLevel As Integer)
```

Parameter	Description
<b>logLevel</b>	<p>Specifies the level of log entry detail used by the server when processing API requests. Valid values include:</p> <ul style="list-style-type: none"> <li>• 1 = SYSTEM</li> <li>• 2 = ERROR</li> <li>• 3 = WARNING</li> <li>• 4 = INFORMATION</li> <li>• 5 = DETAIL</li> </ul>

## **reqManagedAccts()**

Call this method to request the list of managed accounts. The list will be returned by the [managedAccounts\(\)](#) event.

**Note:** This request can only be made when connected to a Financial Advisor account.

```
Public Overridable Sub reqManagedAccts()
```

## **reqAccountUpdates()**

Call this method to request account updates. The account data will be fed back through the [updateAccountTime\(\)](#), [updateAccountValue\(\)](#) and [updatePortfolio\(\)](#) events.

```
Public Overridable Sub reqAccountUpdates(ByVal subscribe As Integer,  
 ByVal acctCode As String)
```

Parameter	Description
<b>subscribe</b>	If set to 1, the client will start receiving account and portfolio updates. If set to 2, the client will stop receiving this information.
<b>acctCode</b>	The account code for which to receive account and portfolio updates.

## **requestFA()**

Call this method to request FA configuration information from the server. The data returns in an XML string via the [receiveFA\(\)](#) event.

```
Public Overridable Sub requestFA(ByVal faDataType As Integer)
```

Parameter	Description
<b>faDataType</b>	Specifies the type of Financial Advisor configuration data being requested. Valid values include: <ul style="list-style-type: none"><li>• 1 = GROUPS</li><li>• 2 = PROFILE</li><li>• 3 =ACCOUNT ALIASES</li></ul>

## replaceFA()

Call this method to modify FA configuration information from the API. Note that this can also be done manually.

```
Public Overridable Sub replaceFA(ByVal faDataType As Integer, ByVal cxml  
As String)
```

Parameter	Description
<b>faDataType</b>	Specifies the type of Financial Advisor configuration data being modified via the API. Valid values include: <ul style="list-style-type: none"><li>• 1 = GROUPS</li><li>• 2 = PROFILE</li><li>• 3 =ACCOUNT ALIASES</li></ul>
<b>cxml</b>	The XML string containing the new FA configuration information.

## reqHistoricalDataEx()

Call this method to start receiving historical data results through the historicalData() event.

**Note:** For a information about historical data request limitations, see [Historical Data Limitations](#).

```
Public Overridable Sub reqHistoricalDataEx(ByVal tickerId As Integer,  
    ByVal contract As TWSLib.IContract, ByVal endTime As String, ByVal  
    duration As String, ByVal barSize As String, ByVal whatToShow As String,  
    ByVal useRTH As Integer, ByVal formatDate As Integer)
```

Parameter	Description
<b>tickerId</b>	The Id for the request. Must be a unique value. When the data is received, it will be identified by this Id. This is also used when canceling the historical data request.
<b>contract</b>	This structure contains a description of the contract for which market data is being requested.
<b>endTime</b>	Use the format yyyyymmdd hh:mm:ss tmz, where the time zone is allowed (optionally) after a space at the end.
<b>durationStr</b>	This is the time span the request will cover, and is specified using the format: <integer> <unit>, i.e., 1 D, where valid units are: <ul style="list-style-type: none"><li>• S (seconds)</li><li>• D (days)</li><li>• W (weeks)</li><li>• M (months)</li><li>• Y (years)</li></ul> <b>Note:</b> If no unit is specified, seconds are used. Also, note "years" is currently limited to one.

Parameter	Description																																			
<b>barSize</b>	<p>The size of the bars that will be returned (within IB/TWS limits). Valid values include:</p> <table> <thead> <tr> <th>Bar Size</th> <th>Parametric Value</th> </tr> </thead> <tbody> <tr><td>1 sec</td><td>1</td></tr> <tr><td>5 secs</td><td>2</td></tr> <tr><td>15 secs</td><td>3</td></tr> <tr><td>30 secs</td><td>4</td></tr> <tr><td>1 min.</td><td>5</td></tr> <tr><td>2 mins</td><td>6</td></tr> <tr><td>3 mins</td><td>16</td></tr> <tr><td>5 mins.</td><td>7</td></tr> <tr><td>15 mins</td><td>8</td></tr> <tr><td>30 mins</td><td>9</td></tr> <tr><td>1 hour</td><td>10</td></tr> <tr><td>1 day</td><td>11</td></tr> <tr><td>1 week</td><td>12</td></tr> <tr><td>1 month</td><td>13</td></tr> <tr><td>3 months</td><td>14</td></tr> <tr><td>1 year</td><td>15</td></tr> </tbody> </table>		Bar Size	Parametric Value	1 sec	1	5 secs	2	15 secs	3	30 secs	4	1 min.	5	2 mins	6	3 mins	16	5 mins.	7	15 mins	8	30 mins	9	1 hour	10	1 day	11	1 week	12	1 month	13	3 months	14	1 year	15
Bar Size	Parametric Value																																			
1 sec	1																																			
5 secs	2																																			
15 secs	3																																			
30 secs	4																																			
1 min.	5																																			
2 mins	6																																			
3 mins	16																																			
5 mins.	7																																			
15 mins	8																																			
30 mins	9																																			
1 hour	10																																			
1 day	11																																			
1 week	12																																			
1 month	13																																			
3 months	14																																			
1 year	15																																			
<b>whatToShow</b>	<p>Determines the nature of data being extracted. Valid values include:</p> <ul style="list-style-type: none"> <li>• TRADES</li> <li>• MIDPOINT</li> <li>• BID</li> <li>• ASK</li> <li>• BID_ASK</li> <li>• HISTORICAL_VOLATILITY</li> <li>• OPTION_IMPLIED_VOLATILITY</li> <li>• OPTION_VOLUME</li> </ul>																																			
<b>useRTH</b>	<p>Determines whether to return all data available during the requested time span, or only data that falls within regular trading hours. Valid values include:</p> <ul style="list-style-type: none"> <li>• 0 - all data is returned even where the market in question was outside of its regular trading hours.</li> <li>• 1 - only data within the regular trading hours is returned, even if the requested time span falls partially or completely outside of the RTH.</li> </ul>																																			

Parameter	Description
<b>formatDate</b>	Determines the date format applied to returned bars. Valid values include: <ul style="list-style-type: none"><li>• 1 - dates applying to bars returned in the format: yyyyymmdd{space}{space}hh:mm:dd</li><li>• 2 - dates are returned as a long integer specifying the number of seconds since 1/1/1970 GMT.</li></ul>

## reqHistoricalData()

This method has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated methods.

```
Public Overridable Sub reqHistoricalData(ByVal id As Integer, ByVal symbol As String, ByVal secType As String, ByVal expiry As String, ByVal strike As Double, ByVal right As String, ByVal multiplier As String, ByVal exchange As String, ByVal currency As String, ByVal isExpired As Integer, ByVal endTime As String, ByVal durationStr As String, ByVal barSizeSetting As String, ByVal whatToShow As String, ByVal useRTH As Integer, ByVal formatDate As Integer)
```

Parameter	Description
<b>id</b>	The ticker id. Must be a unique value. When the market data returns, it will be identified by this tag. This is also used when canceling the market data.
<b>symbol</b>	The symbol of the underlying for which you are requesting market data.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"> <li>• STK</li> <li>• OPT</li> <li>• FUT</li> <li>• IND</li> <li>• FOP</li> <li>• CASH</li> </ul>
<b>expiry</b>	The expiration date. Use the format YYYYMM.
<b>strike</b>	The strike price.
<b>right</b>	Specifies a Put or Call. Valid values are: P, PUT, C, CALL.
<b>multiplier</b>	Allows you to specify a futures or options multiplier in cases where multiple possibilities exist.
<b>exchange</b>	The order destination, such as Smart.
<b>currency</b>	Specifies the currency. Ambiguities may require that this field be specified, for example, when SMART is the exchange and IBM is being requested (IBM can trade in GBP or USD). Given the existence of this kind of ambiguity, it is a good idea to always specify the currency.
<b>endTime</b>	Defines a query end date and time at any point during the past 6 mos. Valid values include any date/time within the past six months in the format: yyyyymmdd HH:mm:ss zzz where "zzz" is the optional time zone.
<b>durationStr</b>	Set the query duration up to one week, using a time unit of seconds, days or weeks. Valid values include any integer followed by a space and then S (seconds), D (days) or W (week). If no unit is specified, seconds is used.

Parameter	Description																																	
<b>barSizeSetting</b>	<p>Specifies the size of the bars that will be returned (within IB/TWS limits). Valid values include:</p> <p>Specifies the size of the bars that will be returned (within IB/TWS limits). Valid values include:</p> <table> <thead> <tr> <th>Bar Size</th> <th>Parametric Value</th> </tr> </thead> <tbody> <tr> <td>1 sec</td> <td>1</td> </tr> <tr> <td>5 secs</td> <td>2</td> </tr> <tr> <td>15 secs</td> <td>3</td> </tr> <tr> <td>30 secs</td> <td>4</td> </tr> <tr> <td>1 min.</td> <td>5</td> </tr> <tr> <td>2 mins</td> <td>6</td> </tr> <tr> <td>5 mins</td> <td>7</td> </tr> <tr> <td>15 mins</td> <td>8</td> </tr> <tr> <td>30 mins</td> <td>9</td> </tr> <tr> <td>1 hour</td> <td>10</td> </tr> <tr> <td>1 day</td> <td>11</td> </tr> <tr> <td>1 week</td> <td>12</td> </tr> <tr> <td>1 month</td> <td>13</td> </tr> <tr> <td>3 months</td> <td>14</td> </tr> <tr> <td>1 year</td> <td>15</td> </tr> </tbody> </table>		Bar Size	Parametric Value	1 sec	1	5 secs	2	15 secs	3	30 secs	4	1 min.	5	2 mins	6	5 mins	7	15 mins	8	30 mins	9	1 hour	10	1 day	11	1 week	12	1 month	13	3 months	14	1 year	15
Bar Size	Parametric Value																																	
1 sec	1																																	
5 secs	2																																	
15 secs	3																																	
30 secs	4																																	
1 min.	5																																	
2 mins	6																																	
5 mins	7																																	
15 mins	8																																	
30 mins	9																																	
1 hour	10																																	
1 day	11																																	
1 week	12																																	
1 month	13																																	
3 months	14																																	
1 year	15																																	
<b>whatToShow</b>	<p>Determines the nature of data being extracted. Valid values include:</p> <ul style="list-style-type: none"> <li>• TRADES</li> <li>• MIDPOINT</li> <li>• BID</li> <li>• ASK</li> <li>• BID_ASK</li> </ul>																																	
<b>useRTH</b>	<p>Determines whether to return all data available during the requested time span, or only data that falls within regular trading hours. Valid values include:</p> <ul style="list-style-type: none"> <li>• 0 - all data is returned even where the market in question was outside of its regular trading hours.</li> <li>• 1 - only data within the regular trading hours is returned, even if the requested time span falls partially or completely outside of the RTH.</li> </ul>																																	
<b>formatDate</b>	<p>Determines the date format applied to returned bars. Valid values include:</p> <ul style="list-style-type: none"> <li>• 1 - dates applying to bars returned in the format: yyyyymmdd{space}{space}hh:mm:dd</li> <li>• 2 - dates are returned as a long integer specifying the number of seconds since 1/1/1970 GMT.</li> </ul>																																	

## exerciseOptionsEx()

Call this method to exercise options.

**Note:** Please note that SMART is not an allowed exchange in [exerciseOptionsEx\(\)](#) calls, and that TWS does a request for the position in question whenever any API initiated exercise or lapse is attempted.

```
Public Overridable Sub exerciseOptionsEx(ByVal tickerId As Integer,
 ByVal contract As TWSLib.IContract, ByVal exerciseAction As Integer,
 ByVal exerciseQuantity As Integer, ByVal account As String, ByVal
 override As Integer)
```

Parameter	Description
<b>tickerId</b>	The Id for the exercise request
<b>contract</b>	This structure contains a description of the contract for which market data is being requested.
<b>exerciseAction</b>	This can have two values: <ul style="list-style-type: none"> <li>• 1 = exercise</li> <li>• 2 = lapse</li> </ul>
<b>exerciseQuantity</b>	The number of contracts to be exercised
<b>account</b>	For institutional orders. Specifies the IB account.
<b>override</b>	Specifies whether your setting will override the system's natural action. For example, if your action is "exercise" and the option is not in-the-money, by natural action the option would not exercise. If you have override set to "yes" the natural action would be overridden and the out-of-the money option would be exercised. Values are: <ul style="list-style-type: none"> <li>• 0 = do not override</li> <li>• 1 = override</li> </ul>

## **exerciseOptions()**

**Note:** This method has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated methods.

```
Public Overridable Sub exerciseOptions(ByVal id As Integer, ByVal symbol As String, ByVal secType As String, ByVal expiry As String, ByVal strike As Double, ByVal right As String, ByVal multiplier As String, ByVal exchange As String, ByVal currency As String, ByVal exerciseAction As Integer, ByVal exerciseQuantity As Integer, ByVal override As Integer)
```

Parameter	Description
<b>id</b>	The ticker id. Must be a unique value.
<b>symbol</b>	The symbol of the underlying
<b>secType</b>	The option type. Values include: OPT, FOP, WAR
<b>expiry</b>	The expiration date. Use the format YYYYMM
<b>strike</b>	The strike price
<b>right</b>	The option right, use values P, PUT, C, or CALL
<b>multiplier</b>	Allows you to specify a futures or option multiplier in cases where more than one possibility exists. If no multiplier is specified, a default value of "100" is assumed.
<b>exchange</b>	The order destination.
<b>currency</b>	Specifies the currency. Ambiguities may require that this field be specified, for example, when SMART is the exchange and IBM is being requested (IBM can trade in GBP or USD). Given the existence of this kind of ambiguity, it is a good idea to always specify the currency.
<b>exerciseAction</b>	Specifies whether you want the option to lapse or be exercised. Values are 1 = exercise, 2 = lapse.
<b>exerciseQuantity</b>	The quantity you want to exercise.
<b>override</b>	Specifies whether your setting will override the system's natural action. For example, if your action is "exercise" and the option is not in-the-money, by natural action the option would not exercise. If you have override set to "yes" the natural action would be overridden and the out-of-the money option would be exercised. Values are: 0 = no, 1 = yes.

## **reqScannerParameters()**

Requests an XML string that describes all possible scanner queries.

```
Public Overridable Sub reqScannerParameters()
```

## reqScannerSubscriptionEx()

Call the reqScannerSubscriptionEX() method to start receiving market scanner results through the [scannerDataEx\(\)](#) event.

```
Public Overridable Sub reqScannerSubscriptionEx(ByVal tickerId As Integer, ByVal subscription As TWSLib.IScannerSubscription)
```

Parameter	Description
<b>tickerId</b>	The Id for the subscription. Must be a unique value. When the subscription data is received, it will be identified by this Id. This is also used when canceling the scanner.
<b>subscription</b>	Summary of the scanner subscription parameters including filters.

## reqScannerSubscription()

**Note:** This method has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated methods.

```
Public Overridable Sub reqScannerSubscription(ByVal tickerId As Integer, ByVal numberOfRows As Integer, ByVal instrument As String, ByVal locationCode As String, ByVal scanCode As String, ByVal abovePrice As Double, ByVal belowPrice As Double, ByVal aboveVolume As Integer, ByVal marketCapAbove As Double, ByVal marketCapBelow As Double, ByVal moodyRatingAbove As String, ByVal moodyRatingBelow As String, ByVal spRatingAbove As String, ByVal spRatingBelow As String, ByVal maturityDateAbove As String, ByVal maturityDateBelow As String, ByVal couponRateAbove As Double, ByVal couponRateBelow As Double, ByVal excludeConvertible As Integer, ByVal averageOptionVolumeAbove As Integer, ByVal scannerSettingPairs As String, ByVal stockTypeFilter As String)
```

Parameter	Description
<b>tickerId</b>	The ticker ID. Must be a unique value.
<b>numberOfRows</b>	Defines the number or rows to display.
<b>instrument</b>	Defines the instrument type used in the scan. Values include STK,
<b>locationCode</b>	Currently limited to US Stocks, with NYSE, AMEX, NASDAQ, and OTCBB being sub-types. STK.US is a legal value for this parameter. Available values are listed in the scanner parameters XML document.
<b>scanCode</b>	The type of the scan, such as HIGH_OPT_VOLUME_PUT_CALL_RATIO. Available values are listed in the scanner parameters XML document.
<b>abovePrice</b>	Filter out contracts with a price lower than this value. Can be left blank.
<b>belowPrice</b>	Filter out contracts with a price above than this value. Can be left blank.

Parameter	Description
<b>aboveVolume</b>	Filter out contracts with a volume below this value. Can be left blank.
<b>marketCapAbove</b>	Filter out contracts with a market cap above this value. Can be left blank.
<b>marketCapBelow</b>	Filter out contracts with a market cap below this value. Can be left blank.
<b>moodyRatingAbove</b>	Filter out contracts with a moody rating above this value. Can be left blank.
<b>moodyRatingBelow</b>	Filter out contracts with a moody rating below this value. Can be left blank.
<b>spRatingAbove</b>	Filter out contracts with an S&P rating lower than this value. Can be left blank.
<b>spRatingBelow</b>	Filter out contracts with an S&P rating higher than this value. Can be left blank.
<b>maturityDateAbove</b>	Filter out contracts with a maturity date later than this value. Can be left blank.
<b>maturityDateBelow</b>	Filter out contracts with a maturity date earlier than this value. Can be left blank.
<b>couponRateAbove</b>	Filter out contracts with a coupon rate lower than this value. Can be left blank.
<b>couponRateBelow</b>	Filter out contracts with a coupon rate higher than this value. Can be left blank.
<b>excludeConvertible</b>	Filter out convertible bonds. Can be left blank.
<b>scannerSettingPairs</b>	Used with the scanCode to help further narrow your query return. Scanner Setting Pairs are delimited by slashes, making this parameter open ended. A pair example is "Annual,true" which when used with Top Option Implied Vol % Gainers scan would return annualized volatilities.
<b>stockTypeFilter</b>	defines the stock type to be returned with respect to stocks and ETFs. Valid values include: <ul style="list-style-type: none"> <li>• ALL - excludes nothing</li> <li>• STOCK - which excludes ETFs</li> <li>• ETF - which includes ETFs</li> </ul>
<b>averageOptionVolume Above</b>	Filter out contracts with an average volume lower than this value.

## cancelHistoricalData()

Used if an internet disconnect has occurred or the results of a query are otherwise delayed and the application is no longer interested in receiving the data.

```
Public Overridable Sub cancelHistoricalData(ByVal tickerId As Integer)
```

Parameter	Description
tickerId	The ticker ID. Must be a unique value.

## cancelScannerSubscription()

```
Public Overridable Sub cancelScannerSubscription(ByVal tickerId As Integer)
```

Parameter	Description
tickerId	The ticker ID. Must be a unique value.

## reqRealTimeBarsEx()

Call the reqRealTimeBarsEx() method to start receiving real time bar results through the [realtimeBar\(\)](#) event.

```
Public Overridable Sub reqRealTimeBarsEx(ByVal tickerId As Integer, ByVal contract As TWSLib.IContract, ByVal barSize As Integer, ByVal whatToShow As String, ByVal useRTH As Integer)
```

Parameter	Description
tickerId	The Id for the request. Must be a unique value. When the data is received, it will be identified by this Id. This is also used when canceling the historical data request.
contract	This structure contains a description of the contract for which market data is being requested.
barSize	Currently only 5 second bars are supported, if any other value is used, an exception will be thrown.
whatToShow	Determines the nature of the data extracted. Valid values include: <ul style="list-style-type: none"> <li>• TRADES</li> <li>• BID</li> <li>• ASK</li> <li>• MIDPOINT</li> </ul>
useRTH	Regular Trading Hours only. Valid values include: <ul style="list-style-type: none"> <li>• 0 = all data available during the time span requested is returned, including time intervals when the market in question was outside of regular trading hours.</li> <li>• 1 = only data within the regular trading hours for the product requested is returned, even if the time span falls partially or completely outside.</li> </ul>

## reqRealTimeBars()

Call the reqRealTimeBars() method to start receiving real time bar results through the [realtimeBar\(\)](#) event.

**Note:** This method has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated methods.

```
Public Overridable Sub reqRealTimeBars(ByVal tickerId As Integer, ByVal symbol As String, ByVal secType As String, ByVal expiry As String, ByVal strike As Double, ByVal right As String, ByVal multiplier As String, ByVal exchange As String, ByVal primaryExchange As String, ByVal currency As String, ByVal isExpired As Integer, ByVal barSize As Integer, ByVal whatToShow As String, ByVal useRTH As Integer)
```

Parameter	Description
<b>tickerId</b>	The Id for the request. Must be a unique value. When the data is received, it will be identified by this Id. This is also used when canceling the historical data request.
<b>symbol</b>	This is the symbol of the underlying asset.
<b>secType</b>	This is the security type. Valid values are: <ul style="list-style-type: none"><li>• STK</li><li>• OPT</li><li>• FUT</li><li>• IND</li><li>• FOP</li><li>• CASH</li><li>• BAG</li></ul>
<b>expiry</b>	The expiration date. Use the format YYYYMM.
<b>strike</b>	The strike price.
<b>right</b>	Specifies a Put or Call. Valid values are: P, PUT, C, CALL.
<b>multiplier</b>	Allows you to specify a future or option contract multiplier. This is only necessary when multiple possibilities exist.
<b>exchange</b>	The order destination, such as Smart.
<b>primaryExchange</b>	Identifies the listing exchange for the contract (do not list SMART).
<b>currency</b>	Specifies the currency. Ambiguities may require that this field be specified, for example, when SMART is the exchange and IBM is being requested (IBM can trade in GBP or USD). Given the existence of this kind of ambiguity, it is a good idea to always specify the currency.
<b>isExpired</b>	Specifies that the contract has expired.
<b>barSize</b>	Currently only 5 second bars are supported, if any other value is used, an exception will be thrown.

Parameter	Description
<b>whatToShow</b>	Determines the nature of the data extracted. Valid values include: <ul style="list-style-type: none"> <li>• TRADES</li> <li>• BID</li> <li>• ASK</li> <li>• MIDPOINT</li> </ul>
<b>useRTH</b>	Regular Trading Hours only. Valid values include: <ul style="list-style-type: none"> <li>• 0 = all data available during the time span requested is returned, including time intervals when the market in question was outside of regular trading hours.</li> <li>• 1 = only data within the regular trading hours for the product requested is returned, even if the time span falls partially or completely outside.</li> </ul>

### **cancelRealTimeBars()**

Used if an internet disconnect has occurred or the results of a query are otherwise delayed and the application is no longer interested in receiving the data.

```
Public Overridable Sub cancelRealTimeBars(ByVal tickerId As Integer)
```

Parameter	Description
<b>tickerId</b>	The ticker ID. Must be a unique value.

### **reqCurrentTime()**

Returns the current system time on the server side.

```
Public Overridable Sub reqCurrentTime()
```

### **createComboLegList()**

This factory method is used to create an [IComboLegList](#) COM object.

```
Public Overridable Sub createComboLegList() As TWSLib.IComboLegList
```

You must use the factory “create” methods to create the COM objects in this section. For example, the `createComboLegList()` method creates an `IComboLeg` object. The `IComboLeg` object contains the definition of the leg list.

## **createContract()**

This factory method is used to create an [IContract](#) COM object.

```
Public Overridable Sub createContract() As TWSLib.IContract
```

You must use the factory "create" methods to create the COM objects described in this chapter. The createContract() method creates an IContract object, which contains a description of the contract for which market data is being requested.

## **createExecutionFilter()**

This factory method is used to create an [IExecutionFilter](#) COM object.

```
Public Overridable Sub createExecutionFilter() As  
TWSLib.IExecutionFilter
```

You must use the factory "create" methods to create the COM objects described in this chapter. The createExecutionFilter() method creates an IExecutionFilter object, which contains the filter criteria used to determine which execution reports are returned.

## **createOrder()**

This factory method is used to create an [IOrder](#) COM object.

```
Public Overridable Sub createOrder() As TWSLib.IOrder
```

You must use the factory "create" methods to create the COM objects described in this chapter. The createOrder() method creates an IOrder object, which contains the details of an order.

## **createScannerSubscription()**

This factory method is used to create an [IScannerSubscription](#) COM object.

```
Public Overridable Sub createScannerSubscription() As  
TWSLib.IScannerSubscription
```

You must use the factory "create" methods to create the COM objects described in this chapter. The createScannerSubscription() method creates an ISscannerSubscription object, which contains a summary of the scanner subscription parameters.

## **createTagValueList**

This factory method is used to create [ITagValueList](#) and [ITagValue](#) objects.

```
Public Overridable Function createTagValueList() As TWSLib.ITagValueList
```

You must use the factory "create" methods to create the COM objects described in this chapter.

## createUnderComp()

This factory method is used to create an [IUnderComp](#) COM object.

```
Public Overridable Sub createUnderComp() As TWSLib.IScannerSubscription
```

You must use the factory “create” methods to create the COM objects described in this chapter. The createUnderComp() method creates an IUnderComp object, which is used to define a Delta-Neutral Combo contract.

## reqFundamentalData()

Call this method to receive Reuters global fundamental data. There must be a subscription to Reuters Fundamental set up in Account Management before you can receive this data.

```
Public Overridable Sub reqFundamentalData(ByVal reqId As Integer, ByVal
contract As TWSLib.IContract, ByVal reportType As String)
```

Parameter	Description
<b>reqId</b>	The ID of the data request.
<b>contract</b>	This structure contains a description of the contract for which Reuters Fundamental data is being requested.
<b>reportType</b>	Identifies the report type, which is one of the following: <ul style="list-style-type: none"> <li>• Estimates</li> <li>• Financial Statements</li> <li>• Summary</li> </ul>

## cancelFundamentalData()

Call this method to stop receiving Reuters global fundamental data.

```
Public Overridable Sub cancelFundamentalData(ByVal reqId As Integer)
```

Parameter	Description
<b>reqId</b>	The ID of the data request.

## ActiveX Events

ActiveX events receive information from the system and make it available to an application. This section defines the ActiveX events you can receive via the DTwsEvents interface.

**Note:** Please note that events in red have been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated events. The new "Ex" events have been grouped with their deprecated counterparts for easy identification.

<b>Connection and Server</b>	<b>Contract Details</b>
<a href="#">connectionClosed()</a>	<a href="#">contractDetailsEx()</a>
<a href="#">currentTime()</a>	<a href="#">contractDetails()</a>
<a href="#">errMsg()</a>	<a href="#">contractDetailsEnd()</a>
<b>Market Data</b>	<a href="#">bondContractDetails()</a>
<a href="#">tickPrice()</a>	<b>Executions</b>
<a href="#">tickSize()</a>	<a href="#">execDetailsEx()</a>
<a href="#">tickOptionComputation()</a>	<a href="#">execDetails()</a>
<a href="#">tickGeneric()</a>	<a href="#">execDetailsEnd()</a>
<a href="#">tickString()</a>	<b>Market Depth</b>
<a href="#">tickEFP()</a>	<a href="#">updateMktDepth()</a>
<a href="#">tickSnapshotEnd()</a>	<a href="#">updateMktDepthL2()</a>
<b>Orders</b>	<b>Financial Advisors</b>
<a href="#">orderStatus()</a>	<a href="#">managedAccounts()</a>
<a href="#">openOrderEx()</a>	<a href="#">receiveFA()</a>
<a href="#">openOrder1()</a>	<b>Historical Data</b>
<a href="#">openOrder2()</a>	<a href="#">historicalData()</a>
<a href="#">openOrder3()</a>	<b>Market Scanners</b>
<a href="#">openOrder4()</a>	<a href="#">scannerParameters()</a>
<a href="#">nextValidId()</a>	<a href="#">scannerDataEx()</a>
<a href="#">permId()</a>	<a href="#">scannerData()</a>
<b>Account and Portfolio</b>	<a href="#">scannerDataEnd()</a>
<a href="#">updateAccountValue()</a>	<b>Real Time Bars</b>
<a href="#">updatePortfolioEx()</a>	<a href="#">realtimebar()</a>
<a href="#">updatePortfolio()</a>	<b>Fundamental Data</b>
<a href="#">updateAccountTime()</a>	<a href="#">fundamentalData()</a>
<b>News Bulletins</b>	
<a href="#">updateNewsBulletin()</a>	

## tickPrice()

This function is called when the market data changes. Prices are updated immediately with no delay.

```
Sub tickPrice(ByVal id As Integer, ByVal tickType As Integer, ByVal price
As Double, ByVal canAutoExecute As Integer)
```

Parameter	Description
<b>id</b>	The ticker ID that was specified previously in the call to reqMktData()
<b>tickType</b>	Specifies the type of price. Possible values are: <ul style="list-style-type: none"> <li>• 1 = bid</li> <li>• 2 = ask</li> <li>• 4 = last</li> <li>• 6 = high</li> <li>• 7 = low</li> <li>• 9 = close</li> </ul>
<b>price</b>	The bid, ask or last price, the daily high, daily low or last day close, depending on tickType value.
<b>canAutoExecute</b>	Specifies whether the price tick is available for automatic execution. Possible values are: <ul style="list-style-type: none"> <li>• 0 = not eligible for automatic execution</li> <li>• 1 = eligible for automatic execution</li> </ul>

## tickSize()

This function is called when the market data changes. Sizes are updated immediately with no delay.

```
Sub tickSize(ByVal id As Integer, ByVal tickType As Integer, ByVal size
As Integer)
```

Parameter	Description
<b>id</b>	The ticker ID that was specified previously in the call to reqMktData()
<b>tickType</b>	Specifies the type of price. Possible values are: <ul style="list-style-type: none"> <li>• 0 = bid size</li> <li>• 3 = ask size</li> <li>• 5 = last size</li> <li>• 8 = volume</li> </ul>
<b>size</b>	The bid size, ask size, last size or trading volume, depending on the tickType value.

## tickOptionComputation()

```
Sub tickOptionComputation(ByVal id As Integer, ByVal tickType As Integer, ByVal impliedVol As Double, ByVal delta As Double, ByVal modelPrice As Double, ByVal pvDividend As Double)
```

Parameter	Description
<b>id</b>	The ticker ID that was specified previously in the call to reqMktData()
<b>tickType</b>	Specifies the type of tick. Possible values are: <ul style="list-style-type: none"> <li>• 10 = Bid</li> <li>• 11 = Ask</li> <li>• 12 = Last</li> </ul>
<b>ImpliedVol</b>	The implied volatility calculated by the TWS option modeler, using the specified ticktype value.
<b>delta</b>	The option delta calculated by the TWS option modeler.

## tickGeneric()

This method is called when the market data changes. Values are updated immediately with no delay.

```
Sub tickGeneric(ByVal id As Integer, ByVal tickType As Integer, ByVal value As Double)
```

Parameter	Description
<b>tickerId</b>	The ticker Id that was specified previously in the call to reqMktData()
<b>tickType</b>	Specifies the type of tick. Pass the field value into TickType.getField(int tickType) to retrieve the field description. For example, a field value of 46 will map to shortable, etc.
<b>value</b>	The value of the specified field.

## tickString()

This method is called when the market data changes. Values are updated immediately with no delay.

```
Sub tickString(ByVal id As Integer, ByVal tickType As Integer, ByVal value As String)
```

Parameter	Description
<b>tickerId</b>	The ticker Id that was specified previously in the call to reqMktData()

Parameter	Description
<b>tickType</b>	Specifies the type of tick. Pass the field value into TickType.getField(int tickType) to retrieve the field description. For example, a field value of 45 will map to lastTimestamp, etc.
<b>value</b>	The value of the specified field.

## tickEFP()

This method is called when the market data changes. Values are updated immediately with no delay.

```
Sub tickEFP(ByVal tickerId As Integer, ByVal field As Integer, ByVal basisPoints
As Double, ByVal formattedBasisPoints As String, ByVal totalDividends As Double,
ByVal holdDays As Integer, ByVal futureExpiry As String, ByVal dividendImpact As
Double, ByVal dividendsToExpiry As Double))
```

Parameter	Description
<b>tickerId</b>	The ticker Id that was specified previously in the call to reqMktData().
<b>field</b>	Specifies the type of price. Pass the field value into TickType.getField(int tickType) to retrieve the field description. For example, a field value of 38 will map to bidEFP, etc.
<b>basisPoints</b>	Annualized basis points, which is representative of the financing rate that can be directly compared to broker rates.
<b>formattedBasis Points</b>	Annualized basis points as a formatted string that depicts them in percentage form.
<b>totalDividends</b>	The total expected dividends.
<b>holdDays</b>	The number of hold days until the expiry of the EFP.
<b>futureExpiry</b>	The expiration date of the single stock future.
<b>dividendImpact</b>	The dividend impact upon the annualized basis points interest rate.
<b>dividendsToExpiry</b>	The dividends expected until the expiration of the single stock future.

## tickSnapshotEnd()

This is called when a snapshot market data subscription has been fully handled and there is nothing more to wait for. This also covers the timeout case.

```
Sub tickSnapshotEnd(ByVal reqId As Integer)
```

Parameter	Description
<b>reqID</b>	Id of the data request.

## orderStatus()

This event is called whenever the status of an order changes. It is also fired after reconnecting if the client has any open orders.

```
Sub orderStatus(ByVal id As Integer, ByVal status As String, ByVal filled
As Integer, ByVal remaining As Integer, ByVal avgFillPrice As Double,
ByVal permId As Integer, ByVal parentId As Integer, ByVal lastFillPrice
As Double, ByVal clientId As Integer, ByVal whyHeld As String)
```

Parameter	Description
<b>id</b>	The order ID that was specified previously in the call to placeOrder()
<b>status</b>	<p>The order status. Possible values include:</p> <ul style="list-style-type: none"> <li>• PendingSubmit - indicates that you have transmitted the order, but have not yet received confirmation that it has been accepted by the order destination.</li> <li>• PendingCancel - indicates that you have sent a request to cancel the order but have not yet received cancel confirmation from the order destination. At this point, your order is not confirmed canceled. You may still receive an execution while your cancellation request is pending.</li> </ul> <p><b>Note:</b> PendingSubmit and PendingCancel order statuses are not sent by the system and should be explicitly set by the API developer when an order is canceled.</p> <ul style="list-style-type: none"> <li>• PreSubmitted - indicates that a simulated order type has been accepted by the system and that this order has yet to be elected. The order is held in the system until the election criteria are met. At that time the order is transmitted to the order destination as specified.</li> <li>• Submitted - indicates that your order has been accepted at the order destination and is working.</li> <li>• Cancelled - indicates that the balance of your order has been confirmed canceled by the system. This could occur unexpectedly when the destination has rejected your order.</li> <li>• Filled - indicates that the order has been completely filled.</li> <li>• Inactive - indicates that the order has been accepted by the system (simulated orders) or an exchange (native orders) but that currently the order is inactive due to system, exchange or other issues.</li> </ul>

Parameter	Description
<b>filled</b>	Specifies the number of shares that have been executed.
<b>remaining</b>	Specifies the number of shares still outstanding.
<b>avgFillPrice</b>	The average price of the shares that have been executed. This parameter is valid only if the <b>filled</b> parameter value is greater than zero. Otherwise, the price parameter will be zero.
<b>permId</b>	The id used to identify orders. Remains the same over sessions.
<b>parentId</b>	The order ID of the parent order, used for bracket and auto trailing stop orders.
<b>lastFilledPrice</b>	The last price of the shares that have been executed. Valid only if the filled parameter value is greater than zero. Otherwise, the price parameter will be zero.
<b>clientId</b>	- The ID of the client who placed the order. Note that application orders have a fixed clientId and orderId of 0 that distinguishes them from API orders.

## errMsg()

This event is called when there is an error with the communication or when TWS wants to send a message to the client.

```
Sub errMsg(ByVal id As Integer, ByVal errorCode As Integer, ByVal
errorMsg As String)
```

Parameter	Description
<b>id</b>	This is the orderId or tickerId of the request that generated the error
<b>errorCode</b>	Error codes are documented in the Error Codes topic.
<b>errorMsg</b>	This is the textual description of the error, also documented in the Error Codes topic.

## connectionClosed()

This event is triggered when TWS closes the sockets connection with the ActiveX control, or when TWS is shut down.

```
Sub connectionClosed()
```

## openOrderEx()

This method is called to feed in open orders.

```
Sub openOrderEx(ByVal orderId As Integer, ByVal contract As
TWSLib.IContract, ByVal order As TWSLib.IOrder, ByVal orderState As
TWSLib.IOrderState)
```

Parameter	Description
<b>orderId</b>	The order Id assigned by TWS. Used to cancel or update the order.
<b>contract</b>	The Contract class attributes describe the contract.
<b>order</b>	The Order class attributes define the details of the order.
<b>orderState</b>	The orderState attributes include margin and commissions fields for both pre and post trade data.

## openOrder1()

This event sends the contract description of the open order.

**Note:** This event has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated events.

```
Sub openOrder1(ByVal id As Integer, ByVal symbol As String, ByVal secType As
String, ByVal expiry As String, ByVal strike As Double, ByVal right As String,
ByVal exchange As String, ByVal currency As String, ByVal localSymbol As String)
```

Parameter	Description
<b>id</b>	The assigned order id. Use this id to cancel or modify the order.
<b>symbol</b>	The underlying symbol.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"> <li>• STK</li> <li>• OPT</li> <li>• FUT</li> <li>• IND</li> <li>• FOP</li> <li>• CASH</li> </ul>
<b>expiry</b>	The expiration date. Use the format YYYYMM.
<b>strike</b>	The strike price.
<b>right</b>	Specifies a Put or Call. Valid values are: P, PUT, C, CALL.
<b>exchange</b>	The order destination, such as Smart.
<b>currency</b>	Specifies the currency. This field is only required when the secType = CASH. Otherwise it is ignored.

Parameter	Description
<b>local symbol</b>	The local exchange symbol for the underlying asset.

## openOrder2()

This event sends the order description of the open order.

**Note:** This event has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated events.

```
Sub openOrder2(ByVal id As Integer, ByVal action As String, ByVal
quantity As Integer, ByVal orderType As String, ByVal lmtPrice As Double,
ByVal auxPrice As Double, ByVal tif As String, ByVal ocaGroup As String,
ByVal account As String, ByVal openClose As String, ByVal origin As
Integer, ByVal orderRef As String, ByVal clientId As Integer)
```

Parameter	Description
<b>id</b>	The order id. This is the same value that was passed in openOrder1.
<b>action</b>	Identifies the side. Valid values are: BUY, SELL, SSHORT
<b>quantity</b>	The order quantity.
<b>orderType</b>	Identifies the order type. Valid values are: <ul style="list-style-type: none"> <li>• MKT</li> <li>• MKTCLS</li> <li>• LMT</li> <li>• LMTCLS</li> <li>• PEGMKT</li> <li>• STP</li> <li>• STPLMT</li> <li>• TRAIL</li> <li>• REL</li> <li>• VWAP</li> </ul>
<b>lmtPrice</b>	This is the limit price, used for Limit, Stop Limit and Relative orders. In all other cases specify zero. For relative orders with no limit price, also specify zero.
<b>auxPrice</b>	This is the STOP price for stop limit orders and the offset amount for relative orders. In all other cases, specify 0.
<b>tif</b>	The time in force. Valid values are DAY, GTC, IOC
<b>ocaGroup</b>	Identifies a one-cancels-all group.
<b>account</b>	The account. For institutional customers only.
<b>openClose</b>	Specifies whether the order is an open or close order. For institutional customers only.

Parameter	Description
<b>origin</b>	The order origin. For institutional customers only. Valid values are: 0 = Customer, 1 = Firm.
<b>orderRef</b>	The order reference. For institutional customers only.
<b>clientId</b>	The ID of the requesting client (or TWS) that placed the order. Note that TWS orders have a fixed clientId and orderId of 0 that distinguishes them from API orders.

## openOrder3()

This event sends the contract description of the open order.

**Note:** This event has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated events.

```
Sub openOrder3(ByVal id As Integer, ByVal symbol As String, ByVal secType As String, ByVal expiry As String, ByVal strike As Double, ByVal right As String, ByVal exchange As String, ByVal currency As String, ByVal localSymbol As String, ByVal action As String, ByVal quantity As Integer, ByVal orderType As String, ByVal lmtPrice As Double, ByVal auxPrice As Double, ByVal tif As String, ByVal ocaGroup As String, ByVal account As String, ByVal openClose As String, ByVal origin As Integer, ByVal orderRef As String, ByVal clientId As Integer, ByVal permId As Integer, ByVal sharesAllocation As String, ByVal faGroup As String, ByVal faMethod As String, ByVal faPercentage As String, ByVal faProfile As String, ByVal goodAfterTime As String, ByVal goodTillDate As String)
```

Parameter	Description
<b>id</b>	The assigned order id. Use this id to cancel or modify the order.
<b>symbol</b>	The underlying symbol.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"> <li>• STK</li> <li>• OPT</li> <li>• FUT</li> <li>• IND</li> <li>• FOP</li> <li>• CASH</li> </ul>
<b>expiry</b>	The expiration date. Use the format YYYYMM.
<b>strike</b>	The strike price.
<b>right</b>	Specifies a Put or Call. Valid values are: P, PUT, C, CALL.
<b>exchange</b>	The order destination, such as Smart.
<b>currency</b>	Specifies the currency. This field is only required when the secType = CASH. Otherwise it is ignored.
<b>local symbol</b>	The local exchange symbol for the underlying asset.

Parameter	Description
<b>action</b>	Identifies the side. Valid values are: BUY, SELL, SSHORT
<b>quantity</b>	The size of the order.
<b>orderType</b>	<p>Identifies the order type. Valid values are:</p> <ul style="list-style-type: none"> <li>• MKT</li> <li>• MKTCLS</li> <li>• LMT</li> <li>• LMTCLS</li> <li>• PEGMKT</li> <li>• STP</li> <li>• STPLMT</li> <li>• TRAIL</li> <li>• REL</li> <li>• VWAP</li> </ul>
<b>lmtPrice</b>	This is the limit price, used for Limit, Stop Limit and Relative orders. In all other cases specify zero. For relative orders with no limit price, also specify zero.
<b>AuxPrice</b>	This is the STOP price for stop limit orders, and the offset amount for relative orders. In all other cases, specify zero.

## openOrder4()

This event sends the contract description of the open order.

**Note:** This event has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated events.

```
Sub openOrder4 (ByVal id As Integer, ByVal symbol As String, ByVal secType As String, ByVal expiry As String, ByVal strike As Double, ByVal right As String, ByVal exchange As String, ByVal currency As String, ByVal localSymbol As String, ByVal action As String, ByVal quantity As Integer, ByVal orderType As String, ByVal lmtPrice As Double, ByVal auxPrice As Double, ByVal tif As String, ByVal ocaGroup As String, ByVal account As String, ByVal openClose As String, ByVal origin As Integer, ByVal orderRef As String, ByVal clientId As Integer, ByVal permId As Integer, ByVal sharesAllocation As String, ByVal faGroup As String, ByVal faMethod As String, ByVal faPercentage As String, ByVal faProfile As String, ByVal goodAfterTime As String, ByVal goodTillDate As String, ByVal ocaType As Integer, ByVal rule80A As String, ByVal settlingFirm As String, ByVal allOrNone As Integer, ByVal minQty As Integer, ByVal percentOffset As Double, ByVal eTradeOnly As Integer, ByVal firmQuoteOnly As Integer, ByVal nbboPriceCap As Double, ByVal auctionStrategy As Integer, ByVal startingPrice As Double, ByVal stockRefPrice As Double, ByVal delta As Double, ByVal stockRangeLower As Double, ByVal stockRangeUpper As Double, ByVal blockOrder As Integer, ByVal sweepToFill As Integer, ByVal ignoreRth As Integer, ByVal hidden As Integer, ByVal discretionaryAmt As Double, ByVal displaySize As Integer, ByVal parentId As Integer, ByVal triggerMethod As Integer, ByVal shortSaleSlot As Integer, ByVal designatedLocation As String, ByVal volatility As Double, ByVal volatilityType As Integer, ByVal deltaNeutralOrderType As String, ByVal deltaNeutralAuxPrice As Double, ByVal continuousUpdate As Integer, ByVal referencePriceType As Integer, ByVal trailStopPrice As Double, ByVal basisPoints As Double, ByVal basisPointsType As Integer, ByVal legsStr As String, ByVal scaleInitLevelSize As Integer, ByVal scaleSubsLevelSize As Integer, ByVal scalePriceIncrement As Double)
```

Parameter	Description
<b>id</b>	The assigned order id. Use this id to cancel or modify the order.
<b>symbol</b>	The underlying symbol.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"><li>• STK</li><li>• OPT</li><li>• FUT</li><li>• IND</li><li>• FOP</li><li>• CASH</li></ul>

Parameter	Description
<b>expiry</b>	The expiration date. Use the format YYYYMM.
<b>strike</b>	The strike price.
<b>right</b>	Specifies a Put or Call. Valid values are: P, PUT, C, CALL.
<b>exchange</b>	The order destination, such as Smart.
<b>currency</b>	Specifies the currency. This field is only required when the secType = CASH. Otherwise it is ignored.
<b>local symbol</b>	The local exchange symbol for the underlying asset.
<b>action</b>	Identifies the side. Valid values are: BUY, SELL, SSHORT
<b>quantity</b>	The size of the order.
<b>orderType</b>	<p>Identifies the order type. Valid values are:</p> <ul style="list-style-type: none"> <li>• MKT</li> <li>• MKTCLS</li> <li>• LMT</li> <li>• LMTCLS</li> <li>• PEGMKT</li> <li>• STP</li> <li>• STPLMT</li> <li>• TRAIL</li> <li>• REL</li> <li>• VWAP</li> </ul>
<b>lmtPrice</b>	This is the limit price, used for Limit, Stop Limit and Relative orders. In all other cases specify zero. For relative orders with no limit price, also specify zero.
<b>AuxPrice</b>	This is the STOP price for stop limit orders, and the offset amount for relative orders. In all other cases, specify zero.
<b>tif</b>	The time in force.
<b>ocaGroup</b>	Identifies the order as part of a one-cancels-all group.
<b>ocaType</b>	Cancel on Fill with Block = 1 Reduce on Fill with Block = 2 Reduce on Fill without Block = 3
<b>account</b>	The account to which the order is allocated. For institutional customers only.
<b>openClose</b>	Specifies whether the order is to open or close a position. For institutional customers only.
<b>origin</b>	The order origin. Valid values are: 0 = customer; 1 = firm. For institutional customers only.
<b>orderRef</b>	The order reference. For institutional customers only.
<b>clientId</b>	The ID of the client who placed the order. NOTE: TWS orders have a fixed client ID of "0."

Parameter	Description
<b>permId</b>	The TWS ID used to identify orders. This value remains the same over TWS sessions.
<b>sharesAllocation</b>	Deprecated. Upgrade to new FA functionality must be done.
<b>faGroup</b>	The advisor group to which the trade will be allocated. Use an empty string if not applicable. Valid values include:
<b>faMethod</b>	The advisor method which will be used to allocate the trade. Use an empty string if not applicable.
<b>faPercentage</b>	The percentage of the order for trade allocation.
<b>faProfile</b>	The advisor profile which will be used to allocate the trade. Use an empty string if not applicable. Valid values include:
<b>goodAfterTime</b>	The trade's "Good After Time," format "YYYYMMDD hh:mm:ss (optional time zone)" Use an empty String if not applicable.
<b>goodTillDate</b>	You must enter GTD as the time in force to use this string. The trade's "Good Till Date," format "YYYYMMDD hh:mm:ss (optional time zone)" Use an empty String if not applicable.
<b>rthOnly</b>	1 = yes, 2 = no
<b>rule80A</b>	Values are: <ul style="list-style-type: none"> <li>• Individual = 'I'</li> <li>• Agency = 'A',</li> <li>• AgentOtherMember = 'W'</li> <li>• IndividualPTIA = 'J'</li> <li>• AgencyPTIA = 'U'</li> <li>• AgentOtherMemberPTIA = 'M'</li> <li>• IndividualPT = 'K'</li> <li>• AgencyPT = 'Y'</li> <li>• AgentOtherMemberPT = 'N'</li> </ul>
<b>settlingFirm</b>	Institution only.
<b>allOrNone</b>	Identifies the order as having the all or none attribute, which means the entire quantity must execute or the order will be cancelled. Values are: 0 = no, 1 = yes
<b>minQty</b>	Identifies a minimum quantity order type.
<b>percentOffset</b>	The percent offset amount for relative orders.
<b>eTradeOnly</b>	Trade with electronic quotes. Values are: 0 = no, 1 = yes
<b>firmQuoteOnly</b>	Trade with firm quotes. Values are: 0 = no, 1 = yes
<b>nbboPriceCap</b>	Maximum smart order distance from the NBBO.

Parameter	Description
<b>auctionStrategy</b>	Values are: <ul style="list-style-type: none"> <li>• match = 1</li> <li>• improvement = 2</li> <li>• transparent = 3</li> </ul> For orders on BOX only.
<b>startingPrice</b>	The auction starting price. For orders on BOX only.
<b>stockRefPrice</b>	The stock reference price. The reference price is used for VOL orders to compute the limit price sent to an exchange (whether or not Continuous Update is selected), and for price range monitoring.
<b>delta</b>	The stock delta. For orders on BOX only.
<b>stockRangeLower</b>	The lower value for the acceptable underlying stock price range. For price improvement option orders on BOX and VOL orders with dynamic management.
<b>stockRangeUpper</b>	The upper value for the acceptable underlying stock price range. For price improvement option orders on BOX and VOL orders with dynamic management.
<b>blockOrder</b>	If set to true, specifies that the order is an ISE Block order.
<b>sweepToFill</b>	If set to true, specifies that the order is a Sweep-to-Fill order.
<b>ignoreRth</b>	If set to true, allows triggering of orders outside of regular trading hours.
<b>hidden</b>	If set to true, the order will not be visible when viewing the market depth. This option only applies to orders routed to the ISLAND exchange.
<b>discretionaryAmt</b>	The amount off the limit price allowed for discretionary orders.
<b>displaySize</b>	The publicly disclosed order size, used when placing Iceberg orders.
<b>parentId</b>	The order ID of the parent order, used for bracket and auto trailing stop orders.
<b>triggerMethod</b>	Specifies how Simulated Stop, Stop-Limit and Trailing Stop orders are triggered. Valid values are: <ul style="list-style-type: none"> <li>• <b>0</b> - The default value. The "double bid/ask" method will be used for orders for OTC stocks and US options. All other orders will used the "last" method.</li> <li>• <b>1</b> - use "double bid/ask" method, where stop orders are triggered based on two consecutive bid or ask prices.</li> <li>• <b>2</b> - "last" method, where stop orders are triggered based on the last price.</li> </ul>
<b>shortSaleSlot</b>	Valid values are 1 or 2.
<b>designatedLocation</b>	Use only when shortSaleSlot value = 2.

Parameter	Description
<b>volatility</b>	The option price in volatility, as calculated by TWS' Option Analytics. This value is expressed as a percent and is used to calculate the limit price sent to the exchange.
<b>volatilityType</b>	Values are: <ul style="list-style-type: none"> <li>• 1 = Daily volatility</li> <li>• 2 = Annual volatility</li> </ul>
<b>deltaNeutralOrderType</b>	VOL orders only. Enter an order type to instruct TWS to submit a delta neutral trade on full or partial execution of the VOL order. For no hedge delta order to be sent, specify NONE.
<b>deltaNeutralAuxPrice</b>	VOL orders only. Use this field to enter a value if the value in the <i>deltaNeutralOrderType</i> field is an order type that requires an Aux price, such as a REL order.
<b>continuousUpdate</b>	VOL orders only. Specifies whether TWS will automatically update the limit price of the order as the underlying price moves.
<b>referencePriceType</b>	VOL orders only. Specifies how you want TWS to calculate the limit price for options, and for stock range price monitoring. Valid values include: <ul style="list-style-type: none"> <li>• 1 = Average of NBBO</li> <li>• 2 = NBB or the NBO depending on the action and right.</li> </ul>

## updateAccountValue()

This event updates a single account value.

```
Sub updateAccountValue(ByVal key As String, ByVal value As String, ByVal currency As String, ByVal accountName As String)
```

Parameter	Description
<b>key</b>	<p>A string that indicates one type of account value. Below are some of the keys sent by TWS.</p> <ul style="list-style-type: none"> <li>• Account Type</li> <li>• Account Code</li> <li>• Available Funds</li> <li>• Buying Power</li> <li>• CashBalance - Account cash balance</li> <li>• Currency - Currency string</li> <li>• DayTradesRemaining - Number of day trades left</li> <li>• EquityWithLoanValue - Equity with Loan Value</li> <li>• Excess Liquidity</li> <li>• Full Available Funds</li> <li>• Full Excess Liquidity</li> <li>• Full Init Margin Req</li> <li>• Full Maint Margin Req</li> <li>• Future Option Value</li> <li>• Futures PNL</li> <li>• Gross Position Value</li> <li>• InitMarginReq - Current initial margin requirement</li> <li>• Leverage</li> <li>• Look Ahead Available Funds</li> <li>• Look Ahead Next Change</li> <li>• Look Ahead Excess Liquidity</li> <li>• Look Ahead Margin Req</li> <li>• Look Ahead Maint Margin Req</li> <li>• LongOptionValue - Long option value</li> </ul>

Parameter	Description
<b>key</b>	Values, continued: <ul style="list-style-type: none"> <li>• MaintMarginReq - Current maintenance margin</li> <li>• NetLiquidation - Net liquidation value</li> <li>• OptionMarketValue - Option market value</li> <li>• Realized PNL</li> <li>• Settled Cash</li> <li>• ShortOptionValue - Short option value</li> <li>• StockMarketValue - Stock market value</li> <li>• Total Cash Balance</li> <li>• Total Cash Value</li> <li>• UnalteredInitMarginReq - Overnight initial margin requirement</li> <li>• UnalteredMaintMarginReq - Overnight maintenance margin requirement</li> <li>• Unrealized PNL</li> </ul>
<b>value</b>	The value associated with the key.
<b>currency</b>	Defines the currency of the value, if the value is a monetary amount.
<b>account</b>	states the account the message applies to. Useful for Financial Advisor sub-account messages.

## updatePortfolioEx()

This callback is made in response to the [reqAccountUpdates\(\)](#) method.

```
Sub updatePortfolioEx(ByVal contract As TWSLib.IContract, ByVal position As Integer, ByVal marketPrice As Double, ByVal marketValue As Double, ByVal averageCost As Double, ByVal unrealizedPNL As Double, ByVal realizedPNL As Double, ByVal accountName As String)
```

Parameter	Description
<b>contract</b>	This object contains a description of the contract which is being traded. The exchange field in a contract is not set for portfolio update.
<b>position</b>	This integer indicates the position on the contract. If the position is 0, it means the position has just cleared.
<b>marketPrice</b>	The unit price of the instrument.
<b>marketValue</b>	The total market value of the instrument.
<b>averageCost</b>	The average cost per share is calculated by dividing your cost (execution price + commission) by the quantity of your position.

Parameter	Description
<b>unrealizedPNL</b>	The difference between the current market value of your open positions and the average cost, or Value - Average Cost.
<b>realizedPNL</b>	Shows your profit on closed positions, which is the difference between your entry execution cost (execution price + commissions to open the position) and exit execution cost ((execution price + commissions to close the position)
<b>accountName</b>	The name of the account the message applies to. Useful for Financial Advisor sub-account messages.

## updatePortfolio()

This event updates a single holding in your portfolio.

**Note:** This event has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated events.

```
Sub updatePortfolio(ByVal symbol As String, ByVal secType As String,
ByVal expiry As String, ByVal strike As Double, ByVal right As String,
ByVal currency As String, ByVal localSymbol As String, ByVal position As
Integer, ByVal marketPrice As Double, ByVal marketValue As Double, ByVal
averageCost As Double, ByVal unrealizedPNL As Double, ByVal realizedPNL
As Double, ByVal accountName As String)
```

Parameter	Description
<b>symbol</b>	The symbol of the underlying asset.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"> <li>• STK</li> <li>• OPT</li> <li>• FUT</li> <li>• IND</li> <li>• FOP</li> <li>• CASH</li> <li>• BAG</li> </ul>
<b>expiry</b>	The expiration date. Use the format YYYYMM.
<b>strike</b>	The strike price.
<b>right</b>	Specifies a put or call. Valid values are: P, PUT, C, CALL
<b>currency</b>	Specifies the currency. This field is only required when the secType = CASH. Otherwise it is ignored.
<b>localSymbol</b>	The local exchange symbol of the asset.
<b>position</b>	Indicates the position on the contract. If the value is 0, it means the position has just cleared.
<b>marketPrice</b>	The unit price of the instrument.
<b>marketValue</b>	The total market value of the instrument.

Parameter	Description
<b>averageCost</b>	The average cost per share is calculated by dividing your cost (execution price + commission) by the quantity of your position.
<b>unrealizedPNL</b>	The difference between the current market value of your open positions and the average cost, or (Value - Average Cost).
<b>realizedPNL</b>	Shows your profit on closed positions, which is the difference between your entry execution cost (execution price + commissions to open the position) and exit execution costs (execution price + commissions to close the position).
<b>account</b>	States the account the message applies to. Useful for Financial Advisor sub-account messages.

### updateAccountTime()

This event sends the time at which the account values and portfolio market prices were calculated.

```
Sub updateAccountTime(ByVal timeStamp As String)
```

Parameter	Description
<b>timeStamp</b>	This indicates the last update time of the account information.

### nextValidId()

This event is called after a successful connection to TWS.

```
Sub nextValidId(ByVal id As Integer)
```

Parameter	Description
<b>id</b>	The next available order ID received from TWS upon connection. Increment all successive orders by one based on this ID.

### permId()

This event is always received after an order Status event. It gives the permId for the specified order id. The permId will remain the same from session to session.

```
Sub permId(ByVal id As Integer, ByVal permId As Integer)
```

Parameter	Description
<b>id</b>	The next available order ID received from TWS upon connection. Increment all successive orders by one based on this ID.
<b>permId</b>	This id will remain the same from session to session

## contractDetailsEx()

This event is called only in response to the [reqContractDetailsEx\(\)](#) method having been called.

```
Sub contractDetailsEx(ByVal reqId As Integer, ByVal contractDetails As
TWSLib.IContractDetails)
```

Parameter	Description
<b>reqId</b>	The ID of the data request. Ensures that responses are matched to requests if several requests are in process.
<b>contractDetails</b>	This object contains a full description of the contract being looked up.

## contractDetails()

This event is fired when the `reqContractDetails()` or `reqContractDetails2()` methods are invoked.

**Note:** This event has been deprecated as of API release 9,4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated events.

```
Sub contractDetails(ByVal symbol As String, ByVal secType As String,
ByVal expiry As String, ByVal strike As Double, ByVal right As String,
ByVal exchange As String, ByVal currency As String, ByVal localSymbol As
String, ByVal marketName As String, ByVal tradingClass As String, ByVal
conId As Integer, ByVal minTick As Double, ByVal priceMagnifier As
Integer, ByVal multiplier As String, ByVal orderTypes As String, ByVal
validExchanges As String)
```

Parameter	Description
<b>symbol</b>	The underlying symbol.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"> <li>• STK</li> <li>• OPT</li> <li>• FUT</li> <li>• IND</li> <li>• FOP</li> <li>• CASH</li> </ul>
<b>expiry</b>	The expiration date. Use the format YYYYMM.
<b>strike</b>	The strike price.
<b>right</b>	Specifies a put or call. Valid values are: P, PUT, C, CALL
<b>exchange</b>	The order destination, such as Smart.
<b>currency</b>	Specifies the currency. This field is only required when the <code>secType = CASH</code> . Otherwise it is ignored.
<b>localSymbol</b>	The local exchange symbol of the asset.

Parameter	Description
<b>marketName</b>	The market name for this contract.
<b>tradingClass</b>	The trading class name for this contract.
<b>conId</b>	The unique contract identifier.
<b>minTick</b>	The minimum price tick.
<b>priceMagnifier</b>	Allows execution and strike prices to be reported consistently with market data, historical data and the order price, i.e. Z on LIFFE is reported in index points and not GBP.
<b>multiplier</b>	The order size multiplier.
<b>orderTypes</b>	The list of valid order types for this contract.
<b>validExchanges</b>	The list of exchanges on which this contract is traded.

### contractDetailsEnd()

This method is called once all contract details for a given request are received. This helps to define the end of an option chain.

```
Sub contractDetailsEnd(ByVal reqId As Integer)
```

Parameter	Description
<b>reqID</b>	Id of the data request.

### execDetailsEx()

This method is called when the [reqExecutionsEx\(\)](#) method is invoked, or when an order is filled.

```
Sub execDetailsEx(ByVal reqId As Integer, ByVal contract As
TWSLib.IContract, ByVal execution As TWSLib.IExecution)
```

Parameter	Description
<b>orderId</b>	The order Id that was specified previously in the call to <a href="#">placeOrderEx()</a> .
<b>contract</b>	This object contains a full description of the contract that was executed.
<b>execution</b>	This structure contains addition order execution details.

## execDetails()

This event is fired when the reqExecutions() methods is invoked, or when an API order is filled or partially filled.

**Note:** This event has been deprecated as of API release 9,4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated events.

```
Sub execDetails(ByVal id As Integer, ByVal symbol As String, ByVal secType As String, ByVal expiry As String, ByVal strike As Double, ByVal right As String, ByVal cExchange As String, ByVal currency As String, ByVal localSymbol As String, ByVal execId As String, ByVal time As String, ByVal acctNumber As String, ByVal eExchange As String, ByVal side As String, ByVal shares As Integer, ByVal price As Double, ByVal permId As Integer, ByVal clientId As Integer, ByVal isLiquidation As Integer)
```

Parameter	Description
<b>id</b>	The order id. This is the same value that was passed in openOrder1. Note: TWS orders have a fixed orderId of 0, which distinguishes them from API client orders.
<b>symbol</b>	The underlying symbol.
<b>secType</b>	The security type. Valid values are: <ul style="list-style-type: none"> <li>• STK</li> <li>• OPT</li> <li>• FUT</li> <li>• IND</li> <li>• FOP</li> <li>• CASH</li> </ul>
<b>expiry</b>	The expiration date. Use the format YYYYMM.
<b>strike</b>	The strike price.
<b>right</b>	Specifies a put or call. Valid values are: P, PUT, C, CALL
<b>cExchange</b>	The order destination, such as Smart.
<b>currency</b>	Specifies the currency. This field is only required when the secType = CASH. Otherwise it is ignored.
<b>localSymbol</b>	The local exchange symbol for the underlying asset.
<b>execId</b>	The order execution id.
<b>time</b>	The time of order execution.
<b>acctNumber</b>	The customer account number.
<b>eExchange</b>	--
<b>side</b>	Specifies if the transaction was a purchase or a sale. Valid values are: <ul style="list-style-type: none"> <li>• BOT</li> <li>• SLD</li> </ul>

Parameter	Description
<b>shares</b>	The number of shares filled.
<b>price</b>	The order execution price.
<b>permId</b>	The TWS id used to identify orders, remains the same over TWS sessions.
<b>permId</b>	The ID of the client that placed the order. Note that TWS orders have a fixed ID of 0.
<b>isliquidation</b>	Specifies whether an execution is the result of a liquidation. Possible values are: <ul style="list-style-type: none"> <li>• 0 = execution is not the result of liquidation</li> <li>• 1 = execution is the result of a liquidation</li> </ul>

## execDetailsEnd()

This method is called once all executions have been sent to a client in response to reqExecutionsEx()

```
Sub execDetailsEnd(ByVal reqId As Integer)
```

Parameter	Description
<b>reqID</b>	Id of the data request.

## updateMktDepth()

This function is called when the market depth changes.

```
Sub updateMktDepth(ByVal id As Integer, ByVal position As Integer, ByVal operation As Integer, ByVal side As Integer, ByVal price As Double, ByVal size As Integer)
```

Parameter	Description
<b>id</b>	The ticker ID that was specified previously in the call to reqMktDepth()
<b>position</b>	Specifies the row ID of this market depth entry.
<b>operation</b>	Identifies how this order should be applied to the market depth. Valid values are: <ul style="list-style-type: none"> <li>• 0 = insert (insert this new order into the row identified by 'position')</li> <li>• 1 = update (update the existing order in the row identified by 'position')</li> <li>• 2 = delete (delete the existing order at the row identified by 'position')</li> </ul>
<b>side</b>	The side of the book to which this order belongs. Valid values are: <ul style="list-style-type: none"> <li>• 0 = ask</li> <li>• 1 = bid</li> </ul>

Parameter	Description
<b>price</b>	The order price.
<b>size</b>	The order size.

## updateMktDepthL2()

This function is called when the Level II market depth changes.

```
Sub updateMktDepthL2(ByVal id As Integer, ByVal position As Integer,
ByVal marketMaker As String, ByVal operation As Integer, ByVal side As
Integer, ByVal price As Double, ByVal size As Integer)
```

Parameter	Description
<b>id</b>	The ticker ID that was specified previously in the call to reqMktDepth()
<b>position</b>	Specifies the row id of this market depth entry.
<b>marketMaker</b>	Specifies the exchange hosting this order.
<b>operation</b>	Identifies the how this order should be applied to the market depth. Valid values are: <ul style="list-style-type: none"> <li>• 0 = insert (insert this new order into the row identified by 'position')</li> <li>• 1 = update (update the existing order in the row identified by 'position')</li> <li>• 2 = delete (delete the existing order at the row identified by 'position')</li> </ul>
<b>side</b>	Identifies the side of the book that this order belongs to. Valid values are: <ul style="list-style-type: none"> <li>• 0 = ask</li> <li>• 1 = bid</li> </ul>
<b>price</b>	The order price.
<b>size</b>	The order size.

## updateNewsBulletin()

This event is triggered for each new bulletin if the client has subscribed (i.e. by calling the reqNewsBulletins() method).

```
Sub updateNewsBulletin(ByVal msgId As Short, ByVal msgType As Short,
ByVal message As String, ByVal origExchange As String)
```

Parameter	Description
<b>msgId</b>	The bulletin ID, increments for each new bulletin.

Parameter	Description
<b>msgType</b>	Specifies the type of bulletin. Valid values include: <ul style="list-style-type: none"> <li>• 1 = Regular IB news bulletin</li> <li>• 2 = Exchange no longer available for trading.</li> <li>• 3 = Exchange is available for trading.</li> </ul>
<b>message</b>	The bulletins message text.
<b>origExchange</b>	The exchange from which this message originated.

## managedAccounts()

This event id is fired when a successful connection is made to a Financial Advisor account. It is also fired when the reqManagedAccts() method is invoked.

```
Sub managedAccounts(ByVal accountsList As String)
```

Parameter	Description
accountsList	The comma delimited list of FA-managed accounts.

## receiveFA()

This event receives previously requested FA configuration information from TWS.

```
Sub receiveFA(ByVal faDataType As Integer, ByVal cxml As String)
```

Parameter	Description
<b>faDataType</b>	Specifies the type of Financial Advisor configuration data being received from TWS. Valid values include: <ul style="list-style-type: none"> <li>• 1 = GROUPS</li> <li>• 2 = PROFILE</li> <li>• 3 =ACCOUNT ALIASES</li> </ul>
<b>cxml</b>	The XML string containing the previously requested FA configuration information.

## historicalData()

This event receives requested historical data from TWS.

```
Sub historicalData(ByVal reqId As Integer, ByVal date As String, ByVal open As Double, ByVal high As Double, ByVal low As Double, ByVal close As Double, ByVal volume As Integer, ByVal barCount As Integer, ByVal WAP As Double, ByVal hasGaps As Integer)
```

Parameter	Description
<b>reqId</b>	The ticker ID of the request to which this bar is responding.
<b>date</b>	The date-time stamp of the start of the bar. The format is determined by the reqHistoricalData() formatDate parameter.

Parameter	Description
<b>open</b>	The bar opening price.
<b>high</b>	The high price during the time covered by the bar.
<b>low</b>	The low price during the time covered by the bar.
<b>close</b>	The bar closing price.
<b>volume</b>	The volume during the time covered by the bar.
<b>WAP</b>	The weighted average price during the time covered by the bar.
<b>hasGaps</b>	Identifies whether or not there are gaps in the data.

## bondContractDetails()

```
Sub bondContractDetails(ByVal symbol As String, ByVal secType As String,
ByVal cusip As String, ByVal coupon As Double, ByVal maturity As String,
ByVal issueDate As String, ByVal ratings As String, ByVal bondType As
String, ByVal couponType As String, ByVal convertible As Integer, ByVal
callable As Integer, ByVal putable As Integer, ByVal descAppend As
String, ByVal exchange As String, ByVal currency As String, ByVal
marketName As String, ByVal tradingClass As String, ByVal conId As
Integer, ByVal minTick As Double, ByVal orderTypes As String, ByVal
validExchanges As String, ByVal nextOptionDate As String, ByVal
nextOptionType As String, ByVal nextOptionPartial As Integer, ByVal
notes As String)
```

Parameter	Description
<b>symbol</b>	The bond symbol.
<b>secType</b>	BOND
<b>cusip</b>	The nine-character bond CUSIP, or 12 character SEDOL.
<b>coupon</b>	The interest rate used to calculate the amount you will receive in interest payments over the course of the year.
<b>maturity</b>	The date on which the issuer must repay the face value of the bond.
<b>issueDate</b>	The date on which the bond was issued.
<b>ratings</b>	Identifies the credit rating of the issuer. A higher credit rating generally indicates a less risky investment. Bond ratings are from Moody's and S&P respectively.
<b>bondType</b>	The type of the bond, such as "Corp" for corporate.
<b>couponType</b>	The type of the coupon, such as "FIXED."
<b>convertible</b>	Values are: True or False. If true, the bond can be converted to stock under certain conditions.
<b>callable</b>	Values are: True or False. If true, the bond can be called by the issuer under certain conditions.
<b>putable</b>	Values are: True or False. If true, the bond can be sold back to the issuer under certain conditions.
<b>descAppend</b>	Description string containing further descriptive information about the bond.
<b>exchange</b>	The exchange on which the BOND trades.
<b>currency</b>	The currency in which the bond trades.
<b>marketName</b>	The market name for this contract.
<b>tradingClass</b>	The trading class name for this contract.
<b>conId</b>	The IB contract ID of the bond.
<b>minTick</b>	The minimum price increment of the bond.
<b>orderTypes</b>	The order types that apply to this bond.
<b>validExchanges</b>	A comma-delimited string of exchanges on which this bond trades.

Parameter	Description
<b>nextOptionDate</b>	Next option date. Applies only to bonds with embedded options.
<b>nextOptionType</b>	Next option type. Applies only to bonds with embedded options.
<b>nextOptionPartial</b>	Next option partial. Applies only to bonds with embedded options (is the next option full or partial?).
<b>notes</b>	Bond notes, if populated for the bond in IB's database.

## scannerParameters()

```
Sub scannerParameters(ByVal xml As String)
```

Parameter	Description
<b>xml</b>	An XML document that describes the valid parameters that a scanner parameter can have.

## scannerDataEx()

This event receives the requested market scanner data results.

```
Sub scannerDataEx(ByVal reqId As Integer, ByVal rank As Integer, ByVal
contractDetails As TWSLib.IContractDetails, ByVal distance As String,
ByVal benchmark As String, ByVal projection As String, ByVal legsStr As
String)
```

Parameter	Description
<b>reqId</b>	The ID of the request to which this row is responding.
<b>rank</b>	The ranking within the response of this bar.
<b>contractDetails</b>	This object contains a full description of the contract.
<b>distance</b>	Varies based on query.
<b>benchmark</b>	Varies based on query.
<b>projection</b>	Varies based on query.
<b>legsStr</b>	Describes combo legs when scan is returning EFP.

## scannerData()

**Note:** This event has been deprecated as of API release 9.4 (December 2007) and will soon be removed. Please update your software to use the "Ex" counterparts to these deprecated events.

```
Sub scannerData(ByVal reqId As Integer, ByVal rank As Integer, ByVal
symbol As String, ByVal secType As String, ByVal expiry As String, ByVal
strike As Double, ByVal right As String, ByVal exchange As String, ByVal
currency As String, ByVal localSymbol As String, ByVal marketName As
```

```
String, ByVal tradingClass As String, ByVal distance As String, ByVal
benchmark As String, ByVal projection As String, ByVal legsStr As String)
```

Parameter	Description
<b>reqId</b>	The ticker ID of the request to which this row is responding.
<b>rank</b>	The ranking within the response of this particular bar.
<b>symbol</b>	The symbol to which this bar pertains.
<b>secType</b>	The security type of that symbol.
<b>expiry</b>	If a derivative, the expiration date.
<b>strike</b>	If an option, it's strike price.
<b>right</b>	If an option, call or put.
<b>exchange</b>	The exchange on which the symbol trades.
<b>currency</b>	The currency the symbol trades in on the exchange.
<b>localSymbol</b>	Identifier assigned to derivatives that uniquely identifies them on an exchange.
<b>marketName</b>	The market name of the contract.
<b>tradingClass</b>	The trading class name of the contract.
<b>distance</b>	Varies based on the query.
<b>benchmark</b>	Varies based on the query.
<b>projection</b>	Varies based on the query.

### scannerDataEnd()

This function is called when the snapshot is received and marks the end of one scan.

```
Sub scannerDataEnd(ByVal reqId As Integer)
```

Parameter	Description
<b>reqId</b>	The ID of the market data snapshot request being closed by this parameter.

### realtimeBar()

This method receives the real-time bars data results.

```
Sub realtimeBar(ByVal tickerId As Integer, ByVal time As Integer, ByVal
open As Double, ByVal high As Double, ByVal low As Double, ByVal close As
Double, ByVal volume As Integer, ByVal WAP As Double, ByVal Count As
Integer)
```

Parameter	Description
<b>reqId</b>	The ticker Id of the request to which this bar is responding.
<b>time</b>	The date-time stamp of the start of the bar. The format is determined by the reqHistoricalData() formatDate parameter.
<b>open</b>	The bar opening price.

Parameter	Description
<b>high</b>	The high price during the time covered by the bar.
<b>low</b>	The low price during the time covered by the bar.
<b>close</b>	The bar closing price.
<b>volume</b>	The volume during the time covered by the bar.
<b>wap</b>	The weighted average price during the time covered by the bar.
<b>count</b>	When TRADES historical data is returned, represents the number of trades that occurred during the time period the bar covers.

### **currentTime()**

This method receives the current system time on the server side.

```
Sub currentTime(ByVal time As Integer)
```

Parameter	Description
<b>time</b>	The current system time on the server side.,

### **fundamentalData()**

This method is called to receive Reuters global fundamental market data. There must be a subscription to Reuters Fundamental set up in Account Management before you can receive this data.

```
Sub fundamentalData(ByVal reqId As Integer, ByVal data As String)
```

Parameter	Description
<b>reqId</b>	The ID of the data request.
<b>data</b>	One of three XML reports: <ul style="list-style-type: none"><li>• Estimates (estimates)</li><li>• Financial statements (finstat)</li><li>• Summary (snapshot)</li></ul>

## ActiveX COM Objects

The tables below define properties for the following objects:

- [\*\*IExecution\*\*](#)
- [\*\*IExecutionFilter\*\*](#)
- [\*\*IContract\*\*](#)
- [\*\*IContractDetails\*\*](#)
- [\*\*IComboLeg\*\*](#)
- [\*\*IComboLegList\*\*](#)
- [\*\*IOrder\*\*](#)
- [\*\*IOrderState\*\*](#)
- [\*\*IScannerSubscription\*\*](#)
- [\*\*ITagValueList\*\*](#)
- [\*\*ITagValue\*\*](#)
- [\*\*IUnderComp\*\*](#)

You must use the factory “create” methods to create the COM objects in this section. Once a COM object has been created by a factory method, the COM object is tied to a corresponding TWS COM object (an instance of the COM object). Do not try to pass a COM object to another instance of a TWS COM object.

## IExecution

Property	Description
orderId() As Integer	The order id. <b>Note:</b> TWS orders have a fixed order id of "0."
clientId() As Integer	The id of the client that placed the order. <b>Note:</b> TWS orders have a fixed client id of "0."
execId() As String	Unique order execution id.
time() As String	The order execution time.
acctNumber() As String	The customer account number.
exchange() As String	Exchange that executed the order.
side() As String	Specifies if the transaction was a sale or a purchase. Valid values are: <ul style="list-style-type: none"> <li>• BOT</li> <li>• SLD</li> </ul>
shares() As Integer	The number of shares filled.
price() As Double	The order execution price.
permId() As Integer	The TWS id used to identify orders, remains the same over TWS sessions.
liquidation() As Integer	Identifies the position as one to be liquidated last should the need arise.
cumQty() As Integer	Cumulative quantity. Used in regular trades, combo trades and legs of the combo.
avgPrice() As Double	Average price. Used in regular trades, combo trades and legs of the combo.

## IExecutionFilter

Property	Description
clientId() As Integer	Filter the results of the reqExecutionsEx() method based on the clientId.
acctCode() As String	Filter the results of the reqExecutionsEx() method based on an account code. Note: this is only relevant for Financial Advisor (FA) accounts.
time() As String	Filter the results of the reqExecutionsEx() method based on execution reports received after the specified time. The format for timeFilter is "yyyymmdd-hh:mm:ss"
symbol() As String	Filter the results of the reqExecutionsEx() method based on the order symbol.
secType() String	Filter the results of the reqExecutionsEx() method based on the order security type. <b>Note:</b> Refer to the Contract object for the list of valid security types.

Property	Description
exchange() As String	Filter the results of the reqExecutionsEx() method based on the order exchange.
side() As String	Filter the results of the reqExecutionsEx() method based on the order action. <b>Note:</b> Refer to the Order object for the list of valid order actions.

## IContract

Property	Description
symbol() As String	This is the symbol of the underlying asset.
secType() As String	This is the security type. Valid values are: <ul style="list-style-type: none"> <li>• STK</li> <li>• OPT</li> <li>• FUT</li> <li>• IND</li> <li>• FOP</li> <li>• CASH</li> <li>• BAG</li> </ul>
expiry() As String	The expiration date. Use the format YYYYMM.
strike() As Double	The strike price.
right() As String	Specifies a Put or Call. Valid values are: P, PUT, C, CALL.
multiplier() As String	Allows you to specify a future or option contract multiplier. This is only necessary when multiple possibilities exist.
exchange() As String	The order destination, such as Smart.
currency() As String	Specifies the currency. Ambiguities may require that this field be specified, for example, when SMART is the exchange and IBM is being requested (IBM can trade in GBP or USD). Given the existence of this kind of ambiguity, it is a good idea to always specify the currency.
localSymbol() As String	This is the local exchange symbol of the underlying asset.
primaryExch() As String	Identifies the listing exchange for the contract (do not list SMART).
includeExpired() As Integer	If set to true, contract details requests and historical data queries can be performed pertaining to expired contracts.  Note: Historical data queries on expired contracts are limited to the last year of the contracts life, and are initially only supported for expired futures contracts,
comboLegsDescrip() As String	Description for combo legs
comboLegs() As Object	Dynamic memory structure used to store the leg definitions for this contract.
conId() As Integer	The unique contract identifier.

Property	Description
secIdType As String	Security identifier, when querying contract details or when placing orders. Supported identifiers are: <ul style="list-style-type: none"><li>• ISIN (Example: Apple: US0378331005)</li><li>• CUSIP (Example: Apple: 037833100)</li><li>• SEDOL (Consists of 6-AN + check digit. Example: BAE: 0263494)</li><li>• RIC (Consists of exchange-independent RIC Root and a suffix identifying the exchange. Example: AAPL.O for Apple on NASDAQ.)</li></ul>
secId as String	Unique identifier for the secIdType.

## IContractDetails

Property	Description
summary() As Object	A contract summary.
marketName() String	The market name for this contract.
tradingClass() As String	The trading class name for this contract.
minTick() As Double	The minimum price tick.
priceMagnifier() Integer	Allows execution and strike prices to be reported consistently with market data, historical data and the order price, i.e. Z on LIFFE is reported in index points and not GBP.
orderTypes() As String	The list of valid order types for this contract.
validExchanges() As String	The list of exchanges this contract is traded on.
underConId() As String	The underlying contract ID.
longName() As String	The descriptive name of the asset.
cusip() As String	For Bonds. The nine-character bond CUSIP or the 12-character SEDOL.
ratings() As String	For Bonds. Identifies the credit rating of the issuer. A higher credit rating generally indicates a less risky investment. Bond ratings are from Moody's and S&P respectively.
descAppend() As String	For Bonds. A description string containing further descriptive information about the bond.
bondType() As String	For Bonds. The type of bond, such as "CORP."
couponType() As String	For Bonds. The type of bond coupon, such as "FIXED."
callable() As Integer	For Bonds. Values are True or False. If true, the bond can be called by the issuer under certain conditions.
putable() As Integer	For Bonds. Values are True or False. If true, the bond can be sold back to the issuer under certain conditions.
coupon() As Double	For Bonds. The interest rate used to calculate the amount you will receive in interest payments over the course of the year.
convertible() As Integer	For Bonds. Values are True or False. If true, the bond can be converted to stock under certain conditions.
maturity() As String	For Bonds. The date on which the issuer must repay the face value of the bond.
issueDate() As String	For Bonds. The date the bond was issued.
nextOptionDate() As String	For Bonds, applies to bonds with embedded options.
nextOptionType() As String	For Bonds, applies to bonds with embedded options.
nextOptionPartial() As Integer	For Bonds, applies to bonds with embedded options.
notes() As String	For Bonds, if populated for the bond in IB's database
contractMonth() As String	The contract month. Typically the contract month of the underlying for a futures contract.

Property	Description
industry() As String	The industry classification of the underlying/product. For example, Financial.
category() As String	The industry category of the underlying. For example, InvestmentSvc.
subcategory() As String	The industry subcategory of the underlying. For example, Brokerage.
timeZoneId() As String	The ID of the time zone for the trading hours of the product. For example, EST.
tradingHours() As String	The trading hours of the product. For example, 20090507:0700-1830,1830-2330;20090508:CLOSED.
liquidHours() As String	The liquid trading hours of the product. For example, 20090507:0930-1600;20090508:CLOSED.

## IComboLeg

Property	Description
conId() As Integer	The unique contract identifier specifying the security.
action() As String	The side (buy or sell) for the leg you are constructing.
ratio() As Integer	Select the relative number of contracts for the leg you are constructing. To help determine the ratio for a specific combination order, refer to the Interactive Analytics section of the User's Guide.
exchange() As String	The exchange to which the complete combination order will be routed.
openClose() As Integer	Specifies whether the order is an open or close order. Valid values are: <ul style="list-style-type: none"> <li>• 0 - Same as the parent security. This is the only option for retail customers.</li> <li>• 1 - Open. This value is only valid for institutional customers.</li> <li>• 2 - Close. This value is only valid for institutional customers.</li> <li>• Unknown - (3)</li> </ul>
shortSaleSlot() Integer	For institutional customers only. <ul style="list-style-type: none"> <li>• 0 - inapplicable (i.e. retail customer or not short leg)</li> <li>• 1 - clearing broker</li> <li>• 2 - third party. If this value is used, you must enter a designated location.</li> </ul>
designatedLocation() As String	If shortSaleSlot == 2, the designatedLocation must be specified. Otherwise leave blank or orders will be rejected.

## IComboLegList

Property	Description
Add() As Object	Adds combo legs to a combo leg list.
Count() As Integer	Leg count.
Item(Integer) As Object	Get leg by index.

## IOrder

Property	Description
orderId() As Integer	The id for this order.
clientId() As Integer	The id of the client that placed this order.
permid() As Integer	The TWS id used to identify orders, remains the same over TWS sessions.
action() As String	Identifies the side. Valid values are: BUY, SELL, SSHORT
totalQuantity() As Integer	The order quantity.
orderType() As String	Identifies the order type. Valid values are: <ul style="list-style-type: none"> <li>• MKT</li> <li>• MKTCLS</li> <li>• LMT</li> <li>• LMTCLS</li> <li>• PEGMKT</li> <li>• SCALE</li> <li>• STP</li> <li>• STPLMT</li> <li>• TRAIL</li> <li>• REL</li> <li>• VWAP</li> <li>• TRAILLIMIT</li> </ul>
lmtPrice() As Double	This is the LIMIT price, used for limit, stop-limit and relative orders. In all other cases specify zero. For relative orders with no limit price, also specify zero.
auxPrice() As Double	This is the STOP price for stop-limit orders, and the offset amount for relative orders. In all other cases, specify zero.
timeInForce() As String	The time in force. Valid values are: DAY, GTC, IOC, GTD.

<b>Property</b>	<b>Description</b>
ocaGroup() As String	Identifies an OCA (one cancels all) group.
ocaType() As Integer	<p>Tells how to handle remaining orders in an OCA group when one order or part of an order executes. Valid values include:</p> <ul style="list-style-type: none"> <li>• 1 = Cancel all remaining orders with block</li> <li>• 2 = Remaining orders are proportionately reduced in size with block</li> <li>• 3 = Remaining orders are proportionately reduced in size with no block</li> </ul> <p>If you use a value "with block" gives your order has overfill protection. This means that only one order in the group will be routed at a time to remove the possibility of an overfill.</p>
account() As String	The account. For institutional customers only.
openClose() As String	Specifies whether the order is an open or close order. For institutional customers only. Valid values are O, C.
origin() As Integer	The order origin. For institutional customers only. Valid values are 0 = customer, 1 = firm
orderRef() String	The order reference. For institutional customers only.
transmit() As Integer	Specifies whether the order will be transmitted by TWS. If set to false, the order will be created at TWS but will not be sent.
parentId() As Integer	The order ID of the parent order, used for bracket and auto trailing stop orders.
blockOrder() As Integer	If set to true, specifies that the order is an ISE Block order.
sweepToFill() As Integer	If set to true, specifies that the order is a Sweep-to-Fill order.
displaySize() As Integer	The publicly disclosed order size, used when placing Iceberg orders.

Property	Description
triggerMethod() As Integer	<p>Specifies how Simulated Stop, Stop-Limit and Trailing Stop orders are triggered. Valid values are:</p> <ul style="list-style-type: none"> <li>• <b>0</b> - The default value. The "double bid/ask" method will be used for orders for OTC stocks and US options. All other orders will used the "last" method.</li> <li>• <b>1</b> - use "double bid/ask" method, where stop orders are triggered based on two consecutive bid or ask prices.</li> <li>• <b>2</b> - "last" method, where stop orders are triggered based on the last price.</li> <li>• <b>3</b> double last method.</li> <li>• <b>4</b> bid/ask method.</li> <li>• <b>7</b> last or bid/ask method.</li> <li>• <b>8</b> mid-point method.</li> </ul>
outsideRth() As Integer	If set to true, allows orders to also trigger or fill outside of regular trading hours.
hidden() As Integer	If set to true, the order will not be visible when viewing the market depth. This option only applies to orders routed to the ISLAND exchange.
discretionaryAmt() As Double	The amount off the limit price allowed for discretionary orders.
goodAfterTime() As String	The trade's "Good After Time," format "YYYYMMDD hh:mm:ss (optional time zone)" Use an empty String if not applicable.
goodTillDate() As String	You must enter a tif value of GTD. The trade's "Good Till Date," format is: YYYYMMDD hh:mm:ss (optional time zone) Use an empty String if not applicable.
faGroup() As String	The Financial Advisor group the trade will be allocated to -- use an empty String if not applicable.
faProfile() As String	The Financial Advisor allocation profile the trade will be allocated to -- use an empty String if not applicable.
faMethod() As String	The Financial Advisor allocation method the trade will be allocated with -- use an empty String if not applicable.
faPercentage() As String	The Financial Advisor percentage concerning the trade's allocation -- use an empty String if not applicable.
shortSaleSlot() As Integer	Values are 1 or 2.

<b>Property</b>	<b>Description</b>
designatedLocation() As String	Use only when shortSaleSlot value = 2.
ocaType() As Integer	Cancel on Fill with Block = 1 Reduce on Fill with Block = 2 Reduce on Fill without Block = 3
overridePercentageConstraints() As Integer	Precautionary constraints are defined on the TWS Presets page, and help ensure that your price and size order values are reasonable. Orders sent from the API are also validated against these safety constraints, and may be rejected if any constraint is violated. To override validation, set this parameter's value to True. Valid values include: <ul style="list-style-type: none"> <li>• 0 = False</li> <li>• 1 = True</li> </ul>
rule80A() As String	Valid values are: <ul style="list-style-type: none"> <li>• Individual = 'I'</li> <li>• Agency = 'A',</li> <li>• AgentOtherMember = 'W'</li> <li>• IndividualPTIA = 'J'</li> <li>• AgencyPTIA = 'U'</li> <li>• AgentOtherMemberPTIA = 'M'</li> <li>• IndividualPT = 'K'</li> <li>• AgencyPT = 'Y'</li> <li>• AgentOtherMemberPT = 'N'</li> </ul>
settlingFirm() As String	Institutional only.
clearingAccount() As String	For IBExecution customers: Specifies the true beneficiary of the order. This value is required for FUT/FOP orders for reporting to the exchange.
clearingIntent() As String	For IBExecution customers: Valid values are: IB, Away, and PTA (post trade allocation).
allOrNone() As Integer	yes=1, no=0
minQty() As Integer	Identifies a minimum quantity order type.
percentOffset() As Double	The percent offset amount for relative orders.
eTradeOnly() As Integer	Trade with electronic quotes. yes = 1, no = 0
firmQuoteOnly() As Integer	Trade with firm quotes. yes = 1, no = 0
nbboPriceCap() As Double	The maximum Smart order distance from the NBBO.

Property	Description
auctionStrategy() As Integer	<p>Valid values are:</p> <ul style="list-style-type: none"> <li>• match = 1</li> <li>• improvement = 2</li> <li>• transparent = 3</li> </ul> <p>For BOX exchange only.</p>
startingPrice() As Double	The starting price. Valid on BOX orders only.
stockRefPrice() As Double	The stock reference price. The reference price is used for VOL orders to compute the limit price sent to an exchange (whether or not Continuous Update is selected), and for price range monitoring.
delta() As Double	The stock delta. Valid on BOX orders only.
stockRangeLower() As Double	The lower value for the acceptable underlying stock price range. For price improvement option orders on BOX and VOL orders with dynamic management.
stockRangeUpper() As Double	The upper value for the acceptable underlying stock price range. For price improvement option orders on BOX and VOL orders with dynamic management.
volatility() Double	What the price is, computed via TWS's Options Analytics. For VOL orders, the limit price sent to an exchange is not editable, as it is the output of a function. Volatility is expressed as a percentage.
volatilityType() Integer	<p>How the volatility is calculated.</p> <ul style="list-style-type: none"> <li>• Daily = 1</li> <li>• Annual = 2</li> </ul>
deltaNeutralOrderType() As String	VOL orders only. Enter an order type to instruct TWS to submit a delta neutral trade on full or partial execution of the VOL order. For no hedge delta order to be sent, specify NONE.
deltaNeutralAuxPrice() As Double	VOL orders only. Use this field to enter a value if the value in the <i>deltaNeutralOrderType</i> field is an order type that requires an Aux price, such as a REL order.

<b>Property</b>	<b>Description</b>
continuousUpdate() As Integer	Used for dynamic management of volatility orders. Determines whether TWS is supposed to update the order price as the underlying moves. If selected, the limit price sent to an exchange is modified by TWS if the computed price of the option changes enough to warrant doing so. This is very helpful in keeping the limit price sent to the exchange up to date as the underlying price changes.
ireferencePriceType() As Integer	Used for dynamic management of volatility orders. Set to <ul style="list-style-type: none"> <li>• 1 = Average of National Best Bid or Ask, or set to</li> <li>• 2 = National Best Bid when buying a call or selling a put; and National Best Ask when selling a call or buying a put.</li> </ul>
trailStopPrice() As Double	For TRAILLIMIT orders only
scaleInitLevelSize() As Integer	For Scale orders: Defines the size of the first, or initial, order component.
scaleSubsLevelSize() As Integer	For Scale orders: Defines the order size of the subsequent scale order components. Used in conjunction with scaleInitLevelSize().
scalePriceIncrement() As Double	For Scale orders: Defines the price increment between scale components. This field is required.
basisPoints() As Integer	For EFP orders only
basisPointsType() As Double	For EFP orders only
whatIf() As Integer	Use to request pre-trade commissions and margin information. If set to true, margin and commissions data is received back via the IOrderState() object for the openOrder() callback.

## IOrderState

Property	Description
status() As String	Displays the order status.
initMargin() As String	Shows the impact the order would have on your initial margin.
maintMargin() As String	Shows the impact the order would have on your maintenance margin.
equityWithLoan() As String	Shows the impact the order would have on your equity with loan value.
commission() As Double	Shows the commission amount on the order.
minCommission() As Double	Used in conjunction with the maxCommission field, this defines the lowest end of the possible range into which the actual order commission will fall.
maxCommission() As Double	Used in conjunction with the minCommission field, this defines the highest end of the possible range into which the actual order commission will fall.
commissionCurrency() As String	Shows the currency of the commission value.
warningText() As String	Displays a warning message if warranted.

## IScannerSubscription

Property	Description
numberOfRows() As Integer	Defines the number of rows of data to return for a query.
instrument() As String	Defines the instrument type for the scan.
locations() As String	The location, currently the only valid location is US stocks.
scanCode() As String	Can be left blank.
priceAbove() As Double	Filter out contracts with a price lower than this value. Can be left blank.
priceBelow() As Double	Filter out contracts with a price higher than this value. Can be left blank.
volumeAbove() As Integer	Filter out contracts with a volume lower than this value. Can be left blank.
marketCapAbove() As Double	Filter out contracts with a market cap lower than this value. Can be left blank.
marketCapBelow() As Double	Filter out contracts with a market cap above this value. Can be left blank.
moodyRatingAbove() As String	Filter out contracts with a Moody rating below this value. Can be left blank.
moodyRatingBelow() As String	Filter out contracts with a Moody rating above this value. Can be left blank.
spRatingAbove() As String	Filter out contracts with an S&P rating below this value. Can be left blank.
spRatingBelow() As String	Filter out contracts with an S&P rating above this value. Can be left blank.
maturityDateAbove() As String	Filter out contracts with a maturity date earlier than this value. Can be left blank.
maturityDateBelow() As String	Filter out contracts with a maturity date later than this value. Can be left blank.
couponRateAbove() As String	Filter out contracts with a coupon rate lower than this value. Can be left blank.
couponRateBelow() As String	Filter out contracts with a coupon rate higher than this value. Can be left blank.
excludeConvertible() As Integer	Filter out convertible bonds. Can be left blank.
scannerSettingPairs() As String	Can leave empty. For example, a pairing "Annual, true" used on the "top Option Implied Vol % Gainers" scan would return annualized volatilities.
averageOptionVolumeAbove () As Integer	Can leave empty.

Property	Description
stockTypeFilter() As String	Values are: <ul style="list-style-type: none"> <li>• ALL (excludes nothing)</li> <li>• STOCK (excludes ETFs)</li> <li>• ETF (includes ETFs)</li> </ul>

## ITagValueList

Property	Description
Count() As Integer	The number of tag-value pairs (IBAlgo parameters).
Item(Integer) As Object	A tag-value pair (IBAlgo parameter). For more information, see <a href="#">IBAlgo Parameters</a> .

## ITagValue

Property	Description
tag() As String	An IBAlgo order parameter. For more information, see <a href="#">IBAlgo Parameters</a> .
value() As String	The value of the IBAlgo parameter.

## IUnderComp

Property	Description
conId() As Integer	The unique contract identifier specifying the security. Used for Delta-Neutral Combo contracts.
delta() As Double	The underlying stock or future delta. Used for Delta-Neutral Combo contracts.
price() As Double	The price of the underlying. Used for Delta-Neutral Combo contracts.

## ActiveX Properties

The table below defines properties you can use when connecting to a server using ActiveX.

**Note:** All of these properties have been deprecated as of API release 9.4 (December 2007), except *String TwsConnectionTime* and *long serverVersion*) and will soon be removed. Please update your software to use the properties in the ActiveX [IOrder](#) COM object (except where otherwise noted) instead of these deprecated properties.

Property	Description
<code>String TwsConnectionTime</code>	Connection time.
<code>long serverVersion</code>	Server Version.
<b>The following properties have been deprecated. They have all been replaced by properties in the IOrder COM object except where otherwise noted.</b>	
<code>String tif</code>	The time in force. Valid values are: DAY, GTC, IOC, GTD
<code>boolean transmit</code>	Specifies whether the order will be transmitted by the server. If set to false, the order will be created but will not be sent.
<code>String oca</code>	Identifies an OCA (one cancels all) group.
<code>String account</code>	Identifies the account. <b>For institutional customers only.</b>
<code>String orderRef</code>	Customer-defined order identification field. Enter a reference string to help identify orders.
<code>long origin</code>	Identifies the order origin. <b>For institutional customers only.</b> Valid values are: <ul style="list-style-type: none"> <li>• 0 = customer</li> <li>• 1 = firm</li> </ul>
<code>String openClose</code>	Specifies whether the order is an open or close. <b>For institutional customers only.</b> Valid values are O, C.
<code>long parentId</code>	The order ID of the parent order, used for bracket and auto trailing stop orders.
<code>boolean blockOrder</code>	If set to true, specifies that the order is an ISE Block order.
<code>boolean sweepToFill</code>	If set to true, specifies that the order is a Sweep-to-Fill order.
<code>long displaySize</code>	The publicly disclosed order size, used when placing Iceberg orders.

Property	Description
long triggerMethod	<p>Specifies how Simulated Stop, Stop-Limit and Trailing Stop orders are triggered. Valid values are:</p> <ul style="list-style-type: none"> <li>• <b>0</b> - The default value. The "double bid/ask" method will be used for orders for OTC stocks and US options. All other orders will used the "last" method.</li> <li>• <b>1</b> - use "double bid/ask" method, where stop orders are triggered based on two consecutive bid or ask prices.</li> <li>• <b>2</b> - "last" method, where stop orders are triggered based on the last price.</li> </ul>
boolean ignoreRth	If set to true, allows triggering of orders outside of regular trading hours.
boolean hidden	If set to true, the order will not be visible when viewing the market depth. this option only works with orders routed to the ISLAND exchange.
long clientIdFilter	<p>Filter the results of the reqExecutions() method based on the clientId.</p> <p><b>Note:</b> This property has been deprecated. Use the corresponding property in <a href="#">IExecutionFilter</a>.</p>
String acctCodeFilter	<p>Filter the results of the reqExecutions() method based on an account code. This is only relevant for FA managed accounts.</p> <p><b>Note:</b> This property has been deprecated. Use the corresponding property in <a href="#">IExecutionFilter</a>.</p>
String timeFilter	<p>Filter the results of the reqExecutions() method based on execution reports received after the specified time.</p> <p><b>Note:</b> This property has been deprecated. Use the corresponding property in <a href="#">IExecutionFilter</a>.</p>
String symbolFilter	<p>Filter the results of the reqExecutions() method based on the order symbol.</p> <p><b>Note:</b> This property has been deprecated. Use the corresponding property in <a href="#">IExecutionFilter</a>.</p>
String secTypeFilter	<p>Filter the results of the reqExecutions() method based on the order security type. Refer to the ActiveX methods section for the list of valid security types.</p> <p><b>Note:</b> This property has been deprecated. Use the corresponding property in <a href="#">IExecutionFilter</a>.</p>
String exchangeFilter	<p>Filter the results of the reqExecutions() method based on the order exchange.</p> <p><b>Note:</b> This property has been deprecated. Use the corresponding property in <a href="#">IExecutionFilter</a>.</p>

Property	Description
String sideFilter	Filter the results of the reqExecutions() method based on the order action. Refer to the ActiveX methods sections for the list of valid order actions. <b>Note:</b> This property has been deprecated. Use the corresponding property in <a href="#">IExecutionFilter</a> .
double discretionaryAmt	Set an order's discretionary amount.
long shortSaleSlot	For institutional customers only. <ul style="list-style-type: none"> <li>• 0 - inapplicable (i.e. retail customer or not short leg)</li> <li>• 1 - clearing broker</li> <li>• 2 - third party. If this value is used, you must enter a designated location.</li> </ul> <b>Note:</b> This property has been deprecated. Use the corresponding property in <a href="#">IComboLeg</a> .
String designatedLocation	Only valid when shortSaleSlot value = 2. Otherwise leave blank or orders will be rejected.
String goodAfterTime	The trade's "Good After Time," format "YYYYMMDD hh:mm:ss (optional time zone)" Use an empty String if not applicable. This property has been deprecated. Use the corresponding property in <a href="#">IExecutionFilter</a> .
String goodTillDate	You must enter a tif value of GTD. The trade's "Good Till Date," format "YYYYMMDD hh:mm:ss (optional time zone)" Use an empty String if not applicable.
long ocaType	Cancel on Fill with Block = 1 Reduce on Fill with Block = 2 Reduce on Fill without Block = 3
boolean rthOnly	Regular trading hours only. 0 = no 1 = yes.
String rule80A	Individual = 'I' Agency = 'A', AgentOtherMember = 'W' IndividualPTIA = 'J' AgencyPTIA = 'U' AgentOtherMemberPTIA = 'M' IndividualPT = 'K' AgencyPT = 'Y' AgentOtherMemberPT = 'N'
String settlingFirm	Institutional only.
boolean allOrNone	<ul style="list-style-type: none"> <li>• 0 = no</li> <li>• 1 = yes</li> </ul>
long minQty	Identifies a minimum quantity order type.

Property	Description
double percentOffset	The percent offset for relative orders.
boolean eTradeOnly	Trade with electronic quotes. <ul style="list-style-type: none"> <li>• 0 = no</li> <li>• 1 = yes</li> </ul>
boolean firmquoteOnly	Trade with firm quotes. <ul style="list-style-type: none"> <li>• 0 = no</li> <li>• 1 = yes</li> </ul>
double nbboPriceCap	Maximum Smart order distance from the NBBO.
long auctionStrategy	Valid values are: <ul style="list-style-type: none"> <li>• match = 1</li> <li>• improvement = 2</li> <li>• transparent = 3</li> </ul> For orders on BOX only.
double startingPrice	Starting price. For BOX orders only.
double stockRefPrice	Used for VOL orders to compute the limit price sent to an exchange (whether or not Continuous Update is used), and for price range monitoring. Also used for price improvement option orders.
double delta	The stock delta. For BOX orders only.
double stockRangeLower	The lower value for the acceptable underlying stock price range. For price improvement option orders on BOX and VOL orders with dynamic management.
double stockRangeUpper	The upper value for the acceptable underlying stock price range. For price improvement option orders on BOX and VOL orders with dynamic management.
bool overridePercentageConstraints	If set, allows you to override TWS order price percentage constraints set to reject orders that deviate too far from the NBBO. This was created to avoid transmitting orders with an incorrect price.
double volatility	The option price in volatility, as calculated by TWS' Option Analytics. This value is expressed as a percent and is used to calculate the limit price sent to the exchange.
long volatilityType	Select: <ul style="list-style-type: none"> <li>• 1 = Daily volatility</li> <li>• 2 = Annual volatility</li> </ul>
string deltaNeutralOrderType	VOL orders only. Enter an order type to instruct TWS to submit a delta neutral trade on full or partial execution of the VOL order. For no hedge delta order to be sent, specify NONE.
int deltaNeutralAuxPrice	VOL orders only. Use this field to enter a value if the value in the <i>deltaNeutralOrderType</i> field is an order type that requires an Aux price, such as a REL order.

<b>Property</b>	<b>Description</b>
bool continuousUpdate	VOL orders only. Specifies whether TWS will automatically update the limit price of the order as the underlying price moves.
long referencePriceType	VOL orders only. Specifies how you want TWS to calculate the limit price for options, and for stock range price monitoring. Valid values include: <ul style="list-style-type: none"> <li>• 1 = Average of NBBO</li> <li>• 2 = NBB or the NBO depending on the action and right.</li> </ul>
int scaleNumComponents	For Scale orders: Defines the number of component orders into which the parent order will be split, thereby backing into the number of units within each component.
int scaleComponentSize	For Scale orders: Defines the number of units per component, backing into the number of components into which the parent order is split.
double scalePriceIncrement	For Scale orders: Defines the price increment per scale component.
double basisPoints	For EFP orders.
int basisPointsType	For EFP orders.

## Placing a Combination Order

A combination order is a special type of order that is constructed of many separate legs but executed as a single transaction. Submit combo orders such as calendar spreads, conversions and straddles using the BAG security type (defined in the *Contract* object). The key to implementing a successful API combination order using the API is to knowing how to place the same order using Trader Workstation. If you are familiar with placing combination orders in TWS, then it will be easier to place the same order using the API, because the API only imitates the behavior of TWS.

### Example

In this example, a customer places a BUY order on a calendar spread for GOOG. To buy one calendar spread means:

**Leg 1: Sell 1 GOOG OPT SEP 18 '09 150.0 CALL (100)**

**Leg 2: Buy 1 GOOG OPT JAN 21 '11 150.0 CALL (100)**

Here is a summary of the steps required to place a combo order using the API:

- Obtain the contract id (conId) for each leg. Get this number by invoking the [reqContractDetailsEx\(\)](#) method.
- Include each leg on the [IComboLeg](#) COM object by populating the related fields.
- Implement the [placeOrderEx\(\)](#) method with the [IContract](#) and [IOrder](#) COM objects.

### To place this combo order

- 1 Get the Contract IDs for both leg definitions:

```

'First Leg
Dim con1 As TWSLib.IContract
con1 = Tws1.createContract

con1.symbol = "GOOG"
con1.secType = "OPT"
con1.expiry = "200909"
con1.strike = 150.0
con1.right = "C"
con1.multiplier = "100"
con1.exchange = "SMART"
con1.currency = "USD"

Tws1.reqContractDetailsEx(1, con1)

'Second Leg
Dim con2 As TWSLib.IContract
con2 = Tws1.createContract

con2.symbol = "GOOG"
con2.secType = "OPT"
```

```

con2.expiry = "201101"
con2.strike = 150.0
con2.right = "C"
con2.multiplier = "100"
con2.exchange = "SMART"
con2.currency = "USD"

Tws1.reqContractDetailsEx(2, con2)

'All conId numbers are delivered by the ContractDetail()

Private Sub Tws1_contractDetailsEx(ByVal sender As Object, ByVal
e As AxTWSLib._DTwsEvents_contractDetailsExEvent) Handles
Tws1.contractDetailsEx

    Dim contractDetails As TWSLib.IContractDetails
    contractDetails = e.contractDetails

    Dim contract As TWSLib.IContract
    contract = contractDetails.summary

    'reqId = 1 is corresponding to the first request or first leg
    'reqId = 2 is corresponding to the second request or second leg

    If e.reqId = 1 Then
        leg1 = contract.conId 'to obtain conId for the first leg
    End If

    If e.reqId = 2 Then
        leg2 = contract.conId 'to obtain conId for the second leg
    End If

End Sub

```

- 2** Once the program has acquired the conId value for each leg, include it in the ComboLeg object:

```

TWSLib.IComboLegList
addAllLegs = Tws1.createComboLegList

'First Combo leg
Dim Leg1 As TWSLib.IComboLeg
Leg1 = addAllLegs.Add()

Leg1.conId = leg1_conId
Leg1.ratio = 1
Leg1.action = "SELL"
Leg1.exchange = "SMART"
Leg1.openClose = 0
Leg1.shortSaleSlot = 0
Leg1.designatedLocation = ""

' Second Combo leg
Dim Leg2 As TWSLib.IComboLeg

```

```
Leg2 = addAllLegs.Add()

Leg1.conId = leg2_conId
Leg1.ratio = 1
Leg1.action = "BUY"
Leg1.exchange = "SMART"
Leg1.openClose = 0
Leg1.shortSaleSlot = 0
Leg1.designatedLocation = ""
```

- 3** Invoke the `placeOrder()` method with the appropriate contract and order objects:

```
Dim contract As TWSLib.IContract
contract = Tws1.createContract

contract.symbol = "USD"
contract.secType = "BAG"
contract.exchange = "SMART"
contract.currency = "USD"
contract.comboLegs = addAllLegs

Dim order As TWSLib.IOrder
order = Tws1.createOrder

order.action = "BUY"
order.totalQuantity = 1
order.orderType = "MKT"

Tws1.placeOrderEx(OrderId, contract, order)
```

**Note:** For more information on combination orders, see the TWS Users Guide topics [About Combination Orders](#) and [Notes on Combination Orders](#).



This chapter describes the C++ API, including the following topics:

- [\*\*Linking to TWS using the TwsSocketClient.dll\*\*](#)
- [\*\*Using the C++ TestSocketClient Sample Program\*\*](#)
- [\*\*Class EClientSocket Functions\*\*](#)
- [\*\*Class EWrapper Functions\*\*](#)
- [\*\*SocketClient Properties\*\*](#)
- [\*\*Placing a Combination Order\*\*](#)

## Linking to TWS using the *TwsSocketClient.dll*

### To link to TWS using the *TwsSocketClient.dll*

- 1** Create a Windows application using MS Visual Studio (version 5.0 or higher).
- 2** Add the full path to the `\SocketClient\include` directory in your API installation folder to your project's include path. This should be done for any individual project that accesses the *TwsSocketClient* library's header files.
- 3** Add the full path to the `\SocketClient\lib\TwsSocketclient.lib` file in your API installation directory to your project's libraries path. This should be done for any individual project that accesses the *TwsSocketClient* library.
- 4** Include `EWrapper.h` and `EClientSocket.h` in any Visual C++ source code that accesses their functionality and data structures.
- 5** Subclass the `EWrapper` class.
- 6** Override the following functions:

Ewrapper Function	Description
<code>tickPrice()</code>	Handles market data.
<code>tickSize()</code>	
<code>tickOptionComputation()</code>	
<code>tickGeneric()</code>	
<code>tickString()</code>	
<code>tickEFP()</code>	
<code>orderStatus()</code>	Receives order status.
<code>openOrder()</code>	Receives open orders.
<code>error()</code>	Receives error information.
<code>connectionClosed()</code>	Notifies when TWS terminates the connection.
<code>updateAccountValue()</code>	Receives current account values.
<code>updateAccountTime()</code>	Receives the last time account information was updated.
<code>updatePortfolio()</code>	Receives current portfolio information.
<code>nextValidId()</code>	Receives the next valid order ID upon connection.
<code>contractDetails()</code>	Receives contract information.
<code>contractDetailsEnd()</code>	Identifies the end of a given contract details request.
<code>bondContractDetails()</code>	Receives bond contract information.
<code>execDetails()</code>	Receives execution report information.
<code>updateMktDepth()</code>	Receives market depth information.

Ewrapper Function	Description
updateMktDepthL2()	Receives Level II market depth information.
updateNewsBulletin()	Receives IB news bulletins.
managedAccounts()	Receives a list of Financial Advisor (FA) managed accounts.
receiveFA()	Receives FA configuration information.
historicalData()	Receives historical data results.
scannerParameters()	Receives an XML document that describes the valid parameters of a scanner subscription.
scannerData()	Receives market scanner results.
realTimeBar()	Receives real-time bars.
currentTime()	Receives the current system time on the server.
fundamentalData()	Receives Reuters global fundamental market data.

- 7** Instantiate the EClientSocket class.
- 8** Call the following functions:
  - a** Import com.ib.client.\* into your source code file.
  - b** Implement the EWrapper interface. This class will receive messages from the socket.

**c** Override the following functions:

Ewrapper Functions	Description
tickPrice()	Handles market data.
tickSize()	
tickOptionComputation()	
tickGeneric()	
tickString()	
tickEFP()	
orderStatus()	Receives order status.
openOrder()	Receives open orders.
error()	Receives error information.
connectionClosed()	Notifies when TWS terminates the connection.
updateAccountValue()	Receives current account values.
updateAccountTime()	Receives the last time account information was updated.
updatePortfolio()	Receives current portfolio information.
nextValidId()	Receives the next valid order ID upon connection.
contractDetails()	Receives contract information.
contractDetailsEnd()	Identifies the end of a given contract details request.
bondContractDetails()	Receives bond contract information.
execDetails()	Receives execution report information.
updateMktDepth()	Receives market depth information.
updateMktDepthL2()	Receives Level II market depth information.
updateNewsBulletin()	Receives IB news bulletins.
managedAccounts()	Receives a list of Financial Advisor (FA) managed accounts.
receiveFA()	Receives FA configuration information.
historicalData()	Receives historical data results.
scannerParameters()	Receives an XML document that describes the valid parameters of a scanner subscription.
scannerData()	Receives market scanner results.
realTimeBar()	Receives real-time bars.
currentTime()	Receives the current system time on the server.
fundamentalData()	Receives Reuters global fundamental market data.

- d** Instantiate the EClientSocket class. This object will be used to send messages to TWS.
- e** Call the following functions:

EClientSocket Functions	Description
eConnect()	Connects to TWS.
eDisconnect()	Disconnects from TWS.
reqMktData()	Requests market data.
cancelMktData()	Cancels market data.
reqMktDepth()	Requests market depth.
cancelMktDepth()	Cancels market depth.
reqContractDetails()	Requests contract details.
placeOrder()	Places an order.
cancelOrder()	Cancels an order.
reqAccountUpdates()	Requests account values, portfolio, and account update time information.
reqExecutions()	Requests a list of the day's execution reports.
reqOpenOrders()	Requests a list of current open orders for the requesting client and associates TWS open orders with the client. The association only occurs if the requesting client has a Client ID of 0.
reqAllOpenOrders()	Requests a list of all open orders.
reqAutoOpenOrders()	Automatically associates a new TWS with the client. The association only occurs if the requesting client has a Client ID of 0.
reqNewsBulletin()	Requests IB news bulletins.
cancelNewsBulletins()	Cancels IB news bulletins.
setServerLogLevel()	Sets the level of API request and processing logging.
reqManagedAccts()	Requests a list of Financial Advisor (FA) managed account codes.
requestFA()	Requests FA configuration information from TWS.
replaceFA()	Modifies FA configuration information from the API.
reqScannerParameters()	Requests an XML document that describes the valid parameters of a scanner subscription.
reqScannerSubscription()	Requests market scanner results.
cancelScannerSubscription()	Cancels a scanner subscription.

<b>EClientSocket Functions</b>	<b>Description</b>
reqHistoricalData()	Requests historical data.
cancelHistoricalData()	Cancels historical data.
reqRealTimeBars()	Requests real-time bars.
cancelRealTimeBars()	Cancels real-time bars.
exerciseOptions()	Exercises options.
reqCurrentTime()	Requests the current server time.
serverVersion()	Returns the version of the TWS instance to which the API application is connected.
TwsConnectionTime()	Returns the time the API application made a connection to TWS.
reqFundamentalData()	Requests Reuters global fundamental data. There must be a subscription to Reuters Fundamental set up in Account Management before you can receive this data.
cancelFundamentalData()	Cancels Reuters global fundamental data.

To run the program, ensure that the *TwsSocketClient.dll* is in the same directory as your executable, or in your path. By default, the *TwsSocketClient.dll* file installs into your C:\Windows\system or C:\WINNT\system32 directory.

## Using the C++ TestSocketClient Sample Program

You can access TWS through a Microsoft Windows-based C++ application using the TWSSocketClient.dll component. Before you can connect to TWS using the SocketClient component, you must:

- Install the socket client component and sample programs
- Configure TWS to support the API components
- Have Microsoft Visual Studio (Visual C++ 5.0 or higher) installed on your PC.

The TestSocketClient program is a sample program that shows you how to use sockets to connect to TWS from a Microsoft Windows-based C++ application.

### To run the pre-built sample application

We've included a complete C++ client with our API software. To run this sample application, go to your TWS API installation folder, then open the \TestSocketClient\Release folder. Run the file named client2.exe.

### To run the TestSocketClient program from Microsoft Visual Studio 2008

- 1 From Microsoft Visual Studio 2008, open the project file *client2.dsp* from the \TestSocketClient directory in your API installation folder.
- 2 If you are prompted to convert *client2.dsp* to the current Visual C++ project format, click *Yes to All*.
- 3 Build that project, and run it.

#### In case of errors:

If the C++ sample application won't build or run due to errors, try the following steps:

- 1 In the Visual Studio 2008 Solution Explorer, remove the following files from the project (select the item from the list and press **Delete**):
  - Remove *DlgShareAllocation.h* from the list of Header Files.
  - Remove *DlgShareAllocation.cpp* from the list of Source Files.
  - Remove *TwsSocketClient.lib* from the project.
- 2 In the Solution Explorer, add the following existing items from the *TestSocketClient* folder in your TWS API installation folder (right-click the Header Files/Source Files folder, then select **Add > Existing Item**):
  - Add *DlgUnderComp.h* to the list of Header Files.
  - Add *DlgUnderComp.cpp* to the list of Source Files.

**Note:** If you are running TWS API Version 9.6 or higher, you must also add *DlgAlgoParams.h* to the list of Header Files and *DlgAlgoParams.cpp* to the list of Source Files in the project.

- 3** In the Solution Explorer, add the following existing items from the *SocketClient/src* folder in your TWS API installation folder to the list of Source Files (right-click the Source Files folder, then select **Add > Existing Item**):
  - *EClientSocket.cpp*
  - *MySocket.cpp*
- 4** Build and run the sample client.

## Class EClientSocket Functions

The list below define the class EClientSocket functions you can use when connecting to TWS.  
The list of functions includes:

<b>Connection and Server</b>	<b>Contract Details</b>
<code>EClientSocket()</code>	
<code>eConnect()</code>	<code>reqContractDetails()</code>
<code>eDisconnect()</code>	<b>Market Depth</b>
<code>isConnected()</code>	<code>reqMktDepth()</code>
<code>reqCurrentTime()</code>	<code>cancelMktDepth()</code>
<code>serverVersion()</code>	<b>News Bulletins</b>
<code>TwsConnectionTime()</code>	<code>reqNewsBulletins()</code>
<code>setLogLevel()</code>	<code>cancelNewsBulletins()</code>
<code>checkMessages()</code>	<b>Financial Advisors</b>
<b>Market Data</b>	<code>reqManagedAccts()</code>
<code>reqMktData()</code>	<code>requestFA()</code>
<code>cancelMktData()</code>	<code>replaceFA()</code>
<b>Orders</b>	<b>Historical Data</b>
<code>placeOrder()</code>	<code>reqHistoricalData()</code>
<code>cancelOrder()</code>	<code>cancelHistoricalData()</code>
<code>reqOpenOrders()</code>	<b>Market Scanners</b>
<code>reqAllOpenOrders()</code>	<code>reqScannerParameters()</code>
<code>reqAutoOpenOrders()</code>	<code>reqScannerSubscription()</code>
<code>reqIDs()</code>	<code>cancelScannerSubscription()</code>
<code>exerciseOptions()</code>	<b>Real Time Bars</b>
<b>Account</b>	<code>reqRealTimeBars()</code>
<code>reqAccountUpdates()</code>	<code>cancelRealTimeBars()</code>
<b>Executions</b>	<b>Fundamental Data</b>
<code>reqExecutions()</code>	<code>reqFundamentalData()</code>
	<code>cancelFundamentalData()</code>

### **EClientSocket()**

This is the constructor.

```
EClientSocket( EWrapper *ptr)
```

<b>Parameter</b>	<b>Description</b>
<b>ptr</b>	The pointer to an object that was derived from the EWrapper base class.

## eConnect()

This function must be called before any other. There is no feedback for a successful connection, but a subsequent attempt to connect will return the message "Already connected."

```
bool eConnect( const char *host, UINT port, int clientId=0)
```

Parameter	Description
<b>host</b>	The host name or IP address of the machine where TWS is running. Leave blank to connect to the local host.
<b>port</b>	Must match the port specified in TWS on the Configure>API>Socket Port field.
<b>clientId</b>	A number used to identify this client connection. All orders placed/modified from this client will be associated with this client identifier. Note: Each client MUST connect with a unique clientId.

## eDisconnect()

Call this function to terminate the connections with TWS. Calling this function does not cancel orders that have already been sent.

```
void eDisconnect()
```

Parameter	Description
<b>ptr</b>	The pointer to an object that was derived from the EWrapper base class.

## isConnected()

Call this function to check if there is a connection with TWS

```
void isConnected()
```

## reqMktData()

Call this function to request market data. The market data will be returned by the tickPrice and tickSize events.

```
void reqMktData(TickerID id, const Contract &contract, CString
genericTicklist, bool snapshot)
```

Parameter	Description
<b>id</b>	The ticker id. Must be a unique value. When the market data returns, it will be identified by this tag. This is also used when canceling the market data.
<b>contract</b>	This structure contains a description of the contract for which market data is being requested.
<b>genericTicklist</b>	A comma delimited list of generic tick types. Tick types can be found in the <a href="#">Generic Tick Types</a> page.
<b>snapshot</b>	Check to return a single snapshot of market data and have the market data subscription cancel. Do not enter any genericTicklist values if you use snapshot.

## cancelMktData()

After calling this function, market data for the specified id will stop flowing.

```
void cancelMktData(TickerID id)
```

Parameter	Description
<b>id</b>	The ID that was specified in the call to reqMktData().

## placeOrder()

Call this function to place an order. The order status will be returned by the orderStatus event.

```
void placeOrder( OrderID id, const Contract &contract, const Order
&order) See Extended Order Attributes
```

Parameter	Description
<b>id</b>	The order id. You must specify a unique value. When the order status returns, it will be identified by this tag. This tag is also used when canceling the order.
<b>contract</b>	This structure contains a description of the contract which is being traded.
<b>order</b>	This structure contains the details of the order. Note: Each client MUST connect with a unique clientId.

## cancelOrder()

Call this function to cancel an order.

```
void cancelOrder(OrderID id)
```

Parameter	Description
<b>id</b>	The order ID that was specified previously in the call to placeOrder()

## checkMessages()

This function should be called frequently (every 1 second) to check for messages received from TWS.

```
void checkMessages()
```

## reqOpenOrders()

Call this function to request the open orders that were placed from this client. Each open order will be fed back through the openOrder() and orderStatus() functions on the EWrapper.

**Note:** The client with a clientId of 0 will also receive the TWS-owned open orders. These orders will be associated with the client and a new orderId will be generated. This association will persist over multiple API and TWS sessions.

```
void reqOpenOrders()
```

## reqAccountUpdates()

Call this function to start getting account values, portfolio, and last update time information.

```
void reqAccountUpdates(bool subscribe, const CString& acctCode)
```

Parameter	Description
<b>subscribe</b>	If set to TRUE, the client will start receiving account and portfolio updates. If set to FALSE, the client will stop receiving this information.
<b>acctCode</b>	The account code for which to receive account and portfolio updates.

## reqExecutions()

When this function is called, the execution reports that meet the filter criteria are downloaded to the client via the execDetails() function.

```
void reqExecutions(int reqID, const ExecutionFilter& filter)
```

Parameter	Description
<b>reqId</b>	The ID of the data request. Ensures that responses are matched to requests if several requests are in process.
<b>filter</b>	This object contains attributes that describe the filter criteria used to determine which execution reports are returned.

## **reqIDs()**

Call this function to request from TWS the next valid ID that can be used when placing an order. After calling this function, the nextValidId() event will be triggered, and the id returned is that next valid ID. That ID will reflect any autobinding that has occurred (which generates new IDs and increments the next valid ID therein).

```
void reqIDs(int numIds)
```

Parameter	Description
<b>numIds</b>	The number of ids you want to reserve.

## **reqContractDetails()**

Call this function to download all details for a particular underlying. The contract details will be received via the contractDetails() function on the EWrapper.

```
void reqContractDetails (const Contract &contract)
```

Parameter	Description
<b>reqId</b>	The ID of the data request. Ensures that responses are matched to requests if several requests are in process.
<b>Contract</b>	The summary description of the contract being looked up.

## **reqMktDepth()**

Call this function to request market depth for a specific contract. The market depth will be returned by the updateMktDepth() and updateMktDepthL2() events.

```
void reqMktDepth (TickerID id, const Contract &contract, long numRows)
```

Parameter	Description
<b>id</b>	The ticker id. Must be a unique value. When the market depth data returns, it will be identified by this tag. This is also used when canceling the market depth
<b>contract</b>	This structure contains a description of the contract for which market depth data is being requested.
<b>numRows</b>	- specified the number of market depth rows to display.

## **cancelMktDepth()**

After calling this function, market depth data for the specified id will stop flowing.

```
void cancelMktDepth (TickerID id)
```

Parameter	Description
<b>id</b>	The ID that was specified in the call to reqMktDepth().

**reqNewsBulletins()**

Call this function to start receiving news bulletins. Each bulletin will be returned by the updatedNewsBulletin() event.

```
void reqNewsBulletins(bool allMsgs)
```

Parameter	Description
<b>allMsgs</b>	If set to TRUE, returns all the existing bulletins for the current day and any new ones. If set to FALSE, will only return new bulletins.

**cancelNewsBulletins()**

Call this function to stop receiving news bulletins.

```
void cancelNewsBulletins()
```

**setLogLevel()**

The default detail level is ERROR. For more details, see “API Logging” on page -18.

```
void setLogLevel(int logLevel)
```

Parameter	Description
<b>logLevel</b>	<p>Specifies the level of log entry detail used by the server (TWS) when processing API requests. Valid values include:</p> <ul style="list-style-type: none"> <li>• 1 = SYSTEM</li> <li>• 2 = ERROR</li> <li>• 3 = WARNING</li> <li>• 4 = INFORMATION</li> <li>• 5 = DETAIL</li> </ul>

**reqAllOpenOrders()**

Call this function to request the open orders placed from all clients and also from TWS. Each open order will be fed back through the openOrder() and orderStatus() functions on the EWrapper.

**Note:** No association is made between the returned orders and the requesting client.

```
void reqAllOpenOrders()
```

## reqAutoOpenOrders()

Call this function to request that newly created TWS orders be implicitly associated with the client. When a new TWS order is created, the order will be associated with the client, and fed back through the openOrder() and orderStatus() functions on the EWrapper.

**Note:** This request can only be made from a client with clientId of 0.

```
reqAutoOpenOrders (bool bAutoBind)
```

Parameter	Description
<b>bAutoBind</b>	If set to TRUE, newly created TWS orders will be implicitly associated with the client. If set to FALSE, no association will be made.

## reqManagedAccts()

Call this function to request the list of managed accounts. The list will be returned by the managedAccounts() function on the EWrapper.

**Note:** This request can only be made when connected to a FA managed account.

```
void reqManagedAccts ()
```

## requestFA()

Call this function to request FA configuration information from TWS. The data returns in an XML string via a "receiveFA" ActiveX event.

```
requestFA(long faDataType)
```

Parameter	Description
<b>faDataType</b>	Specifies the type of Financial Advisor configuration data being requested. Valid values include: <ul style="list-style-type: none"> <li>• 1 = GROUPS</li> <li>• 2 = PROFILE</li> <li>• 3 = ACCOUNT ALIASES</li> </ul>

## replaceFA()

Call this function to modify FA configuration information from the API. Note that this can also be done manually in TWS itself.

```
replaceFA(long faDataType, string XML)
```

Parameter	Description
<b>faDataType</b>	Specifies the type of Financial Advisor configuration data being modified via the API. Valid values include: <ul style="list-style-type: none"> <li>• 1 = GROUPS</li> <li>• 2 = PROFILE</li> <li>• 3 =ACCOUNT ALIASES</li> </ul>
<b>XML</b>	The XML string containing the new FA configuration information.

## reqHistoricalData()

```
void reqHistoricalData (TickerID id, const Contract &contract, String endTime, String durationStr, long barSizeSetting String whatToShow, int useRTH, int formatDate)
```

Parameter	Description
<b>id</b>	The id of the request. Must be a unique value. When the market data returns, it will be identified by this tag. This is also used when canceling the market data.
<b>contract</b>	This object contains a description of the contract for which market data is being requested.
<b>endTime</b>	Defines a query end date and time at any point during the past 6 mos. Valid values include any date/time within the past six months in the format: yyymmdd HH:mm:ss ttt where "ttt" is the optional time zone.
<b>durationStr</b>	Set the query duration up to one week, using a time unit of seconds, days or weeks. Valid values include any integer followed by a space and then S (seconds), D (days) or W (week). If no unit is specified, seconds is used.

Parameter	Description																																		
<b>barSizeSetting</b>	<p>Specifies the size of the bars that will be returned (within IB/TWS limits). Valid values include:</p> <table style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; width: 30%;">Bar Size</th><th style="text-align: left;">Parametric Value</th></tr> </thead> <tbody> <tr><td>1 sec</td><td>1</td></tr> <tr><td>5 secs</td><td>2</td></tr> <tr><td>15 secs</td><td>3</td></tr> <tr><td>30 secs</td><td>4</td></tr> <tr><td>1 min.</td><td>5</td></tr> <tr><td>2 mins</td><td>6</td></tr> <tr><td>3 mins</td><td>16</td></tr> <tr><td>5 mins.</td><td>7</td></tr> <tr><td>15 mins</td><td>8</td></tr> <tr><td>30 mins</td><td>9</td></tr> <tr><td>1 hour</td><td>10</td></tr> <tr><td>1 day</td><td>11</td></tr> <tr><td>1 week</td><td>12</td></tr> <tr><td>1 month</td><td>13</td></tr> <tr><td>3 months</td><td>14</td></tr> <tr><td>1 year</td><td>15</td></tr> </tbody> </table>	Bar Size	Parametric Value	1 sec	1	5 secs	2	15 secs	3	30 secs	4	1 min.	5	2 mins	6	3 mins	16	5 mins.	7	15 mins	8	30 mins	9	1 hour	10	1 day	11	1 week	12	1 month	13	3 months	14	1 year	15
Bar Size	Parametric Value																																		
1 sec	1																																		
5 secs	2																																		
15 secs	3																																		
30 secs	4																																		
1 min.	5																																		
2 mins	6																																		
3 mins	16																																		
5 mins.	7																																		
15 mins	8																																		
30 mins	9																																		
1 hour	10																																		
1 day	11																																		
1 week	12																																		
1 month	13																																		
3 months	14																																		
1 year	15																																		
<b>whatToShow</b>	<p>Determines the nature of data being extracted. Valid values include:</p> <ul style="list-style-type: none"> <li>• TRADES</li> <li>• MIDPOINT</li> <li>• BID</li> <li>• ASK</li> <li>• BID_ASK</li> <li>• HISTORICAL_VOLATILITY</li> <li>• OPTION_IMPLIED_VOLATILITY</li> <li>• OPTION_VOLUME</li> </ul>																																		
<b>useRTH</b>	<p>Determines whether to return all data available during the requested time span, or only data that falls within regular trading hours. Valid values include:</p> <ul style="list-style-type: none"> <li>• 0 - all data is returned even where the market in question was outside of its regular trading hours.</li> <li>• 1 - only data within the regular trading hours is returned, even if the requested time span falls partially or completely outside of the RTH.</li> </ul>																																		

Parameter	Description
<b>formatDate</b>	Determines the date format applied to returned bars. Valid values include: <ul style="list-style-type: none"> <li>• 1 - dates applying to bars returned in the format: yyyyymmdd{space}{space}hh:mm:dd</li> <li>• 2 - dates are returned as a long integer specifying the number of seconds since 1/1/1970 GMT.</li> </ul>

**Note:** For a information about historical data request limitations, see [Historical Data Limitations](#).

### exerciseOptions()

```
void exerciseOptions(long id, const Contract &contract, int
exerciseAction, int exerciseQuantity, const CString &account, int
override)
```

Parameter	Description
<b>id</b>	The ticker id. Must be a unique value.
<b>contract</b>	This structure contains a description of the contract for which market depth data is being requested.
<b>exerciseAction</b>	Specifies whether you want the option to lapse or be exercised. Values are 1 = exercise, 2 = lapse.
<b>exerciseQuantity</b>	The quantity you want to exercise.
<b>override</b>	Specifies whether your setting will override the system's natural action. For example, if your action is "exercise" and the option is not in-the-money, by natural action the option would not exercise. If you have override set to "yes" the natural action would be overridden and the out-of-the money option would be exercised. Values are: 0 = no, 1 = yes.

### reqScannerParameters()

Requests an XML string that describes all possible scanner queries.

```
void reqScannerParameters()
```

### reqScannerSubscription()

```
void reqScannerSubscription(int tickerId, const
ScannerSubscription&subscription)
```

Parameter	Description
<b>tickerId</b>	The ticker ID. Must be a unique value.
<b>ScannerSubscription</b>	This structure contains possible parameters used to filter results.

## **cancelHistoricalData()**

Used if an internet disconnect has occurred or the results of a query are otherwise delayed and the application is no longer interested in receiving the data.

```
void cancelHistoricalData (int tickerId)
```

Parameter	Description
<b>tickerId</b>	The ticker ID. Must be a unique value.

## **cancelScannerSubscription()**

```
void cancelScannerSubscription(int tickerId)
```

Parameter	Description
<b>tickerId</b>	The ticker ID. Must be a unique value.

## **reqRealTimeBars()**

Call the reqRealTimeBars() function to start receiving real time bar results through the realtimeBar() EWrapper function.

```
void reqRealTimeBars(int tickerId, Contract contract, int barSize,
String whatToShow, boolean useRTH)
```

Parameter	Description
<b>tickerId</b>	The Id for the request. Must be a unique value. When the data is received, it will be identified by this Id. This is also used when canceling the request.
<b>contract</b>	This object contains a description of the contract for which real time bars are being requested
<b>barSize</b>	Currently only 5 second bars are supported, if any other value is used, an exception will be thrown.
<b>whatToShow</b>	Determines the nature of the data extracted. Valid values include: <ul style="list-style-type: none"> <li>• TRADES</li> <li>• BID</li> <li>• ASK</li> <li>• MIDPOINT</li> </ul>
<b>useRTH</b>	Regular Trading Hours only. Valid values include: <ul style="list-style-type: none"> <li>• 0 = all data available during the time span requested is returned, including time intervals when the market in question was outside of regular trading hours.</li> <li>• 1 = only data within the regular trading hours for the product requested is returned, even if the time span falls partially or completely outside.</li> </ul>

**cancelRealTimeBars()**

Call the cancelRealTimeBars() function to stop receiving real time bar results.

```
void cancelRealTimeBars (int tickerId)
```

Parameter	Description
<b>tickerId</b>	The Id that was specified in the call to reqRealTimeBars().

**reqCurrentTime()**

Returns the current system time on the server side.

```
void reqCurrentTime()
```

**serverVersion()**

Returns the version of the TWS instance to which the API application is connected.

```
serverVersion()
```

**TwsConnectionTime()**

Returns the time the API application made a connection to TWS.

```
TwsConnectionTime()
```

**reqFundamentalData()**

Call this function to receive Reuters global fundamental data. There must be a subscription to Reuters Fundamental set up in Account Management before you can receive this data.

```
void reqFundamentalData(int reqId, const Contract &contract, String reportType)
```

Parameter	Description
<b>reqId</b>	The ID of the data request. Ensures that responses are matched to requests if several requests are in process.
<b>contract</b>	This structure contains a description of the contract for which Reuters Fundamental data is being requested.
<b>reportType</b>	Identifies the report type, which is one of the following: <ul style="list-style-type: none"> <li>• Estimates</li> <li>• Financial Statements</li> <li>• Summary</li> </ul>

## **cancelFundamentalData()**

Call this function to stop receiving Reuters global fundamental data.

```
void cancelFundamentalData(int reqId)
```

Parameter	Description
<b>reqId</b>	The ID of the data request.

## Class EWrapper Functions

The tables below define the class EWrapper functions you can use when connecting to TWS. These functions receive events from TWS. The list of functions includes:

<b>Connection and Server</b>	<b>Contract Details</b>
<a href="#">winError()</a>	<a href="#">contractDetails()</a>
<a href="#">error()</a>	<a href="#">contractDetailsEnd()</a>
<a href="#">connectionClosed()</a>	<a href="#">bondContractDetails()</a>
<a href="#">currentTime()</a>	
<b>Market Data</b>	<b>Executions</b>
<a href="#">tickPrice()</a>	<a href="#">execDetails()</a>
<a href="#">tickSize()</a>	<a href="#">execDetailsEnd()</a>
<a href="#">tickOptionComputation()</a>	
<a href="#">tickGeneric()</a>	<b>Market Depth</b>
<a href="#">tickString()</a>	<a href="#">updateMktDepth()</a>
<a href="#">tickEFP()</a>	<a href="#">updateMktDepthL2()</a>
<a href="#">tickSnapshotEnd()</a>	
<b>Orders</b>	<b>Financial Advisors</b>
<a href="#">orderStatus()</a>	<a href="#">managedAccounts()</a>
<a href="#">openOrder()</a>	<a href="#">receiveFA()</a>
<a href="#">nextValidId()</a>	
<b>Account and Portfolio</b>	<b>Historical Data</b>
<a href="#">updateAccountValue()</a>	<a href="#">historicalData()</a>
<a href="#">updatePortfolio()</a>	
<a href="#">updateAccountTime()</a>	<b>Market Scanners</b>
<b>News Bulletins</b>	<a href="#">scannerParameters()</a>
<a href="#">updateNewsBulletin()</a>	<a href="#">scannerData()</a>
	<a href="#">scannerDataEnd()</a>
	<b>Real Time Bars</b>
	<a href="#">realtimeBar()</a>
	<b>Fundamental Data</b>
	<a href="#">fundamentalData()</a>

## tickPrice()

This function is called when the market data changes. Prices are updated immediately with no delay.

```
virtual void tickPrice(TickerID id, TickType field, double price, int canAutoExecute)
```

Parameter	Description
<b>id</b>	The ticker ID that was specified previously in the call to reqMktData()
<b>tickType</b>	Specifies the type of price. Possible values are: <ul style="list-style-type: none"> <li>• 1 = bid</li> <li>• 2 = ask</li> <li>• 4 = last</li> <li>• 6 = high</li> <li>• 7 = low</li> <li>• 9 = close</li> </ul>
<b>price</b>	- could be the bid, ask, last price, daily high, daily low or last day close, depending on tickType value.
<b>canAutoExecute</b>	Specifies whether the price tick is available for automatic execution. Possible values are: <ul style="list-style-type: none"> <li>• 0 = not eligible for automatic execution</li> <li>• 1 = eligible for automatic execution</li> </ul>

## tickSize()

This function is called when the market data changes. Sizes are updated immediately with no delay.

```
virtual void tickSize(TickerID id, TickType field, int size)
```

Parameter	Description
<b>id</b>	The ticker ID that was specified previously in the call to reqMktData()
<b>tickType</b>	Specifies the type of size. Possible values are: <ul style="list-style-type: none"> <li>• 0 = bid size</li> <li>• 3 = ask size</li> <li>• 5 = last size</li> <li>• 8 = volume</li> </ul>
<b>size</b>	Could be the bid size, ask size, last size or trading volume, depending on the tickType value.

## tickOptionComputation()

This function is called when the market in an option or its underlier moves. TWS's option model volatilities, prices, and deltas, along with the present value of dividends expected on that options underlier are received.

```
virtual void tickOptionComputation(TickerID tickerId, TickType tickType,
double impliedVol, double delta, double modelPrice, double pvDividend)
```

Parameter	Description
<b>id</b>	The ticker ID that was specified previously in the call to reqMktData().
<b>tickType</b>	Specifies the type of tick. Possible values are: <ul style="list-style-type: none"> <li>• 10 = Bid</li> <li>• 11 = Ask</li> <li>• 12 = Last</li> </ul>
<b>impliedVol</b>	The implied volatility calculated by the TWS option modeler, using the specified ticktype value.
<b>delta</b>	The option delta calculated by the TWS option modeler.
<b>modelPrice</b>	The model price.
<b>pvDividend</b>	The present value of dividends expected on the options underlying instrument.

## tickGeneric()

This function is called when the market data changes. Values are updated immediately with no delay.

```
virtual void tickGeneric(TickerId tickerId, TickType tickType, double
value)
```

Parameter	Description
<b>tickerId</b>	The ticker Id that was specified previously in the call to reqMktData().
<b>tickType</b>	Specifies the type of price. Pass the field value into TickType.getField(int tickType) to retrieve the field description. For example, a field value of 46 will map to shortable, etc.
<b>value</b>	The value of the specified field.

## tickString()

This function is called when the market data changes. Values are updated immediately with no delay.

```
virtual void tickString(TickerId tickerId, TickType tickType, const
CString& value)
```

Parameter	Description
<b>tickerId</b>	The ticker Id that was specified previously in the call to reqMktData().
<b>field</b>	Specifies the type of price. Pass the field value into TickType.getField(int tickType) to retrieve the field description. For example, a field value of 45 will map to lastTimestamp, etc.
<b>value</b>	The value of the specified field.

## tickEFP()

This function is called when the market data changes. Values are updated immediately with no delay.

```
virtual void tickEFP(TickerId tickerId, TickType tickType, double
basisPoints, const CString& formattedBasisPoints, double totalDividends,
int holdDays, const CString& futureExpiry, double dividendImpact, double
dividendsToExpiry)
```

Parameter	Description
<b>tickerId</b>	The ticker Id that was specified previously in the call to reqMktData()
<b>field</b>	Specifies the type of price. Pass the field value into TickType.getField(int tickType) to retrieve the field description. For example, a field value of 38 will map to bidEFP, etc.
<b>basisPoints</b>	Annualized basis points, which is representative of the financing rate that can be directly compared to broker rates.
<b>formattedBasisPoint s</b>	Annualized basis points as a formatted string that depicts them in percentage form.
<b>impliedFuture</b>	Implied futures price.
<b>holdDays</b>	Number of hold days until the expiry of the EFP.
<b>futureExpiry</b>	Expiration date of the single stock future.
<b>dividendImpact</b>	The dividend impact upon the annualized basis points interest rate.
<b>dividendsToExpiry</b>	The dividends expected until the expiration of the single stock future.

## tickSnapshotEnd()

This is called when a snapshot market data subscription has been fully handled and there is nothing more to wait for. This also covers the timeout case.

```
virtual void tickSnapshotEnd(int reqId)
```

Parameter	Description
<b>reqID</b>	Id of the data request.

## orderStatus()

This event is called whenever the status of an order changes. It is also fired after reconnecting to TWS if the client has any open orders.

```
virtual void orderStatus(OrderId id, const CString &status, int filled,
int remaining, double avgFillPrice, int permId, int parentId, double
lastFillPrice, int clientId, const CString& whyHeld)
```

Parameter	Description
<b>id</b>	The order ID that was specified previously in the call to placeOrder()
<b>status</b>	<p>The order status. Possible values include:</p> <ul style="list-style-type: none"> <li>PendingSubmit - indicates that you have transmitted the order, but have not yet received confirmation that it has been accepted by the order destination. <b>NOTE:</b> This order status is not sent by TWS and should be explicitly set by the API developer when an order is submitted.</li> <li>PendingCancel - indicates that you have sent a request to cancel the order but have not yet received cancel confirmation from the order destination. At this point, your order is not confirmed canceled. You may still receive an execution while your cancellation request is pending. <b>NOTE:</b> This order status is not sent by TWS and should be explicitly set by the API developer when an order is canceled.</li> <li>PreSubmitted - indicates that a simulated order type has been accepted by the IB system and that this order has yet to be elected. The order is held in the IB system until the election criteria are met. At that time the order is transmitted to the order destination as specified.</li> <li>Submitted - indicates that your order has been accepted at the order destination and is working.</li> <li>Cancelled - indicates that the balance of your order has been confirmed canceled by the IB system. This could occur unexpectedly when IB or the destination has rejected your order.</li> <li>Filled - indicates that the order has been completely filled.</li> <li>Inactive - indicates that the order has been accepted by the system (simulated orders) or an exchange (native orders) but that currently the order is inactive due to system, exchange or other issues.</li> </ul>
<b>filled</b>	Specifies the number of shares that have been executed.
<b>remaining</b>	Specifies the number of shares still outstanding.

Parameter	Description
<b>avgFillPrice</b>	The average price of the shares that have been executed. This parameter is valid only if the <b>filled</b> parameter value is greater than zero. Otherwise, the price parameter will be zero.
<b>permId</b>	The TWS id used to identify orders. Remains the same over TWS sessions.
<b>parentId</b>	The order ID of the parent order, used for bracket and auto trailing stop orders.
<b>lastFilledPrice</b>	The last price of the shares that have been executed. This parameter is valid only if the filled parameter value is greater than zero. Otherwise, the price parameter will be zero.
<b>clientId</b>	The ID of the client (or TWS) that placed the order. Note that TWS orders have a fixed clientId and orderId of 0 that distinguishes them from API orders.
<b>whyHeld</b>	This field is used to identify an order held when TWS is trying to locate shares for a short sell. The value used to indicate this is 'locate'.

## error()

This event is called when there is an error with the communication or when TWS wants to send a message to the client.

```
virtual void error(const int id, const int errorCode, const CString
errorString)
```

Parameter	Description
<b>id</b>	This is the orderId or tickerId of the request that generated the error.
<b>errorCode</b>	Error codes are documented in the <a href="#">API Message Codes</a> topic.
<b>errorString</b>	This is the textual description of the error, also documented in the Error Codes topic.

## winError()

This event is called when there is an error on the client side.

```
virtual void winError(const CString &str, int lastError)
```

Parameter	Description
<b>str</b>	This is the error message text.
<b>lastError</b>	The error code returned by GetLastError().

## **connectionClosed()**

This function is called when TWS closes the sockets connection with the ActiveX control, or when TWS is shut down.

```
virtual void connectionClosed()
```

## **managedAccounts()**

This function is called when a successful connection is made to a Financial Advisor account. It is also called when the reqManagedAccts() function is invoked.

```
virtual void managedAccounts(const CString& accountsList)
```

Parameter	Description
<b>accountsList</b>	The comma delimited list of FA managed accounts.

## **openOrder()**

This function is called to feed in open orders.

```
virtual void openOrder(OrderId orderId, const Contract &contract, const Order &order, const OrderState &orderState)
```

For more information, see [Extended Order Attributes](#).

Parameter	Description
<b>orderId</b>	The order ID assigned by TWS. Use to cancel or update the order.
<b>contract</b>	The Contract class attributes describe the contract.
<b>order</b>	The Order class gives the details of the open order.
<b>orderState</b>	The orderState class includes attributes used for both pre and post trade margin and commission data.

**updateAccountValue()**

This function is called only when ReqAccountUpdates on EClientSocket object has been called.

```
virtual void updateAccountValue(const CString& key, const CString&
value, const CString& currency, const CString& accountName)
```

Parameter	Description
<b>key</b>	- A string that indicates one type of account value. Below is a set of keys sent by TWS. <ul style="list-style-type: none"> <li>• CashBalance - Account cash balance</li> <li>• Currency - Currency string</li> <li>• DayTradesRemaining - Number of day trades left</li> <li>• EquityWithLoanValue - Equity with Loan Value</li> <li>• InitMarginReq - Current initial margin requirement</li> <li>• LongOptionValue - Long option value</li> <li>• MaintMarginReq - Current maintenance margin</li> <li>• NetLiquidation - Net liquidation value</li> <li>• OptionMarketValue - Option market value</li> <li>• ShortOptionValue - Short option value</li> <li>• StockMarketValue - Stock market value</li> <li>• UnalteredInitMarginReq - Overnight initial margin requirement</li> <li>• UnalteredMaintMarginReq - Overnight maintenance margin requirement</li> </ul>
<b>value</b>	The value associated with the key.
<b>currency</b>	Defines the currency type, in case the value is a currency type.
<b>account</b>	States the account to which the message applies. Useful for Financial Advisor sub-account messages.

## updatePortfolio()

This function is called only when reqAccountUpdates on EClientSocket object has been called.

```
virtual void updatePortfolio(const Contract& contract, int position,
double marketPrice, double marketValue, double averageCost, double
unrealizedPNL, double realizedPNL, const CString& accountName)
```

Parameter	Description
<b>contract</b>	This structure contains a description of the contract which is being traded. The exchange field in a contract is not set for portfolio update.
<b>position</b>	This integer indicates the position on the contract. If the position is 0, it means the position has just cleared.
<b>marketPrice</b>	Unit price of the instrument.
<b>marketValue</b>	The total market value of the instrument.
<b>averageCost</b>	The average cost per share is calculated by dividing your cost (execution price + commission) by the quantity of your position.
<b>unrealizedPNL</b>	The difference between the current market value of your open positions and the average cost, or Value - Average Cost.
<b>realizedPNL</b>	Shows your profit on closed positions, which is the difference between your entry execution cost (execution price + commissions to open the position) and exit execution cost ((execution price + commissions to close the position))
<b>accountName</b>	States the account to which the message applies. Useful for Financial Advisor sub-account messages.

## updateAccountTime()

This function is called only when reqAccountUpdates on EClientSocket object has been called.

```
virtual void updateAccountTime(const CString& timeStamp)
```

Parameter	Description
<b>timeStamp</b>	This indicates the last update time of the account information.

## nextValidId()

This function is called after a successful connection to TWS.

```
virtual void nextValidId(OrderID orderId)
```

Parameter	Description
<b>orderId</b>	The next available order ID received from TWS upon connection. Increment all successive orders by one based on this ID.

## contractDetails()

This function is called only when reqContractDetails function on the EClientSocket object has been called.

```
virtual void contractDetails(const ContractDetails &contractDetails)
```

Parameter	Description
<b>reqId</b>	The ID of the data request. Ensures that responses are matched to requests if several requests are in process.
<b>contractDetails</b>	This structure contains a full description of the contract being looked up.

## contractDetailsEnd()

This function is called once all contract details for a given request are received. This helps to define the end of an option chain.

```
void contractDetailsEnd(int reqId)
```

Parameter	Description
<b>reqID</b>	The ID of the data request.

## execDetails()

This event is fired when the reqExecutions() functions is invoked, or when an order is filled.

```
virtual void execDetails( OrderId orderId, const Contract& contract,
const Execution& execution, long liquidation)
```

Parameter	Description
<b>id</b>	The order ID that was specified previously in the call to placeOrder().
<b>contract</b>	This structure contains a full description of the contract that was executed.
<b>execution</b>	This structure contains addition order execution details.

## execDetailsEnd()

This function is called once all executions have been sent to a client in response to reqExecutions().

```
virtual void execDetailsEnd(int reqId)
```

Parameter	Description
<b>reqID</b>	The Id of the data request.

## updateMktDepth()

This function is called when the market depth changes.

```
virtual void updateMktDepth(TickerId id, int position, int operation,
int side, double price, int size)
```

Parameter	Description
<b>id</b>	The ticker ID that was specified previously in the call to reqMktDepth()
<b>position</b>	Specifies the row id of this market depth entry.
<b>operation</b>	Identifies the how this order should be applied to the market depth. Valid values are: <ul style="list-style-type: none"> <li>• 0 = insert (insert this new order into the row identified by 'position')</li> <li>• 1 = update (update the existing order in the row identified by 'position')</li> <li>• 2 = delete (delete the existing order at the row identified by 'position')</li> </ul>
<b>side</b>	Identifies the side of the book that this order belongs to. Valid values are: <ul style="list-style-type: none"> <li>• 0 = ask</li> <li>• 1 = bid</li> </ul>
<b>price</b>	The order price.
<b>size</b>	The order size.

## updateMktDepthL2()

This function is called when the Level II market depth changes.

```
virtual void updateMktDepthL2(TickerId id, int position, CString
marketMaker, int operation, int side, double price, int size)
```

Parameter	Description
<b>id</b>	The ticker ID that was specified previously in the call to reqMktDepth()
<b>position</b>	Specifies the row id of this market depth entry.
<b>marketMaker</b>	Specifies the exchange hosting this order.

Parameter	Description
<b>operation</b>	Identifies the how this order should be applied to the market depth. Valid values are:· <ul style="list-style-type: none"> <li>• 0 = insert (insert this new order into the row identified by 'position')·</li> <li>• 1 = update (update the existing order in the row identified by 'position')·</li> <li>• 2 = delete (delete the existing order at the row identified by 'position')</li> </ul>
<b>side</b>	Identifies the side of the book that this order belongs to. Valid values are: <ul style="list-style-type: none"> <li>• 0 = ask</li> <li>• 1 = bid</li> </ul>
<b>price</b>	The order price.
<b>size</b>	The order size.

### **updateNewsBulletin()**

This event is triggered for each new bulletin if the client has subscribed (i.e. by calling the `reqNewsBulletins()` function.

```
virtual void updateNewsBulletin(int msgId, int msgType, const CString&
message, const CString& origExchange
```

Parameter	Description
<b>msgId</b>	The bulletin ID, incrementing for each new bulletin.
<b>msgType</b>	Specifies the type of bulletin. Valid values include: <ul style="list-style-type: none"> <li>• 1 = Regular news bulletin</li> <li>• 2 = Exchange no longer available for trading</li> <li>• 3 = Exchange is available for trading</li> </ul>
<b>message</b>	The bulletin's message text.
<b>origExchange</b>	The exchange from which this message originated.

## receiveFA()

This event receives previously requested FA configuration information from TWS.

```
virtual receiveFA(long faDataType, string XML)
```

Parameter	Description
<b>faDataType</b>	Specifies the type of Financial Advisor configuration data being received from TWS. Valid values include: <ul style="list-style-type: none"> <li>• 1 = GROUPS</li> <li>• 2 = PROFILE</li> <li>• 3 =ACCOUNT ALIASES</li> </ul>
<b>XML</b>	The XML string containing the previously requested FA configuration information.

## bondContractDetails()

This function is called only when reqContractDetails function on the EClientSocket object has been called for bonds.

```
virtual void bondContractDetails(const contractDetails&contractDetails)
```

Parameter	Description
<b>reqId</b>	The ID of the data request.
<b>contractDetails</b>	This structure contains a description of the contract which is being traded. The exchange field in a contract is not set for portfolio update.

## historicalData()

This function receives the requested historical data results.

```
virtual void historicalData(TickerId reqId, const CString& date, double open, double high, double low, double close, int volume, double WAP, int hasGaps)
```

Parameter	Description
<b>reqId</b>	The ticker ID of the request to which this bar is responding.
<b>date</b>	The date-time stamp of the start of the bar. The format is determined by the reqHistoricalData() formatDate parameter.
<b>open</b>	The bar opening price.
<b>high</b>	The high price during the time covered by the bar.
<b>low</b>	The low price during the time covered by the bar.
<b>close</b>	The bar closing price.
<b>volume</b>	The volume during the time covered by the bar.
<b>WAP</b>	The weighted average price during the time covered by the bar.

Parameter	Description
<b>count</b>	When TRADES historical data is returned, represents the number of trades that occurred during the time period the bar covers.
<b>hasGaps</b>	Reports whether or not there are gaps in the data.

### scannerParameters()

This function receives an XML document that describes the valid parameters that a scanner subscription can have.

```
virtual void scannerParameters(String xml)
```

Parameter	Description
<b>xml</b>	An XML document that describes the valid parameters for queries.

### scannerData()

This function receives the requested market scanner data results.

```
virtual void scannerData(int reqId, int rank, const ContractDetails
&contractDetails, String distance, String benchmark, String projection)
```

Parameter	Description
<b>reqId</b>	The ticker ID of the request to which this row is responding.
<b>rank</b>	The ranking within the response of this bar.
<b>contractDetails</b>	This object contains a full description of the contract.
<b>distance</b>	Varies based on query.
<b>benchmark</b>	Varies based on query.
<b>projection</b>	Varies based on query.
<b>legsStr</b>	Describes combo legs when scan is returning EFP.

### scannerDataEnd()

This function is called when the snapshot is received and marks the end of one scan.

```
virtual void scannerDataEnd(int reqId)
```

Parameter	Description
<b>reqId</b>	The ID of the market scanner request being closed by this parameter.

## realtimeBar()

This function receives the real-time bars data results.

```
virtual void realtimeBar(TickerId reqId, long time, double open, double
high, double low, double close, long volume, double wap, int count)
```

Parameter	Description
<b>reqId</b>	The ticker Id of the request to which this bar is responding.
<b>time</b>	The date-time stamp of the start of the bar. The format is determined by the reqHistoricalData() formatDate parameter.
<b>open</b>	The bar opening price.
<b>high</b>	The high price during the time covered by the bar.
<b>low</b>	The low price during the time covered by the bar.
<b>close</b>	The bar closing price.
<b>volume</b>	The volume during the time covered by the bar.
<b>wap</b>	The weighted average price during the time covered by the bar.
<b>count</b>	When TRADES historical data is returned, represents the number of trades that occurred during the time period the bar covers.

## currentTime()

This function receives the current system time on the server side.

```
virtual void currentTime(long time)
```

Parameter	Description
<b>time</b>	The current system time on the server side.

## fundamentalData()

This function is called to receive Reuters global fundamental market data. There must be a subscription to Reuters Fundamental set up in Account Management before you can receive this data.

```
virtual void fundamentalData(int reqId, string data)
```

Parameter	Description
<b>reqId</b>	The ID of the data request.
<b>data</b>	One of three XML reports: <ul style="list-style-type: none"> <li>• Estimates (estimates)</li> <li>• Financial statements (finstat)</li> <li>• Summary (snapshot)</li> </ul>

## SocketClient Properties

The tables below define properties for the Execution, Contract and Order classes, and classes that are closely related to them.

- [\*\*Execution\*\*](#)
- [\*\*ExecutionFilter\*\*](#)
- [\*\*Contract\*\*](#)
- [\*\*ContractDetails\*\*](#)
- [\*\*ComboLeg\*\*](#)
- [\*\*Order\*\*](#)
- [\*\*OrderState\*\*](#)
- [\*\*ScannerSubscription\*\*](#)
- [\*\*UnderComp\*\*](#)

## Execution

Property	Description
CString execId	Unique order execution id.
CString time	The order execution time.
CString acctNumber	The customer account number.
CString exchange	Exchange that executed the order.
CString side	Specifies if the transaction was a sale or a purchase. Valid values are: <ul style="list-style-type: none"> <li>• BOT</li> <li>• SLD</li> </ul>
int shares	The number of shares filled.
double price	The order execution price.
int permId	The TWS id used to identify orders, remains the same over TWS sessions.
long clientId	The id of the client that placed the order. <b>Note:</b> TWS orders have a fixed client id of 0.
long orderId	The order id. <b>Note:</b> TWS orders have a fixed order id of 0.
int liquidation	Identifies the position as one to be liquidated last should the need arise.
int cumQty	Cumulative quantity. Used in regular trades, combo trades and legs of the combo.
double avgPrice	Average price. Used in regular trades, combo trades and legs of the combo.

## ExecutionFilter

Property	Description
long clientId	Filter the results of the reqExecutions() function based on the clientId.
CString acctCode	Filter the results of the reqExecutions() function based on an account code. Note: this is only relevant for FA managed accounts.
CString time	Filter the results of the reqExecutions() function based on execution reports received after the specified time. The format for timeFilter is "yyyymmdd-hh:mm:ss"
CString symbol	Filter the results of the reqExecutions() function based on the order symbol.
CString secType	Filter the results of the reqExecutions() function based on the order security type. Note: Refer to the Contract struct for the list of valid security types.

<b>Property</b>	<b>Description</b>
CString exchange	Filter the results of the reqExecutions() function based on the order exchange.
CString side	Filter the results of the reqExecutions() function based on the order action. Note: Refer to the Order struct for the list of valid order actions.

## Contract

Property	Description
CString symbol	This is the symbol of the underlying asset.
CString secType	<p>This is the security type. Valid values are</p> <ul style="list-style-type: none"> <li>• STK</li> <li>• OPT</li> <li>• FUT</li> <li>• IND</li> <li>• FOP</li> <li>• CASH</li> <li>• BAG</li> </ul>
CString expiry	The expiration date. Use the format YYYYMM.
double strike	The strike price.
CString right	Specifies a Put or Call. Valid values are: P, PUT, C, CALL.
CString multiplier	Allows you to specify a futures or options multiplier. This is only necessary when multiple possibilities exist.;
CString exchange	The order destination, such as Smart.
CString currency	Specifies the currency. Ambiguities may require that this field be specified, for example, when SMART is the exchange and IBM is being requested (IBM can trade in GBP or USD). Given the existence of this kind of ambiguity, it is a good idea to always specify the currency.
CString localSymbol	This is the local exchange symbol of the underlying asset.
vector<ComboLeg*>* comboLegs;	Dynamic memory structure used to store the leg definitions for this contract.
CString comboLegsDescrip	Description for combo legs.
CString primaryExchange	To clarify any ambiguity for Smart-routed contracts, include the primary exchange, along with the Smart designation, for the destination.
bool includeExpired	<p>If set to true, contract details requests and historical data queries can be performed pertaining to expired contracts.</p> <p><b>Note:</b> Historical data queries on expired contracts are limited to the last year of the contracts life, and are initially only supported for expired futures contracts.</p>
int conId	The unique contract identifier.

Property	Description
CString secIdType	Security identifier, when querying contract details or when placing orders. Supported identifiers are: <ul style="list-style-type: none"><li>• ISIN (Example: Apple: US0378331005)</li><li>• CUSIP (Example: Apple: 037833100)</li><li>• SEDOL (Consists of 6-AN + check digit. Example: BAE: 0263494)</li><li>• RIC (Consists of exchange-independent RIC Root and a suffix identifying the exchange. Example: AAPL.O for Apple on NASDAQ.)</li></ul>
CString secId	Unique identifier for the secIdType.

## ContractDetails

Property	Description
Contract summary	A contract structure.
CString marketName	The market name for this contract.
CString tradingClass	The trading class name for this contract.
double minTick	The minimum price tick.
CString orderTypes	The list of valid order type for this contract
CString validExchanges	The list of exchanges on which this contract is traded.
CString underConId	The underlying contract ID.
CString longName	The descriptive name of the asset.
CString cusip	For Bonds. The nine-character bond CUSIP or the 12-character SEDOL.
CString ratings	For Bonds. Identifies the credit rating of the issuer. A higher credit rating generally indicates a less risky investment. Bond ratings are from Moody's and S&P respectively.
CString descAppend	For Bonds. A description string containing further descriptive information about the bond.
CString bondType	For Bonds. The type of bond, such as "CORP."
CString couponType	For Bonds. The type of bond coupon, such as "FIXED."
bool callable	For Bonds. Values are True or False. If true, the bond can be called by the issuer under certain conditions.
bool putable	For Bonds. Values are True or False. If true, the bond can be sold back to the issuer under certain conditions.
double coupon	For Bonds. The interest rate used to calculate the amount you will receive in interest payments over the course of the year.
bool convertible	For Bonds. Values are True or False. If true, the bond can be converted to stock under certain conditions.
CString maturity	For Bonds. The date on which the issuer must repay the face value of the bond.
CString issueDate	For Bonds. The date the bond was issued.
CString nextOptionDate	For Bonds, relevant if the bond has embedded options.
CString nextOptionType	For Bonds, relevant if the bond has embedded options.
Bool nextOptionPartial	For Bonds, relevant if the bond has embedded options, i.e., is the next option full or partial?
CString notes	For Bonds, if populated for the bond in IB's database
long priceMagnifier	Allows execution and strike prices to be reported consistently with market data, historical data and the order price, i.e. Z on LIFFE is reported in index points and not GBP.
CString contractMonth	The contract month. Typically the contract month of the underlying for a futures contract.
CString industry	The industry classification of the underlying/product. For example, Financial.

Property	Description
CString category	The industry category of the underlying. For example, InvestmentSvc.
CString subcategory	The industry subcategory of the underlying. For example, Brokerage.
CString timeZoneId	The ID of the time zone for the trading hours of the product. For example, EST.
CString tradingHours	The trading hours of the product. For example, 20090507:0700-1830,1830-2330;20090508:CLOSED.
CString liquidHours	The liquid trading hours of the product. For example, 20090507:0930-1600;20090508:CLOSED.

## ComboLeg

Property	Description
long conId	The unique contract identifier specifying the security.
long ratio	Select the relative number of contracts for the leg you are constructing. To help determine the ratio for a specific combination order, refer to the Interactive Analytics section of the User's Guide.
CString action	The side (buy or sell) for the leg you are constructing.
CString exchange	The exchange to which the complete combination order will be routed.
long openClose	Specifies whether the order is an open or close order. Valid values are: <ul style="list-style-type: none"> <li>• Same - (0) same as the parent security. This is the only option for retail customers.</li> <li>• Open - (1) only valid for institutional customers.</li> <li>• Close - (2) only valid for institutional customers.</li> <li>• Unknown</li> </ul>
int shortSaleSlot	For institutional customers only. <ul style="list-style-type: none"> <li>• 0 - inapplicable (i.e. retail customer or not short leg)</li> <li>• 1 - clearing broker</li> <li>• 2 - third party. If this value is used, you must enter a designated location.</li> </ul>
CString designatedLocation	If shortSaleSlot == 2, the designatedLocation must be specified. Otherwise leave blank or orders will be rejected.

## Order

Property	Description
long orderId	The id for this order.
long clientId	The id of the client that placed this order.
long permId	The TWS id used to identify orders, remains the same over TWS sessions.
CString action	Identifies the side. Valid values are: BUY, SELL, SSHORT
long totalQuantity	The order quantity.
CString orderType	Identifies the order type. Valid values are: <ul style="list-style-type: none"> <li>• MKT</li> <li>• MKTCLS</li> <li>• LMT</li> <li>• LMTCLS</li> <li>• PEGMKT</li> <li>• SCALE</li> <li>• STP</li> <li>• STPLMT</li> <li>• TRAIL</li> <li>• REL</li> <li>• VWAP</li> <li>• TRAILLIMIT</li> </ul>
double lmtPrice	This is the LIMIT price, used for limit, stop-limit and relative orders. In all other cases specify zero. For relative orders with no limit price, also specify zero.
double auxPrice	This is the STOP price for stop-limit orders, and the offset amount for relative orders. In all other cases, specify zero.
CString.tif	The time in force. Valid values are: DAY, GTC, IOC, GTD.
CString ocaGroup	Identifies an OCA (one cancels all) group.
int ocaType	Tells how to handle remaining orders in an OCA group when one order or part of an order executes. Valid values include: <ul style="list-style-type: none"> <li>• 1 = Cancel all remaining orders with block</li> <li>• 2 = Remaining orders are proportionately reduced in size with block</li> <li>• 3 = Remaining orders are proportionately reduced in size with no block</li> </ul> If you use a value "with block" gives your order has overfill protection. This means that only one order in the group will be routed at a time to remove the possibility of an overfill.
CString account	The account. For institutional customers only.

Property	Description
CString openClose	For institutional customers only. Valid values are O, C.
int origin	The order origin. For institutional customers only. Valid values are 0 = customer, 1 = firm
CString orderRef	The order reference. For institutional customers only.
bool transmit	Specifies whether the order will be transmitted by TWS. If set to false, the order will be created at TWS but will not be sent.
long parentId	The order ID of the parent order, used for bracket and auto trailing stop orders.
bool blockOrder	If set to true, specifies that the order is an ISE Block order.
bool sweepToFill	If set to true, specifies that the order is a Sweep-to-Fill order.
int displaySize	The publicly disclosed order size, used when placing Iceberg orders.
int triggerfunction	Specifies how Simulated Stop, Stop-Limit and Trailing Stop orders are triggered. Valid values are: <ul style="list-style-type: none"> <li>• <b>0</b> - The default value. The "double bid/ask" function will be used for orders for OTC stocks and US options. All other orders will used the "last" function.</li> <li>• <b>1</b> - use "double bid/ask" function, where stop orders are triggered based on two consecutive bid or ask prices.</li> <li>• <b>2</b> - "last" function, where stop orders are triggered based on the last price.</li> <li>• <b>3</b> double last function.</li> <li>• <b>4</b> bid/ask function.</li> <li>• <b>7</b> last or bid/ask function.</li> <li>• <b>8</b> mid-point function.</li> </ul>
boolean outsideRth()	If set to true, allows orders to also trigger or fill outside of regular trading hours.
boolean hidden	If set to true, the order will not be visible when viewing the market depth. This option only applies to orders routed to the ISLAND exchange.
double discretionaryAmt	The amount off the limit price allowed for discretionary orders.
int shortSaleSlot	Valid values are 1 or 2.
CString designatedLocation	Used only when shortSaleSlot = 2.
CString goodAfterTime	The trade's "Good After Time," format "YYYYMMDD hh:mm:ss (optional time zone)" Use an empty String if not applicable.

Property	Description
CString goodTillDate	You must enter GTD as the time in force to use this string. The trade's "Good Till Date," format "YYYYMMDD hh:mm:ss (optional time zone)" Use an empty String if not applicable.
CString rule80A	Values include: <ul style="list-style-type: none"> <li>• Individual = 'I'</li> <li>• Agency = 'A',</li> <li>• AgentOtherMember = 'W'</li> <li>• IndividualPTIA = 'J'</li> <li>• AgencyPTIA = 'U'</li> <li>• AgentOtherMemberPTIA = 'M'</li> <li>• IndividualPT = 'K'</li> <li>• AgencyPT = 'Y'</li> <li>• AgentOtherMemberPT = 'N'</li> </ul>
CString settlingFirm	Institutional only.
CString clearingIntent	For IBExecution customers: Valid values are: IB, Away, and PTA (post trade allocation).
CString clearingAccount	For IBExecution customers: Specifies the true beneficiary of the order. This value is required for FUT/FOP orders for reporting to the exchange.
bool allOrNone	0 = no, 1 = yes
int minQty	Identifies a minimum quantity order type.
double percentOffset	The percent offset amount for relative orders.
bool eTradeOnly	Trade with electronic quotes. 0 = no, 1 = yes
bool firmQuoteOnly	Trade with firm quotes. 0 = no, 1 = yes
double nbboPriceCap	Maximum smart order distance from the NBBO.
int auctionStrategy	Values include: <ul style="list-style-type: none"> <li>• match = 1</li> <li>• improvement = 2</li> <li>• transparent = 3</li> </ul> For orders on BOX only.
double startingPrice	The auction starting price. For orders on BOX only.
double stockRefPrice	The stock reference price. The reference price is used for VOL orders to compute the limit price sent to an exchange (whether or not Continuous Update is selected), and for price range monitoring.
double delta	The stock delta. For orders on BOX only.

Property	Description
double stockRangeLower	The lower value for the acceptable underlying stock price range. For price improvement option orders on BOX and VOL orders with dynamic management.
double stockRangeUpper	The upper value for the acceptable underlying stock price range. For price improvement option orders on BOX and VOL orders with dynamic management.
CString faGroup	The Financial Advisor group the trade will be allocated to -- use an empty String if not applicable.
CString faProfile	The Financial Advisor allocation profile the trade will be allocated to -- use an empty String if not applicable.
CString fafunction	The Financial Advisor allocation function the trade will be allocated with -- use an empty String if not applicable.
CString faPercentage	The Financial Advisor percentage concerning the trade's allocation -- use an empty String if not applicable.
bool overridePercentageConstraints	Precautionary constraints are defined on the TWS Presets page, and help ensure that your price and size order values are reasonable. Orders sent from the API are also validated against these safety constraints, and may be rejected if any constraint is violated. To override validation, set this parameter's value to True.  Valid values include: <ul style="list-style-type: none"><li>• 0 = False</li><li>• 1 = True</li></ul>
double volatility	The option price in volatility, as calculated by TWS' Option Analytics. This value is expressed as a percent and is used to calculate the limit price sent to the exchange.
int volatilityType	Values include: <ul style="list-style-type: none"><li>• 1 = Daily volatility</li><li>• 2 = Annual volatility</li></ul>
CString deltaNeutralOrderType	VOL orders only. Enter an order type to instruct TWS to submit a delta neutral trade on full or partial execution of the VOL order. For no hedge delta order to be sent, specify NONE.
int deltaNeutralAuxPrice	VOL orders only. Use this field to enter a value if the value in the <i>deltaNeutralOrderType</i> field is an order type that requires an Aux price, such as a REL order.
bool continuousUpdate	VOL orders only. Specifies whether TWS will automatically update the limit price of the order as the underlying price moves.
int referencePriceType	VOL orders only. Specifies how you want TWS to calculate the limit price for options, and for stock range price monitoring.  Valid values include: <ul style="list-style-type: none"><li>• 1 = Average of NBBO</li><li>• 2 = NBB or the NBO depending on the action and right.</li></ul>

Property	Description
double trailStopPrice	For TRAILLIMIT orders only
int scaleInitLevelSize	For Scale orders: Defines the size of the first, or initial, order component.
int scaleSubsLevelSize	For Scale orders: Defines the order size of the subsequent scale order components. Used in conjunction with scaleInitLevelSize().
double scalePriceIncrement	For Scale orders: Defines the price increment between scale components. This field is required.
double basisPoints	For EFP orders only
int basisPointsType	For EFP orders only
bool whatIf	Use to request pre-trade commissions and margin information. If set to true, margin and commissions data is received back via the OrderState() object for the openOrder() callback.

## OrderState

Property	Description
CString status	Displays the order status.
CString initMargin	Shows the impact the order would have on your initial margin.
CString maintMargin	Shows the impact the order would have on your maintenance margin.
CString equityWithLoan	Shows the impact the order would have on your equity with loan value.
double commission	Shows the commission amount on the order.
double minCommission	Used in conjunction with the maxCommission field, this defines the lowest end of the possible range into which the actual order commission will fall.
double maxCommission	Used in conjunction with the minCommission field, this defines the highest end of the possible range into which the actual order commission will fall.
CString commissionCurrency	Shows the currency of the commission value.
CString warningText	Displays a warning message if warranted.

## ScannerSubscription

Property	Description
int numberOfRows	Defines the number of rows of data to return for a query.
CString instrument	Defines the instrument type for the scan.
CString locationCode	The location, currently the only valid location is US stocks.
CString scanCode	Can be left blank.
double abovePrice	Filter out contracts with a price lower than this value. Can be left blank.
double belowPrice	Filter out contracts with a price higher than this value. Can be left blank.
int aboveVolume	Filter out contracts with a volume lower than this value. Can be left blank.
double marketCapAbove	Filter out contracts with a market cap lower than this value. Can be left blank.
double marketCapBelow	Filter out contracts with a market cap above this value. Can be left blank.
CString moodyRatingAbove	Filter out contracts with a Moody rating below this value. Can be left blank.
CString moodyRatingBelow	Filter out contracts with a Moody rating above this value. Can be left blank.
CString spRatingAbove	Filter out contracts with an S&P rating below this value. Can be left blank.
CString spRatingBelow	Filter out contracts with an S&P rating above this value. Can be left blank.
CString maturityDateAbove	Filter out contracts with a maturity date earlier than this value. Can be left blank.
CString maturityDateBelow	Filter out contracts with a maturity date later than this value. Can be left blank.
double couponRateAbove	Filter out contracts with a coupon rate lower than this value. Can be left blank.
double couponRateBelow	Filter out contracts with a coupon rate higher than this value. Can be left blank.
CString excludeConvertible	Filter out convertible bonds. Can be left blank.
CString scannerSettingPairs	Can leave empty. For example, a pairing "Annual, true" used on the "top Option Implied Vol % Gainers" scan would return annualized volatilities.
int averageOptionVolumeAbove	Can leave empty.

Property	Description
CString stockTypeFilter	Values include: <ul style="list-style-type: none"><li>• ALL (excludes nothing)</li><li>• STOCK (excludes ETFs)</li><li>• ETF (includes ETFs)</li></ul>

## UnderComp

Attribute	Description
int conId	The unique contract identifier specifying the security. Used for Delta-Neutral Combo contracts.
double delta	The underlying stock or future delta. Used for Delta-Neutral Combo contracts.
double price	The price of the underlying. Used for Delta-Neutral Combo contracts.

## Placing a Combination Order

A combination order is a special type of order that is constructed of many separate legs but executed as a single transaction. Submit combo orders such as calendar spreads, conversions and straddles using the BAG security type (defined in the *Contract* object). The key to implementing a successful API combination order using the API is to knowing how to place the same order using Trader Workstation. If you are familiar with placing combination orders in TWS, then it will be easier to place the same order using the API, because the API only imitates the behavior of TWS.

### Example

In this example, a customer places a BUY order for a CLK9 futures contract and a SELL order for a CLM9 futures contract. In this procedure, the customer must invoke `reqContractDetails()` to obtain the `conId` for both CLK9 and CLM9 contracts.

#### **Leg 1: Buy 1 CLK9 futures contract**

#### **Leg 2: Sell 1 CLM9 futures contract**

Here is a summary of the steps required to place a combo order using the API:

- Obtain the contract id (`conId`) for each leg. Get this number by invoking the [reqContractDetails\(\)](#) method.
- Include each leg on the [ComboLeg](#) object by populating the related fields.
- Implement the [placeOrder\(\)](#) method with the [Contract](#) and [Order](#) socket client properties.

### To place this combo order

- 1 Get the Contract IDs for both leg definitions. Request 1 is assigned to CLK9 and Request 2 is assigned to CLM9.

```
con1.localSymbol = "CLK9";
con1.secType = "FUT";
con1.exchange = "NYMEX";
con1.currency = "USD";

m_client->reqContractDetails(1, con1->getContract()); // request
    1

con2.m_localSymbol = "CLM9";
con2.m_secType = "FUT";
con2.m_exchange = "NYMEX";
con2.m_currency = "USD";

m_client->reqContractDetails(2, con2->getContract()); // request
    2
```

The `conId` values are delivered by the following event. If `reqId` is equal to 1, then the `conid` is for the CLK9 contract. If `reqId` is equal to 2, then the `conId` is for CLM9.

```

::contractDetails( int reqId, const ContractDetails
&contractDetails)
{

// to obtain conId for CLK9
if (reqId == 1)

...
// to obtain conid for CLM9
if (reqId == 2)
...
}

```

- 2** Assign all the related values for combo orders and combine them:

```

leg1.conId = Leg1_conId;
leg1.ratio = 1;
leg1.action = "BUY";
leg1.exchange = "NYMEX";
leg1.openClose = 0;
leg1.shortSaleSlot = 0;
leg1.designatedLocation = "";

leg2.conId = Leg2_conId;
leg2.ratio = 1;
leg2.action = "SELL";
leg2.exchange = "NYMEX";
leg2.openClose = 0;
leg2.shortSaleSlot = 0;
leg2.designatedLocation = "";

```

- 3** Invoke the placeOrder() method with the appropriate contract and order objects. As shown below, it includes the addAllLegs declaration in the contract object.

```

contract.symbol = "USD"; // arbitrary value only combo orders
contract.secType = "BAG"; // BAG is the security type for COMBO
order
contract.exchange = "NYMEX";
contract.currency = "USD";
contract.comboLegs = addAllLegs; //including combo order in
contract object

order.m_action = "BUY";
order.m_totalQuantity = 1;
order.m_orderType = "MKT";

m_client->placeOrder(Orderid, contract->getContract(),
order->getOrder());

```

**Note:** For more information on combination orders, see the TWS Users Guide topics [About Combination Orders](#) and [Notes on Combination Orders](#).



# Java

This chapter describes the Java API, including the following topics:

- [\*\*Linking to TWS using the Java API\*\*](#)
- [\*\*Running the Java Test Client Sample Program\*\*](#)
- [\*\*Java Test Client Overview\*\*](#)
- [\*\*Java API Overview\*\*](#)
- [\*\*Java EClientSocket Methods\*\*](#)
- [\*\*Java EWrapper Methods\*\*](#)
- [\*\*Java SocketClient Properties\*\*](#)
- [\*\*Placing a Combination Order\*\*](#)

## Linking to TWS using the Java API

### To link to TWS using the Java API

- 1** Import com.ib.client.\* into your source code file.
- 2** Implement the EWrapper interface. This class will receive messages from the socket.
- 3** Override the following methods:

Ewrapper Method	Description
tickPrice()	Handles market data.
tickSize()	
tickOptionComputation()	
tickGeneric()	
tickString()	
tickEFP()	
orderStatus()	Receives order status.
openOrder()	Receives open orders.
error()	Receives error information.
connectionClosed()	Notifies when TWS terminates the connection.
updateAccountValue()	Receives current account values.
updateAccountTime()	Receives the last time account information was updated.
updatePortfolio()	Receives current portfolio information.
nextValidId()	Receives the next valid order ID upon connection.
contractDetails()	Receives contract information.
contractDetailsEnd()	Identifies the end of a given contract details request.
bondContractDetails()	Receives bond contract information.
execDetails()	Receives execution report information.
updateMktDepth()	Receives market depth information.
updateMktDepthL2()	Receives Level II market depth information.
updateNewsBulletin()	Receives IB news bulletins.
managedAccounts()	Receives a list of Financial Advisor (FA) managed accounts.
receiveFA()	Receives FA configuration information.
historicalData()	Receives historical data results.

Ewrapper Method	Description
scannerParameters()	Receives an XML document that describes the valid parameters of a scanner subscription.
scannerData()	Receives market scanner results.
realTimeBar()	Receives real-time bars.
currentTime()	Receives the current system time on the server.
fundamentalData()	Receives Reuters global fundamental market data.

**4** Instantiate the EClientSocket class. This object will be used to send messages to TWS.

**5** Call the following methods:

EClientSocket Method	Description
eConnect()	Connects to TWS.
eDisconnect()	Disconnects from TWS.
reqMktData()	Requests market data.
cancelMktData()	Cancels market data.
reqMktDepth()	Requests market depth.
cancelMktDepth()	Cancels market depth.
reqContractDetails()	Requests contract details.
placeOrder()	Places an order.
cancelOrder()	Cancels an order.
reqAccountUpdates()	Requests account values, portfolio, and account update time information.
reqExecutions()	Requests a list of the day's execution reports.
reqOpenOrders()	Requests a list of current open orders for the requesting client and associates TWS open orders with the client. The association only occurs if the requesting client has a Client ID of 0.
reqAllOpenOrders()	Requests a list of all open orders.
reqAutoOpenOrders()	Automatically associates a new TWS with the client. The association only occurs if the requesting client has a Client ID of 0.
reqNewsBulletin()	Requests IB news bulletins.
cancelNewsBulletins()	Cancels IB news bulletins.
setServerLogLevel()	Sets the level of API request and processing logging.
reqManagedAccts()	Requests a list of Financial Advisor (FA) managed account codes.

<b>EClientSocket Method</b>	<b>Description</b>
requestFA()	Requests FA configuration information from TWS.
replaceFA()	Modifies FA configuration information from the API.
reqScannerParameters()	Requests an XML document that describes the valid parameters of a scanner subscription.
reqScannerSubscription()	Requests market scanner results.
cancelScannerSubscription()	Cancels a scanner subscription.
reqHistoricalData()	Requests historical data.
cancelHistoricalData()	Cancels historical data.
reqRealTimeBars()	Requests real-time bars.
cancelRealTimeBars()	Cancels real-time bars.
exerciseOptions()	Exercises options.
reqCurrentTime()	Requests the current server time.
serverVersion()	Returns the version of the TWS instance to which the API application is connected.
TwsConnectionTime()	Returns the time the API application made a connection to TWS.
reqFundamentalData()	Requests Reuters global fundamental data. There must be a subscription to Reuters Fundamental set up in Account Management before you can receive this data.
cancelFundamentalData()	Cancels Reuters global fundamental data.

## Running the Java Test Client Sample Program

You can access TWS through a Java application using the socket client component. Before you can connect to TWS, you must:

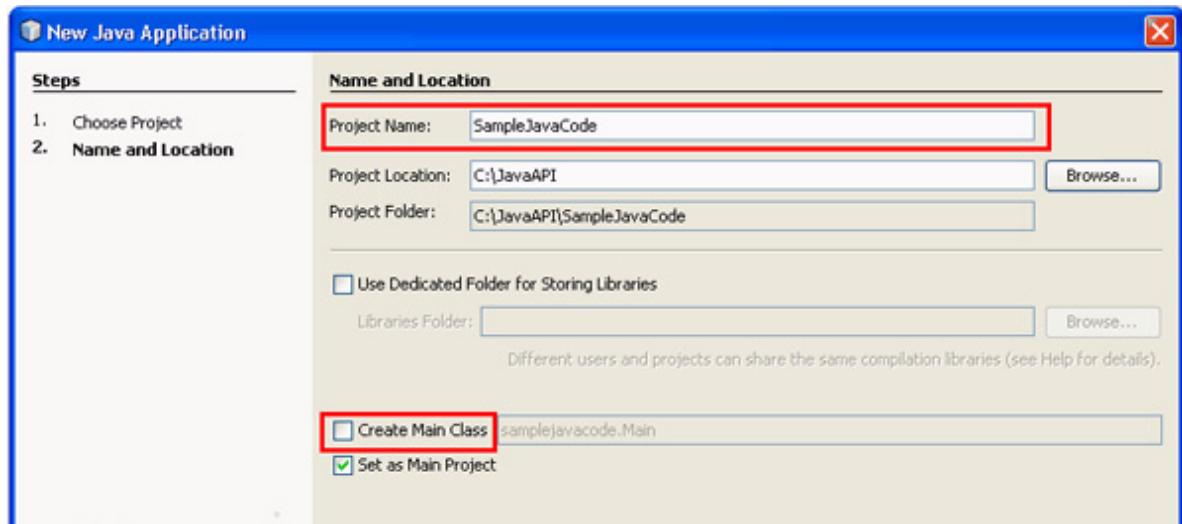
- Install the socket client component and sample programs.
- [Configure TWS to support the API components.](#)
- Have NetBeans with the J2SE development kit installed on your PC. You can download the bundle at the [Sun Website](#), or NetBeans at [netbeans.org](#).

### To run the Java Test Client sample program on Windows

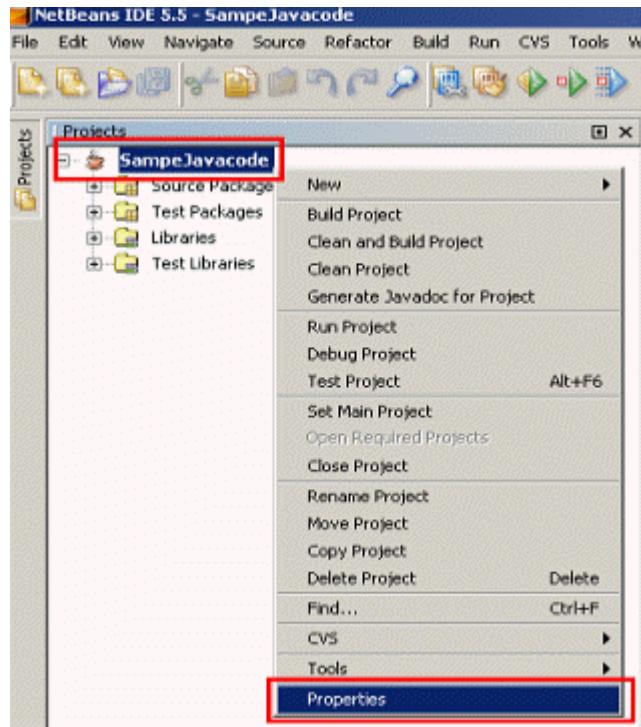
- 1 From Windows Explorer, navigate to the Jts:\Java folder.
- 2 Run the file *run.bat*.

### To run the Java Test Client sample program from a new project in NetBeans

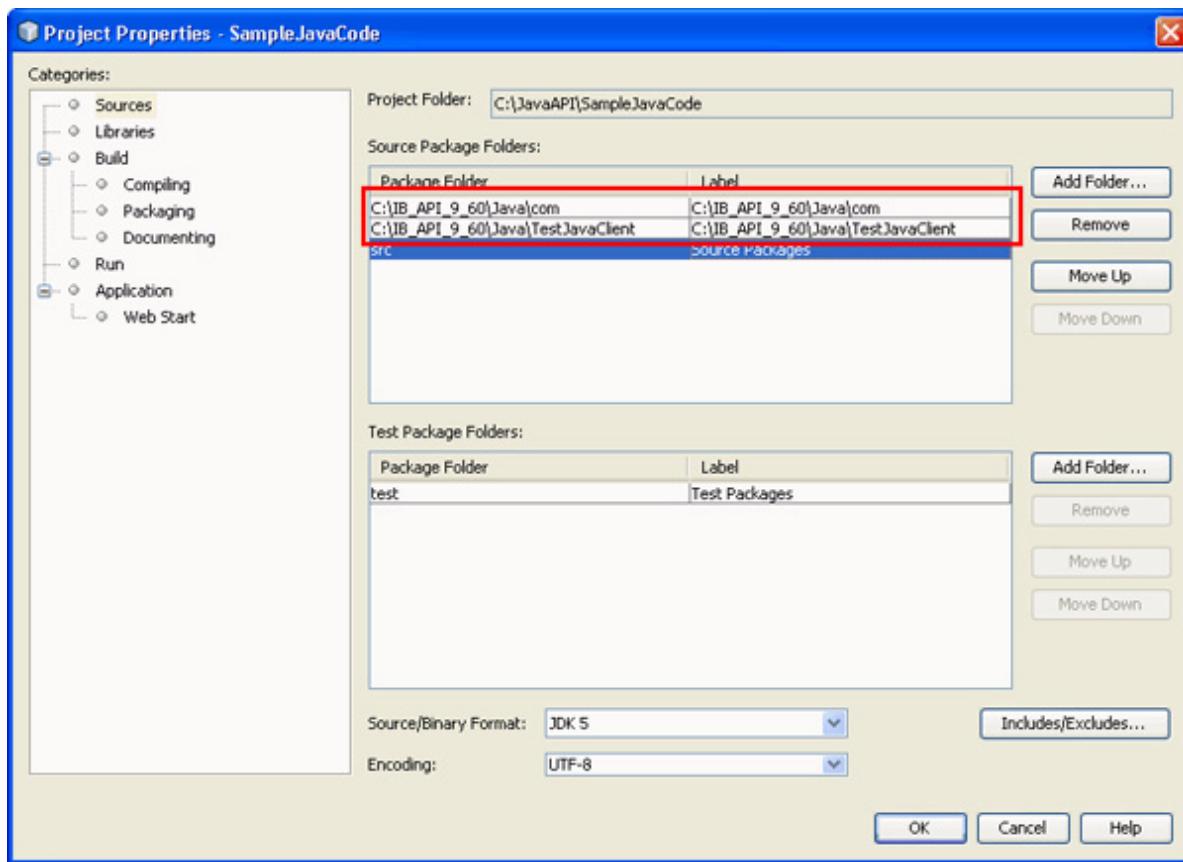
- 1 Open NetBeans and click *New Project* to start the wizard.
- 2 In the Projects area, select *Java Application* and click **Next**.
- 3 In the **New Java Application** window, name your project, choose a location, and uncheck the check box for *Create Main Class*.
- 4 Click **Finish**.



- 5 To set up Java to use the API, right-click **SampleJavacode** and select Properties.



- 6 From the source category click *Add Folder*.
- 7 Navigate to the folder where the TWS API is installed. The folders you want to add are called *com* and *TestJavaClient*. Click **OK**.



- 8** Press **F6** to run the sample java project. When the message says "Project Samplejavacode does not have main class set" select *TestJavaclient.Main* and click **OK**.

The Java Test Client's sample application window is pictured below.



## Java Test Client Overview

This section describes the package, classes and methods used in the Java Test Client, the sample Java program included in the TWS API.

**Note:** The classes and methods used in the Java Test Client were developed for the sample program and are not part of the TWS Java API.

### Package

The Java Test Client includes the package `TestJavaClient`. This package includes all the classes and methods required by the Java Test Client sample program.

### TestJavaClient Classes

`TestJavaClient` includes the following classes:

Class	Description
<code>AccountDlg</code>	Defines the Account/Portfolio dialog, which the Java Test Client sample program uses to display a customer's account and portfolio information.
<code>AcctUpdatesDlg</code>	Defines the Account Updates dialog, which the Java Test Client sample program uses to let FA customers subscribe to account updates.
<code>ComboLegDlg</code>	Defines the Combination Legs dialog, which the Java Test Client sample program uses to let customers add and remove combo legs.
<code>ConnectDlg</code>	Defines the Connect dialog, which the Java Test Client sample program uses to let customers connect to TWS
<code>ExecFilterDlg</code>	Defines the Execution Report Filter dialog, which the Java Test Client sample program uses to let customers enter filter criteria for execution reports.
<code>ExtOrdDlg</code>	Defines the Extended Order Info dialog, which the Java Test Client sample program uses to let customers enter values for extended order attributes.
<code>FAAllocationInfoDlg</code>	Defines the FA Allocation Info dialog, which the Java Test Client sample program uses to let Financial Advisor customers enter information about allocation profiles and account groups.
<code>FinancialAdvisorDlg</code>	Defines the Financial Advisor dialog, which Financial Advisor customers use in the Java Test Client sample program.
<code>LogConfigDlg</code>	Defines the Log Configuration dialog, which the Java Test Client sample program uses to let customers specify the level of log entries in the log file.

Class	Description
MktDepthDlg	Defines the Market Depth dialog, which the Java Test Client sample program uses to let customers view market depth for a specified contracts.
NewsBulletinDlg	Defines the IB News Bulletin Subscription dialog, which the Java Test Client sample program uses to let customers subscribe and unsubscribe to news bulletins.
OrderDlg	Defines the Sample dialog, which the Java Test Client sample program uses to let customers enter contract and order information for orders and requests for contract data, market depth, market data, options exercise, and historical data queries.
SampleFrame	Defines the Java Test Client sample program main window. All of the text panels and buttons on the main window are defined in this class.
ScannerDlg	Defines the Market Scanner dialog (also called the Sample dialog in the test client), which the Java Test Client sample program uses to let customers subscribe and unsubscribe to markets scans, as well as request market scan parameters.

**Note:** For more information about the Java code in the Java Test Client sample program, see the [Getting Started with the TWS Java API guide](#).

## Java API Overview

The TWS Java API contains the package **com.ib.client**, which contains the following classes:

Class	Description
<b>EWrapper</b>	This interface is responsible for receiving messages from TWS.
<b>ComboLeg</b>	This class contains attributes used to describe combo legs.
<b>Contract</b>	This class contains attributes used to describe a contract.
<b>ContractDetails</b>	This class contains attributes used to describe contract details, including bond information.
<b>EClientSocket</b>	This class is responsible for sending messages to TWS.
<b>Execution</b>	This class contains attributes used to describe a trade.
<b>ExecutionFilter</b>	This class contains attributes used to describe execution filter criteria.
<b>Order</b>	This class contains attributes used to describe an order.
<b>OrderState</b>	This class contains attributes used to describe the status of an order.
<b>ScannerSubscription</b>	This class contains attributes used to describe the elements of a market scan.
<b>TickType</b>	This class defines the generic tick types and their tick values.

The methods and attributes of these classes are described in the rest of this chapter.

## Java EClientSocket Methods

This section describes the class EClientSocket methods you can use when connecting to TWS. The list of methods includes:

Connection and Server	Market Depth
<a href="#">EClientSocket()</a> <a href="#">eConnect()</a> <a href="#">eDisconnect()</a> <a href="#">isConnected()</a> <a href="#">setServerLogLevel()</a> <a href="#">reqCurrentTime()</a> <a href="#">serverVersion()</a> <a href="#">TwsConnectionTime()</a>	<a href="#">reqMktDepth()</a> <a href="#">cancelMktDepth()</a>
<b>Market Data</b>	<b>News Bulletins</b>
<a href="#">reqMktData()</a> <a href="#">cancelMktData()</a>	<a href="#">reqNewsBulletins()</a> <a href="#">cancelNewsBulletins()</a>
<b>Orders</b>	<b>Financial Advisors</b>
<a href="#">placeOrder()</a> <a href="#">cancelOrder()</a> <a href="#">reqOpenOrders()</a> <a href="#">reqAllOpenOrders()</a> <a href="#">reqAutoOpenOrders()</a> <a href="#">exerciseOptions()</a>	<a href="#">reqManagedAccts()</a> <a href="#">requestFA()</a> <a href="#">replaceFa()</a>
<b>Account</b>	<b>Market Scanners</b>
<a href="#">reqAccountUpdates()</a>	<a href="#">reqScannerParameters()</a> <a href="#">reqScannerSubscription()</a> <a href="#">cancelScannerSubscription()</a>
<b>Executions</b>	<b>Historical Data</b>
<a href="#">reqExecutions()</a>	<a href="#">reqHistoricalData()</a> <a href="#">cancelHistoricalData()</a>
<b>Contract Details</b>	<b>Real Time Bars</b>
<a href="#">reqContractDetails()</a>	<a href="#">reqRealTimeBars()</a> <a href="#">cancelRealTimeBars()</a>
	<b>Fundamental Data</b>
	<a href="#">reqFundamentalData()</a> <a href="#">cancelFundamentalData()</a>

## **EClientSocket()**

This is the constructor.

```
EClientSocket(AnyWrapper anyWrapper)
```

Parameter	Description
<b>anyWrapper</b>	The reference to an object that was derived from the AnyWrapper base interface. Note EWrapper extends AnyWrapper.

## **eConnect()**

This function must be called before any other. There is no feedback for a successful connection, but a subsequent attempt to connect will return the message "Already connected."

```
void eConnect( String host, int port, int clientId)
```

Parameter	Description
<b>host</b>	The host name or IP address of the machine where TWS is running. Leave blank to connect to the local host.
<b>port</b>	Must match the port specified in TWS on the Configure>API>Socket Port field.
<b>clientId</b>	A number used to identify this client connection. All orders placed/modified from this client will be associated with this client identifier. Note: Each client MUST connect with a unique clientId.

## **eDisconnect()**

Call this method to terminate the connections with TWS. Calling this method does not cancel orders that have already been sent.

```
void eDisconnect()
```

## **isConnected()**

Call this method to check if there is a connection with TWS.

```
void isConnected()
```

## reqMktData()

Call this method to request market data. The market data will be returned by the [tickPrice\(\)](#), [tickSize\(\)](#), [tickOptionComputation\(\)](#), [tickGeneric\(\)](#), [tickString\(\)](#) and [tickEFP\(\)](#) methods.

```
void reqMktData(int tickerId, Contract contract, String genericTicklist,
boolean snapshot)
```

Parameter	Description
<b>tickerId</b>	The ticker id. Must be a unique value. When the market data returns, it will be identified by this tag. This is also used when canceling the market data.
<a href="#">contract</a>	This class contains attributes used to describe the contract.
<b>genericTicklist</b>	A comma delimited list of generic tick types. Tick types can be found in the <a href="#">Generic Tick Types</a> page.
<b>snapshot</b>	Check to return a single snapshot of market data and have the market data subscription cancel. Do not enter any genericTicklist values if you use snapshot.

## cancelMktData()

After calling this method, market data for the specified Id will stop flowing.

```
void cancelMktData(int tickerId)
```

Parameter	Description
<b>tickerId</b>	The Id that was specified in the call to reqMktData().

## placeOrder()

```
void placeOrder( int id, Contract contract, Order order)
```

Parameter	Description
<b>id</b>	The order Id. You must specify a unique value. When the order status returns, it will be identified by this tag. This tag is also used when canceling the order.
<b>contract</b>	This class contains attributes used to describe the contract.
<b>order</b>	This structure contains the details of the order. Note: Each client MUST connect with a unique clientId.

## cancelOrder()

Call this method to cancel an order.

```
void cancelOrder(int id)
```

Parameter	Description
<b>id</b>	The order Id that was specified previously in the call to placeOrder()

## reqOpenOrders()

Call this method to request any open orders that were placed from this API client. Each open order will be fed back through the [openOrder\(\)](#) and [orderStatus\(\)](#) methods on the EWrapper.

**Note:** The client with a clientId of "0" will also receive the TWS-owned open orders. These orders will be associated with the client and a new orderId will be generated. This association will persist over multiple API and TWS sessions.

```
void reqOpenOrders()
```

## reqAccountUpdates()

Call this function to start getting account values, portfolio, and last update time information. The account data will be fed back through the [updateAccountTime\(\)](#), [updateAccountValue\(\)](#) and [updatePortfolio\(\)](#) EWrapper methods.

```
void reqAccountUpdates (boolean subscribe, String acctCode)
```

Parameter	Description
<b>subscribe</b>	If set to TRUE, the client will start receiving account and portfolio updates. If set to FALSE, the client will stop receiving this information.
<b>acctCode</b>	The account code for which to receive account and portfolio updates.

## reqExecutions()

When this method is called, the execution reports that meet the filter criteria are downloaded to the client via the [execDetails\(\)](#) method.

```
void reqExecutions (ExecutionFilter filter)
```

Parameter	Description
<b>filter</b>	The filter criteria used to determine which execution reports are returned.

## reqContractDetails()

Call this method to download all details for a particular contract. The contract details will be received via the [contractDetails\(\)](#) method on the EWrapper.

```
void reqContractDetails (int reqId, Contract contract)
```

Parameter	Description
<b>reqId</b>	The ID of the data request. Ensures that responses are matched to requests if several requests are in process.
<b>contract</b>	This class contains attributes used to describe the contract.

## reqMktDepth()

Call this method to request market depth for a specific contract. The market depth will be returned by the [updateMktDepth\(\)](#) and [updateMktDepthL2\(\)](#) methods.

```
void reqMktDepth(int tickerId, Contract contract, int numRows)
```

Parameter	Description
<b>tickerId</b>	The ticker Id. Must be a unique value. When the market depth data returns, it will be identified by this tag. This is also used when canceling the market depth.
<b>contract</b>	This class contains attributes used to describe the contract.
<b>numRows</b>	Specifies the number of market depth rows to return.

## cancelMktDepth()

After calling this method, market depth data for the specified Id will stop flowing.

```
void cancelMktDepth(int id)
```

Parameter	Description
<b>tickerId</b>	The Id that was specified in the call to reqMktDepth().

## reqNewsBulletins()

Call this method to start receiving news bulletins. Each bulletin will be returned by the [updateNewsBulletin\(\)](#) method.

```
void reqNewsBulletins(boolean allMsgs)
```

Parameter	Description
<b>allMsgs</b>	If set to TRUE, returns all the existing bulletins for the current day and any new ones. IF set to FALSE, will only return new bulletins.

## cancelNewsBulletins()

Call this method to stop receiving news bulletins.

```
void cancelNewsBulletins()
```

## setServerLogLevel()

The default level is ERROR. Refer to the [API logging](#) page for more details.

```
void setServerLogLevel(int logLevel)
```

Parameter	Description
<b>logLevel</b>	<p>Specifies the level of log entry detail used by the server (TWS) when processing API requests. Valid values include:</p> <ul style="list-style-type: none"> <li>• 1 = SYSTEM</li> <li>• 2 = ERROR</li> <li>• 3 = WARNING</li> <li>• 4 = INFORMATION</li> <li>• 5 = DETAIL</li> </ul>

## reqAllOpenOrders

Call this method to request all open orders that were placed from all API clients linked to one TWS, and also from the TWS. Note that you can run up to 8 API clients from a single TWS. Each open order will be fed back through the [openOrder\(\)](#) and [orderStatus\(\)](#) methods on the EWrapper.

**Note:** No association is made between the returned orders and the requesting client.

```
void reqAllOpenOrders()
```

## reqAutoOpenOrders()

Call this method to request that newly created TWS orders be implicitly associated with the client. When a new TWS order is created, the order will be associated with the client and automatically fed back through the [openOrder\(\)](#) and [orderStatus\(\)](#) methods on the EWrapper.

**Note:** TWS orders can only be bound to clients with a clientId of 0.

```
void reqAutoOpenOrders(boolean bAutoBind)
```

Parameter	Description
<b>bAutoBind</b>	If set to TRUE, newly created TWS orders will be implicitly associated with the client. If set to FALSE, no association will be made.

## reqManagedAccts()

Call this method to request the list of managed accounts. The list will be returned by the [managedAccounts\(\)](#) method on the EWrapper.

**Note:** This request can only be made when connected to a Financial Advisor (FA) account

```
void reqManagedAccts()
```

## requestFA()

Call this method to request FA configuration information from TWS. The data returns in an XML string via the [receiveFA\(\)](#) method.

```
void requestFA(long faDataType)
```

Parameter	Description
<b>faDataType</b>	Specifies the type of Financial Advisor configuration data being requested. Valid values include: <ul style="list-style-type: none"> <li>• 1 = GROUPS</li> <li>• 2 = PROFILE</li> <li>• 3 = ACCOUNT ALIASES</li> </ul>

### **replaceFA()**

Call this method to request new FA configuration information from TWS. The data returns in an XML string via a "receiveFA" method.

```
void replaceFA(long faDataType, string xml)
```

Parameter	Description
<b>faDataType</b>	Specifies the type of Financial Advisor configuration data being requested. Valid values include: <ul style="list-style-type: none"> <li>1 = GROUPS</li> <li>2 = PROFILE</li> <li>3 = ACCOUNT ALIASES</li> </ul>
<b>xml</b>	The XML string containing the new FA configuration information.

### **reqScannerParameters()**

Call the reqScannerParameters() method to receive an XML document that describes the valid parameters that a scanner subscription can have.

```
void reqScannerParameters()
```

## reqScannerSubscription()

Call the reqScannerSubscription() method to start receiving market scanner results through the [scannerData\(\)](#) EWrapper method.

```
void reqScannerSubscription(int tickerId, ScannerSubscription  
subscription)
```

Parameter	Description
<b>tickerId</b>	The Id for the subscription. Must be a unique value. When the subscription data is received, it will be identified by this Id. This is also used when canceling the scanner.
<b>subscription</b>	Summary of the scanner subscription parameters including filters.

## cancelScannerSubscription()

Call the cancelScannerSubscription() method to stop receiving market scanner results.

```
void cancelScannerSubscription(int tickerId)
```

Parameter	Description
<b>tickerId</b>	The Id that was specified in the call to reqScannerSubscription().

## reqHistoricalData()

Call the reqHistoricalData() method to start receiving historical data results through the [historicalData\(\)](#) EWrapper method.

```
void reqHistoricalData (int id, Contract contract, String endTime,
String durationStr, String barSizeSetting, String whatToShow, int
useRTH, int formatDate)
```

Parameter	Description																																		
<b>tickerId</b>	The Id for the request. Must be a unique value. When the data is received, it will be identified by this Id. This is also used when canceling the historical data request.																																		
<b>contract</b>	This class contains attributes used to describe the contract.																																		
<b>endTime</b>	Use the format yyyyMMdd hh:mm:ss tmz, where the time zone is allowed (optionally) after a space at the end.																																		
<b>durationStr</b>	<p>This is the time span the request will cover, and is specified using the format: &lt;integer&gt; &lt;unit&gt;, i.e., 1 D, where valid units are:</p> <ul style="list-style-type: none"> <li>• " S (seconds)</li> <li>• " D (days)</li> <li>• " W (weeks)</li> <li>• " M (months)</li> <li>• " Y (years)</li> </ul> <p>If no unit is specified, seconds are used. Also, note "years" is currently limited to one.</p>																																		
<b>barSizeSetting</b>	<p>Specifies the size of the bars that will be returned (within IB/TWS limits). Valid values include:</p> <table> <thead> <tr> <th>Bar Size</th> <th>Parametric Value</th> </tr> </thead> <tbody> <tr> <td>1 sec</td> <td>1</td> </tr> <tr> <td>5 secs</td> <td>2</td> </tr> <tr> <td>15 secs</td> <td>3</td> </tr> <tr> <td>30 secs</td> <td>4</td> </tr> <tr> <td>1 min.</td> <td>5</td> </tr> <tr> <td>2 mins</td> <td>6</td> </tr> <tr> <td>3 mins</td> <td>16</td> </tr> <tr> <td>5 mins</td> <td>7</td> </tr> <tr> <td>15 mins</td> <td>8</td> </tr> <tr> <td>30 mins</td> <td>9</td> </tr> <tr> <td>1 hour</td> <td>10</td> </tr> <tr> <td>1 day</td> <td>11</td> </tr> <tr> <td>1 week</td> <td>12</td> </tr> <tr> <td>1 month</td> <td>13</td> </tr> <tr> <td>3 months</td> <td>14</td> </tr> <tr> <td>1 year</td> <td>15</td> </tr> </tbody> </table>	Bar Size	Parametric Value	1 sec	1	5 secs	2	15 secs	3	30 secs	4	1 min.	5	2 mins	6	3 mins	16	5 mins	7	15 mins	8	30 mins	9	1 hour	10	1 day	11	1 week	12	1 month	13	3 months	14	1 year	15
Bar Size	Parametric Value																																		
1 sec	1																																		
5 secs	2																																		
15 secs	3																																		
30 secs	4																																		
1 min.	5																																		
2 mins	6																																		
3 mins	16																																		
5 mins	7																																		
15 mins	8																																		
30 mins	9																																		
1 hour	10																																		
1 day	11																																		
1 week	12																																		
1 month	13																																		
3 months	14																																		
1 year	15																																		

Parameter	Description
<b>whatToShow</b>	Determines the nature of data being extracted. Valid values include: <ul style="list-style-type: none"> <li>• TRADES</li> <li>• MIDPOINT</li> <li>• BID</li> <li>• ASK</li> <li>• BID_ASK</li> <li>• HISTORICAL_VOLATILITY</li> <li>• OPTION_IMPLIED_VOLATILITY</li> <li>• OPTION_VOLUME</li> </ul>
<b>useRTH</b>	Determines whether to return all data available during the requested time span, or only data that falls within regular trading hours. Valid values include: <ul style="list-style-type: none"> <li>• 0 - all data is returned even where the market in question was outside of its regular trading hours.</li> <li>• 1 - only data within the regular trading hours is returned, even if the requested time span falls partially or completely outside of the RTH.</li> </ul>
<b>formatDate</b>	Determines the date format applied to returned bars. Valid values include: <ul style="list-style-type: none"> <li>• 1 - dates applying to bars returned in the format: yyyyymmdd{space}{space}hh:mm:ss</li> <li>• 2 - dates are returned as a long integer specifying the number of seconds since 1/1/1970 GMT.</li> </ul>

**Note:** For a information about historical data request limitations, see [Historical Data Limitations](#).

### **cancelHistoricalData()**

Call the cancelHistoricalData() method to stop receiving historical data results.

```
void cancelHistoricalData (int tickerId)
```

Parameter	Description
<b>tickerId</b>	The Id that was specified in the call to reqHistoricalData().

## reqRealTimeBars()

Call the reqRealTimeBars() method to start receiving real time bar results through the [realtimeBar\(\)](#) EWrapper method.

```
void reqRealTimeBars(int tickerId, Contract contract, int barSize,
String whatToShow, boolean useRTH)
```

Parameter	Description
<b>tickerId</b>	The Id for the request. Must be a unique value. When the data is received, it will be identified by this Id. This is also used when canceling the historical data request.
<b>contract</b>	This class contains attributes used to describe the contract.
<b>barSize</b>	Currently only 5 second bars are supported, if any other value is used, an exception will be thrown.
<b>whatToShow</b>	Determines the nature of the data extracted. Valid values include: <ul style="list-style-type: none"> <li>• - TRADES</li> <li>• - BID</li> <li>• - ASK</li> <li>• - MIDPOINT</li> </ul>
<b>useRTH</b>	Regular Trading Hours only. Valid values include: <ul style="list-style-type: none"> <li>• 0 = all data available during the time span requested is returned, including time intervals when the market in question was outside of regular trading hours.</li> <li>• 1 = only data within the regular trading hours for the product requested is returned, even if the time span falls partially or completely outside.</li> </ul>

## cancelRealTimeBars()

Call this method to stop receiving real time bar results.

```
void cancelRealTimeBars (int tickerId)
```

Parameter	Description
<b>tickerId</b>	The Id that was specified in the call to reqRealTimeBars().

## exerciseOptions()

Call the exerciseOptions() method to exercise options.

**Note:** SMART is not an allowed exchange in exerciseOptions() calls, and TWS does a request for the position in question whenever any API initiated exercise or lapse is attempted.

```
void exerciseOptions(int tickerId, Contract contract, int
exerciseAction, int exerciseQuantity, String account, int override)
```

Parameter	Description
<b>tickerId</b>	The Id for the exercise request
<b>contract</b>	This class contains attributes used to describe the contract.
<b>exerciseAction</b>	this can have two values: <ul style="list-style-type: none"> <li>• 1 = exercise</li> <li>• 2 = lapse</li> </ul>
<b>exerciseQuantity</b>	The number of contracts to be exercised
<b>account</b>	For institutional orders. Specifies the IB account.
<b>override</b>	Specifies whether your setting will override the system's natural action. For example, if your action is "exercise" and the option is not in-the-money, by natural action the option would not exercise. If you have override set to "yes" the natural action would be overridden and the out-of-the money option would be exercised. Values are: <ul style="list-style-type: none"> <li>• 0 = do not override</li> <li>• 1 = override</li> </ul>

## reqCurrentTime()

Returns the current system time on the server side via the [currentTime\(\)](#) EWrapper method.

```
void reqCurrentTime()
```

## serverVersion()

Returns the version of the TWS instance to which the API application is connected.

```
void serverVersion()
```

## TwsConnectionTime()

Returns the time the API application made a connection to TWS.

```
void TwsConnectionTime ()
```

## reqFundamentalData()

Call this method to receive Reuters global fundamental data. There must be a subscription to Reuters Fundamental set up in Account Management before you can receive this data.

```
void reqFundamentalData(int reqId, Contract contract, string reportType)
```

Parameter	Description
<b>reqId</b>	The ID of the data request. Ensures that responses are matched to requests if several requests are in process.
<b>contract</b>	This structure contains a description of the contract for which Reuters Fundamental data is being requested.
<b>reportType</b>	Identifies the report type, which is one of the following: <ul style="list-style-type: none"><li>• Estimates</li><li>• Financial Statements</li><li>• Summary</li></ul>

### **cancelFundamentalData()**

Call this method to stop receiving Reuters global fundamental data.

```
void cancelFundamentalData(int reqId)
```

Parameter	Description
<b>reqId</b>	The ID of the data request.

## Java EWrapper Methods

This section describes the class EWrapper methods you can use when connecting to TWS. The list of methods includes:

<b>Connection and Server</b>	<b>Executions</b>
<code>currentTime()</code> <code>error()</code> <code>connectionClosed()</code>	<code>execDetails()</code> <code>execDetailsEnd()</code>
<b>Market Data</b>	<b>Market Depth</b>
<code>tickPrice()</code> <code>tickSize()</code> <code>tickOptionComputation()</code> <code>tickGeneric()</code> <code>tickString()</code> <code>tickEFP()</code> <code>tickSnapshotEnd()</code>	<code>updateMktDepth()</code> <code>updateMktDepthL2()</code>
<b>Orders</b>	<b>News Bulletins</b>
<code>orderStatus()</code> <code>openOrder()</code> <code>nextValidId()</code>	<code>updateNewsBulletin()</code>
<b>Account and Portfolio</b>	<b>Financial Advisors</b>
<code>updateAccountValue()</code> <code>updatePortfolio()</code> <code>updateAccountTime()</code>	<code>managedAccounts()</code> <code>receiveFA()</code>
<b>Contract Details</b>	<b>Historical Data</b>
<code>contractDetails()</code> <code>contractDetailsEnd()</code> <code>bondContractDetails()</code>	<code>historicalData()</code>
	<b>Market Scanners</b>
	<code>scannerParameters()</code> <code>scannerData()</code> <code>scannerDataEnd()</code>
	<b>Real Time Bars</b>
	<code>realtimeBar()</code>
	<b>Fundamental Data</b>
	<code>fundamentalData()</code>

**tickPrice()**

This method is called when the market data changes. Prices are updated immediately with no delay.

```
void tickPrice(int tickerId, int field, double price, int canAutoExecute)
```

Parameter	Description
<b>tickerId</b>	The ticker Id that was specified previously in the call to reqMktData()
<b>field</b>	Specifies the type of price. Pass the field value into TickType.getField(int tickType) to retrieve the field description. For example, a field value of 1 will map to bidPrice, a field value of 2 will map to askPrice, etc. <ul style="list-style-type: none"> <li>• 1 = bid</li> <li>• 2 = ask</li> <li>• 4 = last</li> <li>• 6 = high</li> <li>• 7 = low</li> <li>• 9 = close</li> </ul>
<b>price</b>	Specifies the price for the specified field
<b>canAutoExecute</b>	Specifies whether the price tick is available for automatic execution. Possible values are: <ul style="list-style-type: none"> <li>• 0 = not eligible for automatic execution</li> <li>• 1 = eligible for automatic execution</li> </ul>

## tickSize()

This method is called when the market data changes. Sizes are updated immediately with no delay.

```
void tickSize(int tickerId, int field, int size)
```

Parameter	Description
<b>tickerId</b>	The ticker Id that was specified previously in the call to reqMktData()
<b>field</b>	Specifies the type of price. Pass the field value into TickType.getField(int tickType) to retrieve the field description. For example, a field value of 0 will map to bidSize, a field value of 3 will map to askSize, etc. <ul style="list-style-type: none"> <li>• 0 = bid size</li> <li>• 3 = ask size</li> <li>• 5 = last size</li> <li>• 8 = volume</li> </ul>
<b>size</b>	Specifies the size for the specified field

## tickOptionComputation()

This method is called when the market in an option or its underlier moves. TWS's option model volatilities, prices, and deltas, along with the present value of dividends expected on that options underlier are received.

```
void tickOptionComputation(int tickerId, int field, double impliedVol,
double delta, double modelPrice, double pvDividend)
```

Parameter	Description
<b>tickerId</b>	The ticker Id that was specified previously in the call to reqMktData()
<b>field</b>	Specifies the type of option computation. Pass the field value into TickType.getField(int tickType) to retrieve the field description. For example, a field value of 13 will map to modelOptComp, etc. <ul style="list-style-type: none"> <li>• 10 = Bid</li> <li>• 11 = Ask</li> <li>• 12 = Last</li> </ul>
<b>impliedVol</b>	The implied volatility calculated by the TWS option modeler, using the specified ticktype value.
<b>delta</b>	The option delta calculated by the TWS option modeler.
<b>modelPrice</b>	The model price
<b>pvDividend</b>	The present value of dividends expected on the options underlier

## tickGeneric()

This method is called when the market data changes. Values are updated immediately with no delay.

```
void tickGeneric(int tickerId, int tickType, double value)
```

Parameter	Description
<b>tickerId</b>	The ticker Id that was specified previously in the call to reqMktData()
<b>tickType</b>	Specifies the type of price. Pass the field value into TickType.getField(int tickType) to retrieve the field description. For example, a field value of 46 will map to shortable, etc.
<b>value</b>	The value of the specified field

## tickString()

This method is called when the market data changes. Values are updated immediately with no delay.

```
void tickString(int tickerId, int tickType, String value)
```

Parameter	Description
<b>tickerId</b>	The ticker Id that was specified previously in the call to reqMktData()
<b>field</b>	Specifies the type of price. Pass the field value into TickType.getField(int tickType) to retrieve the field description. For example, a field value of 45 will map to lastTimestamp, etc.
<b>value</b>	The value of the specified field

## tickEFP()

This method is called when the market data changes. Values are updated immediately with no delay.

```
void tickEFP(int tickerId, int tickType, double basisPoints, String formattedBasisPoints, double impliedFuture, int holdDays, String futureExpiry, double dividendImpact, double dividendsToExpiry)
```

Parameter	Description
<b>tickerId</b>	The ticker Id that was specified previously in the call to reqMktData()
<b>field</b>	Specifies the type of price. Pass the field value into TickType.getField(int tickType) to retrieve the field description. For example, a field value of 38 will map to bidEFP, etc.

Parameter	Description
<b>basisPoints</b>	Annualized basis points, which is representative of the financing rate that can be directly compared to broker rates
<b>formattedBasisPoints</b>	Annualized basis points as a formatted string that depicts them in percentage form
<b>impliedFuture</b>	Implied futures price
<b>holdDays</b>	The number of hold days until the expiry of the EFP
<b>futureExpiry</b>	The expiration date of the single stock future
<b>dividendImpact</b>	The dividend impact upon the annualized basis points interest rate
<b>dividendsToExpiry</b>	The dividends expected until the expiration of the single stock future

### **tickSnapshotEnd()**

This is called when a snapshot market data subscription has been fully handled and there is nothing more to wait for. This also covers the timeout case.

```
void tickSnapshotEnd(int reqId)
```

Parameter	Description
<b>reqID</b>	Id of the data request.

## orderStatus()

This method is called whenever the status of an order changes. It is also fired after reconnecting to TWS if the client has any open orders.

```
void orderStatus(int orderId, String status, int filled, int remaining,
double avgFillPrice, int permId, int parentId, double lastFillPrice, int
clientId, String whyHeld)
```

Parameter	Description
<b>id</b>	The order Id that was specified previously in the call to placeOrder()
<b>status</b>	<p>The order status. Possible values include:</p> <ul style="list-style-type: none"> <li>• PendingSubmit - indicates that you have transmitted the order, but have not yet received confirmation that it has been accepted by the order destination. <b>NOTE:</b> This order status is not sent by TWS and should be explicitly set by the API developer when an order is submitted.</li> <li>• PendingCancel - indicates that you have sent a request to cancel the order but have not yet received cancel confirmation from the order destination. At this point, your order is not confirmed canceled. You may still receive an execution while your cancellation request is pending. <b>NOTE:</b> This order status is not sent by TWS and should be explicitly set by the API developer when an order is canceled.</li> <li>• PreSubmitted - indicates that a simulated order type has been accepted by the IB system and that this order has yet to be elected. The order is held in the IB system until the election criteria are met. At that time the order is transmitted to the order destination as specified .</li> <li>• Submitted - indicates that your order has been accepted at the order destination and is working.</li> <li>• Cancelled - indicates that the balance of your order has been confirmed canceled by the IB system. This could occur unexpectedly when IB or the destination has rejected your order.</li> <li>• Filled - indicates that the order has been completely filled.</li> <li>• Inactive - indicates that the order has been accepted by the system (simulated orders) or an exchange (native orders) but that currently the order is inactive due to system, exchange or other issues.</li> </ul>
<b>filled</b>	Specifies the number of shares that have been executed.
<b>remaining</b>	Specifies the number of shares still outstanding.

Parameter	Description
<b>avgFillPrice</b>	The average price of the shares that have been executed. This parameter is valid only if the <b>filled</b> parameter value is greater than zero. Otherwise, the price parameter will be zero.
<b>permId</b>	The TWS id used to identify orders. Remains the same over TWS sessions.
<b>parentId</b>	The order ID of the parent order, used for bracket and auto trailing stop orders.
<b>lastFilledPrice</b>	The last price of the shares that have been executed. This parameter is valid only if the filled parameter value is greater than zero. Otherwise, the price parameter will be zero.
<b>clientId</b>	The ID of the client (or TWS) that placed the order. Note that TWS orders have a fixed clientId and orderId of 0 that distinguishes them from API orders.
<b>whyHeld</b>	This field is used to identify an order held when TWS is trying to locate shares for a short sell. The value used to indicate this is 'locate'.

## error()

This method is called when there is an error with the communication or when TWS wants to send a message to the client.

```
void error(int id, int errorCode, String errorMessage)
```

Parameter	Description
<b>id</b>	This is the orderId or tickerId of the request that generated the error.
<b>errorCode</b>	For information on error codes, see <a href="#">Error Codes</a> .
<b>errorMessage</b>	The textual description of the error.

This method is called when exception occurs while handling a request.

```
void error(Exception e)
```

Parameter	Description
<b>e</b>	The exception that occurred

This method is called when TWS wants to send an error message to the client. (V1).

```
void error(String str)
```

Parameter	Description
<b>str</b>	This is the textual description of the error

**connectionClosed()**

This method is called when TWS closes the sockets connection, or when TWS is shut down.

```
void connectionClosed()
```

**managedAccounts()**

This method is called when a successful connection is made to a Financial Advisor account. It is also called when the reqManagedAccts() method is invoked.

```
void managedAccounts (String accountsList)
```

Parameter	Description
<b>accountsList</b>	The comma delimited list of FA managed accounts.

**openOrder()**

This method is called to feed in open orders.

```
void openOrder(int orderId, Contract contract, Order order, OrderState orderState )
```

Parameter	Description
<b>orderId</b>	The order Id assigned by TWS. Used to cancel or update the order.
<b>contract</b>	The Contract class attributes describe the contract.
<b>order</b>	The Order class attributes define the details of the order.
<b>orderState</b>	The orderState attributes include margin and commissions fields for both pre and post trade data.

## **updateAccountValue()**

This method is called only when reqAccountUpdates() method on the EClientSocket object has been called.

```
void updateAccountValue(String key, String value, String currency,
String accountName)
```

Parameter	Description
<b>key</b>	A string that indicates one type of account value. There is a long list of possible keys that can be sent, here are just a few examples: <ul style="list-style-type: none"> <li>• CashBalance - account cash balance</li> <li>• DayTradesRemaining - number of day trades left</li> <li>• EquityWithLoanValue - equity with Loan Value</li> <li>• InitMarginReq - current initial margin requirement</li> <li>• MaintMarginReq - current maintenance margin</li> <li>• NetLiquidation - net liquidation value</li> </ul>
<b>value</b>	The value associated with the key.
<b>currency</b>	Defines the currency type, in case the value is a currency type.
<b>account</b>	States the account the message applies to. Useful for Financial Advisor sub-account messages.

## updatePortfolio()

This method is called only when reqAccountUpdates() method on the EClientSocket object has been called.

```
void updatePortfolio(Contract contract, int position, double marketPrice, double marketValue, double averageCost, double unrealizedPNL, double realizedPNL, String accountName)
```

Parameter	Description
<b>contract</b>	This structure contains a description of the contract which is being traded. The exchange field in a contract is not set for portfolio update.
<b>position</b>	This integer indicates the position on the contract. If the position is 0, it means the position has just cleared.
<b>marketPrice</b>	The unit price of the instrument.
<b>marketValue</b>	The total market value of the instrument.
<b>averageCost</b>	The average cost per share is calculated by dividing your cost (execution price + commission) by the quantity of your position.
<b>unrealizedPNL</b>	The difference between the current market value of your open positions and the average cost, or Value - Average Cost.
<b>realizedPNL</b>	Shows your profit on closed positions, which is the difference between your entry execution cost (execution price + commissions to open the position) and exit execution cost ((execution price + commissions to close the position))
<b>accountName</b>	The name of the account the message applies to. Useful for Financial Advisor sub-account messages.

## updateAccountTime()

This method is called only when reqAccountUpdates() method on the EClientSocket object has been called.

```
void updateAccountTime(String timeStamp)
```

Parameter	Description
<b>timeStamp</b>	This indicates the last update time of the account information

## nextValidId()

This method is called after a successful connection to TWS.

```
void nextValidId(int orderId)
```

Parameter	Description
<b>orderId</b>	The next available order Id received from TWS upon connection. Increment all successive orders by one based on this Id.

## contractDetails()

This method is called only when reqContractDetails method on the EClientSocket object has been called.

```
void contractDetails(int ReqId, ContractDetails contractDetails)
```

Parameter	Description
<b>reqID</b>	The ID of the data request. Ensures that responses are matched to requests if several requests are in process.
<b>contractDetails</b>	This structure contains a full description of the contract being looked up.

## contractDetailsEnd()

This method is called once all contract details for a given request are received. This helps to define the end of an option chain.

```
void contractDetailsEnd(int reqId)
```

Parameter	Description
<b>reqID</b>	The Id of the data request.

## bondContractDetails()

This method is called only when reqContractDetails method on the EClientSocket object has been called for bonds.

```
void bondContractDetails(int reqId, ContractDetails contractDetails)
```

Parameter	Description
<b>reqId</b>	The ID of the data request.
<b>contractDetails</b>	This structure contains a full description of the bond contract being looked up.

## execDetails()

This method is called when the [reqExecutions\(\)](#) method is invoked, or when an order is filled.

```
void execDetails(int orderId, Contract contract, Execution execution)
```

Parameter	Description
<b>orderId</b>	The order Id that was specified previously in the call to placeOrder().
<b>contract</b>	This structure contains a full description of the contract that was executed. <b>Note:</b> Refer to the <a href="#">Java SocketClient Properties</a> page for more information.

Parameter	Description
<b>execution</b>	This structure contains addition order execution details. <b>Note:</b> Refer to the <a href="#">Java SocketClient Properties</a> page for more information.

## execDetailsEnd()

This method is called once all executions have been sent to a client in response to reqExecutions().

```
void execDetailsEnd(int reqId)
```

Parameter	Description
<b>reqID</b>	The Id of the data request.

## updateMktDepth()

This method is called when the market depth changes.

```
void updateMktDepth(int tickerId, int position, int operation, int side,
double price, int size)
```

Parameter	Description
<b>tickerId</b>	The ticker Id that was specified previously in the call to reqMktDepth()
<b>position</b>	Specifies the row Id of this market depth entry.
<b>operation</b>	Identifies how this order should be applied to the market depth. Valid values are:- <ul style="list-style-type: none"> <li>• 0 = insert (insert this new order into the row identified by 'position')</li> <li>• 1 = update (update the existing order in the row identified by 'position')</li> <li>• 2 = delete (delete the existing order at the row identified by 'position')</li> </ul>
<b>side</b>	identifies the side of the book that this order belongs to. Valid values are:- <ul style="list-style-type: none"> <li>• 0 = ask</li> <li>• 1 = bid</li> </ul>
<b>price</b>	The order price.
<b>size</b>	The order size.

## updateMktDepthL2()

This method is called when the Level II market depth changes.

```
void updateMktDepthL2(int tickerId, int position, String marketMaker,
int operation, int side, double price, int size)
```

Parameter	Description
<b>tickerId</b>	The ticker Id that was specified previously in the call to reqMktDepth()
<b>position</b>	Specifies the row id of this market depth entry.
<b>marketMaker</b>	Specifies the exchange hosting this order.
<b>operation</b>	identifies the how this order should be applied to the market depth. Valid values are:- <ul style="list-style-type: none"> <li>• 0 = insert (insert this new order into the row identified by 'position')</li> <li>• 1 = update (update the existing order in the row identified by 'position')</li> <li>• 2 = delete (delete the existing order at the row identified by 'position')</li> </ul>
<b>side</b>	Identifies the side of the book that this order belongs to. Valid values are: <ul style="list-style-type: none"> <li>• 0 = ask</li> <li>• 1 = bid</li> </ul>
<b>price</b>	The order price.
<b>size</b>	The order size.

## updateNewsBulletin()

This method is triggered for each new bulletin if the client has subscribed (i.e. by calling the reqNewsBulletins() method).

```
void updateNewsBulletin(int msgId, int msgType, String message, String
origExchange)
```

Parameter	Description
<b>msgId</b>	The bulletin ID, incrementing for each new bulletin.
<b>msgType</b>	Specifies the type of bulletin. Valid values include: <ul style="list-style-type: none"> <li>• 1 = Regular news bulletin</li> <li>• 2 = Exchange no longer available for trading</li> <li>• 3 = Exchange is available for trading</li> </ul>
<b>message</b>	The bulletin's message text.
<b>origExchange</b>	The exchange from which this message originated.

## receiveFA()

This method receives previously requested FA configuration information from TWS.

```
receiveFA(long faDataType, string xml)
```

Parameter	Description
<b>faDataType</b>	Specifies the type of Financial Advisor configuration data being received from TWS. Valid values include: <ul style="list-style-type: none"> <li>• 1 = GROUPS</li> <li>• 2 = PROFILE</li> <li>• 3 =ACCOUNT ALIASES</li> </ul>
<b>xml</b>	The XML string containing the previously requested FA configuration information.

## historicalData()

This method receives the requested historical data results.

```
void historicalData (int reqId, String date, double open, double high,
double low, double close, int volume, int count, double WAP, boolean
hasGaps)
```

Parameter	Description
<b>reqId</b>	The ticker Id of the request to which this bar is responding.
<b>date</b>	The date-time stamp of the start of the bar. The format is determined by the reqHistoricalData() formatDate parameter.
<b>open</b>	The bar opening price.
<b>high</b>	The high price during the time covered by the bar.
<b>low</b>	The low price during the time covered by the bar.
<b>close</b>	The bar closing price.
<b>volume</b>	The volume during the time covered by the bar.
<b>count</b>	When TRADES historical data is returned, represents the number of trades that occurred during the time period the bar covers
<b>WAP</b>	The weighted average price during the time covered by the bar.
<b>hasGaps</b>	Whether or not there are gaps in the data.

## **scannerParameters()**

This method receives an XML document that describes the valid parameters that a scanner subscription can have.

```
void scannerParameters(String xml)
```

<b>Parameter</b>	<b>Description</b>
<b>xml</b>	A document describing available scanner subscription parameters.

## **scannerData()**

This method receives the requested market scanner data results.

```
void scannerData(int reqId, int rank, ContractDetails contractDetails,
String distance, String benchmark, String projection, String legsStr)
```

<b>Parameter</b>	<b>Description</b>
<b>reqId</b>	The ID of the request to which this row is responding.
<b>rank</b>	The ranking within the response of this bar.
<b>contractDetails</b>	This structure contains a full description of the contract that was executed.
<b>distance</b>	Varies based on query.
<b>benchmark</b>	Varies based on query.
<b>projection</b>	Varies based on query.
<b>legsStr</b>	Describes combo legs when scan is returning EFP.

## **scannerDataEnd()**

This method is called when the snapshot is received and marks the end of one scan.

```
void scannerDataEnd(int reqId)
```

<b>Parameter</b>	<b>Description</b>
<b>reqId</b>	The ID of the market data snapshot request being closed by this parameter.

## realtimeBar()

This method receives the real-time bars data results.

```
void realtimeBar(int reqId, long time, double open, double high, double low, double close, long volume, double wap, int count)
```

Parameter	Description
<b>reqId</b>	The ticker ID of the request to which this bar is responding.
<b>time</b>	The date-time stamp of the start of the bar. The format is determined by the reqHistoricalData() formatDate parameter.
<b>open</b>	The bar opening price.
<b>high</b>	The high price during the time covered by the bar.
<b>low</b>	The low price during the time covered by the bar.
<b>close</b>	The bar closing price.
<b>volume</b>	The volume during the time covered by the bar.
<b>wap</b>	The weighted average price during the time covered by the bar.
<b>count</b>	When TRADES historical data is returned, represents the number of trades that occurred during the time period the bar covers.

## currentTime()

This method receives the current system time on the server side.

```
void currentTime(long time)
```

Parameter	Description
<b>time</b>	The current system time on the server side

## fundamentalData()

This method is called to receive Reuters global fundamental market data. There must be a subscription to Reuters Fundamental set up in Account Management before you can receive this data.

```
void fundamentalData(int reqId, String data)
```

Parameter	Description
<b>reqId</b>	The ID of the data request.
<b>data</b>	One of three XML reports: <ul style="list-style-type: none"> <li>• Estimates (estimates)</li> <li>• Financial statements (finstat)</li> <li>• Summary (snapshot)</li> </ul>

## Java SocketClient Properties

The tables below define attributes for the following classes:

- [\*\*Execution\*\*](#)
- [\*\*ExecutionFilter\*\*](#)
- [\*\*Contract\*\*](#)
- [\*\*ContractDetails\*\*](#)
- [\*\*ComboLeg\*\*](#)
- [\*\*Order\*\*](#)
- [\*\*OrderState\*\*](#)
- [\*\*ScannerSubscription\*\*](#)
- [\*\*UnderComp\*\*](#)

## Execution

Attribute	Description
int m_orderId	The order id. <b>Note:</b> TWS orders have a fixed order id of "0."
int m_clientId	The id of the client that placed the order. Note: TWS orders have a fixed client id of "0."
String m_execId	Unique order execution id.
String m_time	The order execution time.
String m_acctNumber	The customer account number.
String m_exchange	Exchange that executed the order.
String m_side	Specifies if the transaction was a sale or a purchase. Valid values are: <ul style="list-style-type: none"> <li>• BOT</li> <li>• SLD</li> </ul>
int m_shares	The number of shares filled.
double m_price	The order execution price.
int m_permId	The TWS id used to identify orders, remains the same over TWS sessions.
int m_liquidation	Identifies the position as one to be liquidated last should the need arise.
int m_cumQty	Cumulative quantity. Used in regular trades, combo trades and legs of the combo.
double m_avgPrice	Average price. Used in regular trades, combo trades and legs of the combo.

## ExecutionFilter

Attribute	Description
int m_clientId	Filter the results of the reqExecutions() method based on the clientId.
String m_acctCode	Filter the results of the reqExecutions() method based on an account code. Note: this is only relevant for Financial Advisor (FA) accounts.
String m_time	Filter the results of the reqExecutions() method based on execution reports received after the specified time. The format for timeFilter is "yyyymmdd-hh:mm:ss"
String m_symbol	Filter the results of the reqExecutions() method based on the order symbol.
String m_secType	Filter the results of the reqExecutions() method based on the order security type. Note: Refer to the Contract struct for the list of valid security types.

Attribute	Description
String m_exchange	Filter the results of the reqExecutions() method based on the order exchange.
String m_side	Filter the results of the reqExecutions() method based on the order action. Note: Refer to the Order class for the list of valid order actions.

## Contract

Attribute	Description
String m_symbol	This is the symbol of the underlying asset.
String m_secType	This is the security type. Valid values are: <ul style="list-style-type: none"> <li>• STK</li> <li>• OPT</li> <li>• FUT</li> <li>• IND</li> <li>• FOP</li> <li>• CASH</li> <li>• BAG</li> </ul>
String m_expiry	The expiration date. Use the format YYYYMM.
double m_strike	The strike price.
String m_right	Specifies a Put or Call. Valid values are: P, PUT, C, CALL.
String m_multiplier	Allows you to specify a future or option contract multiplier. This is only necessary when multiple possibilities exist.
String m_exchange	The order destination, such as Smart.
String m_currency	Specifies the currency. Ambiguities may require that this field be specified, for example, when SMART is the exchange and IBM is being requested (IBM can trade in GBP or USD). Given the existence of this kind of ambiguity, it is a good idea to always specify the currency.
String m_localSymbol	This is the local exchange symbol of the underlying asset.
String m_primaryExch	Identifies the listing exchange for the contract (do not list SMART).
boolean m_includeExpired	If set to true, contract details requests and historical data queries can be performed pertaining to expired contracts.  Note: Historical data queries on expired contracts are limited to the last year of the contracts life, and are initially only supported for expired futures contracts,
String m_comboLegsDescrip	Description for combo legs
Vector m_comboLegs	Dynamic memory structure used to store the leg definitions for this contract.
int m_conId	The unique contract identifier.

<b>Attribute</b>	<b>Description</b>
String m_secIdType	Security identifier, when querying contract details or when placing orders. Supported identifiers are: <ul style="list-style-type: none"><li>• ISIN (Example: Apple: US0378331005)</li><li>• CUSIP (Example: Apple: 037833100)</li><li>• SEDOL (Consists of 6-AN + check digit. Example: BAE: 0263494)</li><li>• RIC (Consists of exchange-independent RIC Root and a suffix identifying the exchange. Example: AAPL.O for Apple on NASDAQ.)</li></ul>
String m_secId	Unique identifier for the secIdType.

## ContractDetails

Attribute	Description
Contract m_summary	A contract summary.
String m_marketName	The market name for this contract.
String m_tradingClass	The trading class name for this contract.
double m_minTick	The minimum price tick.
String m_priceMagnifier	Allows execution and strike prices to be reported consistently with market data, historical data and the order price, i.e. Z on LIFFE is reported in index points and not GBP.
String m_orderTypes	The list of valid order types for this contract.
String m_validExchanges	The list of exchanges this contract is traded on.
String m_underConId	The underlying contract ID.
String m_longName	Descriptive name of the asset.
String m_cusip	For Bonds. The nine-character bond CUSIP or the 12-character SEDOL.
String m_ratings	For Bonds. Identifies the credit rating of the issuer. A higher credit rating generally indicates a less risky investment. Bond ratings are from Moody's and S&P respectively.
String m_descAppend	For Bonds. A description string containing further descriptive information about the bond.
String m_bondType	For Bonds. The type of bond, such as "CORP."
String m_couponType	For Bonds. The type of bond coupon.
boolean m_callable	For Bonds. Values are True or False. If true, the bond can be called by the issuer under certain conditions.
boolean m_putable	For Bonds. Values are True or False. If true, the bond can be sold back to the issuer under certain conditions.
double m_coupon	For Bonds. The interest rate used to calculate the amount you will receive in interest payments over the course of the year.
boolean m_convertible	For Bonds. Values are True or False. If true, the bond can be converted to stock under certain conditions.
String m_maturity	For Bonds. The date on which the issuer must repay the face value of the bond.
String m_issueDate	For Bonds. The date the bond was issued.
String m_nextOptionDate	For Bonds, only if bond has embedded options.
String m_nextOptionType	For Bonds, only if bond has embedded options.
boolean m_nextOptionPartial	For Bonds, only if bond has embedded options.
String m_notes	For Bonds, if populated for the bond in IB's database
String m_contractMonth	The contract month. Typically the contract month of the underlying for a futures contract.

Attribute	Description
String m_industry	The industry classification of the underlying/product. For example, Financial.
String m_category	The industry category of the underlying. For example, InvestmentSvc.
String m_subcategory	The industry subcategory of the underlying. For example, Brokerage.
String m_timeZoneId	The ID of the time zone for the trading hours of the product. For example, EST.
String m_tradingHours	The trading hours of the product. For example, 20090507:0700-1830,1830-2330;20090508:CLOSED.
String m_liquidHours	The liquid trading hours of the product. For example, 20090507:0930-1600;20090508:CLOSED.

## ComboLeg

Attribute	Description
int m_conId	The unique contract identifier specifying the security.
String m_action	The side (buy or sell) for the leg you are constructing.
int m_ratio	Select the relative number of contracts for the leg you are constructing. To help determine the ratio for a specific combination order, refer to the Interactive Analytics section of the User's Guide.
String m_exchange	The exchange to which the complete combination order will be routed.
int m_openClose	Specifies whether the order is an open or close order. Valid values are: <ul style="list-style-type: none"> <li>• 0 - Same as the parent security. This is the only option for retail customers.</li> <li>• 1 - Open. This value is only valid for institutional customers.</li> <li>• 2 - Close. This value is only valid for institutional customers.</li> <li>• Unknown - (3)</li> </ul>
int m_shortSaleSlot	For institutional customers only. <ul style="list-style-type: none"> <li>• 0 - inapplicable (i.e. retail customer or not short leg)</li> <li>• 1 - clearing broker</li> <li>• 2 - third party. If this value is used, you must enter a designated location.</li> </ul>
String m_designatedLocation	If shortSaleSlot == 2, the designatedLocation must be specified. Otherwise leave blank or orders will be rejected.

## Order

Attribute	Description
int m_orderId	The id for this order.
int m_clientId	The id of the client that placed this order.
int m_permid	The TWS id used to identify orders, remains the same over TWS sessions.
String m_action	Identifies the side. Valid values are: BUY, SELL, SSHORT.
long m_totalQuantity	The order quantity.
String m_orderType	Identifies the order type. Valid values are: <ul style="list-style-type: none"> <li>• MKT</li> <li>• MKTCLS</li> <li>• LMT</li> <li>• LMTCLS</li> <li>• PEGMKT</li> <li>• SCALE</li> <li>• STP</li> <li>• STPLMT</li> <li>• TRAIL</li> <li>• REL</li> <li>• VWAP</li> <li>• TRAILLIMIT</li> </ul>
double m_lmtPrice	This is the LIMIT price, used for limit, stop-limit and relative orders. In all other cases specify zero. For relative orders with no limit price, also specify zero.
double m_auxPrice	This is the STOP price for stop-limit orders, and the offset amount for relative orders. In all other cases, specify zero.
String m_tif	The time in force. Valid values are: DAY, GTC, IOC, GTD.
String m_ocaGroup	Identifies an OCA (one cancels all) group.

Attribute	Description
int m_ocaType	<p>Tells how to handle remaining orders in an OCA group when one order or part of an order executes. Valid values include:</p> <ul style="list-style-type: none"> <li>• 1 = Cancel all remaining orders with block</li> <li>• 2 = Remaining orders are proportionately reduced in size with block</li> <li>• 3 = Remaining orders are proportionately reduced in size with no block</li> </ul> <p>If you use a value "with block" gives your order has overfill protection. This means that only one order in the group will be routed at a time to remove the possibility of an overfill.</p>
String m_account	The account. For institutional customers only.
String m_openClose	Specifies whether the order is an open or close order. For institutional customers only. Valid values are O, C.
int m_origin	The order origin. For institutional customers only. Valid values are 0 = customer, 1 = firm
String m_orderRef	The order reference. For institutional customers only.
bool m_transmit	Specifies whether the order will be transmitted by TWS. If set to false, the order will be created at TWS but will not be sent.
int m_parentId	The order ID of the parent order, used for bracket and auto trailing stop orders.
boolean m_blockOrder	If set to true, specifies that the order is an ISE Block order.
boolean m_sweepToFill	If set to true, specifies that the order is a Sweep-to-Fill order.
int m_displaySize	The publicly disclosed order size, used when placing Iceberg orders.

Attribute	Description
int m_triggerMethod	<p>Specifies how Simulated Stop, Stop-Limit and Trailing Stop orders are triggered. Valid values are:</p> <ul style="list-style-type: none"> <li>• <b>0</b> - The default value. The "double bid/ask" method will be used for orders for OTC stocks and US options. All other orders will used the "last" method.</li> <li>• <b>1</b> - use "double bid/ask" method, where stop orders are triggered based on two consecutive bid or ask prices.</li> <li>• <b>2</b> - "last" method, where stop orders are triggered based on the last price.</li> <li>• <b>3</b> - double last method.</li> <li>• <b>4</b> - bid/ask method.</li> <li>• <b>7</b> - last or bid/ask method.</li> <li>• <b>8</b> - mid-point method.</li> </ul>
boolean m_outsideRth	If set to true, allows orders to also trigger or fill outside of regular trading hours.
boolean m_hidden	If set to true, the order will not be visible when viewing the market depth. This option only applies to orders routed to the ISLAND exchange.
double m_discretionaryAmt	The amount off the limit price allowed for discretionary orders.
String m_goodAfterTime	<p>The trade's "Good After Time," format "YYYYMMDD hh:mm:ss (optional time zone)" Use an empty String if not applicable.</p>
String m_goodTillDate	<p>You must enter a tif value of GTD. The trade's "Good Till Date," format is: YYYYMMDD hh:mm:ss (optional time zone) Use an empty String if not applicable.</p>
String m_faGroup	The Financial Advisor group the trade will be allocated to -- use an empty String if not applicable.
String m_faProfile	The Financial Advisor allocation profile the trade will be allocated to -- use an empty String if not applicable.
String m_faMethod	The Financial Advisor allocation method the trade will be allocated with -- use an empty String if not applicable.
String m_faPercentage	The Financial Advisor percentage concerning the trade's allocation -- use an empty String if not applicable.
int m_shortSaleSlot	Values are 1 or 2.
String m_designatedLocation	Use only when shortSaleSlot value = 2.
int m_ocaType	Cancel on Fill with Block = 1 Reduce on Fill with Block = 2 Reduce on Fill without Block = 3

Attribute	Description
int m_overridePercentageConstraints	<p>Precautionary constraints are defined on the TWS Presets page, and help ensure that your price and size order values are reasonable. Orders sent from the API are also validated against these safety constraints, and may be rejected if any constraint is violated. To override validation, set this parameter's value to True. Valid values include:</p> <ul style="list-style-type: none"> <li>• 0 = False</li> <li>• 1 = True</li> </ul>
string m_rule80A	Valid values are: <ul style="list-style-type: none"> <li>• Individual = 'I'</li> <li>• Agency = 'A',</li> <li>• AgentOtherMember = 'W'</li> <li>• IndividualPTIA = 'J'</li> <li>• AgencyPTIA = 'U'</li> <li>• AgentOtherMemberPTIA = 'M'</li> <li>• IndividualPT = 'K'</li> <li>• AgencyPT = 'Y'</li> <li>• AgentOtherMemberPT = 'N'</li> </ul>
string m_settlingFirm	Institutional only.
string m_clearingAccount	For IBExecution customers: Specifies the true beneficiary of the order. This value is required for FUT/FOP orders for reporting to the exchange.
string m_clearingIntent	For IBExecution customers: Valid values are: IB, Away, and PTA (post trade allocation).
boolean m_allOrNone	yes=1, no=0
int m_minQty	Identifies a minimum quantity order type.
double m_percentOffset	The percent offset amount for relative orders.
boolean m_eTradeOnly	Trade with electronic quotes. yes = 1, no = 0
boolean m_firmQuoteOnly	Trade with firm quotes. yes = 1, no = 0
double m_nbboPriceCap	The maximum Smart order distance from the NBBO.
int m_auctionStrategy	<p>Valid values are:</p> <ul style="list-style-type: none"> <li>• match = 1</li> <li>• improvement = 2</li> <li>• transparent = 3</li> </ul> <p>For BOX exchange only.</p>
double m_startingPrice	The starting price. Valid on BOX orders only.

Attribute	Description
double m_stockRefPrice	The stock reference price. The reference price is used for VOL orders to compute the limit price sent to an exchange (whether or not Continuous Update is selected), and for price range monitoring.
double m_delta	The stock delta. Valid on BOX orders only.
double m_stockRangeLower	The lower value for the acceptable underlying stock price range. For price improvement option orders on BOX and VOL orders with dynamic management.
double m_stockRangeUpper	The upper value for the acceptable underlying stock price range. For price improvement option orders on BOX and VOL orders with dynamic management.
double m_volatility	What the price is, computed via TWSs Options Analytics. For VOL orders, the limit price sent to an exchange is not editable, as it is the output of a function. Volatility is expressed as a percentage.
int m_volatilityType	How the volatility is calculated. <ul style="list-style-type: none"> <li>• Daily = 1</li> <li>• Annual = 2</li> </ul>
string m_deltaNeutralOrderType	VOL orders only. Enter an order type to instruct TWS to submit a delta neutral trade on full or partial execution of the VOL order. For no hedge delta order to be sent, specify NONE.
int m_deltaNeutralAuxPrice	VOL orders only. Use this field to enter a value if the value in the <i>deltaNeutralOrderType</i> field is an order type that requires an Aux price, such as a REL order.
int m_continuousUpdate	Used for dynamic management of volatility orders. Determines whether TWS is supposed to update the order price as the underlying moves. If selected, the limit price sent to an exchange is modified by TWS if the computed price of the option changes enough to warrant doing so. This is very helpful in keeping the limit price sent to the exchange up to date as the underlying price changes.
int m_referencePriceType	Used for dynamic management of volatility orders. Set to <ul style="list-style-type: none"> <li>• 1 = Average of National Best Bid or Ask, or set to</li> <li>• 2 = National Best Bid when buying a call or selling a put; and National Best Ask when selling a call or buying a put.</li> </ul>
double m_trailStopPrice	For TRAILLIMIT orders only
int m_scaleInitLevelSize	For Scale orders: Defines the size of the first, or initial, order component.

Attribute	Description
int m_scaleSubsLevelSize	For Scale orders: Defines the order size of the subsequent scale order components. Used in conjunction with scaleInitLevelSize().
double m_scalePriceIncrement	For Scale orders: Defines the price increment between scale components. This field is required.
double m_basisPoints	For EFP orders only
int m_basisPointsType	For EFP orders only
boolean m_whatIf	Use to request pre-trade commissions and margin information. If set to true, margin and commissions data is received back via the OrderState() object for the openOrder() callback.

## OrderState

Attribute	Description
string m_status	Displays the order status.
String m_initMargin	Shows the impact the order would have on your initial margin.
String m_maintMargin	Shows the impact the order would have on your maintenance margin.
String m_equityWithLoan	Shows the impact the order would have on your equity with loan value.
double m_commission	Shows the commission amount on the order.
double m_minCommission	Used in conjunction with the maxCommission field, this defines the lowest end of the possible range into which the actual order commission will fall.
double m_maxCommission	Used in conjunction with the minCommission field, this defines the highest end of the possible range into which the actual order commission will fall.
String m_commissionCurrency	Shows the currency of the commission value.
String m_warningText	Displays a warning message if warranted.

## ScannerSubscription

Attribute	Description
int m_numberOfRows	Defines the number of rows of data to return for a query.
String m_instrument	Defines the instrument type for the scan.
String m_locationCode	The location, currently the only valid location is US stocks.
String m_scanCode	Can be left blank.
double m_abovePrice	Filter out contracts with a price lower than this value. Can be left blank.
double m_belowPrice	Filter out contracts with a price higher than this value. Can be left blank.
int m_aboveVolume	Filter out contracts with a volume lower than this value. Can be left blank.
double m_marketCapAbove	Filter out contracts with a market cap lower than this value. Can be left blank.
double m_marketCapBelow	Filter out contracts with a market cap above this value. Can be left blank.
String m_moodyRatingAbove	Filter out contracts with a Moody rating below this value. Can be left blank.
String m_moodyRatingBelow	Filter out contracts with a Moody rating above this value. Can be left blank.
String m_spRatingAbove	Filter out contracts with an S&P rating below this value. Can be left blank.
String m_spRatingBelow	Filter out contracts with an S&P rating above this value. Can be left blank.
String m_maturityDateAbove	Filter out contracts with a maturity date earlier than this value. Can be left blank.
String m_maturityDateBelow	Filter out contracts with a maturity date later than this value. Can be left blank.
double m_couponRateAbove	Filter out contracts with a coupon rate lower than this value. Can be left blank.
double m_couponRateBelow	Filter out contracts with a coupon rate higher than this value. Can be left blank.
String m_excludeConvertible	Filter out convertible bonds. Can be left blank.
String m_scannerSettingPairs	Can leave empty. For example, a pairing "Annual, true" used on the "top Option Implied Vol % Gainers" scan would return annualized volatilities.
int m_averageOptionVolumeAbove	Can leave empty.

Attribute	Description
String m_stockTypeFilter	Valid values are: <ul style="list-style-type: none"><li>• ALL (excludes nothing)</li><li>• STOCK (excludes ETFs)</li><li>• ETF (includes ETFs)</li></ul>

## UnderComp

Attribute	Description
int m_conId	The unique contract identifier specifying the security. Used for Delta-Neutral Combo contracts.
double m_delta	The underlying stock or future delta. Used for Delta-Neutral Combo contracts.
double m_price	The price of the underlying. Used for Delta-Neutral Combo contracts.

## Placing a Combination Order

A combination order is a special type of order that is constructed of many separate legs but executed as a single transaction. Submit combo orders such as calendar spreads, conversions and straddles using the BAG security type (defined in the *Contract* object). The key to implementing a successful API combination order using the API is to knowing how to place the same order using Trader Workstation. If you are familiar with placing combination orders in TWS, then it will be easier to place the same order using the API, because the API only imitates the behavior of TWS.

### Example

In this example, a customer places a BUY order on a calendar spread for GOOG. To buy one calendar spread means:

**Leg 1: Sell 1 GOOG OPT SEP 18 '09 150.0 CALL (100)**

**Leg 2: Buy 1 GOOG OPT JAN 21 '11 150.0 CALL (100)**

Here is a summary of the steps required to place a combo order using the API:

- Obtain the contract id (conId) for each leg. Get this number by invoking the [reqContractDetails\(\)](#) method.
- Include each leg on the [ComboLeg](#) object by populating the related fields.
- Implement the [placeOrder\(\)](#) method with the [Contract](#) and [Order](#) socket client properties.

### To place this combo order

- 1** Get the Contract IDs for both leg definitions:

```
//First leg  
  
Contract con1 = new Contract();  
  
con1.m_symbol = "GOOG";  
con1.m_secType = "OPT";  
con1.m_expiry = "200909";  
con1.m_strike = 150.0  
con1.m_right = "C"  
con1.m_multiplier = "100"  
con1.m_exchange = "SMART";  
con1.m_currency = "USD";  
  
.reqContractDetails(1, con1);
```

```

//Second leg

Contract con2 = new Contract();

con2.m_symbol = "GOOG";
con2.m_secType = "OPT";
con2.m_expiry = "201101";
con2.m_strike = 150.0
con2.m_right = "C"
con2.m_multiplier = "100"
con2.m_exchange = "SMART";
con2.m_currency = "USD";

.reqContractDetails(2, con2);

//All conId numbers are delivered by the ContractDetail()

static public String contractDetails(int reqId, ContractDetails
contractDetails) {

Contract contract = contractDetails.m_summary;

/*Base on the request above,
reqId = 1 is corresponding to the first request or first leg
reqId = 2 is corresponding to the second request or second leg*/

if (reqId == 1)
{ Leg1_conId = contract.m_conId;} // to obtain conId for first leg

if (reqId == 2)
{ Leg2_conId = contract.m_conId;} // to obtain conId for second
leg
}

```

- 2** Once the program has acquired the conId value for each leg, include it in the ComboLeg object:

```

ComboLeg leg1 = new ComboLeg(); // for the first leg
ComboLeg leg2 = new ComboLeg(); // for the second leg
Vector addAllLegs = new Vector();

leg1.m_conId = Leg1_conId;
leg1.m_ratio = 1;
leg1.m_action = "SELL";
leg1.m_exchange = "SMART";
leg1.m_openClose = 0;
leg1.m_shortSaleSlot = 0;
leg1.m_designatedLocation = "";

```

```
leg2.m_conId = Leg2_conId;
leg2.m_ratio = 1;
leg2.m_action = "BUY";
leg2.m_exchange = "SMART";
leg2.m_openClose = 0;
leg2.m_shortSaleSlot = 0;
leg2.m_designatedLocation = "";  
  
addAllLegs.add(leg1);
addAllLegs.add(leg2);
```

- 3** Invoke the `placeOrder()` method with the appropriate contract and order objects:

```
Contract contract = new Contract();
Order order = new Order();  
  
contract.m_symbol = "USD";      // For combo order use "USD" as the
                                symbol value all the time
contract.m_secType = "BAG";    // BAG is the security type for
                                COMBO order
contract.m_exchange = "SMART";
contract.m_currency = "USD";
contract.m_comboLegs = addAllLegs; //including combo order in
                                contract object  
  
order.m_action = "BUY";
order.m_totalQuantity = 1;
order.m_orderType = "MKT"
.placeOrder(OrderId, contract, order);
```

**Note:** For more information on combination orders, see the TWS Users Guide topics [About Combination Orders](#) and [Notes on Combination Orders](#).

# Advisors

This chapter describes API functionality for users with Financial Advisor accounts, including the following topics:

- [\*\*Financial Advisor Orders and Account Configuration\*\*](#)
- [\*\*Excel DDE Support\*\*](#)
- [\*\*Support by Other API Technologies\*\*](#)
- [\*\*Improved Financial Advisor Execution Reporting\*\*](#)
- [\*\*Allocation Methods for Account Groups\*\*](#)

## Financial Advisor Orders and Account Configuration

This section assumes familiarity on the part of the reader with TWS Financial Advisor account configuration and order placement.

API FA functionality became significantly more powerful in TWS release 821 and higher, in that the now deprecated "allocation string" method was replaced by the much more powerful Financial Advisor order allocation methods. Prior to those new methods being used, TWS had to be configured to understand the desired FA order groups, profiles, and account aliases. This can be done manually in TWS, or via the API, or via both.

### Excel DDE Support

Starting with TWS release 824, DDE orders now have six Extended Order Attributes: *Good After Time*, *Good Till Date*, *FA Group*, *FA Method*, *FA Percentage*, and *FA Profile*. These can be left empty if they do not apply to an order. TWS Financial Advisor account configuration should be done manually for DDE access.

You can place FA orders on the Advisors page in the most recent release of the TwsDde.xls DDE for Excel API spreadsheet. For more information, see the [Advisors Page](#) topic.

### Support by Other API Technologies

For all ActiveX, Java, or C++ based API technologies, TWS Financial Advisor account configuration is done via two new methods and one new event. The methods are called *replaceFA*, and *requestFA*. The event is called *receiveFA*. These methods and that event pertain to the following three parts of TWS FA account configuration: creating groups, profiles, and account aliases.

- *requestFA(int faDataType)* is a method that is called by an API application to request one of those types of FA configuration data.
- *receiveFA(int faDataType, string XML)* receives the requested data from TWS, via an event that TWS sends that contains the data requested. The event includes an XML string containing the requested information.
- *replaceFA(int faDataType, string XML)* can be called from the API if the API application wishes to replace the previous FA configuration information with a new XML string.

In accordance with the existence of this new functionality, all *placeOrder* methods, whether ActiveX, Java, or C++ based, have four new parameters pertaining to Financial Advisor order placement: *faGroup*, *faMethod*, *faPercentage*, and *faProfile*. When one or more of these new values is not relevant to an order, simply pass in an empty string.

## Improved Financial Advisor Execution Reporting

When using TWS version 823 or higher, the execution messages resulting from a new FA order will report both the initial execution of the order, as well as its being allocated to its various subaccounts. The following example will serve to explain the new reporting scheme:

Assume that 100 shares of IBM is being bought on the NYSE by a Financial Advisor who has three sub-accounts, and who wants them allocated with Equal Quantity to each. The following seven execution messages will occur:

- FA Account: Order filled on NYSE to BUY 100 IBM
- FA Account: allocation of 34 shares out of FA account and into sub account 1. Message says "BUY -34 IBM." The negative quantity reflects the fact that the execution being reported is reducing the purchase.
- SUB1 Account: BUY 34 IBM.
- FA Account: allocation of 33 shares out of FA account and into sub account 2. Message says "BUY -33 IBM."
- SUB2 Account: BUY 33 IBM.
- FA Account: allocation of 33 shares out of FA account and into sub account 3. Message says "BUY -33 IBM."
- SUB3 Account: BUY 33 IBM."

## Allocation Methods for Account Groups

Note that you must type the method name in exactly as appears here, or your order won't work.

### **EqualQuantity Method**

Requires you to specify an order size. This method distributes shares equally between all accounts in the group.

Example: You transmit an order for 400 shares of stock ABC. If your Account Group includes four accounts, each account receives 100 shares. If your Account Group includes six accounts, each account receives 66 shares, and then 1 share is allocated to each account until all are distributed.

### **NetLiq Method**

Requires you to specify an order size. This method distributes shares based on the net liquidation value of each account. The system calculates ratios based on the Net Liquidation value in each account and allocates shares based on these ratios.

Example: You transmit an order for 700 shares of stock XYZ. The account group includes three accounts, A, B and C with Net Liquidation values of \$25,000, \$50,000 and \$100,000 respectively. The system calculates a ratio of 1:2:4 and allocates 100 shares to Client A, 200 shares to Client B, and 400 shares to Client C.

### **AvailableEquity Method**

Requires you to specify an order size. This method distributes shares based on the amount of equity with loan value currently available in each account. The system calculates ratios based on the Equity with Loan value in each account and allocates shares based on these ratios.

Example: You transmit an order for 700 shares of stock XYZ. The account group includes three accounts, A, B and C with available equity in the amounts of \$25,000, \$50,000 and \$100,000 respectively. The system calculates a ratio of 1:2:4 and allocates 100 shares to Client A, 200 shares to Client B, and 400 shares to Client C.

### **PctChange Method**

This method only works when you already hold a position in the selected instrument. Do not specify an order size. Since the quantity is calculated by the system, the order size is displayed in the Quantity field after the order is acknowledged. This method increases or decreases an already existing position. Positive percents will increase a position, negative percents will decrease a position.

**Example 1:** Assume that three of the six accounts in this group hold long positions in stock XYZ. Client A has 100 shares, Client B has 400 shares, and Client C has 200 shares. You want to increase their holdings by 50%, so you enter "50" in the percentage field. The system calculates that your order size needs to be equal to 350 shares. It then allocates 50 shares to Client A, 200 shares to Client B, and 100 shares to Client C.

**Example 2:** You want to close out all long positions for three of the five accounts in a group. You create a sell order and enter "-100" in the Percentage field. The system calculates 100% of

each position for every account in the group that holds a position, and sells all shares to close the positions.

These handy charts make it easy to see how negative and positive percent values will affect long and short positions for both buy and sell orders. Phew, that was a mouthful!

<b>BUY ORDER</b>	<b>Positive Percent</b>	<b>Negative Percent</b>
<b>Long Position</b>	<b>Increases position</b>	<b>No effect</b>
<b>Short Position</b>	<b>No effect</b>	<b>Decreases position</b>

<b>SELL ORDER</b>	<b>Positive Percent</b>	<b>Negative Percent</b>
<b>Long Position</b>	<b>No effect</b>	<b>Decreases position</b>
<b>Short Position</b>	<b>Increases position</b>	<b>No effect</b>



# ActiveX for Excel

This chapter describes the ActiveX for Excel sample spreadsheet, including the following topics:

- [Getting Started with the ActiveX for Excel API](#)
- [Using the ActiveX for Excel Sample Spreadsheet](#)

The ActiveX for Excel sample spreadsheet, **TwsActiveX.xls**, duplicates the functionality of the ActiveX for Excel API spreadsheet but internally uses an ActiveX component, **Tws.ocx**. One of the benefits of using this spreadsheet is that it can connect to a TWS that is running on a remote PC. The DDE for Excel API spreadsheet cannot do this.

The methods, events and COM objects used in the code for the ActiveX for Excel sample spreadsheet are the same as those used in the ActiveX API. See the [ActiveX](#) chapter for complete details about the ActiveX API.

The following figure shows the Tickers page in the ActiveX for Excel API sample spreadsheet.

The screenshot shows the Microsoft Excel interface with the title bar "TwsActiveX.xls [Compatibility Mode] - Microsoft Excel". The ribbon menu includes Home, Insert, Page Layout, Formulas, Data, Review, View, Developer, and Add-ins. The main worksheet is titled "IB Trader Workstation - Tickers". It contains a table with columns: Symbol, Type, Expiry, Strike, P/C, Multiplier, Exchange, Prim Exch, Currency, Combo Legs, Delta-Neutral, Subscription Status, Last Time Stamp, Bid Impl Vol, Bid Delta, Bid Exch, Bid Size, Bid Price, Ask Price, and Ask Size. The table lists various stocks like AAPL, IBM, MSFT, YHOO, GE, GOOG, AMZN, DIA, QQQQ, SPY, IWM, ALLU, and many others. At the bottom of the table, there is a summary row: Average: 1560.669276, Count: 31, Sum: 40577.40117. The status bar at the bottom right shows "75%".

## Getting Started with the ActiveX for Excel API

We have created a sample Excel spreadsheet, **TwsActiveX.xls**, that uses an ActiveX control, **Tws.ocx**. You can use this spreadsheet with TWS as is, or use it to create your own custom TWS API Excel application. It's easy to get started:

- [Download the API components](#), which includes the ActiveX for Excel sample spreadsheet, TwsActiveX.xls.
- Ensure that the application server is running and that it is [configured to support ActiveX](#).
- [Open the spreadsheet](#) and start using the ActiveX for Excel API.

The sample spreadsheet currently comprises several pages complete with sample data and action buttons that make it easy for you to get market data, send orders and view your activity.

### Download the API Components and Spreadsheet

We recommend using the sample Excel spreadsheet that we provide as a starting point toward creating your own ActiveX for Excel API. Follow the steps below to download the sample spreadsheet.

#### To install the ActiveX for Excel sample spreadsheet

- 1 From the [IB homepage](#), select *Application Programming* from the **Software** menu.
- 2 Click the *Proprietary API* tab, then In the column appropriate to your operating system, click *Download latest version*.  
**Note:** Windows users can download the beta test version of the API by using the **Windows Beta** column, or revert to the previous production version by selecting Downgrade to Previous Version.
- 3 Save the installation program to your computer, and if desired, select a different directory. Click **Save**.
- 4 Close any versions of TWS and Excel that you have running.
- 5 Locate the installation program you just saved to your computer, then double-click the file to begin the API installation.
- 6 Follow the instructions in the installation wizard.

Before you can [use the spreadsheet](#), you must have TWS running and configured to support the ActiveX API. See [Configure the Application to Support API Components](#) for detailed instructions.

## Running the ActiveX for Excel API on 64-bit Windows XP Systems

To run the ActiveX for Excel API on 64-bit Windows XP systems, do the following:

- 1 Install Microsoft Visual C++ 2005 SP1 Redistributable Package (x86).
- 2 Install Microsoft Visual J# 2.0 Redistributable Package.
- 3 Download and install the API software.

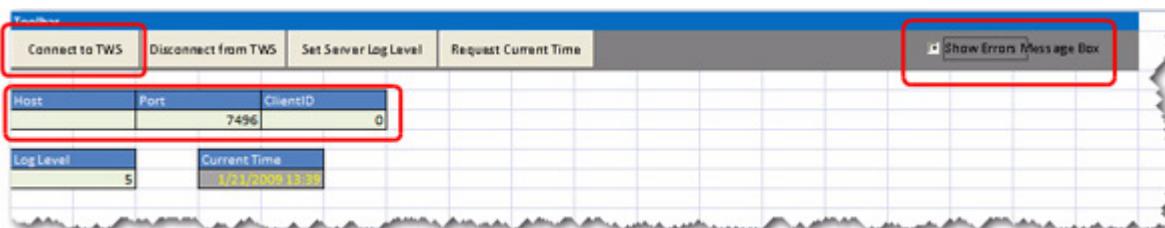
## Open the Sample Spreadsheet

After you have downloaded the sample spreadsheet and configured the application to allow the ActiveX for Excel API to link to it, open the spreadsheet and save it as your personal file.

**Note:** Note that not more than one API application can simultaneously access a single instance. The API application does not need to be running on the same computer on which the application is running.

### To open the sample spreadsheet

- 1 Go to the API installation folder in which the Excel API sample spreadsheet was installed (typically C:\Jts\Excel) and double-click **TwsActiveX.xls**.
- 2 Save the spreadsheet with a different file name. This lets you customize the spreadsheet without changing the original.
- 3 Click the **General** tab.
- 4 Modify the default values in the *Host*, *Port*, and *ClientID* cells, then click **Connect to TWS** on the Toolbar.
  - If you select the **Show Errors Message Box** check box, error messages display when you connect to TWS. In this case, you must click **OK** to dismiss any messages that appear.



## Using the ActiveX for Excel Sample Spreadsheet

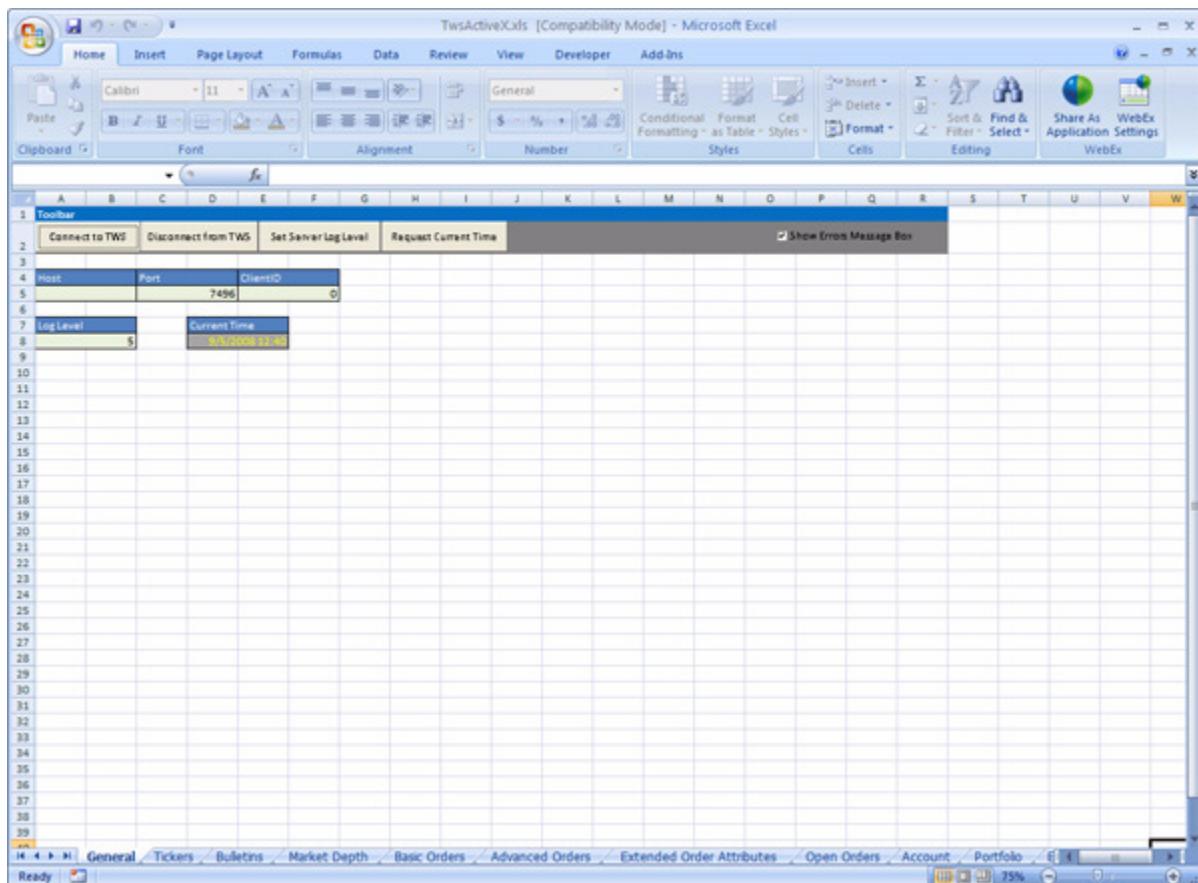
The ActiveX for Excel API sample spreadsheet, TwsDde.xls, includes the following pages (tabs):

Page	Description
<a href="#"><b>General</b></a>	Lets you connect to and disconnect from TWS, set the server log level and request the current time.
<a href="#"><b>Tickers</b></a>	Lets you set up your ticker lines and request market data. You can view market data for all asset types including EFPs and combination orders.
<a href="#"><b>Bulletins</b></a>	Lets you subscribe to and view IB News Bulletins.
<a href="#"><b>Market Depth</b></a>	Lets you view market depth for selected quotes.
<a href="#"><b>Basic Orders</b></a>	Lets you send and modify orders, set up combination orders and EFPs, and request open orders.
<a href="#"><b>Advanced Orders</b></a>	Lets you send and modify advanced orders types that require the use of extended order attributes, such as Bracket, Scale and Trailing Stop Limit orders.
<a href="#"><b>Extended Order Attributes</b></a>	Used in conjunction with the Basic Orders, Advanced Orders, Conditional Orders and Advisors pages, this page lets you change the time in force, create Hidden or Iceberg orders and apply many other order attributes.
<a href="#"><b>Open Orders</b></a>	Shows you all transmitted orders, including those that have been accepted by the IB system, and those that are working at an exchange.
<a href="#"><b>Account</b></a>	Provides up to date account information and displays your portfolio.
<a href="#"><b>Portfolio</b></a>	Displays all your current positions.
<a href="#"><b>Executions</b></a>	Lets you view all execution reports, and includes a filtering box so you can limit your results.
<a href="#"><b>Historical Data</b></a>	Request historical data for an instrument based on data you enter in a query.
<a href="#"><b>Contract Details</b></a>	Lets you collect contract-specific information you will need for other actions, including the conid and supported order types for a contract
<a href="#"><b>Bond Contract Details</b></a>	Lets you collect bond contract-specific information you will need for other actions, including bond coupon and maturity date.
<a href="#"><b>Real Time Bars</b></a>	Lets you request and view real time bars from TWS.
<a href="#"><b>Market Scanner</b></a>	Lets you view market scanner parameters and subscribe to TWS market scanners.
<a href="#"><b>Fundamentals</b></a>	Lets you request and view Fundamentals data from TWS.
<a href="#"><b>Advisors</b></a>	Lets Financial Advisors send and modify FA orders.
<a href="#"><b>Log</b></a>	Lets you view all error messages.

## General Page

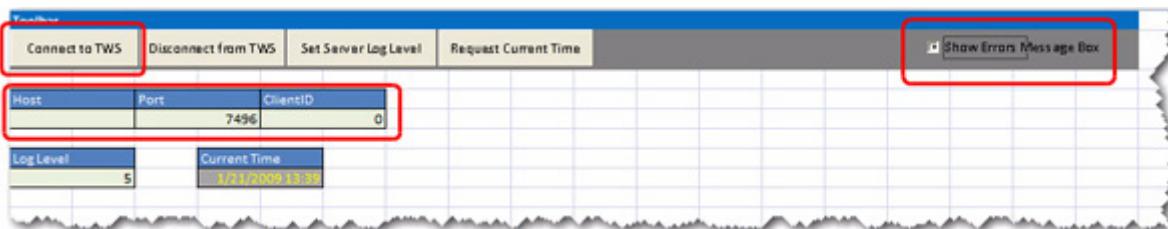
Use the General page to:

- Connect to TWS.
- Disconnect from TWS.
- Set the level of log entry detail used by the server when processing API requests.
- Request the current server time.



## To connect to TWS

- 1 Click **Connect to TWS** in the toolbar.
  - If required, change the values in the *Host*, *Port* and *ClientID* cells.
  - Select the **Show Errors Message Box** to display errors when connecting to TWS.



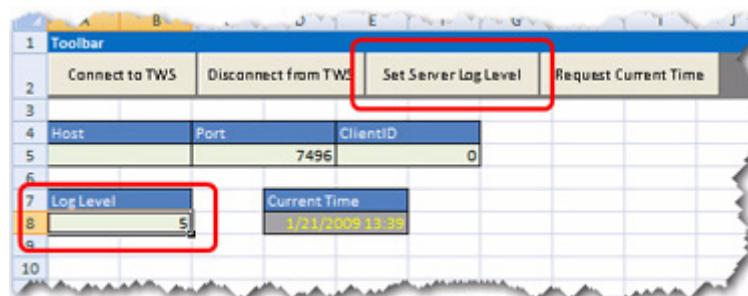
## To disconnect from TWS

- 1 Click **Disconnect from TWS** in the toolbar.

## To set the server log level

- 1 Type one of the following values in the *Log Level* cell:
  - 1 = SYSTEM
  - 2 = ERROR
  - 3 = WARNING
  - 4 = INFORMATION
  - 5 = DETAIL

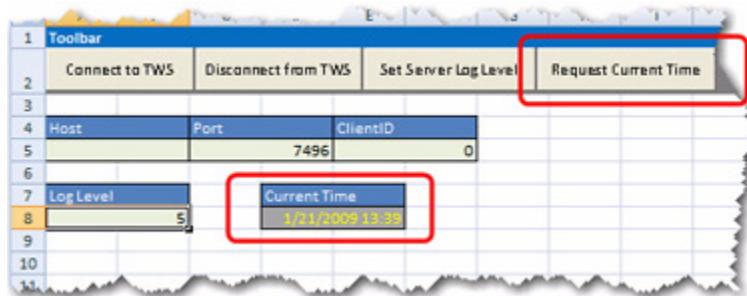
The higher the number, the greater the level of detail and performance overhead.



- 2 Click **Set Server Log Level** in the toolbar.

### To request the current time

- 1 Click **Request Current Time** in the toolbar.



### General Page Toolbar Buttons

The toolbar on the General page includes the buttons described below.

Button	Description
<b>Connect to TWS</b>	Connects to TWS.
<b>Disconnect from TWS</b>	Disconnects from TWS.
<b>Set Server Log Level</b>	Sets the level of detail of entries in the log.txt log file.
<b>Request Current Time</b>	Requests the current server time.

The toolbar also includes the **Show Errors Message Box** check box, which when selected, displays error when connecting to TWS.

## Tickers Page

Use the Tickers page to:

- Create market data (ticker) lines.
- Request market data.
- Create a combination order for options.
- Create market data line for Exchange for Physical (EFP) combination orders.

IB Trader Workstation - Tickers																			
Symbol	Type	Expiry	Strike	P/E	Multiplier	Exchange	Prim Exch	Currency	Combo Legs	Delta-Neutral	Status	Last Time Stamp	Bid Impt Vol	Bid Delta	Bid Exch	Bid Size	Bid Price	Ask Price	Ask Size
Stocks																			
12 IBM	STK					NYSE		USD			Subscribed	1/21/2009 13:27				0	3	-1	
13 MSFT	STK					SMART		USD			Subscribed	1/21/2009 13:27				12	18.67	18.7	
14 YHOO	STK					SMART	ISLAND	USD			Subscribed	1/21/2009 13:25				3	33.38	33.23	
15 GE	STK					SMART	ARCA	USD			Subscribed	1/21/2009 13:25				3	12.57	12.58	
16 GOOG	STK					SMART	ARCA	USD			Subscribed	1/21/2009 13:25				1	285.2	285.39	
17 AMZN	STK					SMART		USD			Subscribed	1/21/2009 13:24				7	48.85	49.06	
18 DIA	STK					SMART		USD			Subscribed	1/21/2009 13:20				7	80.38	80.43	
19 QQQQ	STK					SMART		USD			Subscribed	1/21/2009 13:20				70	24.29	24.3	
20 SPY	STK					SMART		USD			Subscribed	1/21/2009 13:20				84	81.66	81.66	
21 IWM	STK					SMART		USD			Subscribed	1/21/2009 13:21				1	43.73	43.77	
22 ALU	STK					SMART		USD			Subscribed	1/21/2009 13:21				1	2.05	2.03	
23 AAPL	STK					SMART		USD			Subscribed	1/21/2009 13:21				1	79.19	79.25	
24 INTC	STK					SMART		USD											
25 CSCO	STK					SMART		USD											
26 TWTR	STK					SMART		USD											
27 ORCL	STK					SMART		USD											
28 UVLT	STK					SMART		USD											
29 ETC	STK					SMART		USD											
30 LEH	STK					SMART		USD											
31 PFE	STK					SMART		USD											
32 WM	STK					SMART		USD											
33 WB	STK					SMART		USD											
34 C	STK					SMART		USD											
35 JPM	STK					SMART		USD											
36 BAC	STK					SMART		USD											
37 F	STK					SMART		USD											
38 WFC	STK					SMART		USD											
39 XLF	STK					SMART		USD											

## Using the Tickers Page

**Note:** Ensure that TWS is running, and that you have entered your user name in the *User Name* field in the *Which Trader Workstation?* section of all pages in the Excel spreadsheet to properly connect to TWS.

### To create a ticker using the Create Ticker button

- 1 Click the **Tickers** tab at the bottom of the spreadsheet.
- 2 Click the line number to the left of a blank row to select the row. You must have a blank row selected to create a ticker line.

- 3** Click the **Create Ticker** button on the toolbar and enter information in the Create Tickers dialog.
- 4** Click **OK**.

For stocks, you only need to specify the *Symbol*, *Type*, *Exchange* (usually SMART), and *Currency*.



#### To create a ticker on the spreadsheet

- 1** Select a blank cell in the *Symbol* column and enter a symbol.
- 2** Tab through the all contract description fields and enter data where necessary, for example if you are entering a stock ticker, you don't need values in the Expiry, Strike, P/C and Multiplier fields.

**Note:** The Exchange field accepts the following values: SMART (for smart order routing), and any valid exchange acronym.

#### To request market data for a ticker

- 1** Select the ticker row for which you want to request market data by clicking the row number.
- 2** Enter a comma-separated list of generic tick values in the *Generic Tick List* cell. For details about generic tick values, see [Generic Tick Types](#).
- 3** Optionally, click the **Snapshot** check box to request a single snapshot of market data.
- 4** Click **Request Market Data** on the toolbar.

To get market data for a group of tickers, select multiple ticker rows while holding down the **Shift** key, then click **Request Market Data** multiple times until all rows are showing data.

### To set the refresh rate

The market data refresh rate determines how often the link to TWS is refreshed.

- Enter the refresh rate value (in whole numbers, in seconds) in the *Market Data Refresh Rate* cell.

TWS market data updates every 3 seconds by default.

### Tickers Page Toolbar Buttons

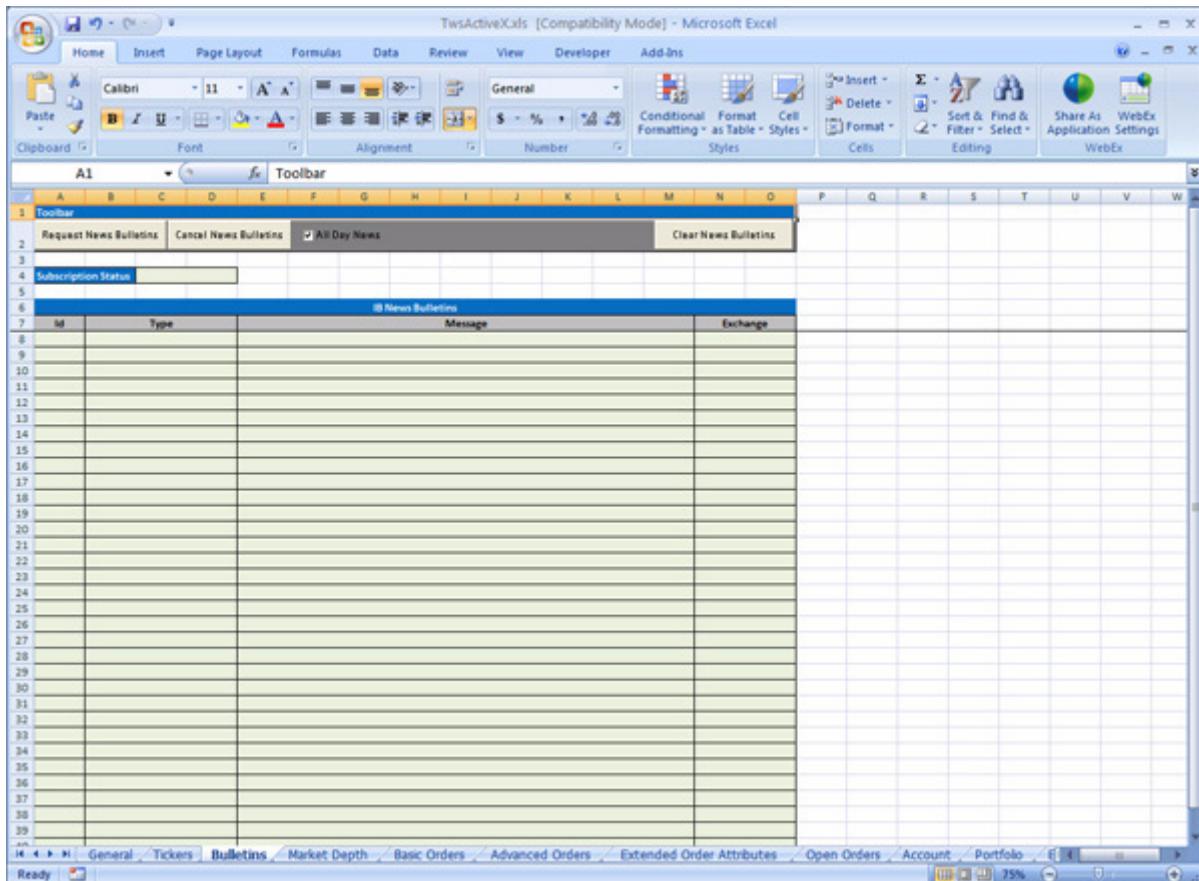
The toolbar on the Tickers page includes the buttons described below.

Button	Description
<b>Create Ticker</b>	Opens the Ticker box. Enter information to create a market data line.
<b>Combo Legs</b>	Opens the Combination Legs box. Enter contract details to create legs of a combination order one by one.
<b>Request Market Data</b>	Select a line and click to get market data for the selected contract.
<b>Cancel Market Data</b>	Cancel market data for the selected ticker.
<b>Clear Market Data</b>	Clears all market data from the page.

The toolbar also includes the **Snapshot** check box, which when selected, requests only a single snapshot of market data.

## Bulletins Page

Use the Bulletins page to request and view IB news bulletins. Simply click **Request News Bulletins** in the toolbar. News bulletins display in the table on the page. To request all the existing bulletins for the current day and any new ones, select the **All Day News** check box on the toolbar. If this check box is not selected, you will receive only new bulletins.



### Bulletins Page Toolbar Buttons

The toolbar on the Tickers page includes the buttons described below.

Button	Description
<b>Request News Bulletins</b>	Requests all the new news bulletins.
<b>Cancel News Bulletins</b>	Cancels receipt of all news bulletins.
<b>Clear News Bulletins</b>	Clears all news bulletins from the page.

The toolbar also includes the **All Day News** check box, which when selected, requests all the existing bulletins for the current day and any new ones.

## Market Depth Page

Use the Market Depth page to view market depth for selected contracts. You can also view market depth for NYSE-listed products through the Open Book Market Data Subscription, and NASDAQ-listed products through the TotalView Market Data Subscriptions, if you have signed up for those subscriptions.

IB Trader Workstation - Tickers																	
Contract Description					Subscription Status	Market Depth											
Symbol	Type	Expiry	Strike	P/C		Multplier	Exchange	Currency	Bid	Ask							
								MM	Price	Size	Cum Size	Avg Price	MM	Price	Size	Cum Size	Avg Price
10 IBM	STK					SMART	USD	Subscribed	85.1	19	19	85.1	85.19	1	1	1	85.19
11									85.03	1	20	85.0945	85.25	1	2	2	85.22
12									83.77	3	25	84.9235					
13																	
14																	
15																	
16																	
17																	
18																	
19																	
20	[For the purpose of this sample we only show a max of 10 market depth rows]																
21 GOOG	OPT	200901	550	C		SMART	USD										
22																	
23																	
24																	
25																	
26																	
27																	
28																	
29																	
30																	
31	[For the purpose of this sample we only show a max of 10 market depth rows]																
32 GOOG	STK					SMART	USD										
33																	
34																	
35																	
36																	
37																	
38																	
39																	

## Using the Market Depth Page

**Note:** Ensure that TWS is running, and that you have connected the spreadsheet to TWS.

### To request market depth for a contract

- 1** Click the **Market Depth** tab at the bottom of the spreadsheet to open the Market Depth page.
- 2** Select the ticker symbol for which you want to request the market depth, or enter a new ticker on a blank line.
- 3** Click the **Request Market Depth** button on the toolbar.

### To reset the market data refresh rate for market depth

- 1** Type the desired market data refresh rate in seconds in the *Market Depth Refresh Rate* cell. The value must be in whole numbers from 1 to 9.

## Market Depth Page Toolbar Buttons

The toolbar on the Market Depth page includes the following buttons:

Button	Description
<b>Request Market Depth</b>	View bid/ask depth prices for the selected contract.
<b>Cancel Market Depth</b>	Cancel market depth for the selected contract.
<b>Clear Market Depth</b>	Clears all market depth data from the page.

The page also includes a *Market Depth Refresh Rate* cell, which lets you rest the market depth refresh rate in seconds, in whole numbers from 1 - 9. The default refresh rate is 3 seconds.

## Basic Orders Page

Use the Basic Orders page to:

- Create an order.
- Place a "what if" order, which shows you the margin and commission information before you place an order.
- Create a "basket" of orders.
- Modify and cancel orders.
- Create combination orders.

The screenshot shows a Microsoft Excel window titled "TwsActiveX.xls [Compatibility Mode] - Microsoft Excel". The ribbon tabs include Home, Insert, Page Layout, Formulas, Data, Review, View, Developer, and Add-Ins. The Home tab is selected, showing font and alignment tools. The main content area displays a table of stock contracts and a separate table for orders.

**Contract Description**

Symbol	Type	Expiry	Strike	P/C	Multplier	Exchange	Prim Exch	Currency	Combo Legs	Delta-Neutral
IBM	STK					NYSE	USD			
MSFT	STK					SMART	USD			
<b>YHOO</b>	<b>STK</b>					SMART	ISLAND	USD		
GE	STK					SMART	ARCA	USD		
GOOG	STK					SMART	ARCA	USD		
AMZN	STK					ARCA	USD			
DIA	STK					SMART	USD			
QQQ	STK					SMART	USD			
SPY	STK					SMART	USD			
IWM	STK					SMART	USD			
ALU	STK					SMART	USD			
AAPL	STK					SMART	USD			
INTC	STK					SMART	USD			
CSCO	STK					SMART	USD			
TWTR	STK					SMART	USD			
ORCL	STK					SMART	USD			
LVLT	STK					SMART	USD			
ETFC	STK					SMART	USD			
LEH	STK					SMART	USD			
PFE	STK					SMART	USD			
WB	STK					SMART	USD			
C	STK					SMART	USD			
JPM	STK					SMART	USD			
BAC	STK					SMART	USD			
F	STK					SMART	USD			
WFC	STK					SMART	USD			
XLF	STK					SMART	USD			
QID	STK					SMART	USD			
SOS	STK					SMART	USD			
XLE	STK					SMART	USD			
GSS	STK					SMART	USD			

**Order Description**

Action	Quantity	Order Type	Lmt Price	Aux Price	Id	Status	Filled	Remain
BUY	1000	LMT	30.00					
BUY	500	LMT	18.00					
BUY	1000	LMT	30.00					
BUY	1000	LMT	30.00					

**Hint:** To place or modify an order, select a cell that overlaps the "Place / Modify Order" button.

## Placing Orders

This topic describes how to place the following types of orders on the Orders page:

- Simple orders
- Basket orders
- Modified orders

**Note:** Ensure that TWS is running, and that you have connected the spreadsheet to TWS.

### To place an order

- 1 Click the **Basic Orders** tab at the bottom of the spreadsheet.
- 2 Define a contract by typing a symbol in a blank *Symbol* field, then entering information in the relevant contract description fields.
- 3 Select a contract and set up the order using the *Order Description* fields.

You must define the *Action* (Buy, Sell or Short Sell), *Quantity*, *Order Type*, *Limit Price* (unless it's a market order) and if necessary, the *Aux. Price* for order types that require it.

- 4 If desired, select the contract (the ticker row) and apply extended order attributes by clicking the **Apply Extended Template** button on the toolbar. This applies all attributes you have defined on the [Extended Order Attributes](#) page to the selected contract.
- 5 Click the **Place/Modify Order** button on the toolbar.

### To place a "basket" of orders

- 1 Click the **Basic Orders** tab at the bottom of the spreadsheet.
- 2 Define a contract by typing a symbol in a blank *Symbol* field, then entering information in the relevant contract description fields.
- 3 Select a contract and set up the order using fields in the *Order Description* section.
- 4 Repeat Steps 1 and 2 for additional orders.
- 5 Select a group of orders.
  - To select a group of contiguous orders, highlight the first order, hold down the **Shift** key, then highlight the last order of the group.
  - To select a group of non-contiguous orders, hold the **Ctrl** key down as you select each order.
- 6 Click the **Place/Modify Order** button.

### To modify an order (or group of orders)

- 1** On the Basic Orders page, change any necessary parameters in an order or group of orders.
- 2** Select the order or a group of orders.
  - To select a group of contiguous orders, highlight the first order, hold down the **Shift** key, then highlight the last order of the group.
  - To select a group of non-contiguous orders, hold the **Ctrl** key down as you select each order.
- 3** Click the **Place/Modify Order** button.

### Placing a Combination Order

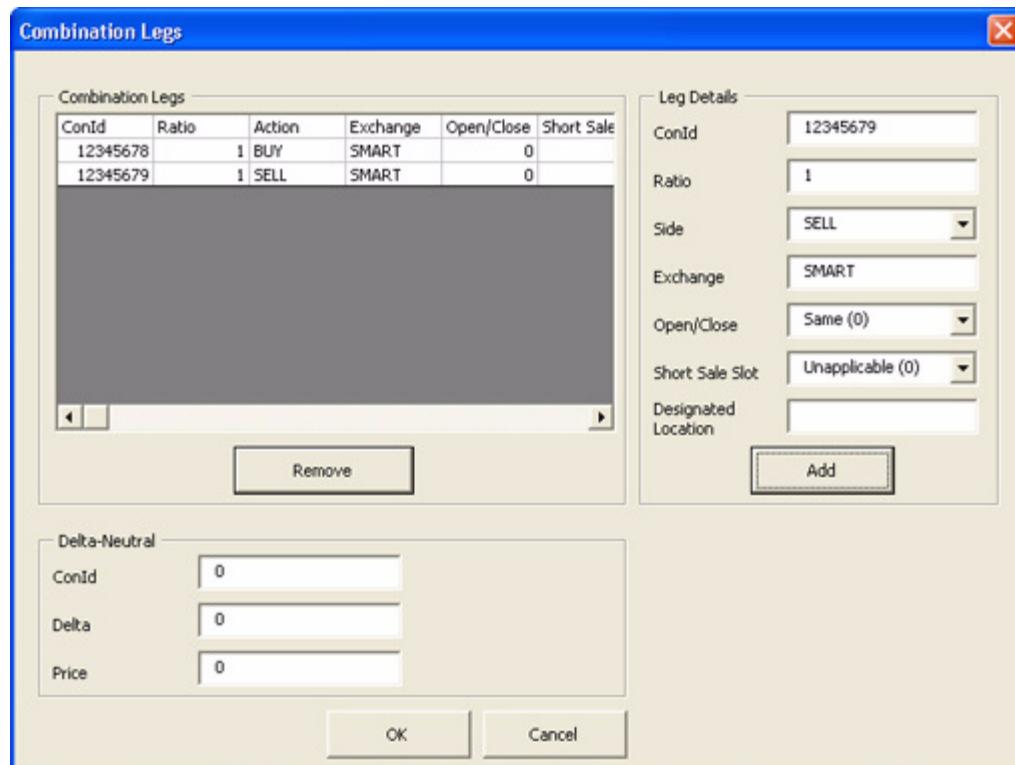
A combination order is a special type of order that is constructed of many separate legs but executed as a single transaction.

To buy a calendar spread, you would:

- Buy 1 OPT JUL03 17.5 CALL (100)
- Sell 1 OPT AUG03 17.5 CALL (100)

### To create a calendar spread order

The following example walks you through the process of placing a hypothetical calendar spread order for XYZ on ISE.



**1** Use the **Contract Details** page to get the contract id for both of the leg definitions.

- The conid for XYZ option JUL08 17.5 CALL on ISE is "12345678".
- The conid for XYZ option AUG08 17.5 CALL on ISE is "12345679".

**2** Click the **Basic Orders** tab to build the combo leg definitions. Click the **Combo Legs** button on the Basic Orders page toolbar and enter leg information. Your leg information is translated into the format:

[CMBLGS]\_[NumOfLegs]\_[Combo Leg Definitions]\_[CMBLGS]

where:

- [CMBLGS] is the delimiter used to identify the start and end of the leg definitions
- [NumOfLegs] is the number of leg definitions
- [Combo Leg Definitions] defines N leg definitions, and each leg definition consists of [conid]\_[ratio]\_[action]\_[exchange]\_[openClose], so the resulting combo substring looks as follows:

CMBLGS\_2\_17496957\_1\_BUY\_EMPTY\_0\_15910089\_1\_SELL\_EMPTY\_0\_CMBLGS

**3** The combination leg definitions must occur before the extended order attributes. The full place order DDE request string will look like this:

```
=acctName|ord!id12345?place?BUY_1_XYZ_BAG_ISE_LMT_1_CMBLGS_2_12345678_1_BUY_EMPTY_0_12345679_1_SELL_EMPTY_0_CMBLGS_DAY_EMPTY_0_O_0_EMPTY_0_EMPTY_0_O_EMPTY_0
```

If the order legs do not constitute a valid combination, one of the following errors will be returned:

- 312 = The combo details are invalid.
- 313 = The combo details for '<leg number>' are invalid.
- 314 = Security type 'BAG' requires combo leg details.
- 315 = Stock combo legs are restricted to SMART exchange.

**Notes:** 1. The exchange for the leg definition must match that of the combination order. The exception is for a STK leg definition, which must specify the SMART exchange.

2. The openClose leg definition value is always 'SAME' (i.e.0) for retail accounts. For institutional accounts, the value may be any of the following: (SAME, OPEN, CLOSE).

## Supported Order Types

The order types currently supported through the ActiveX for Excel API are:

- Limit (LMT)
- Market (MKT)
- Limit if Touched (LIT)
- Market if Touched (MIT)
- Market on Close (MOC)
- Limit on Close (LOC)
- Pegged to Market (PEGMKT)
- Relative (REL)
- Stop (STP)
- Stop Limit (STPLMT)
- Trailing Stop (TRAIL)
- Trailing Stop Limit (TRAILLIMIT)
- Volume-Weighted Average Price (VWAP)
- Volatility orders (VOL)

## Basic Orders Page Toolbar Buttons

The toolbar on the Basic Orders page includes the following buttons:

Button	Description
<b>Create Ticker</b>	Opens the Ticker box. Enter information to create a market data line.
<b>Combo Legs</b>	Opens the Combination Legs box. Enter contract details to create legs of a combination order one by one. You can also enter Delta Neutral information.
<b>Apply Extended Template</b>	Applies the current values on the Extended Order Attributes page to the highlighted order row.
<b>Place/Modify Orders</b>	After you have completed the Order Description fields, and defined any extended attributes, click to create an order for the selected contract.
<b>Cancel Order</b>	This button cancels the selected order(s).
<b>Clear Order Statuses</b>	Clears all order status information from the page.

There is also a **WhatIf** check box on the toolbar. When checked, you will receive margin and commission data as if the order were placed, but the order will NOT be placed.

## Advanced Orders Page

Use the Advanced Orders page to:

- Create complex orders that require the use of extended order attributes, including:
  - Bracket orders
  - VOL orders
  - Trailing Stop Limit Orders
  - Scale Orders
  - Relative Orders

IB Trader Workstation - Advanced Orders												Order St										
	Contract Description								Order Description													
	Symbol	Type	Expiry	Strike	P/C	Multiplier	Exchange	Prim Exch	Currency	Combo Legs	Delta-Neutral	Action	Quantity	Order Type	Lim Price	Aus Price	Id	Status	Filled	Remain		
10	MOT	STK					SMART		USD			BUY	100	LMT	4.50							
11	MOT	STK					SMART		USD			SELL	100	LMT	6.00							
12	MOT	STK					SMART		USD			SELL	100	LMT	4.00							
14	VOL Orders																					
16	GOOG	OPT	200812	270	C	100	SMART		USD			BUY	1	VOL	70.00							
17	MSFT	OPT	200812	27	C	100	SMART	PHLX	USD			BUY	1	VOL	0.15							
21	DYRS	STK					SMART		USD			SELL	100	TRAILLIMIT	15.00	0.30						
26	IBM	STK					SMART		USD			BUY	10000	LMT	80.00							
27	IBM	STK					SMART		USD			BUY	10000	REL	80.00							
31	GOOG	STK					SMART		USD			BUY	100	REL	800.00							

This page includes several example orders with mouseover help to assist you in learning how to place these orders. Simply move your mouse over the red triangle of the corner of cells on the page to display pop-up help.

Contract Description											Order Details		
Symbol	Type	Expiry	Strike	P/C	Multiplier	Exchange	Prim Exch	Currency	Combo Legs	Delta-Neutral	Action	Quantity	Order Type
<b>Jacket Order - BUY LIMIT Order Example</b> Move your mouse over the red corner for help.													
IOT	STK					SI							MT
DT	STK					SI							MT
OT	STK					SI							MT
<b>Orders</b> Move your mouse over the red corner for help.													
G	OPT	200812	270	C	100	SI							DL
F	OPT	200812	27	C	100	SI							DL
<b>Selling Stop Limit Order - SELL Order Example</b> Move your mouse over the red corner for help.													
RYS	STK					SMART		USD			SELL		100 TRAILLIMIT

For more information about using extended order attributes for individual orders or groups of orders, see [Apply Extended Order Attributes to Individual Orders and Groups of Orders](#)

## Placing a Bracket Order

Bracket orders in the ActiveX for Excel sample spreadsheet require the use of the extended order attributes *Transmit* and *Parent Order Id*. You must turn *Transmit* off until the order is completely set up, and you must identify the first order in the bracket as the Parent Order.

### To place a Buy-Limit bracket order

- 1 Click the **Advanced Orders** tab at the bottom of the spreadsheet.
- 2 Enter the contract descriptions and order descriptions for all three orders on three contiguous rows:
  - The first order should be a BUY LMT order.
  - The second order should be a SELL STP order.
  - The third order should be a SELL LMT order.
- 3 Click the **Extended Order Attributes** tab. Change the value for *Transmit* to **0** (row 13 on the Extended Order Attributes page).

This ensures that your orders are not transmitted until you have completed the order setup.
- 4 Click the **Advanced Orders** tab, highlight the first order in the bracket order, then click the **Place/Modify Order** button.

The order is not executed, but the system generates an Order ID.
- 5 Copy the Order ID for the first order, omitting the "id" prefix, then click the **Extended Order Attributes** tab and paste the Order ID into the *Value* field for *Parent Order Id* (row 14). This value will be applied to all subsequent orders until you remove it from the Extended Order Attributes page.

The first order of the bracket order is now the primary order.
- 6 Click the **Advanced Orders** tab, highlight the second order, then click the **Place/Modify Order** button.

The order is not executed but is now associated with the primary order by means of the Parent Order Id extended order attribute.
- 7 Click the **Extended Order Attributes** tab and change the value for *Transmit* back to **1** (row 13).
- 8 Click the **Advanced Orders** tab, highlight the third order in the bracket order, then click the **Place/Modify Order** button. The entire bracket order is transmitted.
- 9 When you are done placing your bracket order, go to the **Extended Order Attributes** page and delete the *Parent Order Id* value you entered. If you do not, this value will be applied to all subsequent orders that you place in the spreadsheet.

## Placing a Volatility Order

In the ActiveX for Excel sample spreadsheet, you place VOL orders by entering values for the following extended order attributes:

- Volatility
- Volatility Type
- Reference Price Type
- Continuous Update
- Underlying Range (Low) - optional
- Underlying Range (High) - optional
- Hedge Delta Order Type - optional
- *Hedge Delta Aux Price* - optional

### To place a VOL order

- 1 Click the **Advanced Orders** tab at the bottom of the spreadsheet.
- 2 Define a contract by typing a symbol in a blank *Symbol* field, then entering information in the relevant contract description fields.
- 3 Select a contract and set up the order using the *Order Description* fields.
  - Enter **VOL** in the *Order Type* field.
- 4 Click the **Extended Order Attributes** tab. Enter values in the *Value* field for the following extended order attributes:
  - *Volatility* - This value represents the volatility to use in calculating a limit price for the option. Enter this value as a percentage, not as the market data is displayed. For example, enter 17.12 instead of .1712.
  - *Volatility Type* - Enter **1** for daily volatility or **2** for annual volatility.
  - *Reference Price Type* - This value is used to compute the limit price sent to an exchange and for stock range price monitoring. Enter **1** to use the average of the best bid and ask; or **2** to use NBB (bid) when buying a call or selling a put, or the NBO (ask) when selling a call or buying a put.
  - *Continuous Update* - Enter **1** to automatically update the option price as the underlying stock price (or futures price, for index options) moves. Enter **0** if you do not want to use this feature.
- 5 On the **Extended Order Attributes** page, enter values in the *Value* field for the following optional extended order attributes:
  - *Underlying Range (Low)* - Enter a low-end acceptable stock price relative to the selected option order. If the price of the underlying instrument falls below the lower stock range price, the option order will be canceled.
  - *Underlying Range (High)* - Enter a high-end acceptable stock price relative to the selected option order. If the price of the underlying instrument rises above the higher stock range price, the option order will be canceled.

- *Hedge Delta Order Type* - Enter **LMT**, **MKT** or **REL**. Enter **NONE** if you do not want to use delta hedging.
  - *Hedge Delta Aux Price* - If you have entered LMT or REL as the *Hedge Delta Order Type*, enter the price as the value for this attribute.
- 6** Click the **Advanced Orders** tab, then highlight the order row.
- 7** Click the **Apply Extended Template** button. The values you entered for the extended order attributes are applied to the order row and displayed in the *Extended Order Attributes* section of the page.
- 8** With the order row highlighted, click the **Place/Modify Order** button.
- 9** When you are done placing VOL orders, go to the **Extended Order Attributes** page and delete the VOL order values you entered. If you do not, these values will be applied to all subsequent orders that you place in the spreadsheet.

## Placing a Trailing Stop Limit Order

In TWS, there are four values that make up a trailing stop limit order:

- trailing amount
- stop price
- limit price
- limit offset

In the ActiveX for Excel API spreadsheet, you enter the trailing amount, stop price and limit price. There is no field or extended order attribute for the limit offset value. You must include the limit offset in the stop price (the *Trail Stop Price* extended order attribute).

### To create a Trailing Stop Limit Order

- 1** Click the **Advanced Orders** tab at the bottom of the spreadsheet.
- 2** Define a contract by typing a symbol in a blank *Symbol* field, then entering information in the relevant contract description fields.
- 3** Select a contract and set up the order using the *Order Description* fields.
  - Enter **BUY** or **SELL** in the *Action* field.
  - Enter the limit price in the *Lmt Price* field.
  - Enter **TRAILLIMIT** in the *Order Type* field.
  - Enter the trailing amount in the *Aux Price* field.
- 4** Click the **Extended Order Attributes** tab. Specify the trailing stop price as an extended order attribute. Type this value in the *Trail Stop Price Value* field.
  - The Trail Stop Price value must include the limit offset.  
For a sell order:

**Trail Stop Price = Limit Price - Trailing Amount - Limit Offset**

For a buy order:

**Trail Stop Price = Limit Price + Trailing Amount + Limit Offset**

- 5 On the **Advanced Orders** page, select the order row and click the **Apply Extended Template** button. The *Trail Stop Price* value is applied to the selected order and displayed in the *Trail Stop Price* field in the *Extended Order Attributes* section of the page.
- 6 Click the **Place/Modify Order** button.
- 7 When you are done placing your order, go to the **Extended Order Attributes** page and delete the *Trail Stop Price* value you entered. If you do not, this value will be applied to all subsequent orders that you place in the spreadsheet.

## Placing a Scale Order

In the ActiveX for Excel sample spreadsheet, you place scale orders by entering values for the following extended order attributes:

- Scale Component Size
- Scale Price Increment

### To place a scale order

- 1 Click the **Advanced Orders** tab at the bottom of the spreadsheet.
- 2 Define a contract by typing a symbol in a blank *Symbol* field, then entering information in the relevant contract description fields.
- 3 Select a contract and set up the order using the *Order Description* fields. The order type should be LMT or REL.
- 4 Click the **Extended Order Attributes** tab. Enter values in the *Value* field for the following extended order attributes:
  - *Scale Component Size* - Enter the size of the first, or initial, order component. For example, if you submit a 10,000-share order with a Scale Component Size value of 1000, the first component will be fore 1000 shares.
  - *Scale Price Increment* - Enter the amount used to calculate the per-unit price of each component in the scale ladder. This cannot be a negative number.

**Note:** As of API Release 9.41, the Scale Num Components not supported.

- 5 On the **Advanced Orders** page, select the order row and click the **Apply Extended Template** button. The scale order values are applied to the selected order and displayed in the *Extended Order Attributes* section of the page.
- 6 Click the **Place/Modify Order** button.
- 7 When you are done placing your order, go to the **Extended Order Attributes** page and delete the scale order values you entered. If you do not, these values will be applied to all subsequent orders that you place in the spreadsheet.

## Placing a Relative Order

In the ActiveX for Excel sample spreadsheet, you place relative orders by entering a value for the *Percent Offset* extended order attribute.

### To place a relative order

- 1 Click the **Advanced Orders** tab at the bottom of the spreadsheet.
- 2 Define a contract by typing a symbol in a blank *Symbol* field, then entering information in the relevant contract description fields.
- 3 Select a contract and set up the order using the *Order Description* fields.
  - Enter **REL** as the order type.
  - Enter the price cap in the *Lmt Price* cell.
- 4 Click the **Extended Order Attributes** tab. Enter a percentage in decimal form in the *Value* field for the *Percent Offset* extended order attribute.
- 5 On the **Advanced Orders** page, select the order row and click the **Apply Extended Template** button. The percent offset value is applied to the selected order and displayed in the *Extended Order Attributes* section of the page.
- 6 Click the **Place/Modify Order** button.
- 7 When you are done placing your order, go to the **Extended Order Attributes** page and delete the *Percent Offset* value you entered. If you do not, this value will be applied to all subsequent orders that you place in the spreadsheet.

## Advanced Orders Page Toolbar Buttons

The toolbar on the Advanced Orders page includes the following buttons:

Button	Description
<b>Create Ticker</b>	Opens the <a href="#">Ticker box</a> . Enter information to create a market data line.
<b>Combo Legs</b>	Opens the <a href="#">Combination Legs box</a> . Enter contract details to create legs of a combination order one by one.
<b>Apply Extended Template</b>	Applies the current values on the Extended Order Attributes page to the highlighted order row.
<b>Place/Modify Orders</b>	After you have completed the Order Description fields, and defined any extended attributes, click to create an order for the selected contract.
<b>Cancel Order</b>	This button cancels the selected order(s).
<b>Clear Order Statuses</b>	Clears all order status information from the page.

There is also a **WhatIf** check box on the toolbar. When checked, you will receive margin and commission data as if the order were placed, but the order will NOT be placed.

## Extended Order Attributes Page

The Extended Order Attributes page includes all of the optional attributes you can use when you send an order, such as setting a display size to create an iceberg order, adding orders to an OCA group, and setting the transmit date for a Good After Time order. Once you define the attributes on this page, you can apply them to a single order or selected group of orders using the **Apply Extended Template** button, which occurs on both the Orders page and the Conditional Orders page. The attributes populate the extended order attributes fields that follow the *Order Status* fields to the far right of the page.

Extended Order Attributes		
Note: This page has a new ability as of API 8.91. It is now a template that can be applied to any order.		
Attribute	Value	Note
7 Time In Force (DAY, GTC, etc.)	DAY	The time in force. Valid values are: DAY, GTC, IOC, GFD.
8 OCA Group		*One Cancels All* group name
9 Account (Institutional only)		
10 Open/Close(OnOpen, OnClose) (Institutional only)	0	
11 Origin (OnCustomer, 1Form) (Institutional only)	0	
12 Order Ref	HelloWorld	
13 Transmit (0=false, 1=true)	1	
14 Parent Order Id		
15 Block Order (0=false, 1=true)	0	
16 Sweep To Fill (0=false, 1=true)	0	
17 Display Size	0	
18 Trigger Method	0=Default, 1=Double_Bid_Ask, 2=Last, 3=Double_Last, 4=Bid_Ask, 7=Last_or_Bid_Ask, 8=Mid-point	
19 Hidden (0=false, 1=true)	0	
20 Discretionary Amount (SMART Routing)		
21 Good After Time	FORMAT: 20080707 08:00:00 (time zone)	
22 Good Till Date	FORMAT: 20080707 08:00:00 (time zone)	
23 FA Group (Financial Advisors only)	NOTE:	
24 FA Method (Financial Advisors only)	THESE FOUR FIELDS	
25 FA Percentage (Financial Advisors only)	ONLY APPLY TO FINANCIAL ADVISOR ORDERS	
26 FA Profile (Financial Advisors only)	LEAVE THEM EMPTY OTHERWISE	
27 Short Sale Slot (Institutional only)	1 - if you hold the shares, 2 - if they will be delivered elsewhere. Only for action = "SHORT".	
28 Short Sale Location (Institutional only)	If shares will be delivered elsewhere, specify where (can be comma-delimited list, no spaces)	
29 OCA Type	1=Cancel on Fill with Block, 2=Reduce on Fill with Block, 3=Reduce on Fill without Block	
30 Rule BOA	"I" - Individual, "A" - Agency, "W" - AgentOtherMember, "J" - IndividualPTA, "M" - AgentOtherMemberPTA, "K" - IndividualPT, "L" - AgencyPT, "N" - AgentOtherMemberPT	
31 Setting Firm (Institutional only)		
32 All Or None (0=false, 1=true)		
33 Minimum Qty		
34 Percent Offset	Relative Orders Only	
35 Electronic Trade Only (0=false, 1=true) (SMART Routing)		
36 Firm Quote Only (0=false, 1=true) (SMART Routing)		
37 NBBO Price Cap (SMART Routing)		
38 Auction Strategy	BID Exchange, 1=match, 2=improvement, 3=transparent	
39 Starting Price	BID Exchange	
40 Stock Ref Price	BID Exchange	

For a complete list of extended order attributes supported by the API, see [Extended Order Attributes](#).

## **Manually Program Extended Order Attributes**

Observe the following guidelines when you manually assign an attribute:

- When appended to orderDescription, the number and order of attributes cannot be changed.
  - For any attribute that is not defined, use the value 'EMPTY' or {}. Since a string length is limited to 255 characters, we recommend using the open/close curly braces {}.
  - A place order message for a simple stock limit day order looks as follows, with the primary exchange "ISLAND" separating the extended attributes:

## Apply Extended Order Attributes to Individual Orders and Groups of Orders

Normally, values that you enter on the Extended Order Attributes page apply to all subsequent orders. However, you also can apply selected attributes to an individual order or a group of orders on the Orders page.

**Note:** You can also use this procedure to apply extended order attributes to orders on the Conditional Orders page.

#### To apply extended order attributes to individual orders or a group of orders

- 1 Enter the value or values on the Extended Order Attributes page that you want to apply to an individual order or group of orders.
  - 2 On the Orders page, select the order or group of orders.
  - 3 Click the **Apply Extended Template** button.

The extended order attributes are applied to the order(s) and the values you entered on the Extended Order Attributes page are added to the corresponding fields in the *Extended Order Attributes* section of the Orders page.

When you place the order or group of orders, the extended order attribute values you entered are applied to the order.

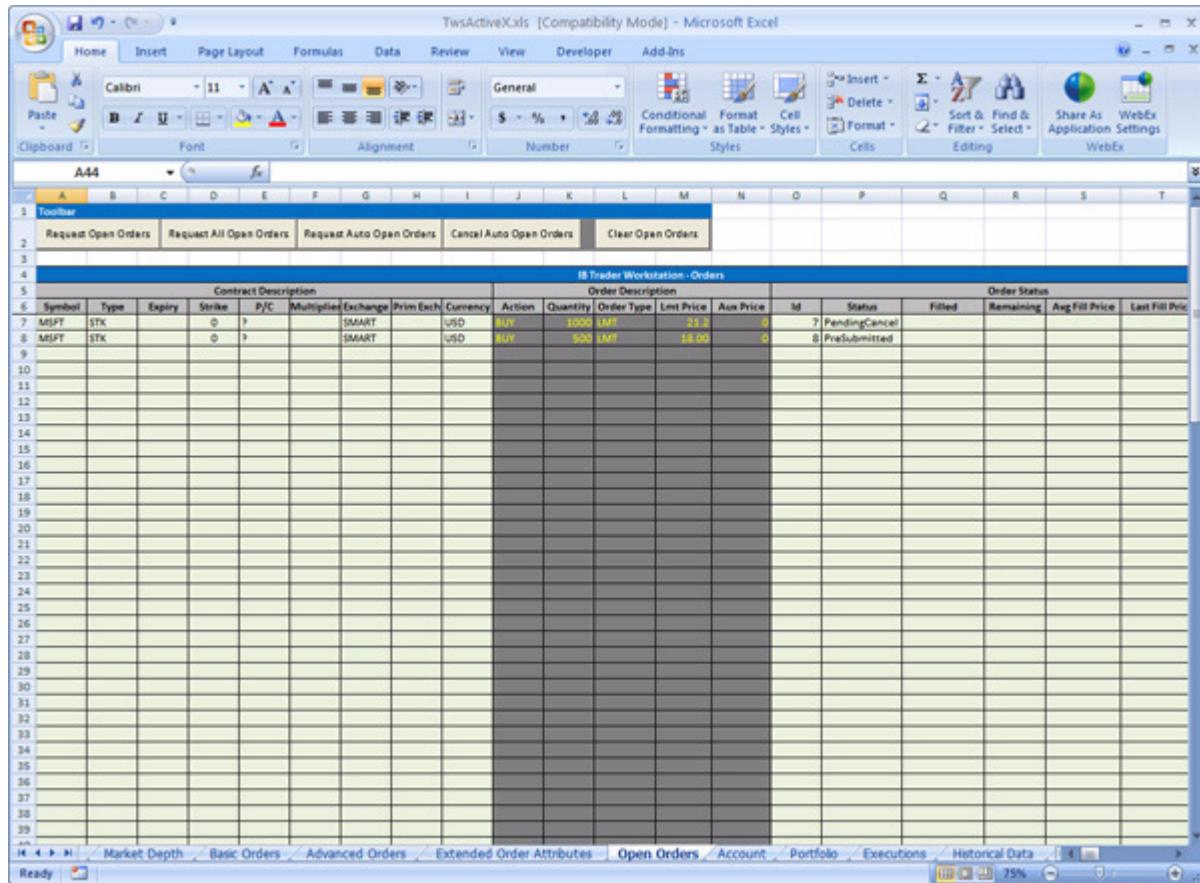
For example, you might want to assign a unique Order Ref number to a group or basket of orders. To do this, you would enter the number for the Order Ref attribute on the Extended Order Attributes page, then select all the orders in the group on the Orders page and click Apply Extended Template.

- 4** Delete the value of the extended order attributes you used for the order from the Extended Order Attributes page. These values will still apply to all subsequent orders that you place from the ActiveX for Excel API spreadsheet unless you remove the value.

## **Open Orders Page**

The Open Orders page shows you all transmitted orders, including those that have been accepted by the IB system, and those that are working at an exchange. Once you have subscribed, the page is updated each time you submit a new order, either through the API or in TWS.

Once an order executes, it remains on the Open Orders page for 30 seconds, with the Status value changed to FILLED. Then the filled order is cleared and you can see it on the Executions page if you subscribed to real-time executions.



## Viewing Open Orders

**Note:** Ensure that TWS is running, and that you have connected the spreadsheet to TWS.

### To view open orders:

- 1 Click the **Open Orders** tab at the bottom of the spreadsheet.
- 2 Do one of the following:
  - To request open orders from the current ActiveX for Excel spreadsheet, click **Subscribe to Open Orders** on the toolbar.
  - To request all open orders for the current account, click **Request All Open Orders** on the toolbar.
  - To associate all newly created TWS orders with the current client, click **Request Auto Open Orders** on the toolbar. Note that the Client ID must be 0.

All of the requested open orders are displayed on the page, including orders you enter in the spreadsheet and in TWS.

Orders that fill remain on the page for 30 seconds with a value of *Fill* in the *Status* field.

### To remove open orders

- 1 Click the **Cancel Open Orders Subscription** button on the toolbar.
- 2 Click the **Clear Open Orders** button.

## Open Orders Tab Toolbar

The toolbar on the Open Orders page includes the following buttons:

Button	Description
<b>Request Open Orders</b>	Queries TWS and returns all open orders. Once you subscribe to open orders, this page updates each time there is a new open order.
<b>Request All Open Orders</b>	Queries TWS and returns all open orders from the current account. Once you subscribe to open orders, this page updates each time there is a new open order.
<b>Request Auto Open Orders</b>	Queries TWS and associate all newly created TWS orders with the current client, which must be Client ID 0.
<b>Cancel Auto Open Orders</b>	Cancels association of newly created TWS orders with the client
<b>Clear Open Orders</b>	Removes all open orders from the page.

## Account Page

Use the Account page to:

- View account details including your current Equity with Loan Value and Available funds.
- View list of advisor-managed account codes.
- Financial Advisors can view FA information.
- View your current portfolio.

Key	Value	Currency	Account Name
AccruedCash	DU55730		DU55730
AccruedCash	TRUE		DU55730
AccountType	UNIVERSAL		DU55730
AccruedCash	0	BASE	DU55730
AccruedCash	0	EUR	DU55730
AccruedCash	0	GBP	DU55730
AccruedCash-C	0	USD	DU55730
AccruedCash-S	0	USD	DU55730
AccruedDividend	0	USD	DU55730
AccruedDividend-C	0	USD	DU55730
AccruedDividend-S	0	USD	DU55730
AvailableFunds	1350500.85	USD	DU55730
AvailableFunds-C	0	USD	DU55730
AvailableFunds-S	1350500.85	USD	DU55730
Billable	0	USD	DU55730
Billable-C	0	USD	DU55730
Billable-S	0	USD	DU55730
BuyingPower	8391695.1	USD	DU55730
CashBalance	984365.71	BASE	DU55730
CashBalance	2.02	EUR	DU55730
CashBalance	1.23	GBP	DU55730
CashBalance	984361.41	USD	DU55730
Currency	BASE	BASE	DU55730
Currency	EUR	EUR	DU55730

## Using the Account Page

**Note:** Ensure that TWS is running, and that you have connected the spreadsheet to TWS.

### To view account information

- 1 Click the **Account** tab at the bottom of the spreadsheet.
- 2 Click **Request Account Updates** on the toolbar.

### To remove account information

- 1 Click the **Account** tab at the bottom of the spreadsheet.
- 2 Click **Cancel Account Updates** on the toolbar to stop receiving account updates.

- 3** Click **Clear Account Data** on the toolbar to clear all data from the page.

#### To request the list of Financial Advisor (FA) managed account codes

- 1** Click the **Account** tab at the bottom of the spreadsheet.

- 2** Click **Request Managed Accounts** on the toolbar.

A comma-separated list of all managed account numbers displays in the *Managed Accounts* cell.

1	Toolbar	d	e	f	g	h	i	j	k	l	m
2	Request Account Updates	Cancel Account Updates	Request Managed Accounts	Request FA							Clear Account Data
3											
4	Parameters										
5	Accr Code			FA Data Type (1=GROUPS, 2=PROFILE, 3=ACCOUNT ALIASES)							
6	DU10			3							
7											
8	Last Update Time				Managed Accounts						
9	TimeStamp		11:03			F98100,U98101,U98102,U98103,U98104,U98105,U98106,U98107,U98108,					
10	Subscription Status										
11											

#### To request Financial Advisor (FA) information

- 1** Click the **Account** tab at the bottom of the spreadsheet.

- 2** In the *Account Code* cell, type the account code for which you want details.

- 3** In the *FA Data Type* cell, enter a numeric value representing the type of data you wish you receive:

- Type **1** for groups.
- Type **2** for profiles.
- Type **3** for account aliases.

- 4** Click **Request FA** on the toolbar.

1	Toolbar	d	e	f	g	h	i	j	k	l	m
2	Request Account Updates	Cancel Account Updates	Request Managed Accounts	Request FA							Clear Account Data
3											
4	Parameters			FA Data Type (1=GROUPS, 2=PROFILE, 3=ACCOUNT ALIASES)							
5	Accr Code			3							
6	DU10				Managed Accounts						
7						F98100,U98101,U98102,U98103,U98104,U98105,U98106,U98107,U98108,					
8	Last Update Time		11:03								
9	TimeStamp										
10	Subscription Status										
11											

## Account Page Toolbar Buttons

The toolbar on the Account page includes the following buttons.

Button	Description
<b>Request Account Updates</b>	Each click gives you data for a specific account value. All blank lines that precede the Account Portfolio section will hold data. Continue to click until all lines are populated.
<b>Cancel Account Updates</b>	Click this button one time for each position you hold. When you get a line of "0's" you know you have downloaded all current positions. These values continue to update in real-time.
<b>Request Managed Accounts</b>	For advisor accounts, receives a list of managed accounts and displays them as a comma-separated list in the <i>Management Accounts</i> cell.
<b>Request FA</b>	For advisor accounts, displays FA data of the type specified in the <i>FA Data Type</i> cell.
<b>Clear Account Data</b>	Clears all information from the page. You must first cancel your subscription before you can clear the data.

# Portfolio Page

Use the Portfolio page to:

- Displays all of your current positions.
  - Exercise options.

**Note:** Ensure that TWS is running, and that you have connected the spreadsheet to TWS.

## **Viewing Your Portfolio**

## To view your portfolio

- 1 Click the **Account** tab on the bottom of the worksheet, then click **Request Account Updates** on the toolbar.
  - 2 Click the **Portfolio** tab at the bottom of the worksheet to view your portfolio.

## To remove portfolio information

- 1 Click the **Account** tab on the bottom of the worksheet, then click **Cancel Account Updates** on the toolbar to stop receiving portfolio updates.
  - 2 Click the **Clear Portfolio Data** button to clear all data from the page.

## **Exercising Options**

You can exercise options or let options lapse on the Portfolio page.

### To exercise an option or let an option lapse

- 1 Click the **Portfolio** tab at the bottom of the worksheet.
  - 2 Enter values in the *Exercise Options Parameters* cells at the far right side of the page:
    - *Exercise Action* - Enter **1** to exercise the selected option; **2** to let the option lapse.
    - *Exercise Quantity* - Enter the number of contracts you wish to exercise or let lapse.
    - *Override* - Enter **1** to override the system's natural action; **2** to not override.

Exercise Options Parameters			
Status	Exercise Action (1,2)	Exercise Quantity	Override (0,1)
		1	10
		1	10

- 3** Click **Exercise Options** in the toolbar. The *Status* column updates.

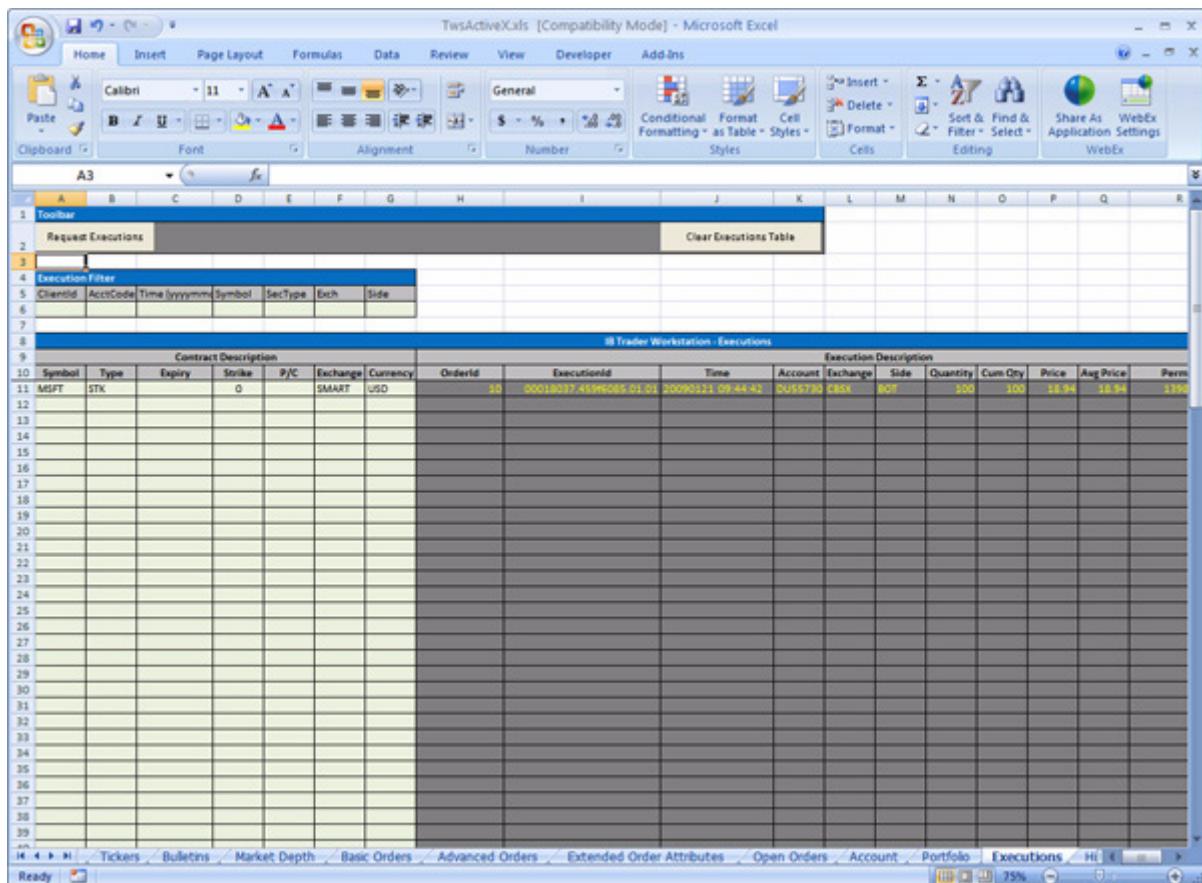
## Portfolio Page Toolbar Buttons

The toolbar on the Portfolio page includes the following buttons.

Button	Description
<b>Exercise Options</b>	Exercises the selected option or lets the selected option lapse, depending on the value in the <i>Exercise Action</i> cell.
<b>Clear Portfolio Data</b>	Removes all data from the page.

## Executions Page

When you subscribe to executions, the Executions page displays information about all completed trades.



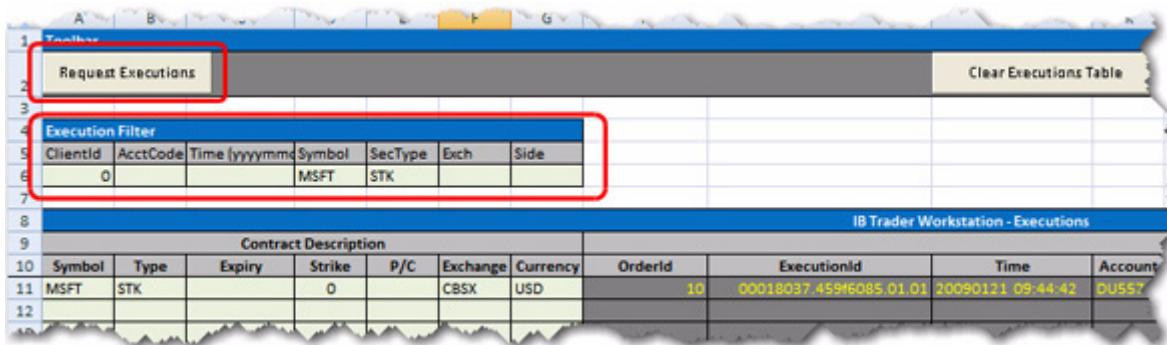
The screenshot shows a Microsoft Excel window titled "TwsActiveX.xls [Compatibility Mode] - Microsoft Excel". The ribbon menu includes Home, Insert, Page Layout, Formulas, Data, Review, View, Developer, and Add-Ins. The ribbon tabs include Font, Alignment, Number, Styles, Cells, Editing, and Application Settings. The main content area displays a table titled "II Trader Workstation - Executions". The table has two main sections: "Contract Description" and "Execution Description". The "Contract Description" section includes columns for Symbol, Type, Expiry, Strike, P/C, Exchange, and Currency. The "Execution Description" section includes columns for OrderId, ExecutionId, Time, Account, Exchange, Side, Quantity, Cum Qty, Price, Avg Price, and Perm. A single row of data is visible, showing details for an execution of MSFT stock (Symbol: MSFT, Type: STK, Expiry: 0, P/C: SMART, Exchange: USD) with OrderId 10, ExecutionId 000180374556005.01.01, Time 20090311 09:44:42, Account DUS6780, Exchange CBX, Side BOT, Quantity 100, Cum Qty 100, Price 18.94, Avg Price 18.94, and Perm 1300.

## Viewing Executions

**Note:** Ensure that TWS is running, and that you have connected the spreadsheet to TWS.

### To view executions

- 1 Click the **Executions** tab at the bottom of the spreadsheet.
- 2 Optionally, filter your executions by entering values in the *Execution Filter* cells:
  - Filter executions by client ID, account code, date/time, symbol, security type, exchange or side.



- 3 Click **Request Executions** on the toolbar.

### To remove execution data

- 1 Click **Clear Executions Table** on the toolbar. All data is removed from the page.

## Executions Page Toolbar Buttons

The toolbar on the Executions page includes the following buttons:

Button	Description
<b>Request Executions</b>	Queries TWS and returns information about all valid executions. After you subscribe to executions, this page updates each time an order executes.
<b>Clear Executions Table</b>	Removes all execution reports from the page.

## Historical Data Page

Use the Historical Data page to request historical data for an instrument based on data you enter in query fields. The query results display on a separate worksheet page and creates a new page for the results if the page doesn't currently exist.

Symbol	Type	Expiry	Strike	P/C	Multipliers	Exchange	Prim Exch	Currency	Combo Legs	Delta-Neutral	Incl Expired	Request Status	Query Specification				
													End Date/Time	Duration	Bar Size	What To Show	
IBM	STK					NYSE		USD				Finished	20081231 18:25:18 GMT	3 M	1 day	MIDPOINT	1
MSFT	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 hour	TRADES	1
YHOO	STK					SMART	ISLAND	USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
GE	STK					SMART	ARCA	USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	TRADES	1
GOOG	STK					SMART	ARCA	USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
AMZN	STK					ARCA		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
DIA	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
QQQ	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
SPY	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
IWM	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
ALU	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
AAPL	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
INTC	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
CSCO	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
TWTR	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
ORCL	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
LVLT	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
ETFC	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
LEH	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
PFE	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
WM	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
WB	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
C	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
JPM	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
BAC	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
F	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
WFC	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
XLF	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
QID	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
SOS	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1
XLE	STK					SMART		USD				Processing ...	20080904 18:25:18 GMT	300 S	1 day	MIDPOINT	1

**Note:** For a information about historical data request limitations, see [Historical Data Limitations](#).

## Viewing Historical Data

**Note:** Ensure that TWS is running, and that you have connected the spreadsheet to TWS.

### To request historical data

- 1 Click the Historical Data tab at the bottom of the spreadsheet.
- 2 Create a ticker by filling in the fields in the *Contract Description* section of the page, or by clicking the **Create Ticker** button on the toolbar and [entering the required information in the Ticker box](#).
- 3 Enter the parameters of your query in the *Query Specification* fields. For complete descriptions of the query fields, [Historical Data Page Query Specification Fields](#).
- 4 Select the line, then click the **Request Historical Data** button. The status of your request displays in the *Request Status* cell.

In the *Activate Page* cell, enter **TRUE** to display the results page on top of the current window. Enter **FALSE** to display the page on a separate tab in the spreadsheet without displaying on top of the current window.

The results are displayed on a new tabbed page in the spreadsheet, the name of which is specified in the *Page Name* cell.

### To request historical data for expired contracts

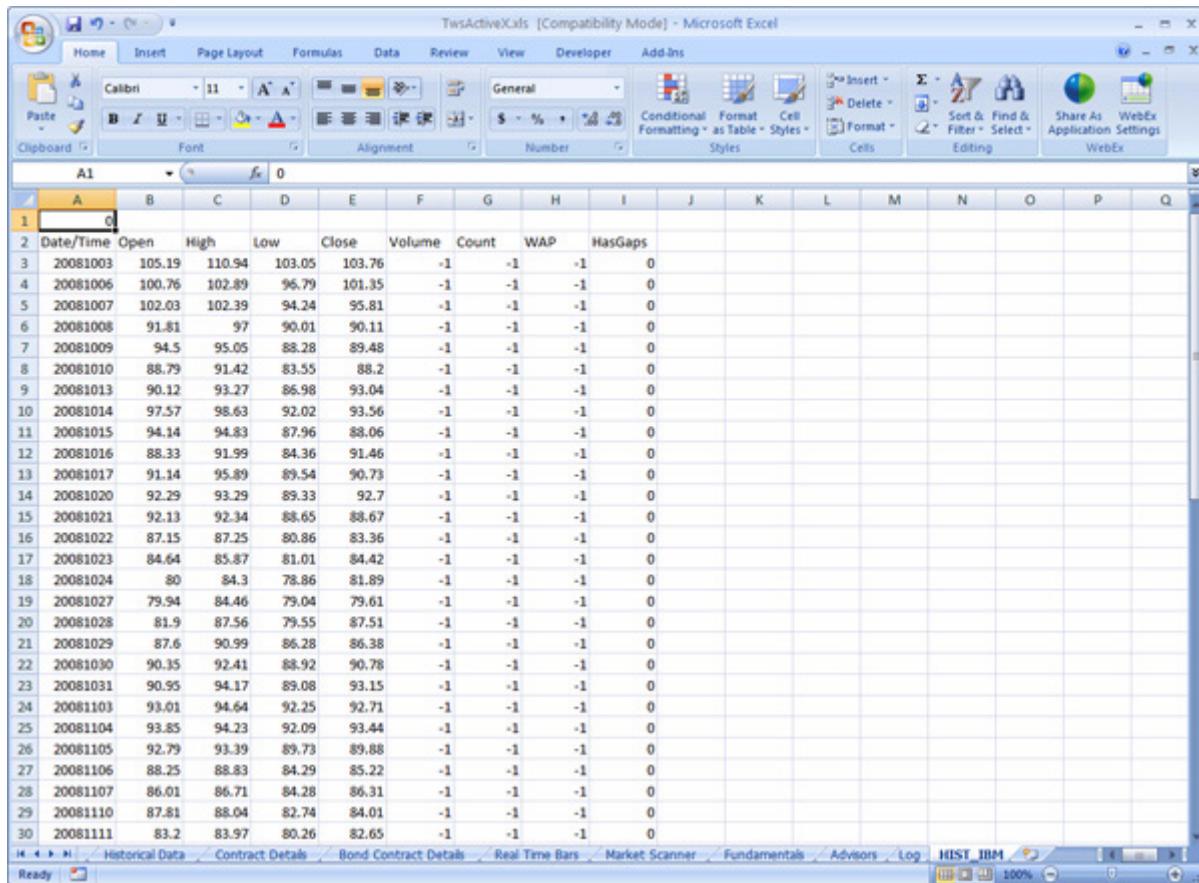
- 1 On the Historical Data page, create a ticker by filling in the fields in the *Contract Description* section of the page, or by clicking the **Create Ticker** button on the toolbar and entering the required information in the Ticker box.
- 2 Enter the parameters of your query in the *Query Specification* fields.
- 3 In the *Incl Expired* cell in the *Query Specification* section, enter **TRUE**.
- 4 Select the line, then click **Request Historical Data** on the toolbar. The status of your request displays in the *Request Status* cell.

In the *Activate Page* cell, enter **TRUE** to display the results page on top of the current window. Enter **FALSE** to display the page on a separate tab in the spreadsheet without displaying on top of the current window.

The results are displayed on a new tabbed page in the spreadsheet, the name of which is specified in the *Page Name* cell.

- 5 Historical data queries on expired contracts are limited to the last year of the life of the contract.

The following figure shows a typical historical data results page.



A screenshot of Microsoft Excel showing a historical data results page. The window title is "TwsActiveX.xls [Compatibility Mode] - Microsoft Excel". The ribbon tabs include Home, Insert, Page Layout, Formulas, Data, Review, View, Developer, and Add-ins. The Home tab is selected. The ribbon also includes Font, Alignment, Number, Styles, Cells, and Editing sections. The main content area displays a table of historical data from 2008. The columns are labeled: Date/Time, Open, High, Low, Close, Volume, Count, WAP, and HasGaps. The data rows show price movements over time, such as 20081003 (Open: 105.19, Close: 103.05) and 20081111 (Open: 83.2, Close: 80.26). The bottom of the screen shows a ribbon bar with tabs like Historical Data, Contract Details, Bond Contract Details, Real Time Bars, Market Scanner, Fundamentals, Advisors, Log, and HIST IBM. The status bar at the bottom right shows "100%".

Date/Time	Open	High	Low	Close	Volume	Count	WAP	HasGaps
20081003	105.19	110.94	103.05	103.76	-1	-1	-1	0
20081006	100.76	102.89	96.79	101.35	-1	-1	-1	0
20081007	102.03	102.39	94.24	95.81	-1	-1	-1	0
20081008	91.81	97	90.01	90.11	-1	-1	-1	0
20081009	94.5	95.05	88.28	89.48	-1	-1	-1	0
20081010	88.79	91.42	83.55	88.2	-1	-1	-1	0
20081013	90.12	93.27	86.98	93.04	-1	-1	-1	0
20081014	97.57	98.63	92.02	93.56	-1	-1	-1	0
20081015	94.14	94.83	87.96	88.06	-1	-1	-1	0
20081016	88.33	91.99	84.36	91.46	-1	-1	-1	0
20081017	91.14	95.89	89.54	90.73	-1	-1	-1	0
20081020	92.29	93.29	89.33	92.7	-1	-1	-1	0
20081021	92.13	92.34	88.65	88.67	-1	-1	-1	0
20081022	87.15	87.25	80.86	83.36	-1	-1	-1	0
20081023	84.64	85.87	81.01	84.42	-1	-1	-1	0
20081024	80	84.3	78.86	81.89	-1	-1	-1	0
20081027	79.94	84.46	79.04	79.61	-1	-1	-1	0
20081028	81.9	87.56	79.55	87.51	-1	-1	-1	0
20081029	87.6	90.99	86.28	86.38	-1	-1	-1	0
20081030	90.35	92.41	88.92	90.78	-1	-1	-1	0
20081031	90.95	94.17	89.08	93.15	-1	-1	-1	0
20081103	93.01	94.64	92.25	92.71	-1	-1	-1	0
20081104	93.85	94.23	92.09	93.44	-1	-1	-1	0
20081105	92.79	93.39	89.73	89.88	-1	-1	-1	0
20081106	88.25	88.83	84.29	85.22	-1	-1	-1	0
20081107	86.01	86.71	84.28	86.31	-1	-1	-1	0
20081110	87.81	88.04	82.74	84.01	-1	-1	-1	0
20081111	83.2	83.97	80.26	82.65	-1	-1	-1	0

## Historical Data Page Query Specification Fields

Parameter	Description																																		
End Date/Time	Use the format <code>yyyymmdd {space}hh:mm:ss{space}tmz</code> where the time zone is allowed (optionally) after a space at the end.																																		
Duration	<p>This is the time span the request will cover, and is specified using the format <i>integer {space} unit</i>, where valid units are:</p> <ul style="list-style-type: none"> <li>• S (seconds)</li> <li>• D (days)</li> <li>• W (weeks)</li> <li>• Y (years)</li> </ul> <p>This unit is currently limited to one. If no unit is specified, seconds are used.</p>																																		
Bar Size	<p>Specifies the size of the bars that will be returned. The following bar sizes may be used, and are specified using the parametric value:</p> <table> <thead> <tr> <th>Bar Size String</th> <th>Integer Value</th> </tr> </thead> <tbody> <tr> <td>1 second</td> <td>1</td> </tr> <tr> <td>5 seconds</td> <td>2</td> </tr> <tr> <td>15 seconds</td> <td>3</td> </tr> <tr> <td>30 seconds</td> <td>4</td> </tr> <tr> <td>1 minutes</td> <td>5</td> </tr> <tr> <td>2 minutes</td> <td>6</td> </tr> <tr> <td>3 minutes</td> <td>16</td> </tr> <tr> <td>5 minutes</td> <td>7</td> </tr> <tr> <td>15 minutes</td> <td>8</td> </tr> <tr> <td>30 minutes</td> <td>9</td> </tr> <tr> <td>1 hour</td> <td>10</td> </tr> <tr> <td>1 day</td> <td>11</td> </tr> <tr> <td>1 week</td> <td>12</td> </tr> <tr> <td>1 month</td> <td>13</td> </tr> <tr> <td>3 months</td> <td>14</td> </tr> <tr> <td>1 year</td> <td>15</td> </tr> </tbody> </table> <p>On the query return page, each "bar" is represented by a line in the spreadsheet. If you specify a duration of 300 seconds, and a bar size of "1" (one second) your return will include 300 lines, and the value in each line is equal to one second, or is a one-second bar. Note that you can use either the Integer value of the Bar Size String or the Integer Value to define the bar sizes.</p>	Bar Size String	Integer Value	1 second	1	5 seconds	2	15 seconds	3	30 seconds	4	1 minutes	5	2 minutes	6	3 minutes	16	5 minutes	7	15 minutes	8	30 minutes	9	1 hour	10	1 day	11	1 week	12	1 month	13	3 months	14	1 year	15
Bar Size String	Integer Value																																		
1 second	1																																		
5 seconds	2																																		
15 seconds	3																																		
30 seconds	4																																		
1 minutes	5																																		
2 minutes	6																																		
3 minutes	16																																		
5 minutes	7																																		
15 minutes	8																																		
30 minutes	9																																		
1 hour	10																																		
1 day	11																																		
1 week	12																																		
1 month	13																																		
3 months	14																																		
1 year	15																																		

Parameter	Description
What to Show	<p>Determines the nature of the data extracted. Valid values include:</p> <ul style="list-style-type: none"> <li>• Trades</li> <li>• Midpoint</li> <li>• Bid</li> <li>• Ask</li> <li>• Bid/Ask</li> </ul> <p>All but the Bid/Ask data contain the <i>start time, open, high, low, close, volume</i> and <i>weighted average price</i> during the time slice queried.</p> <p>For the Bid/Ask query, the <i>open</i> and <i>close</i> values are the time-weighted average bid and the time-weighted average offer, respectively. These bars are identical to the TWS charts' candlestick bars.</p>
RTH Only	<p>Regular Trading Hours only. Valid values include:</p> <ul style="list-style-type: none"> <li>• 0 - all data available during the time span requested is returned, including time intervals when the market in question was outside of regular trading hours.</li> <li>• 1 - only data within the regular trading hours for the product requested is returned, even if the time span falls partially or completely outside.</li> </ul>
Date Format Style	<p>Valid values include:</p> <ul style="list-style-type: none"> <li>• 1 - dates that apply to bars are returned in the format <code>yyyymmdd{space}{space}hh:mm:dd</code> (the same format used when reporting executions).</li> <li>• 2 - the dates are returned as an integer specifying the number of seconds since 1/1/1970 GMT.</li> </ul>
Page Name	The name of the results page. This appears in the tab for the results page at the bottom of the worksheet.
Activate page	Enter TRUE to display the results page on top of the current window. Enter FALSE to display the results on a new page in the spreadsheet without appearing on top of the current window.

Note that the new page is added to the right of the existing tabs on the worksheet.

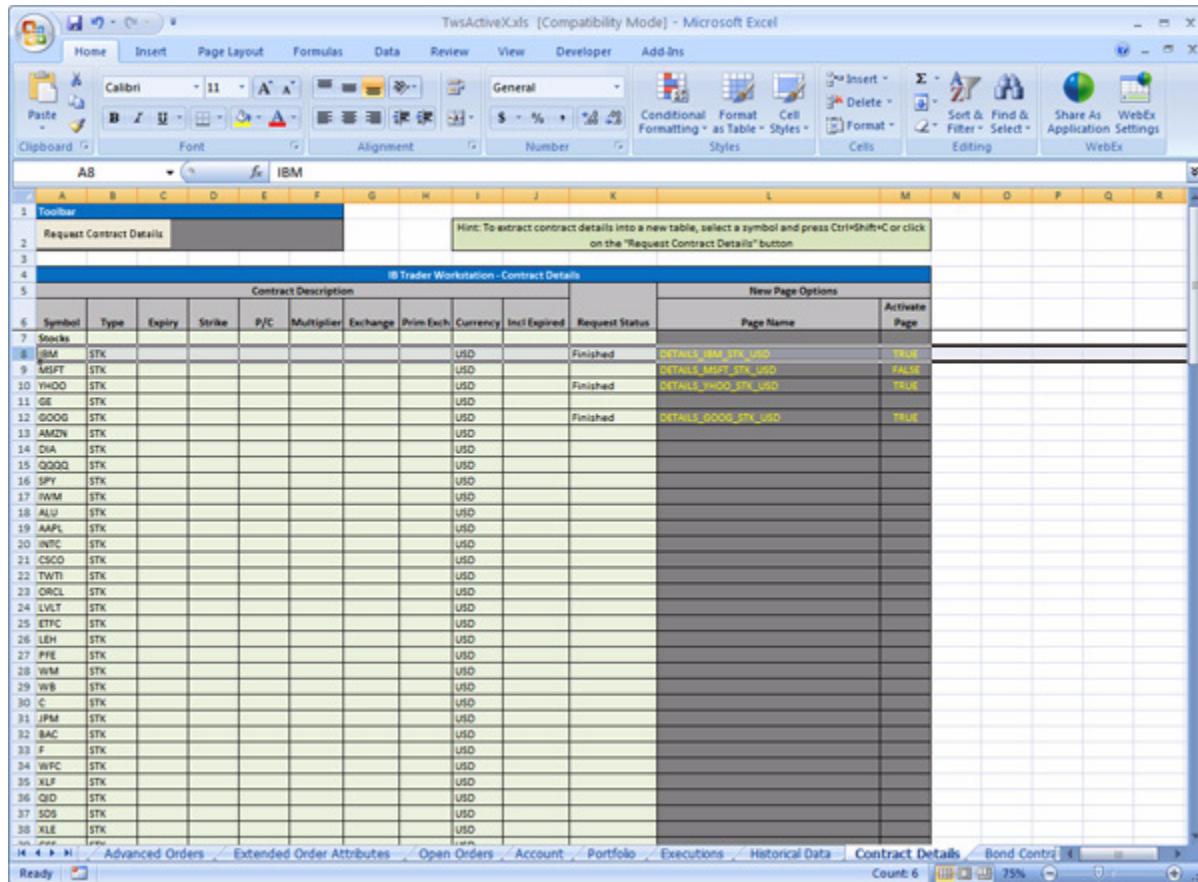
## Historical Data Page Toolbar Buttons

The toolbar on the Historical Data page includes the following buttons.

Button	Description
<b>Create Ticker</b>	Opens the <a href="#">Ticker box</a> . Enter information to create a market data line.
<b>Combo Legs</b>	Opens the <a href="#">Combination Legs box</a> . Enter contract details to create legs of a combination order one by one.
<b>Request Historical Data</b>	Submits your historical data query to TWS and displays the results on a separate worksheet page.
<b>Cancel Historical Data</b>	Cancels the historical data request.

## Contract Details Page

Use the Contract Details page to request contract-specific information such as supported order types, valid exchanges, the contract ID, and so on.



## Requesting Contract Details

**Note:** Ensure that TWS is running, and that you have connected the spreadsheet to TWS.

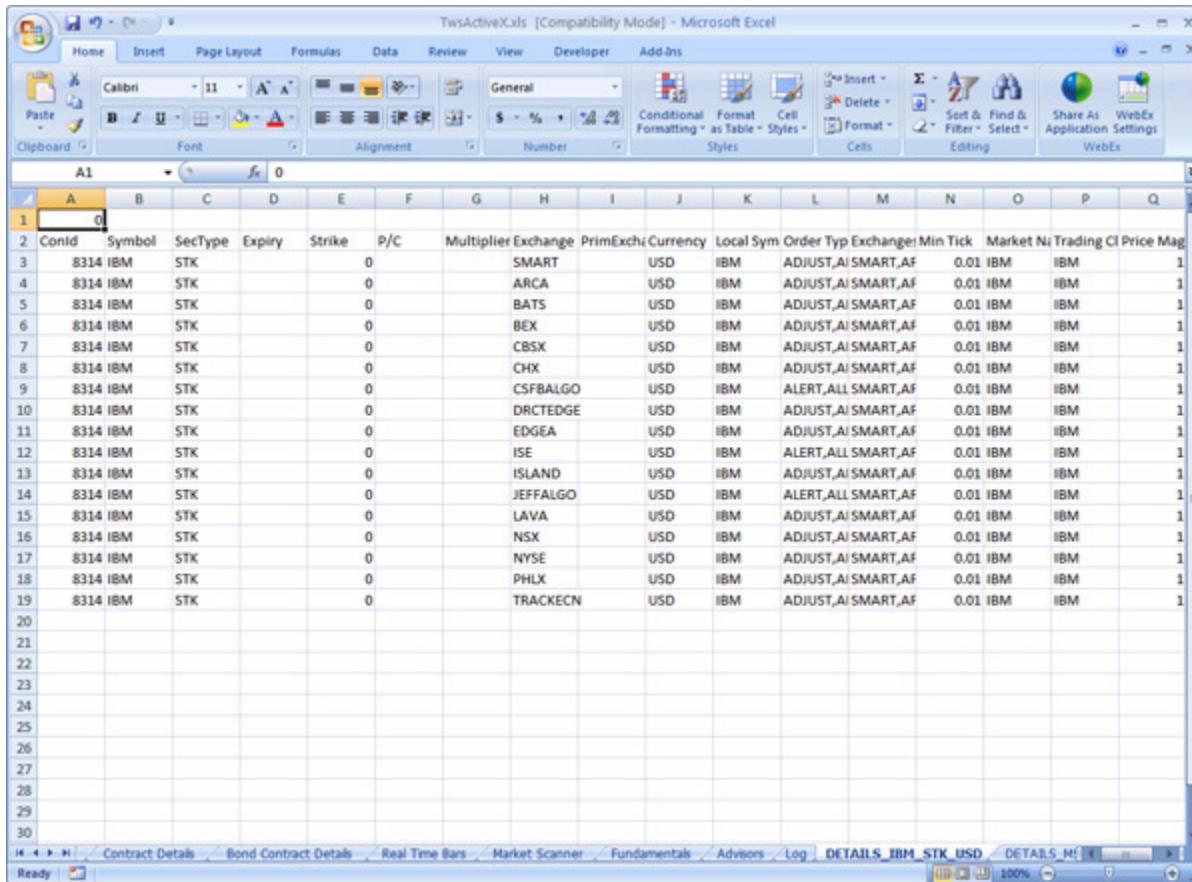
### To request details for a contract

- Click the **Contract Details** tab at the bottom of the spreadsheet to open the Contract Details page.
- Select or enter the ticker symbol for which you want to request contract details.
- To request contract details for an expired contract, type **TRUE** in the *Incl Expired* cell.
- Select the row, then click **Request Contract Details** on the toolbar. The status of your request displays in the *Request Status* cell.

In the *Activate Page* cell, enter **TRUE** to display the results page on top of the current window. Enter **FALSE** to display the page on a separate tab in the spreadsheet without displaying on top of the current window.

The results are displayed on a new tabbed page in the spreadsheet, the name of which is specified in the *Page Name* cell.

The following figure shows a typical contract details page.



The screenshot shows a Microsoft Excel window titled "TwsActiveX.xls [Compatibility Mode] - Microsoft Excel". The ribbon menu includes Home, Insert, Page Layout, Formulas, Data, Review, View, Developer, and Add-Ins. The Home tab is selected, showing standard toolbar icons for Paste, Font, Alignment, Number, Styles, Cells, and Editing. The main content area displays a data grid with approximately 20 rows of contract information. The columns include ConId, Symbol, SecType, Expiry, Strike, P/C, Multiplier, Exchange, PrimExch, Currency, Local Sym, Order Typ, Exchange, Min Tick, Market Ni, Trading Cl, Price Mag, and a blank column at the end. The data shows various stocks (STK) from different exchanges like SMART, ARCA, BATS, BEX, CBSX, CHX, CSFBALGO, DRCTEDGE, EDGEA, ISE, ISLAND, JEFFALGO, LAVA, NSX, NYSE, PHLX, and TRACKCN, with values ranging from 0 to 0.01 IBM.

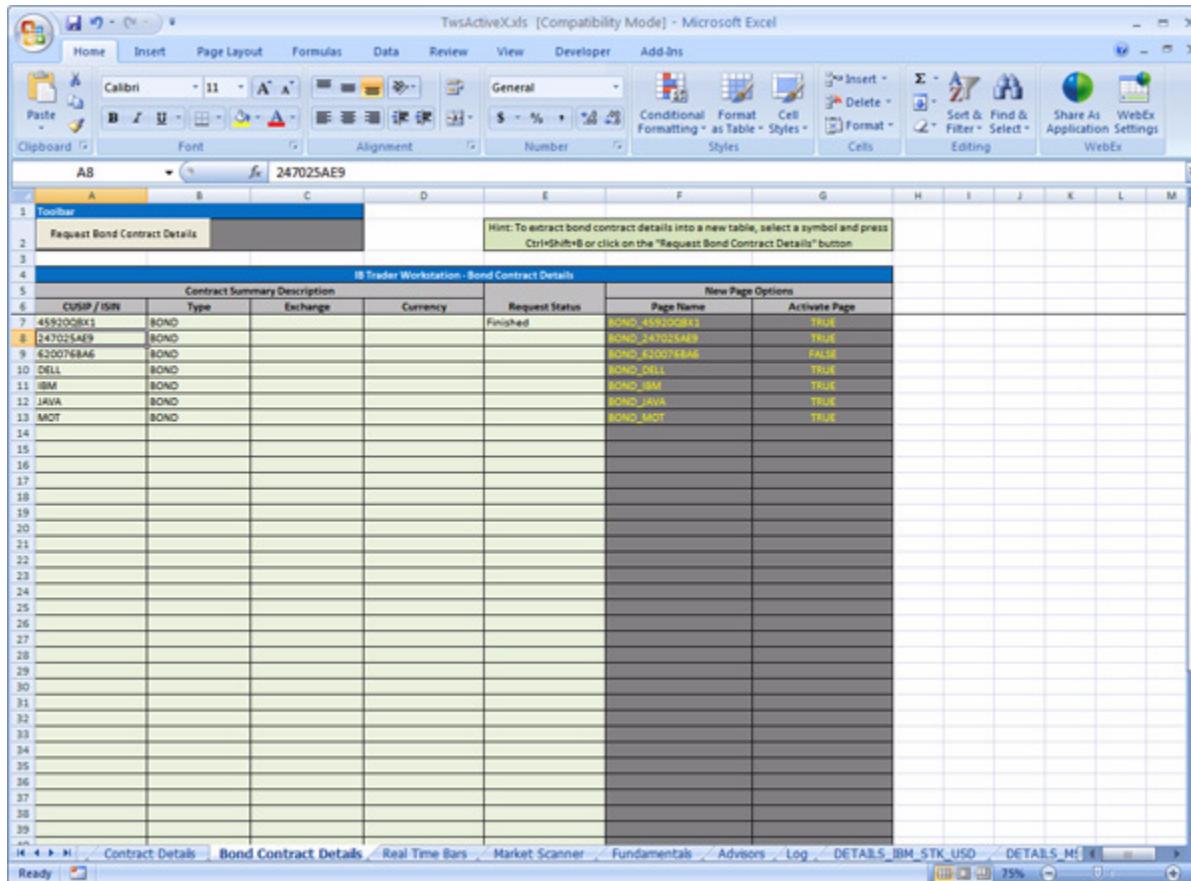
## Contract Details Page Toolbar Buttons

The toolbar on the Contract Details page includes the following button:

Button	Description
<b>Request Contract Details</b>	Returns information on the selected contract.

## Bond Contract Details Page

Use the Bond Contract Details page to request contract-specific information for bonds, including the coupon, ratings, bond type, maturity date, and so on.



## **Requesting Bond Contract Details**

**Note:** Ensure that TWS is running, and that you have connected the spreadsheet to TWS.

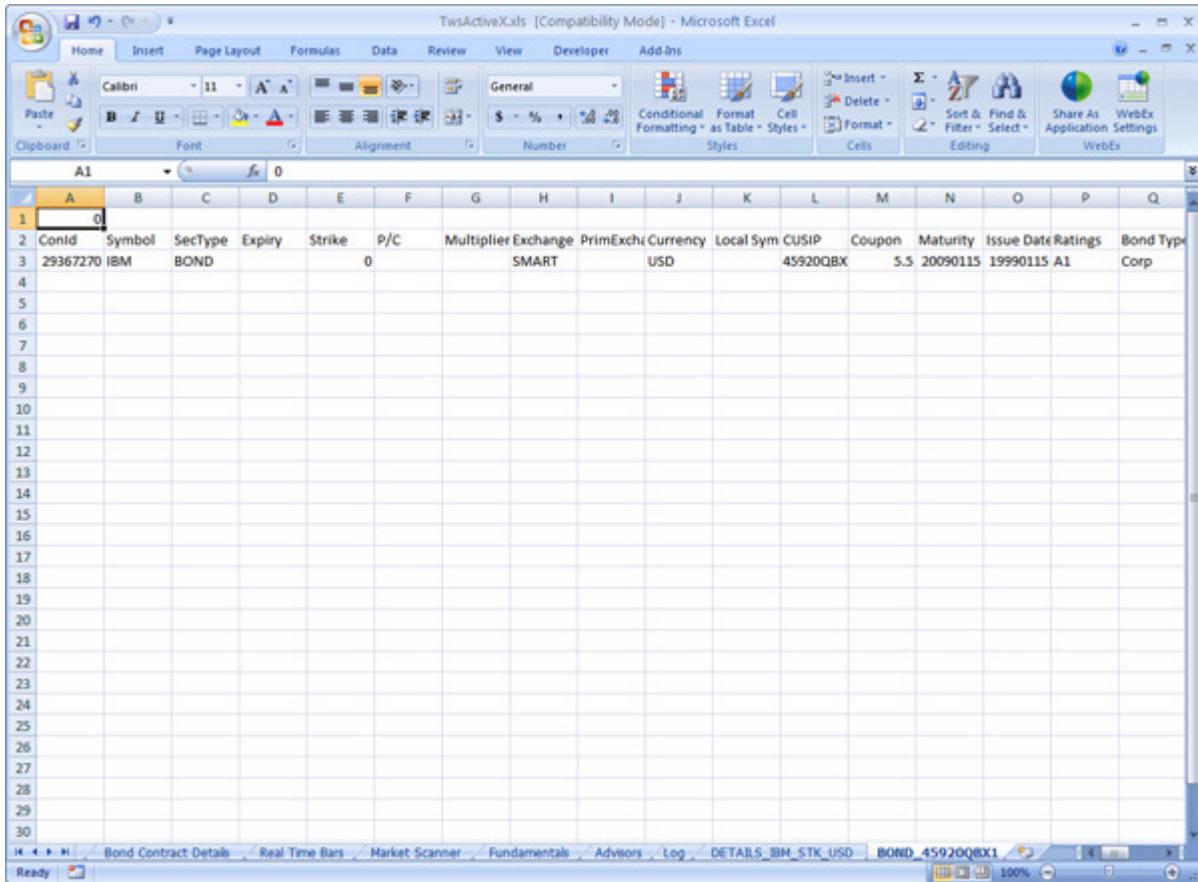
## To request details for a bond contract

- 1 Click the **Bond Contract Details** tab at the bottom of the spreadsheet to open the Bond Contract Details page.
  - 2 Select or enter the ticker symbol for which you want to request bond contract details.
  - 3 Select the row, then click **Request Bond Contract Details** on the toolbar. The status of your request displays in the *Request Status* cell.

In the **Activate Page** cell, enter **TRUE** to display the results page on top of the current window. Enter **FALSE** to display the page on a separate tab in the spreadsheet without displaying on top of the current window.

The results are displayed on a new tabbed page in the spreadsheet, the name of which is specified in the *Page Name* cell.

The following figure shows a typical bond contract details page.



## Bond Contract Details Page Toolbar Buttons

The toolbar on the Bond Contract Details page includes the following button:

Button	Description
<b>Request Bond Contract Details</b>	Gets bond information data for the selected contract.

## Real Time Bars Page

Real time bars allow you to get a summary of real-time market data every five seconds, including the opening and closing price, and the high and the low within that five-second period (using TWS charting terminology, we call these five-second periods "bars"). You can also get data showing trades, midpoints, bids or asks. You request real time bars on the Real Time Bars page, which is shown below.

The screenshot shows a Microsoft Excel window titled "TwsActiveX.xls [Compatibility Mode] - Microsoft Excel". The ribbon tabs include Home, Insert, Page Layout, Formulas, Data, Review, View, Developer, and Add-Ins. The "Developer" tab is selected. The main content area displays a table titled "IB Trader Workstation - Real Time Bars". The table has columns for Symbol, Type, Expiry, Strike, P/C, Multiplier, Exchange, Prim Exch, Currency, What To Show, RTH Only, Subscription Status, Date/Time, Open, High, Low, Close, Volume, WAP, and Count. A row for "SPY" is highlighted in yellow. A tooltip at the top right of the table area says: "Hint: To request real time bars, select a symbol or group of symbols and either press Ctrl+Shift+T or click on the 'Request Real Time Bars' button". The status bar at the bottom shows "Ready" and "75%".

### To request real time bars

**Note:** Ensure that TWS is running, and that you have connected the spreadsheet to TWS.

- 1 Click the **Real Time Bars** tab at the bottom of the spreadsheet to open the Real Time Bars page.
- 2 Select or enter the ticker symbol for which you want to request real time bars.

- 3** Enter the following information for the selected row:
  - In the *What to Show* cell, enter TRADES, BID, ASK or MIDPOINT.
  - In the *RTH Only* cell, enter **0** to return all data available during the time span requested, including time intervals when the market in question was outside of regular trading hours. Enter **1** to return only data within the regular trading hours for the product requested is returned, even if the time span falls partially or completely outside.
- 4** Select the row, then click **Request Real Time Bars** on the toolbar. The status of your request displays in the *Subscription Status* cell.

Results are displayed in the *Real Time Bars* cells on the right side of the page.

## Real Time Bars Page Toolbar Buttons

The toolbar on the Real Time Bars page includes the following buttons.

Button	Description
<b>Create Ticker</b>	Opens the <a href="#">Ticker box</a> . Enter information to create a market data line.
<b>Request Real Time Bars</b>	Submits your real time bars request to TWS and displays the results in the dark gray cells the right side of the page.
<b>Cancel Real Time Bars</b>	Cancels the real time bars request.
<b>Clear Real Time Bars Table</b>	Clears all real time bar data from the page.

## Market Scanner Page

Use the Market Scanner page to subscribe to TWS market scanners. These scanners allow you to define criteria and set filters that return the top x number of underlyings which meet all scan criteria. The scan is continually updated in real time.

You can also display market scanner parameters from this page.

## Starting a Market Scanner Subscription

**Note:** Ensure that TWS is running, and that you have connected the spreadsheet to TWS.

### To start a scanner subscription

- 1 Click the **Market Scanner** tab at the bottom of the spreadsheet.
- 2 Highlight an existing scanner row, or enter information for a different market scanner:
  - a Type the name of the scan results page in the *Page Name* cell.
  - b Type **TRUE** or **FALSE** in the *Activate Page* cell.

Setting this cell to TRUE forces the scan results page to pop to the front of your application every time it updates. To stop this behavior, set the value of this field to FALSE.
  - c Type values for the rest of the scan parameters in the lightly shaded section of the page. You can get all of the scan codes from the market scanner parameters.
- 3 Click **Request Scanner Subscription** on the toolbar. A new page for the scanner is created and is displayed after the subscription is processed.

## Market Scanner Parameters

You can display all of the market scanner parameters from the Market Scanner page. Scanner parameters are returned to the spreadsheet from TWS as an XML file.

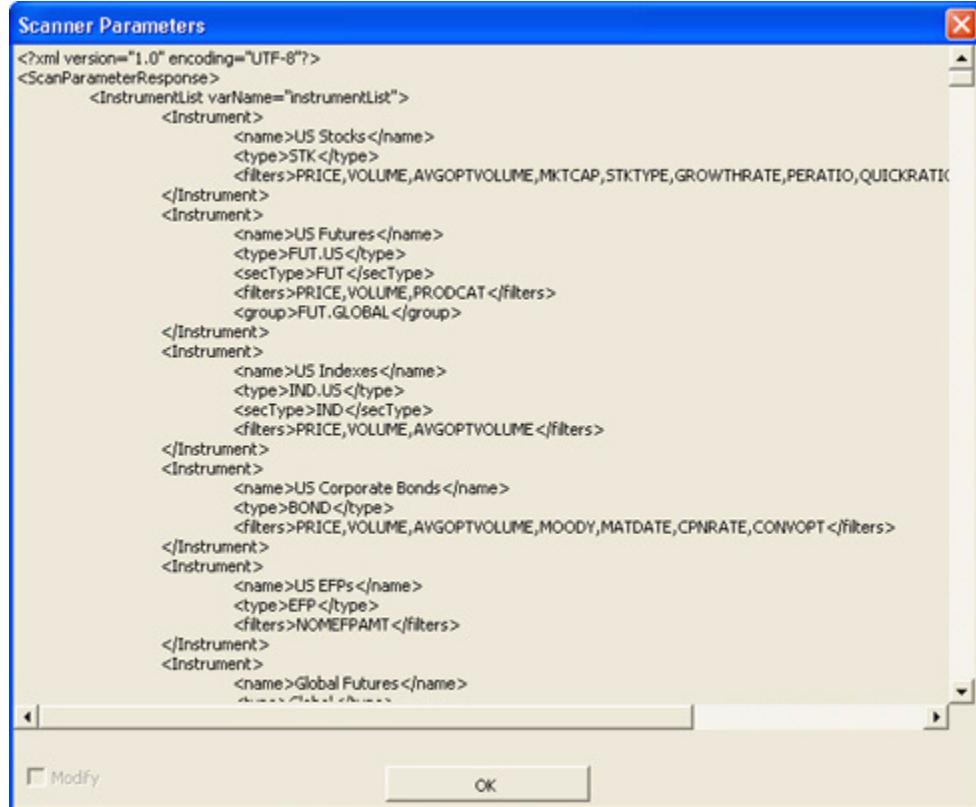
**Note:** Ensure that TWS is running, and that you have connected the spreadsheet to TWS.

### To view scanner parameters

- 1 Click the **Market Scanner** tab at the bottom of the spreadsheet.
- 2 Click **Request Scanner Parameters** on the toolbar.

The entire scanner parameters XML file is displayed in a window.
- 3 To save the parameters in a convenient file on your computer, manually select part or all of the contents of the XML file in the Scanner Parameters window, then copy and paste it into a separate text document.
- 4 Click **OK** to close the Scanner Parameters window.

The Scanner Parameters window is shown on the next page.



## Available Market Scanners

The following table shows the available market scanners in the ActiveX for Excel API spreadsheet.

Market Scanner (Scan Code)	Description
Low Opt Volume P/C Ratio (LOW_OPT_VOL_PUT_CALL_RATIO)*	Put option volumes are divided by call option volumes and the top underlying symbols with the lowest ratios are displayed.
High Option Imp Vol Over Historical (HIGH_OPT_IMP_VOLAT_OVER_HIST)*	Shows the top underlying contracts (stocks or indices) with the largest divergence between implied and historical volatilities.
Low Option Imp Vol Over Historical (LOW_OPT_IMP_VOLAT_OVER_HIST)*	Shows the top underlying contracts (stocks or indices) with the smallest divergence between implied and historical volatilities.
Highest Option Imp Vol (HIGH_OPT_IMP_VOLAT)*	Shows the top underlying contracts (stocks or indices) with the highest vega-weighted implied volatility of near-the-money options with an expiration date in the next two months.
Top Option Imp Vol % Gainers (TOP_OPT_IMP_VOLAT_GAIN)*	Shows the top underlying contracts (stocks or indices) with the largest percent gain between current implied volatility and yesterday's closing value of the 15 minute average of implied volatility.
Top Option Imp Vol % Losers (TOP_OPT_IMP_VOLAT_LOSE)*	Shows the top underlying contracts (stocks or indices) with the largest percent loss between current implied volatility and yesterday's closing value of the 15 minute average of implied volatility.
High Opt Volume P/C Ratio (HIGH_OPT_VOLUME_PUT_CALL_RATIO)	Put option volumes are divided by call option volumes and the top underlying symbols with the highest ratios are displayed.
Low Opt Volume P/C Ratio (LOW_OPT_VOLUME_PUT_CALL_RATIO)	Put option volumes are divided by call option volumes and the top underlying symbols with the lowest ratios are displayed.
Most Active by Opt Volume (OPT_VOLUME_MOST_ACTIVE)	Displays the most active contracts sorted descending by options volume.
Hot by Option Volume (HOT_BY_OPT_VOLUME)	Shows the top underlying contracts for highest options volume over a 10-day average.
High Option Open Interest P/C Ratio (HIGH_OPT_OPEN_INTEREST_PUT_CALL_RATIO)	Returns the top 50 contracts with the <b>highest</b> put/call ratio of outstanding option contracts.
Low Option Open Interest P/C Ratio (LOW_OPT_OPEN_INTEREST_PUT_CALL_RATIO)	Returns the top 50 contracts with the <b>lowest</b> put/call ratio of outstanding option contracts.
Top % Gainers (TOP_PERC_GAIN)	Contracts whose last trade price shows the highest percent increase from the previous night's closing price.

Market Scanner (Scan Code)	Description
Most Active (MOST_ACTIVE)	Contracts with the highest trading volume today, based on units used by TWS (lots for US stocks; contract for derivatives and non-US stocks).  The sample spreadsheet includes two Most Active scans: Most Active List, which displays the most active contracts in the NASDAQ, NYSE and AMEX markets, and Most Active US, which displays the most active stocks in the United States.
Top % Losers (TOP_PERC_LOSE)	Contracts whose last trade price shows the lowest percent increase from the previous night's closing price.
Hot Contracts by Volume (HOT_BY_VOLUME)	Contracts where: <ul style="list-style-type: none"> <li>• today's Volume/avgDailyVolume is highest.</li> <li>• avgDailyVolume is a 30-day exponential moving average of the contract's daily volume.</li> </ul>
Top % Futures Gainers (TOP_PERC_GAIN)	Futures whose last trade price shows the highest percent increase from the previous night's closing price.
Hot Contracts by Price (HOT_BY_PRICE)	Contracts where: <ul style="list-style-type: none"> <li>• (lastTradePrice-prevClose)/avgDailyChange is highest in absolute value (positive or negative).</li> <li>• The avgDailyChange is defined as an exponential moving average of the contract's (dailyClose-dailyOpen)</li> </ul>
Top Trade Count (TOP_TRADE_COUNT)	The top trade count during the day.
Top Trade Rate (TOP_TRADE_RATE)	Contracts with the highest number of trades in the past 60 seconds (regardless of the sizes of those trades).
Top Price Range (TOP_PRICE_RANGE)	The largest difference between today's high and low, or yesterday's close if outside of today's range.
Hot by Price Range (HOT_BY_PRICE_RANGE)	The largest price range (from Top Price Range calculation) over the volatility.
Top Volume Rate (TOP_VOLUME_RATE)	The top volume rate per minute.
Lowest Option Imp Vol (LOW_OPT_IMP_VOLAT)	Shows the top underlying contracts (stocks or indices) with the lowest vega-weighted implied volatility of near-the-money options with an expiration date in the next two months.

Market Scanner (Scan Code)	Description
Most Active by Opt Open Interest (OPT_OPEN_INTEREST_MOST_ACTIVE)	Returns the top 50 underlying contracts with the (highest number of outstanding call contracts) + (highest number of outstanding put contracts)
Not Open (NOT_OPEN)	Contracts that have not traded today.
Halted (HALTED)	Contracts for which trading has been halted.
Top % Gainers Since Open (TOP_OPEN_PERC_GAIN)	Shows contracts with the highest percent price INCREASE between the last trade and opening prices.
Top % Losers Since Open (TOP_OPEN_PERC_LOSE)	Shows contracts with the highest percent price DECREASE between the last trade and opening prices.
Top Close-to-Open % Gainers (HIGH_OPEN_GAP)	Shows contracts with the highest percent price INCREASE between the previous close and today's opening prices.
Top Close-to-Open % Losers (LOW_OPEN_GAP)	Shows contracts with the highest percent price DECREASE between the previous close and today's opening prices.
Lowest Option Imp Vol (LOW_OPT_IMP_VOLAT)*	Shows the top underlying contracts (stocks or indices) with the lowest vega-weighted implied volatility of near-the-money options with an expiration date in the next two months.
Top Option Imp Vol % Gainers (TOP_OPT_IMP_VOLAT_GAIN)*	Shows the top underlying contracts (stocks or indices) with the largest percent gain between current implied volatility and yesterday's closing value of the 15 minute average of implied volatility.
Top Option Imp Vol % Losers (TOP_OPT_IMP_VOLAT_LOSE)*	Shows the top underlying contracts (stocks or indices) with the largest percent loss between current implied volatility and yesterday's closing value of the 15 minute average of implied volatility.
13-Week High (HIGH_VS_13W_HL)	The highest price for the past 13 weeks.
13-Week Low (LOW_VS_13W_HL)	The lowest price for the past 13 weeks.
26-Week High (HIGH_VS_26W_HL)	The highest price for the past 26 weeks.
26-Week Low (LOW_VS_26W_HL)	The lowest price for the past 26 weeks.
52-Week High (HIGH_VS_52W_HL)	The highest price for the past 52 weeks.
52-Week Low (LOW_VS_52W_HL)	The lowest price for the past 52 weeks.

Market Scanner (Scan Code)	Description
<b>EFP</b> - High Synth Bid Rev Yield (HIGH_SYNTH_BID_REV_NAT_YIELD)	Highlights the highest synthetic EFP interest rates available. These rates are computed by taking the price differential between the SSF and the underlying stock and netting dividends to calculate an annualized synthetic implied interest rate over the period of the SSF. The High rates may present an investment opportunity.
<b>EFP</b> - Low Synth Bid Rev Yield (LOW_SYNTH_BID_REV_NAT_YIELD)	Highlights the lowest synthetic EFP interest rates available. These rates are computed by taking the price differential between the SSF and the underlying stock and netting dividends to calculate an annualized synthetic implied interest rate over the period of the SSF. The Low rates may present a borrowing opportunity.

\*30-day (V30) Implied Volatilities:

Implied volatility is calculated using a 100-step binary tree for American style options, and a Black-Scholes model for European style options. Interest rates are calculated using the settlement prices from the day's Eurodollar futures contracts, and dividends are based on historical payouts.

The IB 30-day volatility is the at-market volatility estimated for a maturity thirty calendar days forward of the current trading day. It is based on option prices from two consecutive expiration months. The first expiration month is that which has at least eight calendar days to run. The implied volatility is estimated for the eight options on the four closest to market strikes in each expiry. The implied volatilities are fit to a parabola as a function of the strike price for each expiry. The at-the-market implied volatility for an expiry is then taken to be the value of the fit parabola at the expected future price for the expiry. A linear interpolation (or extrapolation, as required) of the 30-day variance based on the squares of the at-market volatilities is performed. V30 is then the square root of the estimated variance. If there is no first expiration month with less than sixty calendar days to run, we do not calculate a V30.

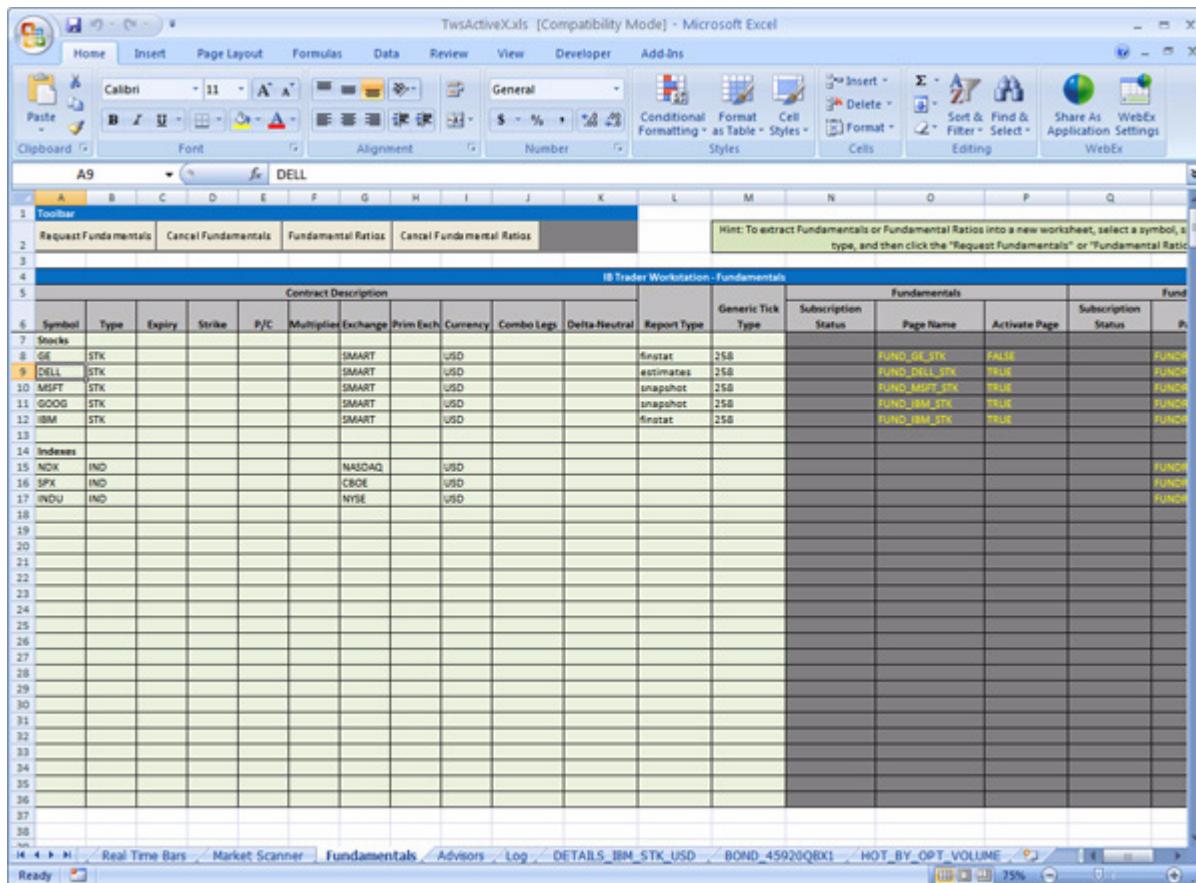
## Market Scanner Page Toolbar Buttons

The toolbar on the Market Scanner page includes the following buttons.

Button	Description
<b>Request Scanner Parameters</b>	Displays all scanner parameters in an XML file in a separate window.
<b>Request Scanner Subscription</b>	Creates and displays a new page for results of the selected market scanner.
<b>Cancel Scanner Subscription</b>	Cancels the market scanner.

## Fundamentals Page

Use the Fundamentals page to receive Reuters global fundamental data and fundamental ratios. There must be a paid subscription to Reuters Fundamental set up in Account Management before you can receive this data.



**To receive Reuters global fundamental data and fundamental ratios**

**Note:** Ensure that TWS is running, and that you have connected the spreadsheet to TWS.

- 1 Click the **Fundamentals** tab at the bottom of the spreadsheet to display the Fundamentals page.
  - 2 Enter information about the contract for which you want fundamentals data or ratios into the *Contract Description* cells.
  - 3 In the *Report Type* cell, enter the type of report you wish to view:
    - **finstat** - Financial Statement
    - **estimates** - Estimates
    - **snapshot** - Summary

- 4 In the *Generic Tick Type* cell, enter **258** as the tick type value. For details on generic tick type values, see [Generic Tick Types](#).
- 5 Enter the following information in the *Fundamentals* section of the page for fundamental data, or the *Fundamental Ratios* section for fundamental ratios:
  - a Fundamental data and ratio results display on a new page in the spreadsheet. Type the name of the results page in the *Page Name* cell.
  - b Type **TRUE** or **FALSE** in the *Activate Page* cell.

Setting this cell to TRUE forces the results page to pop to the front of your application every time it updates. To stop this behavior, set the value of this field to FALSE.
- 6 Do one of the following:
  - Click **Request Fundamentals** on the toolbar to view fundamentals data. A new page for the results is created and is displayed after the request is processed.
  - Click **Fundamentals Ratios** on the toolbar to view fundamentals ratios. A new page for the results is created and is displayed after the request is processed.

The status of your request appears in the *Subscription Status* cell.

## Fundamentals Page Toolbar Buttons

The toolbar on the Market Scanner page includes the following buttons.

Button	Description
<b>Request Fundamentals</b>	Requests fundamentals data, which displays on a new page in the spreadsheet based on the information you enter on the page.
<b>Cancel Fundamentals</b>	Cancels fundamentals data.
<b>Fundamental Ratios</b>	Requests fundamentals ratios, which displays on a new page in the spreadsheet based on the information you enter on the page.
<b>Cancel Fundamental Ratios</b>	Cancels fundamentals ratios.

## Advisors Page

If you are a Financial Advisor and manage multiple accounts, use the Advisors page to create FA orders that:

- allocate shares to a single managed account
- use FA account groups and methods
- use allocation profiles

Symbol	Type	Expiry	Strike	P/C	Multiplied	Exchange	Prim Exch	Currency	Combo Legs	Delta-Neutral	Order Description				Id	Status	Filled	Remaining
											Action	Quantity	Order Type	Limit Price				
IBM	STK					SMART		USD			BUY	1000	LMT	80.00				
GOOG	STK					SMART		USD			BUY	1000	LMT	360.00				
MSFT	STK					SMART		USD			BUY	1000	LMT	21.00				
IBM	STK					SMART		USD			BUY	1000	LMT	80.00				

**Note:**

You must set up your managed accounts, account groups, methods and allocation profiles in TWS before you can place FA orders in the ActiveX for Excel API sample spreadsheet.

## Allocating Shares to a Single Account

You can use the **Advisors** page to set up an order and allocate all shares in the order to a single account.

### To allocate shares to a single account:

**Note:** Ensure that TWS is running, and that you have connected the spreadsheet to TWS.

- 1 Create an account group in TWS.
- 2 Click the **Advisors** tab at the bottom of the spreadsheet.
- 3 Enter the contract information in the *Contract Description* cells, then enter the order information in the *Order Description* cells.
- 4 Click the **Extended Order Attributes** tab. Enter the account code in the *Value* cell for the *Account (Institutional only)* extended order attribute.
- 5 Click the **Advisors** tab.
- 6 Highlight the order row, then click the **Apply Extended** button to apply the *Account* order attribute value to the order. The *Account* value is applied to the selected order and displayed in the *Extended Order Attributes* section of the page.
- 7 Click the **Place/Modify Order** button.
- 8 When you are done allocating shares to the account, delete the *Account* value from the **Extended Order Attributes** page. If you do not delete this value, it will be applied to all subsequent orders placed from the ActiveX for Excel spreadsheet.

Optionally, you can receive margin and commission information that would result from the order if you placed it by selecting the **WhatIf** check box on the toolbar. In this case, your order is not actually placed. Deselect the check box to place your order without seeing the margin and commission information ahead of time.

## Placing an Order using an FA Account Group and Method

You can also use the **Advisors** page to set up an order using an FA account group and FA method.

### To place an order using an FA account group and FA method:

- 1** Create the FA account group(s) and FA method(s) in TWS.
- 2** Click the **Advisors** tab at the bottom of the spreadsheet.
- 3** Enter the contract information in the *Contract Description* cells, then enter the order information in the *Order Description* cells.
- 4** Click the **Extended Order Attributes** tab. Enter values for the following extended order attributes:
  - *FA Group* - Enter the name of the account group.
  - *FA Method* - Enter the name of the allocation method to use for this order.
  - *FA Percentage* - Enter the percentage used by the PctChange allocation method to use for this order. This attribute applies only to FA groups that use this method.
- 5** Click the **Advisors** tab.
- 6** Highlight the order row, then click the **Apply Extended** button to apply the extended order attribute values to the order. The values for *FA Group*, *FA Method* and *FA Percentage* are applied to the selected order and displayed in the *Extended Order Attributes* section of the page.
- 7** Click the **Place/Modify Order** button.
- 8** When you are done allocating shares to the account, delete the values you entered on the **Extended Order Attributes** page. If you do not delete these values, they will be applied to all subsequent orders placed from the ActiveX for Excel spreadsheet.

Optionally, you can receive margin and commission information that would result from the order if you placed it by selecting the **WhatIf** check box on the toolbar. In this case, your order is not actually placed. Deselect the check box to place your order without seeing the margin and commission information ahead of time.

## Placing an Order using an Allocation Profile

You can also use the **Advisors** page to set up an order using an FA allocation profile.

### To place an order using an FA allocation profile:

- 1** Create the FA allocation profile in TWS.
- 2** Click the **Advisors** tab at the bottom of the spreadsheet.
- 3** Enter the contract information in the *Contract Description* cells, then enter the order information in the *Order Description* cells.
- 4** Click the **Extended Order Attributes** tab. Enter the name of the allocation profile in the *Value* field for the *FA Profile* extended order attribute.
- 5** Click the **Advisors** tab.
- 6** Highlight the order row, then click the **Apply Extended** button to apply the extended order attribute value to the order. The value for *FA Profile* is applied to the selected order and displayed in the *Extended Order Attributes* section of the page.
- 7** Click the **Place/Modify Order** button.
- 8** When you are done allocating shares to the account, delete the *FA Profile* value you entered on the **Extended Order Attributes** page. If you do not delete this value, it will be applied to all subsequent orders placed from the ActiveX for Excel spreadsheet.

Optionally, you can receive margin and commission information that would result from the order if you placed it by selecting the **WhatIf** check box on the toolbar. In this case, your order is not actually placed. Deselect the check box to place your order without seeing the margin and commission information ahead of time.

## Advisors Page Toolbar Buttons

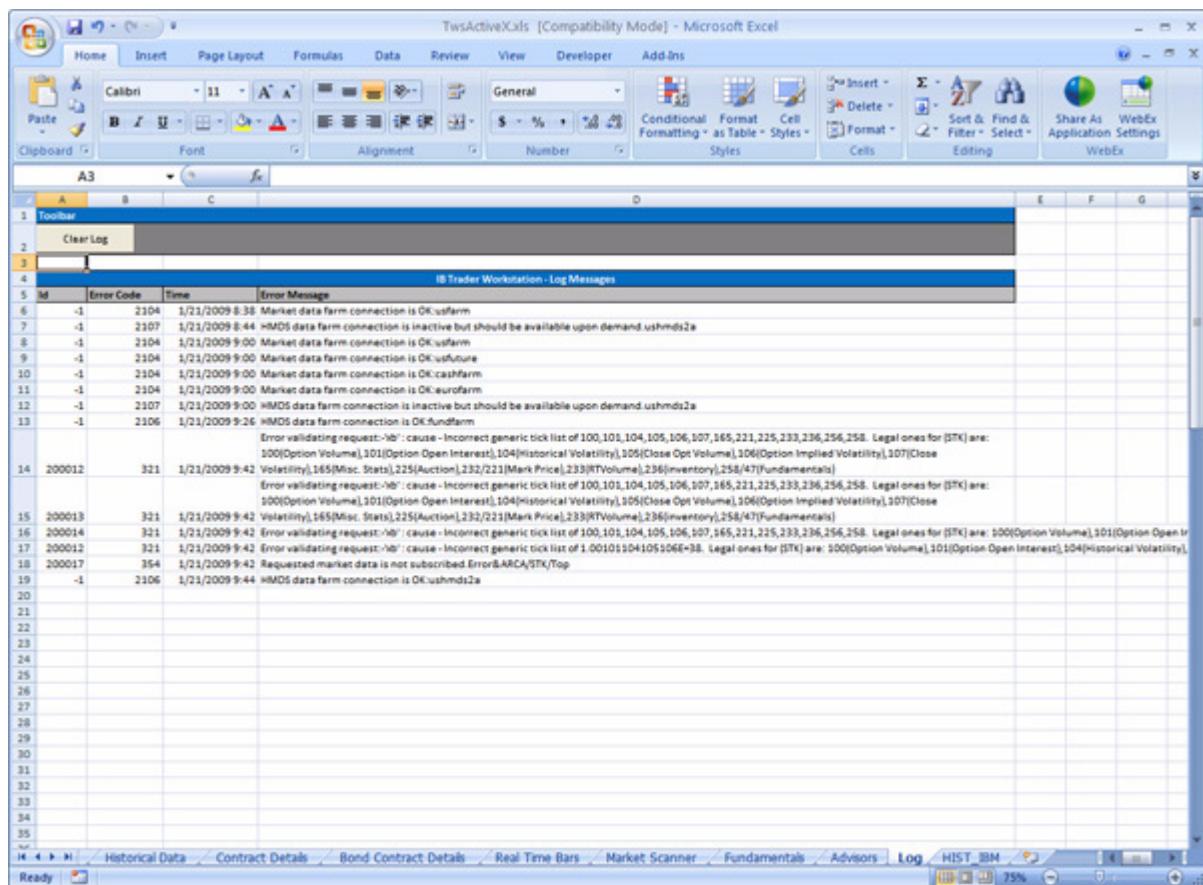
The toolbar on the Basic Orders page includes the following buttons:

Button	Description
<b>Create Ticker</b>	Opens the <a href="#">Ticker box</a> . Enter information to create a market data line.
<b>Combo Legs</b>	Opens the <a href="#">Combination Legs box</a> . Enter contract details to create legs of a combination order one by one.
<b>Apply Extended Template</b>	Applies the current values on the Extended Order Attributes page to the highlighted order row.
<b>Place/Modify Orders</b>	After you have completed the Order Description fields, and defined any extended attributes, click to create an order for the selected contract.
<b>Cancel Order</b>	This button cancels the order(s) you have highlighted.
<b>Clear Order Statuses</b>	Clears all information from the <i>Order Status</i> cells.

## Log Page

The Log page displays all error messages received while logged into TWS and using the ActiveX for Excel spreadsheet. You can clear all the information on the page by clicking **Clear Log** on the toolbar.

Here is an example of a typical Log page:



The screenshot shows a Microsoft Excel window titled "TwsActiveX.xls [Compatibility Mode] - Microsoft Excel". The ribbon tabs include Home, Insert, Page Layout, Formulas, Data, Review, View, Developer, and Add-ins. The developer tab is selected. The main content area displays a table titled "IB Trader Workstation - Log Messages". The table has columns for ID, Error Code, Time, and Error Message. The data in the table is as follows:

ID	Error Code	Time	Error Message
6	-1	1/21/2009 8:38	Market data farm connection is OK;usfarm
7	-1	1/21/2009 8:44	HMD5 data farm connection is inactive but should be available upon demand.ushmdd2a
8	-1	1/21/2009 9:00	Market data farm connection is OK;usfarm
9	-1	1/21/2009 9:00	Market data farm connection is OK;usfuture
10	-1	1/21/2009 9:00	Market data farm connection is OK;cashfarm
11	-1	1/21/2009 9:00	Market data farm connection is OK;eurofarm
12	-1	1/21/2009 9:00	HMD5 data farm connection is inactive but should be available upon demand.ushmdd2a
13	-1	1/21/2009 9:26	HMD5 data farm connection is OK;fundfarm
			Error validating request:<id>; cause: -Incorrect generic tick list of 100,101,104,105,106,107,165,221,225,233,236,256,258. Legal ones for [STK] are: 100[Option Volume],101[Option Open Interest],104[Historical Volatility],105[Close Opt Volume],106[Option Implied Volatility],107[Close Volatility],165[Misc. Stats],225[Auction],232/221[Mark Price],233[RTVolume],236[Inventory],256/47[Fundamentals]
14	200012	321	1/21/2009 9:42 Error validating request:<id>; cause: -Incorrect generic tick list of 100,101,104,105,106,107,165,221,225,233,236,256,258. Legal ones for [STK] are: 100[Option Volume],101[Option Open Interest],104[Historical Volatility],105[Close Opt Volume],106[Option Implied Volatility],107[Close Volatility],165[Misc. Stats],225[Auction],232/221[Mark Price],233[RTVolume],236[Inventory],256/47[Fundamentals]
15	200013	321	1/21/2009 9:42 Volatility,165[Misc. Stats],225[Auction],232/221[Mark Price],233[RTVolume],236[Inventory],256/47[Fundamentals]
16	200014	321	1/21/2009 9:42 Error validating request:<id>; cause: -Incorrect generic tick list of 100,101,104,105,106,107,165,221,225,233,236,256,258. Legal ones for [STK] are: 100[Option Volume],101[Option Open Interest],104[Historical Volatility],105[Close Opt Volume],106[Option Implied Volatility],107[Close Volatility],165[Misc. Stats],225[Auction],232/221[Mark Price],233[RTVolume],236[Inventory],256/47[Fundamentals]
17	200012	321	1/21/2009 9:42 Error validating request:<id>; cause: -Incorrect generic tick list of 1,001,010,104,105,106,107,165,221,225,233,236,256,258. Legal ones for [STK] are: 100[Option Volume],101[Option Open Interest],104[Historical Volatility],105[Close Opt Volume],106[Option Implied Volatility],107[Close Volatility],165[Misc. Stats],225[Auction],232/221[Mark Price],233[RTVolume],236[Inventory],256/47[Fundamentals]
18	200017	354	1/21/2009 9:42 Requested market data is not subscribed.Error&ARCA/STK/Top
19	-1	1/21/2009 9:44	HMD5 data farm connection is OK;ushmdd2a
20			
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
31			
32			
33			
34			
35			

The status bar at the bottom shows "Ready" and "75%".

# POSIX

This chapter describes the POSIX API.

The POSIX API is based on our [C++ API code](#). The C++ code was refactored so it could be built on any POSIX-compliant platform. Use this new POSIX API to build a TWS API on Linux, and on Windows in non-MFC applications.

**Note:** Although the pre-existing public interface has been preserved, you must recompile your client applications.

We also include a POSIX test client. The API installation directory includes these directories for the POSIX API: PosixSocketClient and TestPosixSocketClient. The POSIX test client uses the same methods as the C++ Socket client, plus it exposes several extra methods that clients must call when data is available on a socket for read/write. Refer to `TestPosixSocketClient` as an example. Please note that this test client is greatly simplified. For real POSIX API applications, you will have to use a select system of some kind to manage several sockets and/or asynchronous events.



# Reference Tables

This chapter includes the following TWS API reference information:

- [API Message Codes](#)
- [Tick Types](#)
- [Generic Tick Types](#)
- [TAG Values for FUNDAMENTAL\\_RATIOS tickType](#)
- [Extended Order Attributes](#)
- [Available Market Scanners](#)
- [IBAlgo Parameters](#)

## API Message Codes

This section lists all of the API [Error](#), [System](#) and [Warning](#) message codes and their descriptions.

Message codes shown below that end with a colon ( : ) display additional information.

### Error Codes

Code	Description
100	Max rate of messages per second has been exceeded.
101	Max number of tickers has been reached.
102	Duplicate ticker ID.
103	Duplicate order ID.
104	Can't modify a filled order.
105	Order being modified does not match original order.
106	Can't transmit order ID:
107	Cannot transmit incomplete order.
109	Price is out of the range defined by the <i>Percentage</i> setting at order defaults frame. The order will not be transmitted.
110	The price does not conform to the minimum price variation for this contract.
111	The TIF (Tif type) and the order type are incompatible.
113	The Tif option should be set to DAY for MOC and LOC orders.
114	Relative orders are valid for stocks only.
115	Relative orders for US stocks can only be submitted to SMART, SMART_ECN, INSTINET, or PRIMEX.
116	The order cannot be transmitted to a dead exchange.
117	The block order size must be at least 50.
118	VWAP orders must be routed through the VWAP exchange.
119	Only VWAP orders may be placed on the VWAP exchange.
120	It is too late to place a VWAP order for today.
121	Invalid BD flag for the order. Check "Destination" and "BD" flag.
122	No request tag has been found for order:
123	No record is available for conid:
124	No market rule is available for conid:
125	Buy price must be the same as the best asking price.
126	Sell price must be the same as the best bidding price.
129	VWAP orders must be submitted at least three minutes before the start time.

<b>Code</b>	<b>Description</b>
131	The sweep-to-fill flag and display size are only valid for US stocks routed through SMART, and will be ignored.
132	This order cannot be transmitted without a clearing account.
133	Submit new order failed.
134	Modify order failed.
135	Can't find order with ID =
136	This order cannot be cancelled.
137	VWAP orders can only be cancelled up to three minutes before the start time.
138	Could not parse ticker request:
139	Parsing error:
140	The size value should be an integer:
141	The price value should be a double:
142	Institutional customer account does not have account info
143	Requested ID is not an integer number.
144	Order size does not match total share allocation. To adjust the share allocation, right-click on the order and select "Modify > Share Allocation."
145	Error in validating entry fields -
146	Invalid trigger method.
147	The conditional contract info is incomplete.
148	A conditional order can only be submitted when the order type is set to limit or market.
151	This order cannot be transmitted without a user name.
152	The "hidden" order attribute may not be specified for this order.
153	EFPs can only be limit orders.
154	Orders cannot be transmitted for a halted security.
155	A sizeOp order must have a username and account.
156	A SizeOp order must go to IBSX
157	An order can be EITHER Iceberg or Discretionary. Please remove either the Discretionary amount or the Display size.
158	You must specify an offset amount or a percent offset value.
159	The percent offset value must be between 0% and 100%.
160	The size value cannot be zero.
161	Cancel attempted when order is not in a cancellable state. Order permId =
162	Historical market data Service error message.
163	The price specified would violate the percentage constraint specified in the default order settings.

Code	Description
164	There is no market data to check price percent violations.
165	Historical market Data Service query message.
166	HMDS Expired Contract Violation.
167	VWAP order time must be in the future.
168	Discretionary amount does not conform to the minimum price variation for this contract.

Code	Description
200	No security definition has been found for the request.
201	Order rejected - Reason:
202	Order cancelled - Reason:
203	The security <security> is not available or allowed for this account.

Code	Description
300	Can't find EId with ticker Id:
301	Invalid ticker action:
302	Error parsing stop ticker string:
303	Invalid action:
304	Invalid account value action:
305	Request parsing error, the request has been ignored.
306	Error processing DDE request:
307	Invalid request topic:
308	Unable to create the 'API' page in TWS as the maximum number of pages already exists.
309	Max number (3) of market depth requests has been reached. <b>Note:</b> TWS currently limits users to a maximum of 3 distinct market depth requests. This same restriction applies to API clients, however API clients may make multiple market depth requests for the same security.
310	Can't find the subscribed market depth with tickerId:
311	The origin is invalid.
312	The combo details are invalid.
313	The combo details for leg '<leg number>' are invalid.
314	Security type 'BAG' requires combo leg details.
315	Stock combo legs are restricted to SMART order routing.

<b>Code</b>	<b>Description</b>
316	Market depth data has been HALTED. Please re-subscribe.
317	Market depth data has been RESET. Please empty deep book contents before applying any new entries.
319	Invalid log level <log level>
320	Server error when reading an API client request.
321	Server error when validating an API client request.
322	Server error when processing an API client request.
323	Server error: cause - %s
324	Server error when reading a DDE client request (missing information).
325	Discretionary orders are not supported for this combination of exchange and order type.
326	Unable to connect as the client id is already in use. Retry with a unique client id.
327	Only API connections with clientId set to 0 can set the auto bind TWS orders property.
328	Trailing stop orders can be attached to limit or stop-limit orders only.
329	Order modify failed. Cannot change to the new order type.
330	Only FA or STL customers can request managed accounts list.
331	Internal error. FA or STL does not have any managed accounts.
332	The account codes for the order profile are invalid.
333	Invalid share allocation syntax.
334	Invalid Good Till Date order
335	Invalid delta: The delta must be between 0 and 100.
336	The time or time zone is invalid. The correct format is hh:mm:ss xxx where xxx is an optionally specified time-zone. E.g.: 15:59:00 EST Note that there is a space between the time and the time zone. If no time zone is specified, local time is assumed.
337	The date, time, or time-zone entered is invalid. The correct format is yyyyymmdd hh:mm:ss xxx where yyyyymmdd and xxx are optional. E.g.: 20031126 15:59:00 EST Note that there is a space between the date and time, and between the time and time-zone. If no date is specified, current date is assumed. If no time-zone is specified, local time-zone is assumed.

<b>Code</b>	<b>Description</b>
338	Good After Time orders are currently disabled on this exchange.
339	Futures spread are no longer supported. Please use combos instead.
340	Invalid improvement amount for box auction strategy.
341	Invalid delta. Valid values are from 1 to 100. You can set the delta from the "Pegged to Stock" section of the Order Ticket Panel, or by selecting Page/Layout from the main menu and adding the Delta column.
342	Pegged order is not supported on this exchange.
343	The date, time, or time-zone entered is invalid. The correct format is yyyyymmdd hh:mm:ss xxx where yyyyymmdd and xxx are optional. E.g.: 20031126 15:59:00 EST  Note that there is a space between the date and time, and between the time and time-zone.  If no date is specified, current date is assumed. If no time-zone is specified, local time-zone is assumed.
344	The account logged into is not a financial advisor account.
345	Generic combo is not supported for FA advisor account.
346	Not an institutional account or an away clearing account.
347	Short sale slot value must be 1 (broker holds shares) or 2 (delivered from elsewhere).
348	Order not a short sale -- type must be SSHORT to specify short sale slot.
349	Generic combo does not support "Good After" attribute.
350	Minimum quantity is not supported for best combo order.
351	The "Regular Trading Hours only" flag is not valid for this order.
352	Short sale slot value of 2 (delivered from elsewhere) requires location.
353	Short sale slot value of 1 requires no location be specified.
354	Not subscribed to requested market data.
355	Order size does not conform to market rule.
356	Smart-combo order does not support OCA group.
357	Your client version is out of date.
358	Smart combo child order not supported.
359	Combo order only supports reduce on fill without block(OCA).
360	No whatif check support for smart combo order.
361	Invalid trigger price.
362	Invalid adjusted stop price.

<b>Code</b>	<b>Description</b>
363	Invalid adjusted stop limit price.
364	Invalid adjusted trailing amount.
365	No scanner subscription found for ticker id:
366	No historical data query found for ticker id:
367	Volatility type if set must be 1 or 2 for VOL orders. Do not set it for other order types.
368	Reference Price Type must be 1 or 2 for dynamic volatility management. Do not set it for non-VOL orders.
369	Volatility orders are only valid for US options.
370	Dynamic Volatility orders must be SMART routed, or trade on a Price Improvement Exchange.
371	VOL order requires positive floating point value for volatility. Do not set it for other order types.
372	Cannot set dynamic VOL attribute on non-VOL order.
373	Can only set stock range attribute on VOL or RELATIVE TO STOCK order.
374	If both are set, the lower stock range attribute must be less than the upper stock range attribute.
375	Stock range attributes cannot be negative.
376	The order is not eligible for continuous update. The option must trade on a cheap-to-reroute exchange.
377	Must specify valid delta hedge order aux. price.
378	Delta hedge order type requires delta hedge aux. price to be specified.
379	Delta hedge order type requires that no delta hedge aux. price be specified.
380	This order type is not allowed for delta hedge orders.
381	Your DDE.dll needs to be upgraded.
382	The price specified violates the number of ticks constraint specified in the default order settings.
383	The size specified violates the size constraint specified in the default order settings.
384	Invalid DDE array request.
385	Duplicate ticker ID for API scanner subscription.
386	Duplicate ticker ID for API historical data query.
387	Unsupported order type for this exchange and security type.
388	Order size is smaller than the minimum requirement.
389	Supplied routed order ID is not unique.
390	Supplied routed order ID is invalid.

<b>Code</b>	<b>Description</b>
391	The time or time-zone entered is invalid. The correct format is hh:mm:ss xxx where xxx is an optionally specified time-zone. E.g.: 15:59:00 EST.  Note that there is a space between the time and the time zone.  If no time zone is specified, local time is assumed.
392	Invalid order: contract expired.
393	Short sale slot may be specified for delta hedge orders only.
394	Invalid Process Time: must be integer number of milliseconds between 100 and 2000. Found:
395	Due to system problems, orders with OCA groups are currently not being accepted.
396	Due to system problems, application is currently accepting only Market and Limit orders for this contract.
397	Due to system problems, application is currently accepting only Market and Limit orders for this contract.
398	< > cannot be used as a condition trigger.
399	Order message error

<b>Code</b>	<b>Description</b>
400	Algo order error.
401	Length restriction.
402	Conditions are not allowed for this contract.
403	Invalid stop price.
404	Shares for this order are not immediately available for short sale. The order will be held while we attempt to locate the shares.
405	The child order quantity should be equivalent to the parent order size.
406	The currency < > is not allowed.
407	The symbol should contain valid non-unicode characters only.
408	Invalid scale order increment.
409	Invalid scale order. You must specify order component size.
410	Invalid subsequent component size for scale order.
411	The "Outside Regular Trading Hours" flag is not valid for this order.
412	The contract is not available for trading.
413	What-if order should have the transmit flag set to true.

<b>Code</b>	<b>Description</b>
414	Snapshot market data subscription is not applicable to generic ticks.
415	Wait until previous RFQ finishes and try again.
416	RFQ is not applicable for the contract. Order ID:
417	Invalid initial component size for scale order.
418	Invalid scale order profit offset.
419	Missing initial component size for scale order.
420	Invalid real-time query.
421	Invalid route.
422	The account and clearing attributes on this order may not be changed.
423	Cross order RFQ has been expired. THI committed size is no longer available. Please open order dialog and verify liquidity allocation.
424	FA Order requires allocation to be specified.
425	FA Order requires per-account manual allocations because there is no common clearing instruction. Please use order dialog Adviser tab to enter the allocation.
426	None of the accounts have enough shares.
427	Mutual Fund order requires monetary value to be specified.
428	Mutual Fund Sell order requires shares to be specified.
429	Delta neutral orders are only supported for combos (BAG security type).
430	We are sorry, but fundamentals data for the security specified is not available.
431	What to show field is missing or incorrect.
432	Commission must not be negative.
433	Invalid "Restore size after taking profit" for multiple account allocation scale order.
434	The order size cannot be zero.
435	You must specify an account.
436	You must specify an allocation (either a single account, group, or profile).
437	Order can have only one flag Outside RTH or Allow PreOpen.
438	The application is now locked.
439	Order processing failed. Algorithm definition not found.
440	Order modify failed. Algorithm cannot be modified.
441	Algo attributes validation failed:
442	Specified algorithm is not allowed for this order.
443	Order processing failed. Unknown algo attribute.

<b>Code</b>	<b>Description</b>
444	Volatility Combo order is not yet acknowledged. Cannot submit changes at this time.
445	The RFQ for this order is no longer valid.
446	Missing scale order profit offset.
447	Missing scale price adjustment amount or interval.
448	Invalid scale price adjustment interval.
449	Unexpected scale price adjustment amount or interval.
40	Dividend schedule query failed.

<b>Code</b>	<b>Description</b>
501	Already connected.
502	Couldn't connect to TWS. Confirm that API is enabled in TWS via the Configure>API menu command.
503	Your version of TWS is out of date and must be upgraded.
504	Not connected.
505	Fatal error: Unknown message id.
510	Request market data - sending error:
511	Cancel market data - sending error:
512	Order - sending error:
513	Account update request - sending error:
514	Request for executions - sending error:
515	Cancel order - sending error:
516	Request open order - sending error:
517	Unknown contract. Verify the contract details supplied.
518	Request contract data - sending error:
519	Request market depth - sending error:
520	Cancel market depth - sending error:
521	Set server log level - sending error:
522	FA Information Request - sending error:
523	FA Information Replace - sending error:
524	Request Scanner subscription - sending error:
525	Cancel Scanner subscription - sending error:
526	Request Scanner parameter - sending error:
527	Request Historical data - sending error:
528	Cancel Historical data - sending error:
529	Request real-time bar data - sending error:

<b>Code</b>	<b>Description</b>
530	Cancel real-time bar data - sending error:
531	Request Current Time - Sending error:

### System Message Codes

<b>Code</b>	<b>Description</b>
1100	Connectivity between IB and TWS has been lost.
1101	Connectivity between IB and TWS has been restored- data lost.*
1102	Connectivity between IB and TWS has been restored- data maintained.
1300	TWS socket port has been reset and this connection is being dropped. Please reconnect on the new port - <port_num>
*Market and account data subscription requests must be resubmitted	

### Warning Message Codes

<b>Code</b>	<b>Description</b>
2100	New account data requested from TWS. API client has been unsubscribed from account data.
2101	Unable to subscribe to account as the following clients are subscribed to a different account.
2102	Unable to modify this order as it is still being processed.
2103	A market data farm is disconnected.
2104	A market data farm is connected.
2105	A historical data farm is disconnected.
2106	A historical data farm is connected.
2107	A historical data farm connection has become inactive but should be available upon demand.
2108	A market data farm connection has become inactive but should be available upon demand.
2109	Order Event Warning: Attribute "Outside Regular Trading Hours" is ignored based on the order type and destination. PlaceOrder is now processed.

## Tick Types

The following table lists all possible values for the tickType parameter, which is used in the following [ActiveX events](#), [C++ EWrapper functions](#), and [Java EWrapper methods](#):

- tickPrice()
- tickSize()
- tickOptionComputation()
- tickGeneric()
- tickString()
- tickEFP

<b>Tick Value</b>	<b>Description</b>	<b>Event/Function/Method</b>
0	BID_SIZE	tickSize()
1	BID_PRICE	tickPrice()
2	ASK_PRICE	tickPrice()
3	ASK_SIZE	tickSize()
4	LAST_PRICE	tickPrice()
5	LAST_SIZE	tickSize()
6	HIGH	tickPrice()
7	LOW	tickPrice()
8	VOLUME	tickSize()
9	CLOSE_PRICE	tickPrice()
10	BID_OPTION_COMPUTATION	tickOptionComputation()
11	ASK_OPTION_COMPUTATION	tickOptionComputation()
12	LAST_OPTION_COMPUTATION	tickOptionComputation()
13	MODEL_OPTION	tickOptionComputation()
14	OPEN_TICK	tickPrice()
15	LOW_13_WEEK	tickPrice()
16	HIGH_13_WEEK	tickPrice()
17	LOW_26_WEEK	tickPrice()
18	HIGH_26_WEEK	tickPrice()
19	LOW_52_WEEK	tickPrice()
20	HIGH_52_WEEK	tickPrice()
21	AVG_VOLUME	tickSize()
22	OPEN_INTEREST	tickSize()
23	OPTION_HISTORICAL_VOL	tickGeneric()
24	OPTION_IMPLIED_VOL	tickGeneric()
25	OPTION_BID_EXCH	NOT USED

<b>Tick Value</b>	<b>Description</b>	<b>Event/Function/Method</b>
26	OPTION_ASK_EXCH	NOT USED
27	OPTION_CALL_OPEN_INTEREST	tickSize()
28	OPTION_PUT_OPEN_INTEREST	tickSize()
29	OPTION_CALL_VOLUME	tickSize()
30	OPTION_PUT_VOLUME	tickSize()
31	INDEX_FUTURE_PREMIUM	tickGeneric()
32	BID_EXCH	tickString()
33	ASK_EXCH	tickString()
34	AUCTION_VOLUME	NOT USED
35	AUCTION_PRICE	NOT USED
36	AUCTION_IMBALANCE	NOT USED
37	MARK_PRICE	tickPrice()
38	BID_EFP_COMPUTATION	tickEFP()
39	ASK_EFP_COMPUTATION	tickEFP()
40	LAST_EFP_COMPUTATION	tickEFP()
41	OPEN_EFP_COMPUTATION	tickEFP()
42	HIGH_EFP_COMPUTATION	tickEFP()
43	LOW_EFP_COMPUTATION	tickEFP()
44	CLOSE_EFP_COMPUTATION	tickEFP()
45	LAST_TIMESTAMP	tickString()
46	SHORTABLE	tickGeneric()
47	FUNDAMENTAL RATIOS	tickString()
48	RT_VOLUME	tickGeneric()
49	HALTED	See note below.

**Note:** When trading is halted for a contract, TWS receives a special tick: haltedLast=1. When trading is resumed, TWS receives haltedLast=0. A new tick type, HALTED, tick ID = 49, is now available in regular market data via the API to indicate this halted state. Possible values for this new tick type are:

- 0 = Not halted
- 1 = Halted.

## Generic Tick Types

For all socket-based API technologies, including the socket client library, ActiveX and Java, we provide several types of market data ticks that can be requested as a part of the market data request.

Starting with API version 9.0 (client version 30), version 6 of the REQ\_MKT\_DATA message is sent containing a new field that specifies the requested ticks as a list of comma-delimited integer Ids (generic tick types). Requests for these ticks will be answered if the tick type requested pertains to the contract at issue.

Note that Generic Tick Tags cannot be specified if you elect to use the Snapshot market data subscription.

The generic market data tick types are:

<b>Integer ID Value</b>	<b>Tick Type</b>	<b>Resulting Tick Value (see list below)</b>
100	Option Volume (currently for stocks)	29, 30
101	Option Open Interest (currently for stocks)	27, 28
104	Historical Volatility (currently for stocks)	23
106	Option Implied Volatility (currently for stocks)	24
162	Index Future Premium	31
165	Miscellaneous Stats	15, 16, 17, 18, 19, 20, 21
221	Mark Price (used in TWS P&L computations)	37
225	Auction values (volume, price and imbalance)	34, 35, 36
233	RTVolume	48
236	Shortable	46
256	Inventory	
258	Fundamental Ratios	47

## Using the SHORTABLE Tick

In TWS, there is a SHORTABLE column, as shown below. The column describes number of shares with which the security can be sold short.

Order Management			Shortable	Bid	Bid Size	Ask
Undrlyng	Exch	Description	Key	TIF	Action	Quantity
IBM	SMAR...	Stock (NYSE)		116.90	1	116.
AGG	SMAR...	Stock (ARCA)		98.98	2	99.
GOOG	SMAR...	Stock (NASDAQ)		439.99	1	440.
MS	SMAR...	Stock (NYSE)		27.47	12	27.
GS	SMAR...	Stock (NYSE)		129.71	5	129.
YHOO	SMAR...	Stock (NASDAQ)		18.98	47	18.
SPY	SMAR...	Stock (AMEX)		118.91	97	118.
CSCO	SMAR...	Stock (NASDAQ)		22.92	271	22.

The color GREEN indicates that at least 1000 shares are available to sell short. DARK GREEN indicates that this contract can be sold short but that at the moment there are no shares available for short sale, and that the system is searching for shares. RED indicates that no shares are available for short sale.

With API 9.30 or higher, the shorting indicator is supported for all socket connections. The functionality equates to the SHORTABLE column in the TWS workstation.

When invoking the reqMktDataEx()/reqMktData() methods, you must include generic ticktype 236 in the argument to obtain the shortable value. For example:

The Shortable tick determines if SHORT SELL orders for a contract will be accepted. Analyze the value returned from tickGeneric(int tickerId, int tickType, double value) as follows:

```
if (value > 2.5) { // 3.0
    // There are at least 1000 shares available for a short sale
    // In TWS, this is identical to GREEN status
}
else if (value > 1.5) { // 2.0
    // This contract will be available for short sale if shares can be
    // located
    // In TWS, this is identical to DARK GREEN status
}
else if (value > 0.5) { // 1.0
    // Not available for short sale
    // In TWS, this is identical to RED status
}
else {
    // unknown value
}
```

**Note:** This feature is supported as of server version 33 (872 release of TWS).

## TAG Values for FUNDAMENTAL\_RATIOS tickType

The FUNDAMENTAL\_RATIOS tickType (Tick Value 47) lets you request fundamental ratios in the form TAG=VALUE, TAG2=VALUE2, and so on. These ratios are sent using the tickGeneric() callback. The following table lists all the TAG values for FUNDAMENTAL\_RATIOS.

<b>TAG</b>	<b>Description</b>
NPRICE	<b>Closing Price</b> This is the closing price for the issue from the day it last traded. It is also referred to as the Current Price. Note that some issues may not trade every day, and therefore it is possible for this price to come from a date prior to the last business day.
Three_Year_TTM_Growth	3 year trailing twelve months growth.
TTM_over_TTM	Trailing twelve months over trailing twelve months.
NHIG	<b>High Price</b> This price is the highest price the stock traded at in the last 12 months. This could be an intra-day high.
NLOW	<b>Low Price</b> This price is the lowest price the stock traded at in the last 12 months. This could be an intra-day low.
PDATE	<b>Pricing date</b> The pricing date is the date at which the issue was last priced.
VOL10DAVG	<b>Volume</b> This is the daily average of the cumulative trading volume for the last ten days.
MKTCP	<b>Market capitalization</b> This value is calculated by multiplying the current Price by the current number of shares outstanding.
TTMEPSXCLX	<b>EPS excluding extraordinary items</b> This is the Adjusted Income Available to Common Stockholders for the trailing twelve months divided by the trailing twelve month Diluted Weighted Average Shares Outstanding.
AEPSNORM	<b>EPS Normalized</b> This is the Normalized Income Available to Common Stockholders for the most recent annual period divided by the same period's Diluted Weighted Average Shares Outstanding.
TTMREVPS	<b>Revenue/share</b> This value is the trailing twelve month Total Revenue divided by the Average Diluted Shares Outstanding for the trailing twelve months. <b>Note:</b> Most banks and insurance companies do not report revenues when they announce their preliminary quarterly financial results in the press. When this happens, the trailing twelve month values will not be available (NA).

<b>TAG</b>	<b>Description</b>
QBVPS	<p><b>Book value (Common Equity) per share</b>  This is defined as the Common Shareholder's Equity divided by the Shares Outstanding at the end of the most recent interim period. Book Value is the Total Shareholder's Equity minus Preferred Stock and Redeemable Preferred Stock.</p>
QTANBVPS	<p><b>Book value (tangible) per share</b>  This is the interim Tangible Book Value divided by the Shares Outstanding at the end of the most recent interim period. Tangible Book Value is the Book Value minus Goodwill and Intangible Assets for the same period.</p>
QCSHPS	<p><b>Cash per share</b>  This is the Total Cash plus Short Term Investments divided by the Shares Outstanding at the end of the most recent interim period.  <b>Note:</b> This does NOT include cash equivalents that may be reported under long term assets.</p>
TTMCFSHR	<p><b>Cash Flow per share</b>  This value is the trailing twelve month Cash Flow divided by the trailing twelve month Average Shares Outstanding. Cash Flow is defined as the sum of Income After Taxes minus Preferred Dividends and General Partner Distributions plus Depreciation, Depletion and Amortization.</p>
TTMDIVSHR	<p><b>Dividends per share</b>  This is the sum of the Cash Dividends per share paid to common stockholders during the last trailing twelve month period.</p>
IAD	<p><b>Dividend rate</b>  This value is the total of the expected dividend payments over the next twelve months. It is generally the most recent cash dividend paid or declared multiplied by the dividend payment frequency, plus any recurring extra dividends.</p>
PEEXCLXOR	<p><b>P/E excluding extraordinary items</b>  This ratio is calculated by dividing the current Price by the sum of the Diluted Earnings Per Share from continuing operations BEFORE Extraordinary Items and Accounting Changes over the last four interim periods.</p>
APENORM	<p><b>P/E Normalized</b>  This is the Current Price divided by the latest annual Normalized Earnings Per Share value.</p>

**Reference Tables**

TAG Values for FUNDAMENTAL\_RATIOS tickType

<b>TAG</b>	<b>Description</b>
TMPR2REV	<b>Price to sales</b> This is the current Price divided by the Sales Per Share for the trailing twelve months. If there is a preliminary earnings announcement for an interim period that has recently ended, the revenue (sales) values from this announcement will be used in calculating the trailing twelve month revenue per share. NOTE: Most Banks and Finance companies do not report revenues when they announce their preliminary interim financial results in the press. When this happens, the trailing twelve month values will not be available (NA) until the complete interim filing is released.
PR2TANBK	<b>Price to Tangible Book</b> This is the Current Price divided by the latest annual Tangible Book Value Per Share. Tangible Book Value Per Share is defined as Book Value minus Goodwill and Intangible Assets divided by the Shares Outstanding at the end of the fiscal period.
TTMPRCFPS	<b>Price to Cash Flow per share</b> This is the current Price divided by Cash Flow Per Share for the trailing twelve months. Cash Flow is defined as Income After Taxes minus Preferred Dividends and General Partner Distributions plus Depreciation, Depletion and Amortization.
PRICE2BK	<b>Price to Book</b> This is the Current Price divided by the latest interim period Book Value Per Share.
QCURRATIO	<b>Current ratio</b> This is the ratio of Total Current Assets for the most recent interim period divided by Total Current Liabilities for the same period. NOTE: This item is Not Available (NA) for Banks, Insurance companies and other companies that do not distinguish between current and long term assets and liabilities.
QQUICKRATI	<b>Quick ratio</b> Also known as the Acid Test Ratio, this ratio is defined as Cash plus Short Term Investments plus Accounts Receivable for the most recent interim period divided by the Total Current Liabilities for the same period. NOTE: This item is Not Available (NA) for Banks, Insurance companies and other companies that do not distinguish between current and long term assets and liabilities.
QLTD2EQ	<b>LT debt/equity</b> This ratio is the Total Long Term Debt for the most recent interim period divided by Total Shareholder Equity for the same period.
QTOTD2EQ	<b>Total debt/total equity</b> This ratio is Total Debt for the most recent interim period divided by Total Shareholder Equity for the same period. NOTE: This is Not Meaningful (NM) for banks.

TAG	Description
TTMPAYRAT	<p><b>Payout ratio</b></p> <p>This ratio is the percentage of the Primary/Basic Earnings Per Share Excluding Extraordinary Items paid to common stockholders in the form of cash dividends during the trailing twelve months.</p>
TTMREV	<p><b>Revenue</b></p> <p>This is the sum of all revenue (sales) reported for all operating divisions for the most recent TTM period. NOTE: Most banks and Insurance companies do not report revenues when they announce their preliminary quarterly financial results in the press. When this happens, the quarterly value will not be available (NA).</p>
TTMEBITD	<p><b>EBITD</b></p> <p>Earnings Before Interest, Taxes, Depreciation and Amortization (EBITDA) is EBIT for the trailing twelve months plus the same period's Depreciation and Amortization expenses (from the Statement of Cash Flows). NOTE: This item is only available for Industrial and Utility companies.</p>
TTMEBT	<p><b>Earnings before taxes</b></p> <p>Also known as Pretax Income and Earnings Before Taxes, this is Total Revenue for the most recent TTM period minus Total Expenses plus Non-operating Income (Expenses) for the same period.</p>
TTMNIAC	<p><b>Net Income available to common</b></p> <p>This is the trailing twelve month dollar amount accruing to common shareholders for dividends and retained earnings. Income Available to Common Shareholders is calculated as trailing twelve month Income After Taxes plus Minority Interest and Equity in Affiliates plus Preferred Dividends, General Partner Distributions and US GAAP Adjustments. NOTE: Any adjustment that is negative (ie. Preferred Stock Dividends) would be subtracted from Income After Taxes.</p>
AEBTNORM	<p><b>Earnings before taxes Normalized</b></p> <p>This is the Income Before Tax number excluding the impact of all unusual/one-time/special charges items for the most recent annual period.</p>
ANIACNORM	<p><b>Net Income Available to Common, Normalized</b></p> <p>This is the annual dollar amount accruing to common shareholders for dividends and retained earnings excluding the impact of all unusual/one-time/special charges items.</p>
TTMGROSMGN	<p><b>Gross Margin</b></p> <p>This value measures the percent of revenue left after paying all direct production expenses. It is calculated as the trailing 12 months Total Revenue minus the trailing 12 months Cost of Goods Sold divided by the trailing 12 months Total Revenue and multiplied by 100. NOTE: This item is only available for Industrial and Utility companies.</p>

**Reference Tables**

TAG Values for FUNDAMENTAL\_RATIOS tickType

<b>TAG</b>	<b>Description</b>
TTMNPMSGN	<b>Net Profit Margin %</b> Also known as Return on Sales, this value is the Income After Taxes for the trailing twelve months divided by Total Revenue for the same period and is expressed as a percentage. NOTE: Most Banks and Finance companies do not report revenues when they announce their preliminary quarterly financial results in the press. When this happens, the trailing twelve month value will not be available (NA).
TTMOPMGN	<b>Operating margin</b> This value measures the percent of revenues remaining after paying all operating expenses. It is calculated as the trailing 12 months Operating Income divided by the trailing 12 months Total Revenue, multiplied by 100. Operating Income is defined as Total Revenue minus Total Operating Expenses.
APTMGNPCT	<b>Pretax margin</b> This value represents Income Before Taxes for the most recent fiscal year expressed as a percent of Total Revenue for the most recent fiscal year.
TTMROAPCT	<b>Return on average assets</b> This value is the Income After Taxes for the trailing twelve months divided by the Average Total Assets, expressed as a percentage. Average Total Assets is calculated by adding the Total Assets for the 5 most recent quarters and dividing by 5.
TTMROEPCT	<b>Return on average equity</b> This value is the Income Available to Common Stockholders for the trailing twelve months divided by the Average Common Equity and is expressed as a percentage. Average Common Equity is calculated by adding the Common Equity for the 5 most recent quarters and dividing by 5.
TTMROIPICT	<b>Return on investment</b> This value is the trailing twelve month Income After Taxes divided by the average Total Long Term Debt, Other Long Term Liabilities and Shareholders Equity, expressed as a percentage.
REVCHNGYR	<b>Revenue Change %</b> This value is calculated as the most recent interim period Sales minus the Sales for the same interim period 1 year ago divided by the Sales for the same interim period one year ago, multiplied by 100.
TTMREVCHG	<b>Revenue Change %</b> This is the percent change in the trailing twelve month Sales as compared to the same trailing twelve month period one year ago. It is calculated as the trailing twelve month Sales minus the trailing twelve month Sales one year ago divided by the trailing twelve month Sales one year ago, multiplied by 100.
REVTRENDGR	<b>Revenue growth rate</b> The Five Year Revenue Growth Rate is the annual compounded growth rate of Revenues over the last 5 years.

TAG	Description
EPSCHNGYR	<b>EPS Change %</b> This value is calculated as the most recent interim period EPS minus the EPS for the same interim period 1 year ago divided by the EPS for the same interim period one year ago, multiplied by 100. NOTE: EPS must be positive for both periods. If either EPS value is negative, the result is Not Meaningful (NM).
TTMEPSCHG	<b>EPS Change %</b> This is the percent change in the trailing twelve month EPS as compared to the same trailing twelve month period one year ago. It is calculated as the trailing twelve month EPS minus the trailing twelve month EPS one year ago divided by the trailing twelve month EPS one year ago, multiplied by 100. NOTE: If either value has a negative value, the resulting value will be Not Meaningful (NM).
EPSTRENDGR	<b>EPS growth rate</b> This growth rate is the compound annual growth rate of Earnings Per Share Excluding Extraordinary Items and Discontinued Operations over the last 5 years. NOTE: If the value for either the most recent year or the oldest year is zero or negative, the growth rate cannot be calculated and a 'NA' (Not Available) code will be used.
DIVGRPCT	<b>Growth rate % - dividend</b> The Dividend Growth Rate is the compound annual growth rate in dividends per share. DIVGR% is calculated for 3 years whenever 4 years of dividends are available.

## Extended Order Attributes

The extended order attributes below can be used in all placeOrder functions and Open\_Order events.

Attribute	Possible Values
string m_tif	Day, GTC, IOC, GTD
string m_ocaGroup	Identifies a member of a one-cancels-all group.
string m_account	Institutional only.
string m_openClose	Institutional only.
int m_origin	Institutional only.
string m_orderRef	Customer defined order ID tag.
boolean m_transmit	Specifies whether the order will be transmitted by TWS. If set to false, order is created by not transmitted.
int m_parentId	The order ID of the parent, used for bracket, auto stop and trailing stop orders.
boolean m_blockOrder	If set to true, specifies that the order is a block order.
boolean m_sweepToFill	If set to true, specifies that the order is a Sweep-to-fill order.
int m_displaySize	The publicly disclosed order size to be used when placing iceberg orders.
int m_triggerMethod	Specifies how Simulated Stop, Stop-Limit and Trailing Stop orders are triggered. Valid values are: <ul style="list-style-type: none"> <li>• 0 - the default value. The "double bid/ask" method will be used for orders for OTC stocks and US options. All other orders will use the "last" method.</li> <li>• 1 - use "double bid/ask" method, where stop orders are triggered based on two consecutive bid or ask prices.</li> <li>• 2 - "last" method, where stop orders are triggered based on the last price.</li> <li>• 3 - "double-last" method, where stop orders are triggered based on last two prices.</li> </ul>
boolean m_ignoreRth	If set to true, allows triggering of orders outside of regular trading hours.
boolean m_hidden	If set to true, the order will not be visible when viewing the market depth. The only applies to orders routed to INet.

Attribute	Possible Values
string m_goodAfterTime	Indicates that the trade should be submitted after the time and date set, with format YYYYMMDD HH:MM:SS (seconds are optional). Use an empty string if not applicable.
string m_goodTillDate	Indicates that the trade should remain working until the time and date set, with format YYYYMMDD HH:MM:SS (seconds are optional). You must set the tif to GTD when using this string. Use an empty string if not applicable.
string m_faGroup	The advisor group to which the trade will be allocated. Use an empty string if not applicable.
string m_faProfile	The advisor allocation profile to which the trade will be allocated. Use an empty string if not applicable.
string m_faMethod	The advisor allocation method with which the trade will be allocated. Use an empty string if not applicable.
string m_faPercentage	The advisor percentage concerning the trade's allocation. Use an empty string if not applicable.
string m_primaryExch	To clarify any ambiguity for Smart-routed contracts, include the primary exchange, along with the Smart designation, for the destination.
int m_shortSaleSlot	For institutional customers only. <ul style="list-style-type: none"> <li>• 0 - unapplicable (i.e. retail customer or not short leg)</li> <li>• 1 - clearing broker</li> <li>• 2 - third party. If this value is used, you must enter a designated location.</li> </ul>
string m_designatedLocation	Only valid when shortSaleSlot value = 2. Otherwise leave blank or orders will be rejected.
long ocaType	Cancel on Fill with Block = 1 Reduce on Fill with Block = 2 Reduce on Fill without Block = 3
int rthOnly	Regular trading hours only. yes=1, no=0

**Reference Tables**  
*Extended Order Attributes*

Attribute	Possible Values
String rule80A	Individual = 'I' Agency = 'A', AgentOtherMember = 'W' IndividualPTIA = 'J' AgencyPTIA = 'U' AgentOtherMemberPTIA = 'M' IndividualPT = 'K' AgencyPT = 'Y' AgentOtherMemberPT = 'N'
String settlingFirm	Institutional only
String clearingAccount	The true beneficiary of the order. This value must be sent on FUT/FOP orders for reporting the exchange.
String clearingIntent	IB, Away, or PTA
int allOrNone	yes=1, no=0
long minQty	Identifies a minimum quantity order type.
double percentOffset	The percent offset for relative orders.
int eTradeOnly	Trade with electronic quotes. yes=1, no=0
int firmQuoteOnly	Trade with firm quotes. yes=1, no=0
double nbboPriceCap	Maximum SMART order distance from the NBBO.
long auctionStrategy	match = 1 improvement = 2 transparent = 3 For BOX exchange only.
double startingPrice	Starting price. For BOX exchange only.
double stockRefPrice	The stock reference price. For BOX exchange only.

Attribute	Possible Values
double delta	For BOX exchange only.
double stockRangeLower	The lower value of the acceptable stock range. For BOX exchange only.
double stockRangeUpper	The upper value of the acceptable stock range. For BOX exchange only.
double m_volatility	The option price in volatility, as calculated by TWS' Option Analytics. This value is expressed as a percent and is used to calculate the limit price sent to the exchange.
int m_volatilityType	1 = Daily; 2 = Annual
m_continuousUpdate	0 = false; 1 = True
int m_referencePriceType	1 = Average; 2 = BidOrAsk
String m_deltaNeutralOrderType	Enter an accepted order type such as: MKT, LMT, REL etc.
double m_deltaNeutralAuxPrice	Enter the Aux Price for Hedge Delta order types that require one.
int m_scaleNumComponents	For Scale orders: Defines the number of component orders into which the parent order will be split, thereby backing into the number of units within each component.
int m_scaleComponentSize	For Scale orders: Defines the number of units per component, backing into the number of components into which the parent order is split.
double m_scalePriceIncrement	For Scale orders: Defines the price increment per scale component.
double m_basisPoints	EFP orders
int basisPointsType	EFP orders

## Available Market Scanners

The following table shows a list (current as of July 2008) of the available scanners.

Market Scanner (Scan Code)	Description
Low Opt Volume P/C Ratio (LOW_OPT_VOL_PUT_CALL_RATIO)*	Put option volumes are divided by call option volumes and the top underlying symbols with the lowest ratios are displayed.
High Option Imp Vol Over Historical (HIGH_OPT_IMP_VOLAT_OVER_HIST)*	Shows the top underlying contracts (stocks or indices) with the largest divergence between implied and historical volatilities.
Low Option Imp Vol Over Historical (LOW_OPT_IMP_VOLAT_OVER_HIST)*	Shows the top underlying contracts (stocks or indices) with the smallest divergence between implied and historical volatilities.
Highest Option Imp Vol (HIGH_OPT_IMP_VOLAT)*	Shows the top underlying contracts (stocks or indices) with the highest vega-weighted implied volatility of near-the-money options with an expiration date in the next two months.
Top Option Imp Vol % Gainers (TOP_OPT_IMP_VOLAT_GAIN)*	Shows the top underlying contracts (stocks or indices) with the largest percent gain between current implied volatility and yesterday's closing value of the 15 minute average of implied volatility.
Top Option Imp Vol % Losers (TOP_OPT_IMP_VOLAT_LOSE)*	Shows the top underlying contracts (stocks or indices) with the largest percent loss between current implied volatility and yesterday's closing value of the 15 minute average of implied volatility.
High Opt Volume P/C Ratio (HIGH_OPT_VOLUME_PUT_CALL_RATIO)	Put option volumes are divided by call option volumes and the top underlying symbols with the highest ratios are displayed.
Low Opt Volume P/C Ratio (LOW_OPT_VOLUME_PUT_CALL_RATIO)	Put option volumes are divided by call option volumes and the top underlying symbols with the lowest ratios are displayed.
Most Active by Opt Volume (OPT_VOLUME_MOST_ACTIVE)	Displays the most active contracts sorted descending by options volume.
Hot by Option Volume (HOT_BY_OPT_VOLUME)	Shows the top underlying contracts for highest options volume over a 10-day average.
High Option Open Interest P/C Ratio (HIGH_OPT_OPEN_INTEREST_PUT_CALL_RATIO)	Returns the top 50 contracts with the <b>highest</b> put/call ratio of outstanding option contracts.

Market Scanner (Scan Code)	Description
Low Option Open Interest P/C Ratio (LOW_OPT_OPEN_INTEREST_PUT_CALL_RATIO)	Returns the top 50 contracts with the <b>lowest</b> put/call ratio of outstanding option contracts.
Top % Gainers (TOP_PERC_GAIN)	Contracts whose last trade price shows the highest percent increase from the previous night's closing price.
Most Active (MOST_ACTIVE)	<p>Contracts with the highest trading volume today, based on units used by TWS (lots for US stocks; contract for derivatives and non-US stocks).</p> <p>The sample spreadsheet includes two Most Active scans: Most Active List, which displays the most active contracts in the NASDAQ, NYSE and AMEX markets, and Most Active US, which displays the most active stocks in the United States.</p>
Top % Losers (TOP_PERC_LOSE)	Contracts whose last trade price shows the lowest percent increase from the previous night's closing price.
Hot Contracts by Volume (HOT_BY_VOLUME)	<p>Contracts where:</p> <p>today's Volume/avgDailyVolume is highest.</p> <p>avgDailyVolume is a 30-day exponential moving average of the contract's daily volume.</p>
Top % Futures Gainers (TOP_PERC_GAIN)	Futures whose last trade price shows the highest percent increase from the previous night's closing price.
Hot Contracts by Price (HOT_BY_PRICE)	<p>Contracts where:</p> <p>(lastTradePrice-prevClose)/avgDailyChange is highest in absolute value (positive or negative).</p> <p>The avgDailyChange is defined as an exponential moving average of the contract's (dailyClose-dailyOpen)</p>
Top Trade Count (TOP_TRADE_COUNT)	The top trade count during the day.
Top Trade Rate (TOP_TRADE_RATE)	Contracts with the highest number of trades in the past 60 seconds (regardless of the sizes of those trades).
Top Price Range (TOP_PRICE_RANGE)	The largest difference between today's high and low, or yesterday's close if outside of today's range.

**Reference Tables**  
Available Market Scanners

Market Scanner (Scan Code)	Description
Hot by Price Range (HOT_BY_PRICE_RANGE)	The largest price range (from Top Price Range calculation) over the volatility.
Top Volume Rate (TOP_VOLUME_RATE)	The top volume rate per minute.
Lowest Option Imp Vol (LOW_OPT_IMP_VOLAT)	Shows the top underlying contracts (stocks or indices) with the lowest vega-weighted implied volatility of near-the-money options with an expiration date in the next two months.
Most Active by Opt Open Interest (OPT_OPEN_INTEREST_MOST_ACTIVE)	Returns the top 50 underlying contracts with the (highest number of outstanding call contracts) + (highest number of outstanding put contracts)
Not Open (NOT_OPEN)	Contracts that have not traded today.
Halted (HALTED)	Contracts for which trading has been halted.
Top % Gainers Since Open (TOP_OPEN_PERC_GAIN)	Shows contracts with the highest percent price INCREASE between the last trade and opening prices.
Top % Losers Since Open (TOP_OPEN_PERC_LOSE)	Shows contracts with the highest percent price DECREASE between the last trade and opening prices.
Top Close-to-Open % Gainers (HIGH_OPEN_GAP)	Shows contracts with the highest percent price INCREASE between the previous close and today's opening prices.
Top Close-to-Open % Losers (LOW_OPEN_GAP)	Shows contracts with the highest percent price DECREASE between the previous close and today's opening prices.
Lowest Option Imp Vol (LOW_OPT_IMP_VOLAT)*	Shows the top underlying contracts (stocks or indices) with the lowest vega-weighted implied volatility of near-the-money options with an expiration date in the next two months.
Top Option Imp Vol % Gainers (TOP_OPT_IMP_VOLAT_GAIN)*	Shows the top underlying contracts (stocks or indices) with the largest percent gain between current implied volatility and yesterday's closing value of the 15 minute average of implied volatility.
Top Option Imp Vol % Losers (TOP_OPT_IMP_VOLAT_LOSE)*	Shows the top underlying contracts (stocks or indices) with the largest percent loss between current implied volatility and yesterday's closing value of the 15 minute average of implied volatility.

Market Scanner (Scan Code)	Description
13-Week High (HIGH_VS_13W_HL)	The highest price for the past 13 weeks.
13-Week Low (LOW_VS_13W_HL)	The lowest price for the past 13 weeks.
26-Week High (HIGH_VS_26W_HL)	The highest price for the past 26 weeks.
26-Week Low (LOW_VS_26W_HL)	The lowest price for the past 26 weeks.
52-Week High (HIGH_VS_52W_HL)	The highest price for the past 52 weeks.
52-Week Low (LOW_VS_52W_HL)	The lowest price for the past 52 weeks.
<b>EFP - High Synth Bid Rev Yield</b> (HIGH_SYNTH_BID_REV_NAT_YIELD)	Highlights the highest synthetic EFP interest rates available. These rates are computed by taking the price differential between the SSF and the underlying stock and netting dividends to calculate an annualized synthetic implied interest rate over the period of the SSF. The High rates may present an investment opportunity.
<b>EFP - Low Synth Bid Rev Yield</b> (LOW_SYNTH_BID_REV_NAT_YIELD)	Highlights the lowest synthetic EFP interest rates available. These rates are computed by taking the price differential between the SSF and the underlying stock and netting dividends to calculate an annualized synthetic implied interest rate over the period of the SSF. The Low rates may present a borrowing opportunity.

## IBAlgo Parameters

Beginning with TWS API Release 9.6, the ActiveX, C++ and Java APIs support IBAlgo orders. The following table lists all of the IBAlgo strategies and parameters supported by the API.

Algo Strategy	Parameters
<b>For US Stocks:</b>	
Arrival Price (ArrivalPx)	Max percentage: maxPctVol (double in a range 0.01 to 0.5) Urgency/Risk aversion: riskAversion (Valid values: Get Done; Aggressive; Neutral; Passive) Attempt completion by EOD: forceCompletion (boolean)
Percentage of Volume (PctVol)	Target Percentage: pctVol (double in a range 0.01 to 0.5)
Volume-Weighted Average Price (Vwap)	Max Percentage: maxPctVol (double in a range 0.01 to 0.5)
Time-Weighted Average Price (Twap)	Trade when: strategyType (Valid values: Marketable; Matching Midpoint; Matching Same Side; Matching Last)
<b>For US Options:</b>	
Balance Impact and Risk (BalanceImpactRisk)	Max percentage: maxPctVol (double in a range 0.01 to 0.5) Urgency/Risk aversion: riskAversion (Valid values: Get Done; Aggressive; Neutral; Passive) Attempt completion by EOD: forceCompletion (boolean)
Minimize Impact (MinImpact)	Max Percentage: maxPctVol (double in a range 0.01 to 0.5)

# Index

## A

about the APIs 1-14  
account details in ActiveX for Excel 7-350  
account details in Excel DDE 2-67  
account information, removing in Excel 2-68  
account information, viewing in ActiveX for Excel 7-350  
account information, viewing in Excel 2-68  
Account page 2-67  
    field values 2-69  
    in ActiveX for Excel 7-350  
    toolbar buttons 2-73, 7-352  
Account page, using 2-68  
Account page, using in ActiveX for Excel 7-350  
Active X 3-111  
ActiveX  
    linking to TWS 3-112  
    placing a combination order 3-199  
    registering third-party controls 3-113  
ActiveX API  
    on 64-bit systems 3-113  
ActiveX COM objects 3-178, 3-179, 3-181, 3-183, 3-184, 3-185, 3-191  
ActiveX events 3-148, 3-149, 3-150, 3-151, 3-152, 3-153, 3-154, 3-155, 3-156, 3-158, 3-163, 3-164, 3-165, 3-166, 3-167, 3-168, 3-169, 3-170, 3-171, 3-172, 3-174, 3-175, 3-176, 3-177  
ActiveX factory methods 3-145, 3-146, 3-147  
ActiveX for Excel  
    Account page 7-350  
    Advanced Orders page 7-339  
    Advisors page 7-379  
    allocating shares to a single account 7-380  
    Basic Orders page 7-334  
    Bond Contract Details page 7-365  
    bracket orders in 7-341  
    Bulletins page 7-331  
    connecting to TWS 7-326  
    Contract Details page 7-363  
    disconnecting from TWS 7-326  
    download API components 7-322  
    Executions page 7-355  
    Extended Order Attributes page 7-346  
    FA orders using account group and method 7-381  
    FA orders using allocation profile 7-382  
    Fundamentals page 7-377  
    General page 7-325  
    getting started 7-322  
    Historical Data page 7-357  
    Log page 7-384  
    Market Depth page 7-332  
    Market Scanner page 7-369  
    Open Orders page 7-348  
    opening sample spreadsheet 7-323  
    placing orders in 7-335

Real Time Bars page 7-367  
relative orders in 7-345  
requesting current time 7-327  
scale orders in 7-344  
setting server log level in 7-326  
Tickers page 7-328  
trailing stop limit orders in 7-343  
ActiveX for Excel historical data  
    expired contracts 7-358  
ActiveX for Excel sample spreadsheet  
    using 7-324  
ActiveX methods 3-115, 3-116, 3-131, 3-137, 3-139, 3-140, 3-141, 3-143, 3-144, 3-145  
ActiveX properties 3-194  
ActiveX sample program 3-114  
addComboLeg() 3-130  
Advanced Orders 2-54  
    in ActiveX for Excel 7-339  
Advanced Orders page  
    toolbar buttons 2-60, 7-345  
advisors 6-315  
    financial reporting for 6-317  
Advisors page 2-97  
    in ActiveX for Excel 7-379  
    toolbar buttons 2-101, 7-383  
advisors, Excel DDE support for 6-316  
algo parameters 9-416  
allocation methods for account groups 6-318  
API  
    recommendations 1-17  
API components, downloading 2-28, 7-322  
API components, using 1-14  
API components. configurign in TWS 2-29  
API overview 1-13, 9-387  
API software  
    uninstalling 1-26  
API, about 1-14  
API, for financial advisor accounts 6-315  
Apply Extended Template button 2-45, 7-347  
available market scanners 9-412  
available market scanners in ActiveX for Excel 7-372  
available market scanners in Excel 2-85  
AvailableEquity Method 6-318

## B

bar size settings for historical data 1-21  
Basic Orders page 2-37  
    combination orders 7-336  
    in ActiveX for Excel 7-334  
    modifying orders 7-336  
    placing orders in ActiveX for Excel 7-335  
    toolbar buttons 2-43, 7-338  
basket orders, in ActiveX for Excel 7-335

basket orders, in Excel 2-38  
Bond Contract Details page 2-92  
    in ActiveX for Excel 7-365  
    toolbar buttons 2-93, 7-366  
bond contract details, requesting in ActiveX for Excel 7-365  
bond contract details, requesting in Excel 2-92  
bondContractDetails() 3-174, 4-237, 5-291  
bracket orders  
    in ActiveX for Excel 7-341  
    in DDE for Excel 2-55  
bracket orders, in Excel 2-55  
Bulletins page  
    in ActiveX for Excel 7-331  
    toolbar buttons 7-331

**C**

C++ 4-203  
    Class EClientSocket methods 4-211  
    Class EWrapper functions 4-224  
    combinations orders 4-254  
    linking to TWS 4-204  
    using the sample program 4-209  
C++ SocketClient properties 4-240  
calendar spread order in Excel 2-39, 7-336  
calendar spread order, in C++ 4-254  
cancelHistoricalData() 3-143, 4-221, 5-277  
cancelMktData() 3-119, 4-213, 5-270  
cancelMktDepth() 3-129, 4-215, 5-272  
cancelNewsBulletins() 3-131, 4-216, 5-272  
cancelOrder() 3-123, 4-214, 5-270  
cancelRealTimeBars() 3-145, 4-222, 5-278  
cancelScannerSubscription() 3-143, 4-221, 5-275  
checkMessages() 4-214  
Class EClientSocket methods 4-211, 4-212, 4-213, 4-214, 4-215, 4-216, 4-217, 4-218, 4-220, 4-221, 4-222  
Class EWrapper functions 4-224, 4-225, 4-226, 4-227, 4-229, 4-230, 4-231, 4-232, 4-233, 4-234, 4-235, 4-236, 4-237, 4-238, 4-239  
Clear All Links button 2-35  
clearComboLegs() 3-131  
combination order, in ActiveX 3-199  
combination order, in ActiveX for Excel 7-336  
combination order, in Excel 2-39  
combination orders, in C++ 4-254  
combination orders, in Java 5-312  
ComboLeg 4-246, 5-303  
conditional orders in Excel, examples of 2-51  
Conditional Orders page 2-50  
    toolbar buttons 2-53  
Conditional Orders page, modifying orders 2-52  
conditional orders, in Excel 2-50  
configure TWS 1-15  
configuring TWS 2-29  
connect() 3-116  
connecting to TWS  
    using ActiveX for Excel 7-326  
connectionClosed() 3-153, 4-231, 5-288  
Contract 4-243, 5-300  
Contract Details page 2-90

    in ActiveX for Excel 7-363  
    toolbar buttons 2-91, 7-364  
contract details, requesting in ActiveX for Excel 7-363  
contract details, requesting in Excel 2-90  
ContractDetails 4-245, 5-302  
contractDetails() 3-167, 3-168, 4-234, 5-291  
contractDetailsEx() 3-167  
createComboLegList() 3-145  
createContract() 3-146  
createExecutionFilter() 3-146  
createOrder() 3-146  
createScannerSubscription() 3-146  
createTagValueList 3-146  
createUnderComp() 3-147  
creating a ticker in ActiveX for Excel 7-328  
creating a ticker in Excel 2-33  
current time  
    requesting in ActiveX for Excel 7-327  
currentTime() 3-177, 4-239, 5-296

**D**

DDE for Excel  
    allocating shares to a single account 2-98  
    FA orders using account group and method 2-99  
    FA orders using allocation profile 2-100  
DDE for Excel historical data  
    expired contracts 2-77  
DDE syntax 2-102  
DDE syntax reference 2-105  
DDE, defined 2-27  
delta-neutral RFQs 1-24  
disconnect() 3-116  
disconnecting from TWS  
    using ActiveX for Excel 7-326  
displaying lines of market depth in Excel 2-95  
downloading API components 2-28, 7-322

**E**

EClient Socket methods 5-268, 5-269, 5-270, 5-271, 5-272, 5-273, 5-274, 5-275, 5-276, 5-277, 5-278, 5-279  
EClientSocket functions 4-211  
EClientSocket() 4-211, 5-269  
eConnect() 4-212, 5-269  
eDisconnect() 4-212, 5-269  
EqualQuantity Method 6-318  
errMsg() 3-153  
error codes 9-388  
error messages  
    viewing in ActiveX for Excel 7-384  
error() 4-230, 5-287  
EWrapper methods 5-281  
Excel 2007 Name Manager 2-104  
Excel DDE 2-27, 7-321, 8-385  
    download API components 2-28  
    getting started 2-28  
    macros in 2-104  
    named ranges in 2-103  
    opening sample spreadsheet 2-31  
    pages 2-32, 7-324

- placing orders in 2-38
  - supported order types 2-41, 7-338
  - syntax 2-105
  - viewing sample spreadsheet code 2-102
  - Visual Basic Editor 2-102
  - Excel DDE API reference 2-102
  - Excel DDE sample spreadsheet, installing 2-28, 7-322
  - Excel DDE, extended order attributes 2-44, 7-346
  - Excel DDE, Extended Order attributes page 2-44, 7-346
  - Excel DDE, financial advisor support 6-316
  - Excel DDE, supported order types 2-41, 7-338
  - `execDetails()` 3-169, 4-234, 5-291
  - `execDetailsEnd()` 3-170, 4-235, 5-292
  - `execDetailsEx()` 3-168
  - Execution 4-241, 5-298
    - execution reporting, for financial advisors 6-317
    - execution reporting, in ActiveX for Excel 7-355
    - execution reporting, in Excel DDE 2-63
    - execution reports
      - types of 2-66
    - execution reports, running in Excel 2-66
  - ExecutionFilter 4-241, 5-298
  - Executions page 2-63, 2-65
    - in ActiveX for Excel 7-355
    - toolbar buttons 2-64, 7-356
  - executions reporting, in Excel DDE 2-65
  - executions, viewing in ActiveX for Excel 7-356
  - executions, viewing in Excel 2-63
  - `exerciseOptions()` 3-140, 4-220, 5-279
  - `exerciseOptionsEx()` 3-139
  - exercising options
    - in ActiveX for Excel 7-353
  - extended order attributes
    - applying to individual or groups of orders 2-45, 7-347
    - extended order attributes in Excel 2-46
  - Extended Order Attributes page 2-44
    - in ActiveX for Excel 7-346
  - extended order attributes, manually programming in ActiveX for Excel 7-347
  - extended order attributes, manually programming in Excel 2-45
- F**
- FA information
    - in ActiveX for Excel 7-351
  - FA managed accounts, in ActiveX for Excel 7-351
    - Portfolio page 7-353
  - FA managed accounts, in Excel 2-68
  - FA orders
    - allocating shares to a single account 2-98, 7-380
    - allocation profiles 2-100, 7-382
    - using account group and method 2-99, 7-381
  - FA orders in ActiveX for Excel 7-379
  - FA orders in DDE for Excel 2-97
  - filtering execution reports in Excel DDE 2-65
  - Financial Advisors
    - allocation methods for account groups 6-318
  - financial advisors 6-315
    - execution reporting for 6-317
    - orders and account configuration 6-316
    - financial advisors, Excel DDE support for 6-316
    - financial advisors, support by other API technologies 6-316
    - fundamental data
      - in ActiveX for Excel 7-377
      - report types 7-377
    - fundamental ratios
      - in ActiveX for Excel 7-377
    - FUNDAMENTAL\_RATIOS tickType 9-402
    - Fundamentals page
      - in ActiveX for Excel 7-377
      - toolbar buttons 7-378
- G**
- General page
    - toolbar buttons 7-327
  - generic ticktypes
    - SHORTABLE 9-401
  - getting started
    - with ActiveX for Excel
      - sample spreadsheet 7-322
    - with Excel DDE 2-28
- H**
- historical data
    - duration and bar size settings 1-21
    - viewing in ActiveX for Excel 7-358
    - viewing in Excel 2-77
  - historical data limitations 1-20
  - Historical Data page 2-76
    - in ActiveX for Excel 7-357
    - query specification fields 7-360
    - toolbar buttons 2-81, 7-362
  - historicalData 4-237
  - historicalData() 3-172, 4-237, 5-294
- I**
- IBAlgo parameters 9-416
  - IComboLeg 3-184
  - IContract 3-181
  - IContractDetails 3-183
  - IExecution 3-179
  - IExecutionFilter 3-179
  - if-filled order, in Excel 2-51
  - Index Premium data 1-25
  - installing Excel DDE sample spreadsheet 2-28, 7-322
  - IOrder 3-185
  - IOrderState 3-191
  - isConnected() 4-212, 5-269
- J**
- Java 5-257
    - combination orders 5-312
    - linking to TWS 5-258
    - sample program 5-261
  - Java EClient Socket methods 5-268
  - Java EWrapper methods 3-168, 4-234, 5-281, 5-282, 5-283,

5-284, 5-286, 5-287, 5-288, 5-289, 5-290, 5-291, 5-292, 5-293, 5-294, 5-295, 5-296  
**java sample program** 5-265  
    classes 5-265  
**Java SocketClient properties** 5-297, 5-298, 5-300, 5-302, 5-303, 5-304, 5-309, 5-310  
**Java Test Client** 5-265  
    overview 5-265  
    running 5-261

**L**

**limitations**  
    of historical data requests 1-20  
**linking to TWS, using ActiveX** 3-112  
**log level, setting in Excel** 2-35  
**Log page**  
    in ActiveX for Excel 7-384

**M**

**macros in Excel** 2-104  
**managedAccounts()** 3-172, 4-231, 5-288  
**market depth**  
    requesting in ActiveX for Excel 7-333  
    requesting in Excel 2-95  
**Market Depth page** 2-94  
    in ActiveX for Excel 7-332  
    toolbar buttons 2-96  
    toolbar buttons in ActiveX for Excel 7-333  
    using 2-95  
    using ActiveX for Excel 7-333  
**market depth, displaying more lines in Excel** 2-95  
**Market Scanner page** 2-82  
    in ActiveX for Excel 7-369  
    supported market scanners 9-412  
    toolbar buttons 2-89, 7-376  
**market scanner parameters**  
    in ActiveX for Excel 7-370  
**market scanner subscription, starting in ActiveX for Excel** 7-370  
**market scanner subscription, starting in Excel** 2-83  
**market scanners, available in ActiveX for Excel** 7-372  
**market scanners, available in Excel** 2-85  
**Message codes** 9-397  
**message codes** 9-388  
**modifying orders in ActiveX for Excel** 7-336  
**modifying orders in Excel** 2-39  
**modifying orders, on Conditional Orders page** 2-52  
**modules in Excel VB code** 2-103

**N**

**Name Manager, in Excel 2007** 2-104  
**named ranges** 2-103  
**NetLiq Method** 6-318  
**nextValidId()** 3-166, 4-234, 5-290

**O**

**open orders**  
    removing in Excel 2-62

**Open Orders page** 2-61  
    **ActiveX for Excel** 7-348  
        toolbar buttons 2-62, 7-349  
    **open orders, viewing in ActiveX for Excel** 7-349  
    **open orders, viewing in Excel** 2-62  
    **opening the ActiveX sample program** 3-114  
    **openOrder()** 4-231, 5-288  
    **openOrder1()** 3-154  
    **openOrder2()** 3-155  
    **openOrder3()** 3-156  
    **openOrder4()** 3-158  
    **openOrderEx()** 3-154  
**options**  
    **exercising in ActiveX for Excel** 7-353  
**Order** 4-247, 5-304  
**order attributes in Excel** 2-42  
**order IDs** 1-23  
**order types in Excel DDE** 2-41, 7-338  
**order types, in Excel** 2-41, 7-338  
**orders** 1-22  
    **placing in ActiveX for Excel** 7-335  
        **placing in Excel** 2-38  
**orders and account configuration, for financial advisors** 6-316  
**Orders page**  
    **combination orders** 2-39  
    **modifying orders** 2-39  
    **placing basket orders** 2-55  
        **placing orders** 2-38  
**orders, modifying in ActiveX for Excel** 7-336  
**orders, modifying in Excel** 2-39  
**OrderState** 4-251, 5-309  
**orderStatus()** 3-152, 4-229, 5-286  
    **overview** 1-13, 9-387

**P**

**pages** 2-32, 7-324  
**PctChange Method** 6-318  
**permId()** 3-166  
**placeOrder()** 3-120, 4-213, 5-270  
**placeOrder2()** 3-122  
**placeOrderEx()** 3-119  
**placing orders**  
    **basket** 2-38, 2-55, 7-335  
        **combination order in ActiveX for Excel** 7-336  
        **combination order in Excel** 2-39  
        **conditional orders in Excel** 2-50  
    **placing orders in ActiveX for Excel** 7-335  
    **placing orders in Excel** 2-38  
**portfolio data in Excel DDE** 2-74  
**portfolio data in FA managed accounts, in ActiveX for Excel** 7-353  
**Portfolio page** 2-74  
    **in FA managed accounts, in ActiveX for Excel** 7-353  
        **toolbar buttons** 2-75, 7-354  
**portfolio, viewing in Excel** 2-74  
**portfolio, viewing in FA managed accounts, in ActiveX for Excel** 7-353  
**premium data** 1-25

price-change order, in Excel 2-52  
processing rate, on Tickers page 2-35

**Q**

query specification fields, on Historical Data page in ActiveX for Excel 7-360

**R**

real time bars  
  in ActiveX for Excel 7-367  
Real Time Bars page  
  toolbar buttons 7-368  
realtimeBar() 3-176, 4-239, 5-296  
receiveFA() 3-172, 4-237, 5-294  
recommendations for using API 1-17  
reference tables 9-387  
refresh rate, for market depth in Excel 2-95, 7-333  
refresh rate, on ActiveX for Excel Tickers page 7-330  
refresh rate, on Tickers page 2-35  
registering third-party ActiveX controls 3-113  
relative orders, in ActiveX for Excel 7-345  
relative orders, in DDE for Excel spreadsheet 2-59  
removing open orders, in Excel 2-62  
replaceFA() 4-218, 5-274  
reqAccountUpdates() 3-132, 4-214, 5-271  
reqAllOpenOrders 5-273  
reqAllOpenOrders() 3-124, 4-216  
reqAutoOpenOrders() 3-124, 4-217, 5-273  
reqContractDetails() 3-126, 4-215, 5-271  
reqContractDetails2() 3-127  
reqContractDetailsEx() 3-125  
reqCurrentTime() 3-145, 4-222, 5-279  
reqExecutions() 3-124, 4-214, 5-271  
reqExecutionsEx() 3-124  
reqHistoricalData() 3-137, 4-218, 5-276  
reqHistoricalDataEx() 3-134  
reqIDs() 4-215  
reqIds() 3-125  
reqManagedAccts() 3-132, 4-217, 5-273  
reqMktData() 3-117, 4-213, 5-270  
reqMktData2() 3-118  
reqMktDataEx() 3-116  
reqMktDepth() 3-128, 4-215, 5-272  
reqMktDepth2() 3-129  
reqMktDepthEx() 3-127  
reqNewsBulletins() 3-131, 4-216, 5-272  
reqOpenOrders() 3-124, 4-214, 5-271  
reqRealTimeBars() 3-144, 4-221, 5-278  
reqRealTimeBarsEx() 3-143  
reqScannerParameters() 3-140, 4-220, 5-274  
reqScannerSubscription() 3-141, 4-220, 5-275  
reqScannerSubscriptionEx() 3-141  
Request for Quote 1-24  
requestFA() 3-132, 4-217, 5-273  
requesting bond contract details in ActiveX for Excel 7-365  
requesting bond contract details in Excel 2-92  
requesting contract details in ActiveX for Excel 7-363  
requesting contract details in Excel 2-90

requesting market data, in ActiveX for Excel 7-329  
requesting market data, in Excel 2-34  
requesting market depth

  in ActiveX for Excel 7-333  
  in Excel 2-95

Reuters global fundamentals  
  in ActiveX for Excel 7-377

RFQs 1-24

running execution reports in Excel 2-66

**S**

sample program

  ActiveX 3-114

  C++ 4-209

  Java 5-261

sample spreadsheet

  Account page 2-67

  Advanced Orders page 2-54

  Advisors page 2-97

  Basic Orders page 2-37

  Bond Contract Details page 2-92

  bracket orders in 2-55

  Conditional Orders page 2-50

  Contract Details page 2-90

  Executions page 2-63

  Executions Reporting page 2-65

  Extended Order Attributes page 2-44

  Historical Data page 2-76

  macros in 2-104

  Market Depth page 2-94

  Market Scanner page 2-82

  Open Orders page 2-61

  opening 2-31, 7-323

  pages in 2-32, 7-324

  Portfolio page 2-74

  relative orders in 2-59

  scale orders in 2-58

  Tickers page 2-33

  trailing stop limit orders in 2-57

  using 2-32

  viewing VB code 2-102

sample spreadsheet, installing 2-28, 7-322

scale orders

  in ActiveX for Excel 7-344

scale orders, in DDE for Excel spreadsheet 2-58

scannerData() 3-175, 4-238, 5-295

scannerDataEnd() 4-238, 5-295

scannerDataEx() 3-175

scannerParameters() 3-175, 4-238, 5-295

ScannerSubscription 4-252, 5-310

server log level

  setting in ActiveX for Excel 7-326

serverVersion() 4-222, 5-279

setLogLevel() 4-216

setServerLogLevel() 3-131, 5-272

setting log level, on Tickers page 2-35

setting the processing rate on Tickers page 2-35

SHORTABLE tick 9-401

SocketClient Properties 4-240

SocketClient properties 4-241, 4-243, 4-245, 4-246, 4-247, 4-251, 4-252  
SocketClient properties, in Java API 5-297  
starting market scanner in ActiveX for Excel 7-370  
starting market scanner in Excel 2-83

### T

TAG values for FUNDAMENTAL\_RATIOS tickType 9-402  
TestJavaClient 5-261  
third-party controls, for ActiveX 3-113  
tickEFP() 3-151, 4-227, 5-284  
ticker, creating in ActiveX for Excel 7-328  
ticker, creating in Excel 2-33  
Tickers page 2-33  
    clearing all links 2-35  
    in ActiveX for Excel 7-328  
    requesting market data 2-34  
    requesting market data in ActiveX for Excel 7-329  
    setting log level 2-35  
    setting the processing rate 2-35  
    setting the refresh rate 2-35  
    setting the refresh rate in ActiveX for Excel 7-330  
    toolbar buttons 2-36  
    toolbar buttons in ActiveX for Excel 7-330  
Tickers page, using 2-33  
Tickers page, using ActiveX for Excel 7-328  
tickGeneric() 3-150, 4-226, 5-284  
tickOptionComputation() 3-150, 4-226, 5-283  
tickPrice() 3-149, 5-282  
tickPrice()Class EWrapper Functions 4-225  
tickSize() 3-149, 4-225, 5-283  
tickString() 4-227, 5-284  
toolbar buttons  
    on Account page 2-73  
    on ActiveX for Excel Advanced Orders page 7-345  
    on ActiveX for Excel Advisors page 7-383  
    on ActiveX for Excel Basic Orders page 7-338  
    on ActiveX for Excel Bond Contract Details  
        page 7-366  
    on ActiveX for Excel Contract Details page 7-364  
    on ActiveX for Excel Executions page 7-356  
    on ActiveX for Excel Fundamentals page 7-378  
    on ActiveX for Excel Historical Data page 7-362  
    on ActiveX for Excel Market Depth page 7-333  
    on ActiveX for Excel Market Scanner page 7-376  
    on ActiveX for Excel Open Orders page 7-349  
    on ActiveX for Excel Portfolio page 7-354  
    on ActiveX for Excel Real Time Bars page 7-368  
    on Advanced Orders page 2-60  
    on Advisors page 2-101  
    on Basic Orders page 2-43  
    on Bond Contract Details page 2-93  
    on Conditional Orders page 2-53  
    on Contract Details page 2-91  
    on Executions page 2-64  
    on FA managed accounts, in ActiveX for Excel  
        Account page 7-352  
    on Historical Data page 2-81  
    on Market Depth page 2-96

    on Market Scanner page 2-89  
    on Open Orders page 2-62  
    on Portfolio page 2-75  
toolbar buttons, on ActiveX for Excel Bulletins  
page 7-331  
toolbar buttons, on ActiveX for Excel General  
page 7-327  
toolbar buttons, on ActiveX for Excel Ticklers page 7-330  
toolbar buttons, on Ticklers page 2-36  
trailing stop limit orders, in ActiveX for Excel 7-343  
trailing stop limit orders, in DDE for Excel  
spreadsheet 2-57  
TWS  
    linking from Java 5-258  
TWS precautionary settings 1-22  
TWS, configuring for API 1-15, 2-29  
TWS, linking from C++ 4-204  
TWS, linking using ActiveX 3-112  
TwsConnectionTime() 4-222, 5-279  
TwsSocketClient.dll 4-204  
    linking to 4-204

### U

updateMktDepthL2() 5-293  
uninstalling the API software 1-26  
updateAccountTime() 3-166, 4-233, 5-290  
updateAccountValue() 3-163, 4-232, 5-289  
updateMktDepth() 3-170, 4-235, 5-292  
updateMktDepthL2 5-293  
updateMktDepthL2() 3-171, 4-235  
updateNewsBulletin() 3-171, 4-236, 5-293  
updatePortfolio() 3-165, 4-233, 5-290  
updatePortfolioEx() 3-164  
using Account page in ActiveX for Excel 7-350  
using Account page in Excel 2-68  
using API components 1-14  
using the ActiveX for Excel sample spreadsheet 7-324  
using the ActiveX for Excel Ticklers page 7-328  
using the Market Depth page in ActiveX for Excel 7-333  
using the Market Depth page in Excel 2-95  
using the sample spreadsheet 2-32  
using the Ticklers page 2-33

### V

VB code, in sample spreadsheet 2-102  
VB modules 2-103  
VB\_API\_sample.vbp  
    opening 3-114  
VBE 2-102  
viewing executions, in ActiveX for Excel 7-356  
viewing executions, in Excel 2-63  
viewing historical data in ActiveX for Excel 7-358  
viewing historical data in Excel 2-77  
viewing open orders in ActiveX for Excel 7-349  
viewing open orders in Excel 2-62  
viewing portfolio data in Excel 2-74  
viewing portfolio data in FA managed accounts, in  
ActiveX for Excel 7-353  
Visual Basic Editor 2-102

Visual Basic sample program, for ActiveX 3-114  
VOL orders  
in ActiveX for Excel 7-342

**W**  
winError() 4-230