# QuantGate Client API 1.0

## Overview

The QuantGate Client API delivers the real-time signals and gauge data used to generate spectrums and other QuantGate visual indicators. The QuantGate Client API is based on a The Signal & Strategy API is a Google Protocol Buffers implementation that runs on a WebSocket connection.

## General Considerations

* TLS with SNI (Server Name Indication) is required in order to establish a QuantGate Client API connection.
* All messages sent and received via WebSockets are encoded in a Protocol Buffers (<https://developers.google.com/protocol-buffers>) implementation based on the STOMP Protocol Specification (<https://stomp.github.io/stomp-specification-1.2.html#Abstract>).
* Timestamps should not be considered unique and not be considered as aliases for transaction IDs. Also, the granularity of timestamps is not representative of transaction rates.

## Connection Details

**Connection details to Test Environment:**

|  |  |  |
| --- | --- | --- |
| URL | Scheme | Description |
| test.stealthtrader.com | wss | Test environment |

**Connection details to Production Environment:**

|  |  |  |
| --- | --- | --- |
| URL | Scheme | Description |
| feed.stealthtrader.com | wss | Production environment |

## Authentication

The API client must connect with a JWT token. This token will currently be supplied and updated in the example code.

Note: JWT token generation will be updated in the near future.

## Example API Client

C#: <https://github.com/epcylon/api-samples.git>

## Constructors

**Constructor**

**Description:**

The APIClient *Constructor* creates a new APIClient instance.

Valid *host* values are listed above in the Connection Details section. The *stream* parameter should normally be set to *realtime*, but can be set to *demo* for 24/7 testing with limited symbols.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| host | string | The web address to connect to. |
| port | int | (optional) The port to connect to. If this value is not set, the default port for the protocol (wss vs. ws) |
| stream | enum | (optional) The base DataStream to connect to  (default = realtime). |
| sync | SynchronizationContext | (optional) The synchronization context to return values on (default = SychronizationContext.Current). |

## Properties

**APIClient Properties**

**Description:**

The following public properties are available for the APIClient object.

**Public Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Host | string | The web address to connect to. |
| Port | int | (optional) The port to connect to. If this value is not set, the default port for the protocol (wss vs. ws) |
| Username | string | The currently connected username (client id). |
| Password | string | The password for the current connection (JWT Token). |
| Stream | enum | (optional) The base DataStream to connect to  (default = realtime). |
| IsConnected | bool | Indicates whether the APIClient is currently connected or not. |

## 

## Methods

**Connect Method**

**Description:**

The *Connect* method is used connect to the server with a valid JWT token.

If the connection succeeds, the *Connected* event will be fired. If the connection fails, or is closed, a *Disconnected* event will be fired.

If the connection fails after an initial successful connection, the connection will automatically attempt to reconnect to the servers whenever it becomes available again.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| jwtToken | string | Java Web Token (JWT) to connect with. |

**Disconnect Method**

**Description:**

The *Disconnect* method is used disconnect from the server. Once disconnected the object cannot be used to establish the connection again and should be disposed.

**Request Parameters:**

There are no parameters for this method.

**Dispose Method**

**Description:**

The *Dispose* method is used release all resources used by the APIClient instance. Once disposed, the object cannot be used for any other reason. Calling the *Dispose* method will disconnect the client before releasing the other resources.

**Request Parameters:**

There are no parameters for this method.

**SubscribePerception Method**

**Description:**

The *SubscribePerception* method is used to subscribe to Perception gauge updates for a given symbol. Once a Perception gauge stream has been subscribed to, Perception updates will be received in the *PerceptionUpdated* event of the APIClient.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to calculate the Perception gauge values for. |
| throttleRate | int | Rate to throttle messages at (in ms). Enter 0 for no throttling. |

**ThrottlePerception Method**

**Description:**

The *ThrottlePerception* method is used to change the maximum rate at which the back-end sends Perception gauge updates for a given symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to change the Perception gauge throttle rate for. |
| throttleRate | int | New rate to throttle messages at (in ms). Enter 0 for no throttling. |

**UnsubscribePerception Method**

**Description:**

The *UnsubscribePerception* method is used to unsubscribe from a Perception gauge data stream for a given symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to stop getting Perception data for. |

**SubscribeCommitment Method**

**Description:**

The *SubscribeCommitment* method is used to subscribe to Commitment gauge updates for a given symbol. Once a Commitment gauge stream has been subscribed to, Commitment updates will be received in the *CommitmentUpdated* event of the APIClient.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to calculate the Commitment gauge values for. |
| throttleRate | int | Rate to throttle messages at (in ms). Enter 0 for no throttling. |

**ThrottleCommitment Method**

**Description:**

The *ThrottleCommitment* method is used to change the maximum rate at which the back-end sends Perception gauge updates for a given symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to change the Commitment gauge throttle rate for. |
| throttleRate | int | New rate to throttle messages at (in ms). Enter 0 for no throttling. |

**UnsubscribeCommitment Method**

**Description:**

The *UnsubscribeCommitment* method is used to unsubscribe from a Commitment gauge data stream for a given symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to stop getting Commitment data for. |

**SubscribeEquilibrium Method**

**Description:**

The *SubscribeEquilibrium* method is used to change the maximum rate at which the back-end sends Equilibrium gauge updates for a given symbol at the specified compression.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to change the Equilibrium gauge throttle rate for. |
| compression | string | Compression timeframe being applied to the gauge.  (Default value is 300s.) |
| throttleRate | int | Rate to throttle messages at (in ms). Enter 0 for no throttling. |

**ThrottleEquilibrium Method**

**Description:**

The *ThrottleEquilibrium* method is used to change the maximum rate at which the back-end sends Equilibrium gauge updates for a specific symbol at the specified compression.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to calculate the Equilibrium gauge values for. |
| compression | string | Compression timeframe to apply to the gauge. Default value is 300s. |
| throttleRate | int | New rate to throttle messages at (in ms). Enter 0 for no throttling. |

**UnsubscribeEquilibrium Method**

**Description:**

The *UnsubscribeEquilibrium* method is used to unsubscribe from an Equilibrium gauge data stream at the specified compression for a specific symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to calculate the Equilibrium gauge values for. |
| compression | string | Compression timeframe to apply to the gauge. Default value is 300s. |

**SubscribeSentiment Method**

**Description:**

The *SubscribeSentiment* method is used to subscribe to Sentiment gauge updates for a given symbol. Once a Sentiment gauge stream has been subscribed to, Sentiment updates will be received in the *SentimentUpdated* event of the APIClient.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to calculate the Sentiment gauge values for. |
| compression | string | Compression timeframe to apply to the gauge. Default value is 50t. |
| throttleRate | int | Rate to throttle messages at (in ms). Enter 0 for no throttling. |

**ThrottleSentiment Method**

**Description:**

The *ThrottleSentiment* method is used to Changes the maximum rate at which the back-end sends Sentiment gauge updates for a specific symbol at the specified compression.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to calculate the Sentiment gauge values for. |
| compression | string | Compression timeframe to apply to the gauge. Default value is 50t. |
| throttleRate | int | New rate to throttle messages at (in ms). Enter 0 for no throttling. |

**UnsubscribeSentiment Method**

**Description:**

The *UnsubscribeSentiment* method is used to unsubscribe from Sentiment gauge data stream at the specified compression for a specific symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to calculate the Sentiment gauge values for. |
| compression | string | Compression timeframe to apply to the gauge. Default value is 50t. |

**SubscribeTrigger Method**

**Description:**

The *SubscribeTrigger* method is used to subscribe to Trigger updates for a given symbol. Once a Trigger stream has been subscribed to, Trigger updates will be received in the *TriggerUpdated* event of the APIClient.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to calculate the Trigger values for. |
| throttleRate | int | Rate to throttle messages at (in ms). Enter 0 for no throttling. |

**ThrottleTrigger Method**

**Description:**

The *ThrottleTrigger* method is used to change the maximum rate at which the back-end sends Trigger updates for a given symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to change the Trigger throttle rate for. |
| throttleRate | int | New rate to throttle messages at (in ms). Enter 0 for no throttling. |

**UnsubscribeTrigger Method**

**Description:**

The *UnsubscribeTrigger* method is used to unsubscribe from a Trigger data stream for a given symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to stop getting Trigger data for. |

**SubscribeBookPressure Method**

**Description:**

The *SubscribeBookPressure* method is used to subscribe to Book Pressure gauge updates for a given symbol. Once a Book Pressure gauge stream has been subscribed to, Book Pressure updates will be received in the *BookPressureUpdated* event of the APIClient.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to calculate the Book Pressure gauge values for. |
| throttleRate | int | Rate to throttle messages at (in ms). Enter 0 for no throttling. |

**ThrottleBookPressure Method**

**Description:**

The *ThrottleBookPressure* method is used to change the maximum rate at which the back-end sends Book Pressure gauge updates for a given symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to change the Book Pressure gauge throttle rate for. |
| throttleRate | int | New rate to throttle messages at (in ms). Enter 0 for no throttling. |

**UnsubscribeBookPressure Method**

**Description:**

The *UnsubscribeBookPressure* method is used to unsubscribe from a Book Pressure gauge data stream for a given symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to stop getting Book Pressure data for. |

**SubscribeHeadroom Method**

**Description:**

The *SubscribeHeadroom* method is used to subscribe to Headroom gauge updates for a given symbol. Once a Headroom gauge stream has been subscribed to, Headroom updates will be received in the *HeadroomUpdated* event of the APIClient.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to calculate the Headroom gauge values for. |
| throttleRate | int | Rate to throttle messages at (in ms). Enter 0 for no throttling. |

**ThrottleHeadroom Method**

**Description:**

The *ThrottleHeadroom* method is used to change the maximum rate at which the back-end sends Headroom gauge updates for a given symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to change the Headroom gauge throttle rate for. |
| throttleRate | int | New rate to throttle messages at (in ms). Enter 0 for no throttling. |

**UnsubscribeHeadroom Method**

**Description:**

The *UnsubscribeHeadroom* method is used to unsubscribe from a Headroom gauge data stream for a given symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to stop getting Headroom data for. |

**SubscribeMultiframeEquilibrium Method**

**Description:**

The *SubscribeMultiframeEquilibrium* method is used to subscribe to Multiframe Equilibrium gauge updates for a given symbol. Once a Multiframe Equilibrium gauge stream has been subscribed to, Multiframe Equilibrium updates will be received in the *MultiframeEquilibriumUpdated* event of the APIClient.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to calculate the Multiframe Equilibrium gauge values for. |
| throttleRate | int | Rate to throttle messages at (in ms). Enter 0 for no throttling. |

**ThrottleMultiframeEquilibrium Method**

**Description:**

The *ThrottleMultiframeEquilibrium* method is used to change the maximum rate at which the back-end sends Mutliframe Equilibrium gauge updates for a given symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to change the Multiframe Equilibrium gauge throttle rate for. |
| throttleRate | int | New rate to throttle messages at (in ms). Enter 0 for no throttling. |

**UnsubscribeMultiframeEquilibrium Method**

**Description:**

The *UnsubscribeMultiframeEquilibrium* method is used to unsubscribe from a Multiframe Equilibrium gauge data stream for a given symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to stop getting Multiframe Equilibrium data for. |

**SubscribeStrategy Method**

**Description:**

The *SubscribeStrategy* method is used to subscribe to strategy updates for a given symbol. Once a strategy has been subscribed to, strategy updates will be received in the *StrategyUpdated* event of the APIClient.

The *StrategyId* parameter determines the type of strategy that will be run on the symbol. Clients should currently use “Crb7.6” for the *StrategyId*. Other strategies will be added in the future.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| strategyId | string | Identifies the strategy to subscribe to for the symbol. (Should be “Crb7.6” for pilot signals). |
| symbol | string | The symbol to run the strategy on. |
| throttleRate | int | Rate to throttle messages at (in ms). Enter 0 for no throttling. |

**ThrottleStrategy Method**

**Description:**

The *ThrottleStrategy* method is used to change the maximum rate at which the back-end sends Strategy updates for the given strategy and symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| strategyId | string | The identifier of the strategy to throttle. |
| symbol | string | The symbol to change the Strategy throttle rate for. |
| throttleRate | int | New rate to throttle messages at (in ms). Enter 0 for no throttling. |

**UnsubscribeStrategy Method**

**Description:**

The *UnsubscribeStrategy* method is used to unsubscribe from Stategy data for the given strategy and symbol.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| strategyId | string | The identifier of the strategy to stop running. |
| symbol | string | The symbol to stop getting Strategy data for. |

**SubscribeTopSymbols Method**

**Description:**

The *SubscribeTopSymbols* method is used to subscribe to Top Symbols list updates for a given symbol. Once a Top Symbols stream has been subscribed to, Top Symbols list updates will be received in the *TopSymbolsUpdated* event of the APIClient.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| broker | string | The broker to get the Top Symbols for. Must match a valid broker type string (ex. “paper”, “ib”). |
| instrumentType | enum | The type of instrument to include in the results. (Setting this to None will return the top symbols for all instrument types.) |
| throttleRate | int | Rate to throttle messages at (in ms). Enter 0 for no throttling. |

**ThrottleTopSymbols Method**

**Description:**

The *ThrottleTopSymbols* method is used to change the maximum rate at which the back-end sends Top Symbols updates for the given broker and instrument type.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| broker | string | The broker to throttle the Top Symbols for. |
| instrumentType | enum | The type of instrument to throttle the results for. |
| throttleRate | int | The new throttle rate to set to (in ms). Enter 0 for no throttling. |

**UnsubscribeTopSymbols Method**

**Description:**

The *UnsubscribeTopSymbols* method is used to unsubscribe from Top Symbols data for the given broker and instrument type.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| broker | string | The broker to stop getting the Top Symbols for. |
| instrumentType | enum | The type of instrument to stop including from the results. |

**SearchSymbols Method**

**Description:**

The *SearchSymbols* is used to request symbols that match a specific term and (optionally) a specific broker. Once a Top Symbols stream has been subscribed to, Top Symbols list updates will be received in the *SymbolSearchUpdated* event of the APIClient.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| term | string | The term to search for. |
| broker | string | Broker to get the results for. If supplied, must match a valid broker type string. |

**RequestInstrumentDetails Method**

**Description:**

The *RequestInstrumentDetails* method is used to retrieve Instrument details for a given symbol. After an Instrument has been requested, an Instrument update will be received in the *InstrumentUpdated* event of the APIClient.

Note: If the symbol is invalid, the *InstrumentType* will be *None*, and an *ErrorMessage* will be present.

**Request Parameters:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| symbol | string | The symbol to get the instrument details for. |

## Events

**Connected Event**

**Description:**

The *Connected* event is fired whenever the Client API establishes a connection to the back-end servers.

**EventArg Properties:**

There are no EventArg properties for this event.

**Disconnected Event**

**Description:**

The *Disconnected* event is fired whenever the Client API either fails to connect, or disconnects from the back-end servers. If a valid connection was previously established, the Client API will attempt to reconnect automatically, unless the *Disconnect* method was called.

**EventArg Properties:**

There are no EventArg properties for this event.

**Error Event**

**Description:**

The *Error* event is fired whenever an error message is received from the backend servers.

**EventArg Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Message | string | The error message received from the backend servers. |

**PerceptionUpdated Event**

**Description:**

The *PerceptionUpdated* event is fired whenever the Perception gauge values are updated for a symbol. The parameters will identify the updated symbol, and represent the latest Perception gauge value.

**EventArg Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Symbol | string | The symbol that the gauge is being calculated for. |
| TimeStamp | datetime | Timestamp of the latest update, for when the update occurred. |
| Value | double | Normalized level (between -1 and 1) of the Perception gauge. |
| IsDirty | bool | Whether the data used to generate this gauge value is potentially dirty (values are missing) or stale (not the most recent data). False if clean. |

**CommitmentUpdated Event**

**Description:**

The *CommitmentUpdated* event is fired whenever the Commitment gauge values are updated for a symbol. The parameters will identify the updated symbol, and represent the latest Commitment gauge value.

**EventArg Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Symbol | string | The symbol that the gauge is being calculated for. |
| TimeStamp | datetime | Timestamp of the latest update, for when the update occurred. |
| Value | double | Normalized level (between -1 and 1) of the Commitment gauge. |
| IsDirty | bool | Whether the data used to generate this gauge value is potentially dirty (values are missing) or stale (not the most recent data). False if data is clean. |

**EquilibriumUpdated Event**

**Description:**

The *EquilibriumUpdated* event is fired whenever the Equilibrium gauge values are updated for a symbol. The parameters will identify the updated symbol, and represent the latest Equilibrium gauge values.

**EventArg Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Symbol | string | The symbol that the gauge is being calculated for. |
| TimeStamp | datetime | Timestamp of the latest update, for when the update occurred. |
| Compression | string | Compression timeframe applied to the gauge. |
| EquilibriumPrice | double | The Equilibrium Price (fair market value at time of calculation). |
| GapSize | double | Gap size of each equilibrium deviation. |
| LastPrice | double | Last traded price at the time of calculation. |
| High | double | Position of the high value. |
| Low | double | Position of the low value. |
| Projected | double | Position of the projected value. |
| Bias | double | Equilibrium bias (as determined by the slope). |
| EquilibriumSTD | double | The current equilibrium gauge level in standard deviations from the equilibrium price. |
| IsDirty | bool | Whether the data used to generate this gauge value is potentially dirty (values are missing) or stale (not the most recent data). False if clean. |

**Note:**

The EventArg object for the *EquilibriumUpdated* event also has a method named *EquilibriumBand* that takes a double parameter value called *level*, which calculates the Equilibrium band price at the given *level* of standard deviations from the equilibrium price.

**SentimentUpdated Event**

**Description:**

The *SentimentUpdated* event is fired whenever the Sentiment gauge values are updated for a symbol. The parameters will identify the updated symbol/compression pair, and represent the latest Sentiment gauge values.

**EventArg Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Symbol | string | The symbol that the gauge is being calculated for. |
| TimeStamp | datetime | Timestamp of the latest update, for when the update occurred. |
| Compression | string | Compression timeframe applied to the gauge. |
| Lengths | double[] | Read-only list of the lengths of each bar (with values from -1 to 1), starting with the center bar at index 0, and ending with the further outside bar at index 54. |
| Colors | double[] | Read-only list of the colors of each bar (with values from -1 to 1), starting with the center bar at index 0, and ending with the further outside bar at index 54. |
| AvgLength | double | The normalized average length of all bars (-1 to 1). |
| AvgColor | double | The normalized average color of all bars (-1 to 1). |
| IsDirty | bool | Whether the data used to generate this gauge value is potentially dirty (values are missing) or stale (not the most recent data). False if clean. |

**TriggerUpdated Event**

**Description:**

The *TriggerUpdated* event is fired whenever the Trigger values are updated for a symbol. The parameters will identify the updated symbol, and represent the latest Trigger values.

**EventArg Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Symbol | string | The symbol that the gauge is being calculated for. |
| TimeStamp | datetime | Timestamp of the latest update, for when the update occurred. |
| Bias | double | Overall bias value (long-term bias). |
| Perception | double | Normalized level (between -1 and 1) of the Perception gauge. |
| Sentiment | double | Normalized level (between -1 and 1) of the Sentiment gauge (correlates to the central bar of the Sentiment gauge). |
| Commitment | double | Normalized level (between -1 and 1) of the Commitment gauge. |
| EquilibriumPrice | double | The Equilibrium Price (fair market value at time of calculation). |
| GapSize | double | Gap size of each equilibrium deviation. |
| LastPrice | double | Last traded price at the time of calculation. |
| EquilibriumSTD | double | The current equilibrium gauge level in standard deviations from the equilibrium price. |
| IsDirty | bool | Whether the data used to generate this gauge value is potentially dirty (values are missing) or stale (not the most recent data). False if clean. |

**Note:**

The EventArg object for the *EquilibriumUpdated* event also has a method named *EquilibriumBand* that takes a double parameter value called *level*, which calculates the Equilibrium band price at the given *level* of standard deviations from the equilibrium price.

**BookPressureUpdated Event**

**Description:**

The *BookPressureUpdated* event is fired whenever the Book Pressure gauge values are updated for a symbol. The parameters will identify the updated symbol, and represent the latest Book Pressure gauge value.

**EventArg Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Symbol | string | The symbol that the gauge is being calculated for. |
| TimeStamp | datetime | Timestamp of the latest update, for when the update occurred. |
| Value | double | Normalized level (between -1 and 1) of the Book Pressure gauge. |
| IsDirty | bool | Whether the data used to generate this gauge value is potentially dirty (values are missing) or stale (not the most recent data). False if data is clean. |

**HeadroomUpdated Event**

**Description:**

The *HeadroomUpdated* event is fired whenever the Headroom gauge values are updated for a symbol. The parameters will identify the updated symbol, and represent the latest Headroom gauge value.

**EventArg Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Symbol | string | The symbol that the gauge is being calculated for. |
| TimeStamp | datetime | Timestamp of the latest update, for when the update occurred. |
| Value | double | Normalized level (between -1 and 1) of the Headroom gauge. |
| IsDirty | bool | Whether the data used to generate this gauge value is potentially dirty (values are missing) or stale (not the most recent data). False if data is clean. |

**MultiframeEquilibriumUpdated Event**

**Description:**

The *MultiframeEquilibriumUpdated* event is fired whenever the Multiframe Equilibrium gauge values are updated for a symbol. The parameters will identify the updated symbol, and represent the latest Multiframe Equilibrium gauge values.

**EventArg Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Symbol | string | The symbol that the gauge is being calculated for. |
| TimeStamp | datetime | Timestamp of the latest update, for when the update occurred. |
| Value | double | Normalized level (between -1 and 1) of the Headroom gauge. |
| IsDirty | bool | Whether the data used to generate this gauge value is potentially dirty (values are missing) or stale (not the most recent data). False if data is clean. |

**StrategyUpdated Event**

**Description:**

The *StrategyUpdated* event is fired whenever the strategy values are updated for a strategy that is being run on a symbol. The parameters will identify the symbol/strategy pair, and represent the latest indicator values for the entry progress (%) gauge, as well as the PCES lights, and the current entry signal (long/short) status.

**EventArg Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Symbol | string | The symbol that the strategy is running on. |
| StrategyId | string | Identifies the strategy that was subscribed to for the symbol. (Should be “Crb7.6” for pilot signals). |
| TimeStamp | datetime | Timestamp of the latest update, for when the update occurred. |
| EntryProgress | double | This is the value (between -1 and 1) that is displayed in Pilot as the “percentage” gauge for the strategy (1.0=100%). |
| ExitProgress | double | Similar to EntryProgress, but for exits, rather than entries – not currently implemented. |
| Signal | enum | This is the buy/sell signal (it will indicate 1 for long/buy, or -1 for short/sell) associated with the strategy. |
| PerceptionLevel | double | Normalized level (between -1 and 1) of the strategy-adjusted Perception gauge, if applicable – will be null where not used. |
| PerceptionSignal | enum | Tied to the Perception light for strategy-adjusted Perception signals – will be 0 (“Unknown”) where not used. |
| CommitmentLevel | double | Normalized level (between -1 and 1) of the strategy-adjusted Commitment gauge, if applicable – will be null where not used. |
| CommitmentSignal | enum | Tied to the Commitment light for strategy-adjusted Commitment signals – will be 0 (“Unknown”) where not used. |
| EquilibriumLevel | double | Normalized level (between -1 and 1) of the strategy-adjusted Equilibrium gauge (correlates to the equilibrium deviation value of the Equilibrium gauge), if applicable – will be null where not used. |
| EquilibriumSignal | enum | Tied to the Equilibrium light for strategy-adjusted Equilibrium signals – will be 0 (“Unknown”) where not used. |
| SentimentLevel | double | Normalized level (between -1 and 1) of the strategy-adjusted Sentiment gauge (correlates to the central bar of the Sentiment gauge), if applicable – will be null where not used. |
| SentimentSignal | enum | Tied to the Sentiment light for strategy-adjusted Sentiment signals – will be 0 (“Unknown”) where not used. |

**SymbolSearchUpdated Event**

**Description:**

The *SymbolSearchUpdated* event is fired whenever Symbol Search values are received. The parameters will identify the search term as well as a list of search results.

**EventArg Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Results | List<SearchResult> | The symbol that the gauge is being calculated for. |

**TopSymbol Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Symbol | string | Symbol as listed by the QuantGate servers. |
| Underlying | string | Underlying symbol. |
| Currency | string | Currency the instrument is traded in. |
| Exchange | string | Primary exchange (ISO 10383 MIC) the instrument is traded on. |
| InstrumentType | enum | Type of instrument (stock/future, etc.). |
| DisplayName | string | Display name of the instrument. |

**TopSymbolsUpdated Event**

**Description:**

The *TopSymbolsUpdated* event is fired whenever the Top Symbols list values are updated. The parameters will identify the updated symbol, and represent the latest Multiframe Equilibrium gauge values.

**EventArg Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Symbols | List<TopSymbol> | The symbol that the gauge is being calculated for. |

**TopSymbol Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| TimeStamp | DateTime | Timestamp of the latest update. |
| Symbol | string | Symbol as listed by the QuantGate servers. |
| Underlying | string | Underlying symbol. |
| Currency | string | Currency the instrument is traded in. |
| Exchange | string | Primary exchange (ISO 10383 MIC) the instrument is traded on. |
| InstrumentType | enum | Type of instrument (stock/future, etc.). |
| DisplayName | string | Display name of the instrument. |
| EntryProgress | double | This is the value (between -1 and 1) that is displayed in Pilot as the “percentage” gauge for the strategy (1.0=100%). |
| PerceptionSignal | enum | Tied to the Perception light for the Perception signal. |
| CommitmentSignal | enum | Tied to the Commitment light for the Commitment signal. |
| EquilibriumSignal | enum | Tied to the Equilibrium light for the Equilibrium signal. |
| SentimentSignal | enum | Tied to the Sentiment light for the Sentiment signal. |
| Signal | enum | This is the buy/sell signal (it will indicate 1 for long/buy, or -1 for short/sell) associated with the default strategy. |

**InstrumentUpdated Event**

**Description:**

The *InstrumentUpdated* event is fired whenever the Instrument gauge values are updated for a symbol. The parameters will identify the updated symbol, and represent the latest Multiframe Equilibrium gauge values.

Note: If the symbol is invalid, the *InstrumentType* will be *None*, and an *ErrorMessage* will be present.

**EventArg Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Symbol | string | Symbol as listed by the QuantGate servers. |
| Underlying | string | Underlying symbol. |
| Currency | string | Currency the instrument is traded in. |
| Exchange | string | Primary exchange (ISO 10383 MIC) the instrument is traded on. |
| DisplayName | string | Display name of the instrument. |
| InstrumentType | enum | Type of instrument (stock/future, etc.). |
| PutOrCall | enum | Right of an option, if an option (will be empty otherwise). |
| Strike | double | Strike price of an option, if an option (will be zero otherwise). |
| ExpiryDate | DateTime | Expiry date of the instrument, if applicable. |
| Multiplier | double | Price multiplier (to convert price to value). |
| TimeZone | TimeZoneInfo | Time zone of the primary exchange that the instrument is traded on. |
| TickRanges | List<TickRange> | Tick ranges used to determine price levels.  (See below for parameters.) |
| TradingSessions | List<TradingSession> | Trading session end times and lengths for each day Sunday-Saturday, specified in the time zone of the primary exchange. (See below for parameters, above for time zone.) |
| BrokerSymbols | Dictionary  <string, string> | Map of broker symbols according to broker (ib, cqg, etc.). |
| ErrorMessage | string | Describes an error if the instrument details request failed. |

**TickRange Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Start | double | Start of the tick range. |
| Tick | double | Tick value at this range. |
| Denominator | int | Denominator for fractional formats. |
| Decimals | int | Number of decimals in decimal format. |
| Format | enum | Format to use (Decimal, Fraction, or Tick). |

**TradingSession Properties:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| DayOfWeek | enum | The day of the week that the trading session ends on. |
| Close | TimeSpan | The close time of the trading day. |
| Length | TimeSpan | The length of the trading session. |
| Open | TimeSpan | The open minute of the trading day. |
| StartsOnPrevious | bool | Returns true if the start of the trading session is on the previous day of the week. |