Programming the API: Architecture

EClientSocket and EWrapper Classes

Once the TWS is up and running and actively listening for incoming connections we are ready to write our code. This brings us to the TWS API's two major classes: the **IBApi.EWrapper** interface and the **IBApi.EClientSocket**

Implementing the EWrapper Interface

The **IBApi.EWrapper** interface is the mechanism through which the TWS delivers information to the API client application. By implementing this interface the client application will be able to receive and handle the information coming from the TWS. For further information on how to implement interfaces, refer to your programming language's documentation.

```
    public class EWrapperImpl : EWrapper
    {
```

The EClientSocket Class

The class used to send messages to TWS is **IBApi.EClientSocket**. Unlike EWrapper, this class is not overriden as the provided functions in EClientSocket are invoked to send messages to TWS. To use EClientSocket, first it may be necessary to implement the **IBApi.EWrapper** interface as part of its constructor parameters so that the application can handle all returned messages. Messages sent from TWS as a response to function calls in **IBApi.EClientSocket** require a EWrapper implementation so they can processed to meet the needs of the API client.

Another crucial element is the **IBApi.EReaderSignal** object passed to theEClientSocket's constructor. With the exception of Python, this object is used in APIs to signal a message is ready for processing in the queue. (In Python the Queue class handles this task directly). We will discuss this object in more detail in the **The EReader Thread** section.

```
        EClientSocket clientSocket;
        public readonly EReaderSignal Signal;
```

...

```
        public EWrapperImpl()
        {
            Signal = new EReaderMonitorSignal();
            clientSocket = new EClientSocket(this, Signal);
        }
```