**Release v7.2.0 2024-07-28**

* SecuritiesAccount.Instrument – additional fields discovered for option positions.
* OptionChainParameters class – added documentation comments for all properties
* LevelOneEquities streamer – LevelOneEquity class is “INotifyPropertyChanged” enabled.
  + LevelOneEquitiesService methods Request, Add,Remove now return List<LevelOneEquity>, the list of equities streamed. Use LevelOneEquity.PropertyChanged to notify watchers of a value changed.

LevelOneEquities streamer, LevelOneEquity class is now “INotifyPropertyChanged” enabled, also adding an additional notification level when a specific LevelOneEquity class (any field) changes. These will allow three levels of change notification:

1. Notification callback (what is in place now) whenever LevelOneEquity data response is processed, providing a changed List<LevelOneEquity>.
2. Notification callback whenever a data response is processed for a specific symbol providing a single LevelOneEquity object.
3. A standard INotifyPropertyChanged callback for a specific field from a specific symbol.

Each option has it's own use cases, allowing as fine a level of granularity as needed.

The new "EquityStreamer" wrapper class utilizes these new features. It supports sharing the streamer within your application. It supports multiple sources in your application to connect using any of the three options. It keeps track of connections, and when the last connection is removed for a symbol, it removes the symbol from streaming.

SchwabApiCS\_Test contain an example of how to use the new EquityStreamer class.

Here’s a sample use:

var es = new EquityStreamer(streamer, EquitiesStreamerCallback);

var spyData = es.Add("SPY", SpyStreamCallback, SpyPropertyCallback); // add only one symbol at a time

Any of the 3 callback methose can be null (ignored), but logically at least one must be specified to get any work done.

Sample callback routines:

public void EquitiesStreamerCallback(List<Streamer.LevelOneEquity> levelOneEquities)

{

EquityList.Dispatcher.Invoke(() =>

{

EquityList.ItemsSource = null; // to force refresh

EquityList.ItemsSource = levelOneEquities;

});

}

/// <summary>

/// Whenever a LevelOneEquities response is processed for SPY, this is called.

/// </summary>

/// <param name="data"></param>

public void SpyStreamCallback(LevelOneEquity data)

{

SpyAsk = data.MarkPrice; // updates every time a response is processed for SPY,

// REGARDLESS if changed or not

}

/// <summary>

/// Whenever a LevelOneEquities response is processed for SPY, this is called.

/// It will typically be called multiple times per SPY response,

/// oneceed for every property changed

/// </summary>

/// <param name="data"></param>

/// <param name="propertyName"></param>

public void SpyPropertyCallback(LevelOneEquity data, string propertyName)

{

switch (propertyName)

{

case "MarkPrice": SpyMark = data.MarkPrice; break; // this should only be

// called when data.MarkPrice was changed.

}

}

**Release v7.1.0 2024-07-15**

* Added additional fields to the Quote class that were missing for options, futures, and forex.
* Fixed a bug in LevelOneOptions where only the first option for each expiration date was being returned.
* AccountActivity class: added ConvertLoSignScaleToValue() and logic to preprocess json string. A number of numeric fields were represented as “lo” & “signScale” values:
  + "{fieldname}": { "lo": "5265000", "signScale": 12 } is converted to "{fieldname}": 5.265
  + "{fieldname}": { "lo": "1"} is converted to "{fieldname}": 1
  + "{fieldname}": { "signScale": 12 } is converted to "{fieldname}": null
  + "{fieldname}": {} is converted to "{fieldname}": null
  + "{fieldname}": { "lo": "5", "signScale": 4 } is converted to "{fieldname}": 0.05
  + OrderFillCompleted activity class added
* In SchwabApiTest\Mainwindow.xaml.cs added an extra check to reauthorize if refresh token fails. Normally this shouldn’t happen if RefreshTokenExpires is set for 7 days.

**Release v7.0.0 2024-07-09**

Streaming was the focus of this release. All streaming services that Schwab has implemented are working.

* LevelOneEquities
* LevelOneOptions
* LevelOneFutures
* LevelOneFutures -- Not implemented by Schwab yet
* LevelOneForexes
* NasdaqBooks -- level 2 NASDAQ
* NyseBooks -- level 2 NYSE
* OptionsBooks -- level 2 options
* ChartEquities -- minute candles stream
* ChartFutures -- minute candles stream
* ScreenerEquities -- Not implemented by Schwab yet
* ScreenerOptions -- Not implemented by Schwab yet

**Release v6.0.2 2024-07-05** (patch)

I inadvertently left a line of code setting accountNumber to the first “401” account.

To keep this from happening in the future, and to make it easy for others to set an account number for testing, GetAccountNumberForTesting() was added in Mainwindow.xaml.cs. By default it uses the first account number in the list of accounts (as before) or reads an account number from the optional file AccountNumberForTesting.txt

**Release v6.0.1 2024-07-04** (patch)

I broke the basic rule - always test after making even the most simple change! Last change I made before committing v6.0.0 was changing the version number from 5 to a more conventional v6.0.0. But it was an int, and that didn’t compile.

Other changes:

AggregatedBalance was added to the AccountInfo class. This is a value returned buy the /accounts api that was previously missed.

Added string[] invalidSymbols to the Quote class. This gets set when invalid symbols are passed to GetQuotes.

Added missing futures specific fields to the Quote class.

I believe the futures streamer will throw an exception if futures is not live. I will be testing on Friday after 5p. Check for an update this weekend it this is indeed the case.

**Release v6.0.0 2024-07-03**

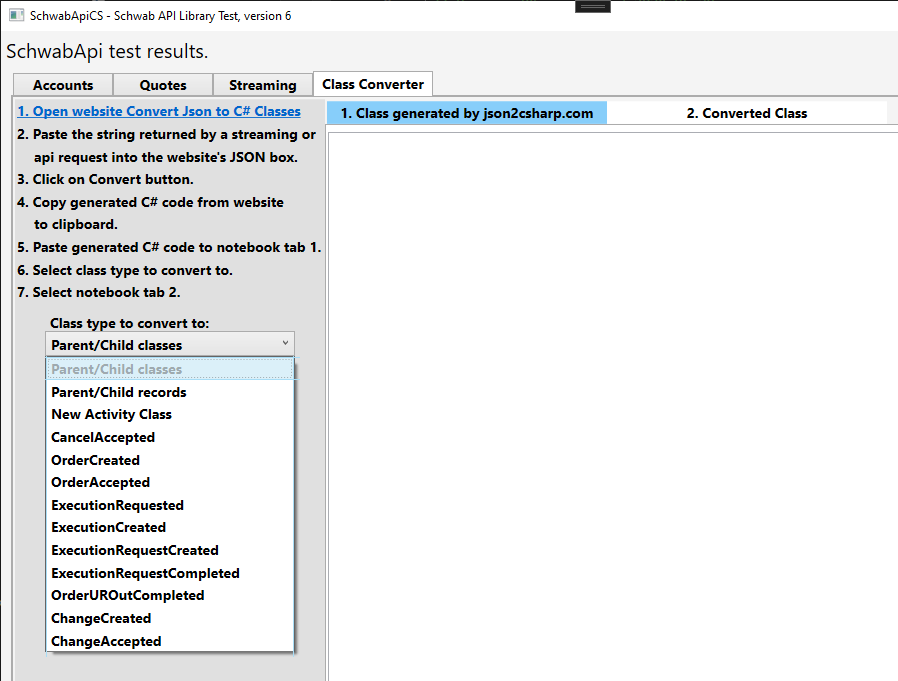
One of the OrderLeg constructors has been depreciated, and replaced with a new OrderLeg constructor that includes parameter “Position” with values “TO\_OPEN” and “TO\_CLOSE”. “Position” is needed to properly handle short equity and selling option orders. Buying and selling calls and puts has been tested, but spreads and other multi-leg orders has not.

Futures streaming is now working.

AccountActivity streamer now recognizes activities CancelAccepted, OrderCreated, OrderAccepted, ExecutionRequested, ExecutionCreated, ExecutionRequestCreated, ExecutionRequestCompleted , OrderUROutCompleted, ChangeCreated, and ChangeAccepted. There may be other undiscovered activity types. If one is encountered (activityObject in the streamer results will be null), post the json string (less any private data) on the discord server SchwabAPICS-C#, channel testing-and-bugs, and we will add it as another Account Activity class.

Added the Class Converter tab. This tool is primary intended for use with the Account Activity streamer. The streamer documentation provided by Schwab doesn’t say what activity types to expect or the contents. I use this tool to generate an activity class from the json string returned by the streamer. See the video on how to use the Class Converter at

https://u.pcloud.link/publink/show?code=XZPad90ZtHiwNmRFlrHT8e84auA6E5Sz0glV



**Release 05 2024-06-28**

Source code license has been changed from Mozilla Public License to the more permissive MIT Public License.

Get the release from <https://github.com/zpmsoftware/SchwabApiCS>.

The streamer class has been refactored and abstracted to make it easier to add new streaming services. This will require a few minor changes to your code:

* Change “streamer.EquitiesRequest()” to “streamer.LevelOneEquities.Request()”,
* Change “streamer. EquitiesAdd()” to “streamer. LevelOneEquities .Add( )”.
* Change “streamer. EquitiesRemove()” to “streamer. LevelOneEquities .Remove( )”.
* Change “streamer. EquitiesView()” to “streamer. LevelOneEquities .View( )”.

New streamer services:

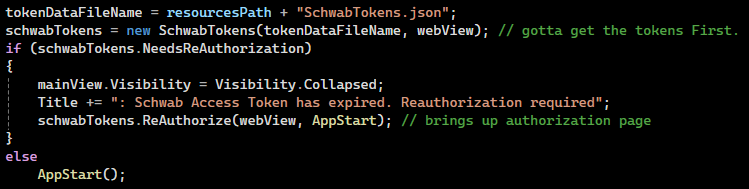
* **LevelOneOptions**. Use streamer. LevelOneOptions.Request(). See example in MainWindow.xaml.cs.

This is ready to use.

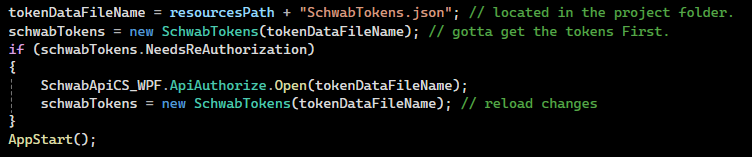
* **LevelOneFutures**. Not ready yet. It acknowledges the request, but no data is sent. It may be because I don’t trade futures. If someone can see data being returned, let us know,
* **AccountActivity**. This is working, sort of. There are some odd values in the streamed data, that may be because it’s still being developed. Only “CancelAccepted” has been fully parsed. It does return the json data string for inspection. It should be useful as is to detect when an order has been filled, rejected, changed or canceled. Better than polling.

All WPF references and related code have been removed from **SchwabApiCS** and moved to **SchwabApiCS\_WPF**. This was done to make SchwabApiCS framework independent. Because of this a few changes need to be made to your application startup:

Change:



To: (grab code from MainWindow.xaml.cs)

****

**SchwabApiAuthorize** is a new project that creates a standalone exe that can be used to open the Schwab authorization page and update SchwabTokens.json file. The exe is stored in the folder SchwabApiAuthorize\bin\Release\net7.0-windows10.0.17763.0\win-x64\publish\win-x64

It’s a large file (170mb) because it contains all needed .net dll files.

**SchwabApiCS\_WPF** is a new project containing WPF specific code used to open the Schwab authorization page and update SchwabTokens.json file whenever the RefreshToken expires. Possible this project could contain other WPF functionality in the future.

**Release 04 2024-06-20**

Detailed exception handling has been added in this release. See the video SchwabApiCS\_ExceptionHandling.mp4 on the download site

<https://u.pcloud.link/publink/show?code=kZDNyV0ZcetC7iSRwjQSbocjSDF9xSDPj4EV>

Order methods will now return the ordered of the order created or replaced.

Level One Equities Streaming has been added. Other streaming options will follow.

Search “streamer” in MainWindow.xaml.cs for sample code.

Order triggers OCO bracket has been added:

var orderOCO = new SchwabApiCS.Order(Order.OrderType.LIMIT, Order.OrderStrategyTypes.TRIGGER,

Order.Session.NORMAL, Order.Duration.GOOD\_TILL\_CANCEL, 180M);

orderOCO.Add(new Order.OrderLeg("AAPL", Order.AssetType.EQUITY, 1));

var orderTriggersOCO = schwabApi.OrderTriggersOCOBracketAsync(accountNumber, orderOCO, 250M, 150M);

orderTriggersOCO.Wait();

First build the initial order, with a TRIGGER order strategy. Alter the order type, session and duration as needed.

Then call OrderTriggersOCOBracket (or with Async) with the limit price (for profit) and stop price (for a loss). This is a good example of how to build your own special order type.

When debugging your own specialized orders, use order.JsonSerialize() to get the json string that will be sent to Schwab. Do that just before calling OrderExecuteNew() or OrderExecuteNewAsync().

**Release 03 2024-06-13**

I haven't tried buying or selling options yet, but don't expect any problems with those.

I will be adding order methods for option spreads, etc in the near future.

OrderFirstTriggersSecondAsync() has been replaced by OrderTriggersSecondAsync()

See example in MainWindow.cs.

hasError has been changed to HasError for consistency.

Hopefully there will be feew of these type of changes going forward as our code base builds.

Order methods will now return the orderId.

Methods that call the Schwab API will return the "Schwab-Client-Correlid" value in the task object.

This can be used by Schwab API support to diagnose an service call (never tried though).

The Order methods and the Order class (in the Orders folder) have been refactored, restructured to make creating new methods much easier and more generic. Added new order methods OrderTriggersSecond() and OrderOCOBracket().

I'm working on OrderTriggersOCOBracket(), but haven't been able to determine the corrrect json string to submit.

If anyone has a json string example of a working order that triggers a OCO bracket order, let me know.

Some methods have been depreciated. They are marked them as obsolete (search on obsolete).

Change the name of the obsolete method slightly (add x at the end) and the compiller let you know where they are used.