**Release 06 2024-07-03**

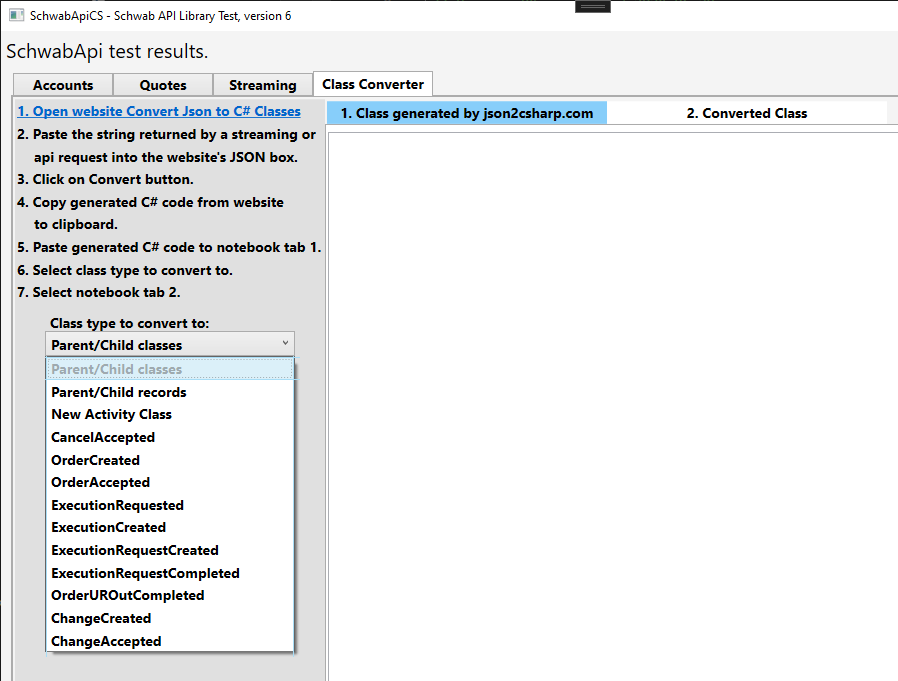
One of the OrderLeg constructors has been depreciated, and replaced with a new OrderLeg constructor that includes parameter “Position” with values “TO\_OPEN” and “TO\_CLOSE”. “Position” is needed to properly handle short equity and selling option orders. Buying and selling calls and puts has been tested, 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:

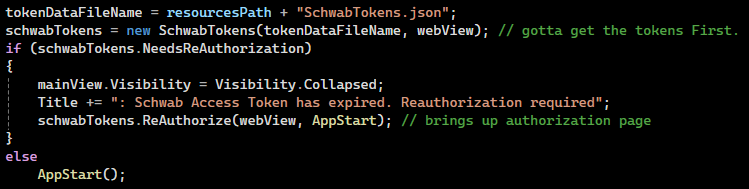
* **LevelOneOptions**. Use streamer. LevelOneOptions.Request(). See example in MainWindow.xaml.cs.

This is ready to use.

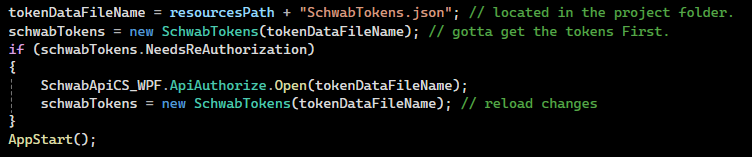
* **LevelOneFutures**. Not ready yet. It acknowledges the request, but no data is sent. It may be because I don’t trade futures. If someone can see data being returned, let us know,
* **AccountActivity**. This is working, sort of. There are some odd values in the streamed data, that may be because it’s 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.