QuantConnect Documentation - Cloud Platform

Created on 06/12/2023

Copyright QuantConnect 2023

Table of Content

- 1 Welcome
- 2 Getting Started
- 3 Security and IP
- 4 Organizations
- 4.1 Getting Started
- 4.2 Tier Features
- 4.3 Resources
- 4.4 Data Storage
- 4.5 Support
- 4.6 Members
- 4.7 Administration
- 4.8 Billing
- 4.9 Credit
- 4.10 Training
- 5 Learning Center
- 5.1 Training
- 5.2 Educators
- 5.3 Course Structure
- 6 Projects
- 6.1 Getting Started
- 6.2 Structure
- 6.3 Files
- 6.4 IDE
- 6.5 Debugging
- 6.6 Collaboration
- 6.7 Code Sessions
- 6.8 Shared Libraries
- 6.9 Package Environments
- 6.10 LEAN Engine Versions
- 7 Research
- 7.1 Getting Started
- 7.2 Deployment
- 8 Backtesting
- 8.1 Getting Started
- 8.2 Research Guide
- 8.3 Deployment
- 8.4 Results
- 8.5 Debugging
- 8.6 Report
- 8.7 Engine Performance
- 9 Datasets
- 9.1 Navigating the Market
- 9.2 Categories
- 9.3 Data Issues
- 9.4 Misconceptions
- 9.5 Licensing
- 9.6 Vendors
- 10 Live Trading
- 10.1 Getting Started
- 10.2 Brokerages
- 10.2.1 Binance
- 10.2.2 Bitfinex
- 10.2.3 Coinbase
- 10.2.4 Interactive Brokers
- 10.2.5 Kraken
- 10.2.6 Oanda
- 10.2.7 Prime Brokerages
- 10.2.8 QuantConnect Paper Trading
- 10.2.9 Samco
- 10.2.10 TD Ameritrade
- 10.2.11 Tradier
- 10.2.12 Trading Technologies
- 10.2.13 Wolverine
- 10.2.14 Zerodha
- 10.2.15 FIX Connections
- 10.2.16 Unsupported Brokerages
- 10.3 Data Feeds
- 10.3.1 US Equities
- 10.3.2 Crypto
- 10.3.3 Crypto Futures
- 10.3.4 CFD
- 10.3.5 Forex
- 10.3.6 Futures
- 10.3.7 Future Options
- 10.3.8 Alternative Data
- 10.3.9 Brokerage Data Feeds
- 10.3.9.1 Interactive Brokers
- 10.3.9.2 Samco
- 10.3.9.3 Tradier
- 10.3.9.4 Zerodha
- 10.4 Deployment
- 10.5 Notifications
- 10.6 Results
- 10.7 Algorithm Control
- 10.8 Reconciliation
- 10.9 Risks
- 11 Optimization
- 11.1 Getting Started
- 11.2 Parameters
- 11.3 Objectives
- 11.4 Strategies
- 11.5 Deployment
- 11.6 Results
- 12 Data Storage
- 13 Community
- 13.1 Code of Conduct
- 13.2 Forum
- 13.3 Discord
- 13.4 Profile
- 13.5 Integration Partners
- 13.6 Affiliates
- 14 API Reference

- 14.1 Authentication
- 14.2 Project Management
- 14.2.1 Create Project
- 14.2.2 Read Project
- 14.2.3 Update Project
- 14.2.4 Delete Project
- 14.2.5 Read Project Nodes
- 14.2.6 Update Project Nodes
- 14.3 File Management
- 14.3.1 Create File
- 14.3.2 Read File
- 14.3.3 Update File
- 14.3.4 Delete File
- 14.4 Compiling Code
- 14.4.1 Create Compilation Job
- 14.4.2 Read Compilation Result
- 14.5 Backtest Management
- 14.5.1 Create Backtest
- 14.5.2 Read Backtest
- 14.5.2.1 Backtest Statistics
- 14.5.2.2 Portfolio
- 14.5.2.3 Orders
- 14.5.3 Update Backtest
- 14.5.4 Delete Backtest
- 14.6 Live Management
- 14.6.1 Create Live Algorithm
- 14.6.2 Read Live Algorithm
- 14.6.2.1 Live Algorithm Statistics
- 14.6.2.2 Logs
- 14.6.2.3 Portfolio State
- 14.6.2.4 Orders
- 14.6.3 Update Live Algorithm
- 14.6.3.1 Liquidate Live Portfolio
- 14.6.3.2 Stop Live Algorithm
- 14.7 Downloading Data
- 14.7.1 Read Downloaded Data
- 14.8 Reports
- 14.8.1 Backtest Report

1 Welcome

QuantConnect is an open-source, community-driven algorithmic trading platform. Our trading engine is powered by LEAN , a cross-platform, multi-asset technology that brings cutting-edge finance to the open-source community. We support Python and C# programming languages.

Our Mission

Quantitative trading infrastructure should be open-source. Millions of financial engineers rewrite the same infrastructure and then keep their work closed to make it harder for competing trading firms. We take a radically open approach to quant finance and let our users focus on alpha, not infrastructure.

The future of finance is automated and we are the open-source infrastructure to power this future. Firms choose our open-source platform as it provides a 10-100x improvement in time to market and substantially reduces the risk of developing your quantitative tools. As our brokerage and data integrations expand, this leverage will be more exceptional.

Everyone should have access to quantitative finance. Algorithmic trading is a powerful tool and we want to open it up to all investors. We do this with transparent, scalable pricing that allows all investors to access quantitative finance. For more information on our mission, check out our Manifesto .

Data Library

We provide an enormous library of data for your backtesting, research, and live trading. The library includes data for Equities, Options, Futures, CFDs, Forex, Crypto, Indices, and alternative data. The library is roughly 400TB in size, contains trade data that spans decades into the past, and comes in tick to daily resolutions. View he Dataset Market to see all of the datasets that we have available, including their respective start dates, end dates, and resolutions. The following image shows the integrated data providers:

Business Model

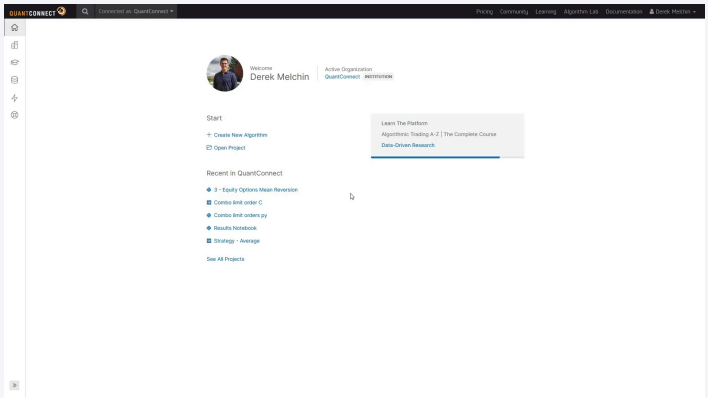
QuantConnect provides cloud infrastructure as a service, similar to many cloud compute vendors. We encourage quants and start up firms to grow within our ecosystem to keep our pricing accessible to individuals and small firms.

For companies interested in running QuantConnect on-premise, we can install it within your corporate firewall and help you get set up with financial data. We charge a set up and maintenance fee for these installations.

2 Getting Started

GET STARTED WITH QUANTCONNECT

Guide through creating a project, running your first backtest, and live algo trading.



Follow these steps to create, backtest, and paper trade a new algorithm:

1. Log in to the Algorithm Lab .

2. On the Projects page, click Create New Algorithm .

Create New Algorithm

3. A new project opens in the Cloud Integrated Development Environment (IDE) .

4. At the top of the IDE, click the Build icon.

5. At the top of the IDE, click the Backtest icon.

The backtest results page displays your algorithmâ€™s performance over the backtest period.

Backtest Results

6. At the top of the IDE, click the Deploy Live icon.

Deploy Live

7. On the Deploy Live page, click the Brokerage field and then click Paper Trading from the drop-down menu.

Deploy Live Page

8. Click Deploy .

The live results page displays your algorithmâ€™s live trading performance.

To deploy a live algorithm with a different brokerage, see the Deploy Live Algorithms section of the brokerage integration documentation .

Live Results Page

3 Security and IP

Introduction

You own all your intellectual property and your code. Your code is private by default unless you explicitly share it with the community in the forums or with another team member via collaboration. You are creating valuable intellectual property, and we respect this and wish to make it easier. We limit the QuantConnect staff members who have access to the database. If we ever need access to your algorithm for debugging, then we will explicitly request permission first.

Intellectual Property

The following sections explain our commitment to protecting your intellectual property.

Ownership

You own all your intellectual property and your code. Your code is private by default unless you explicitly share it with the community in the forums or with another team member via collaboration. You are creating valuable intellectual property; we respect this and wish to make it easier. We document this publicly in our Terms of Service.

Alignment

Beyond words and our legal terms of service, we live this through the alignment of our business model with you, our clients. We aim for you to become more successful, thereby growing within our ecosystem. We've served start-up quant funds as they've grown from $0 to $1B+ AUM.

Reputation and Track Record

Established in 2012, we have a pristine reputation and a 10+ year record of protecting our community's intellectual property. We have served more than 200,000 clients. If we were to violate the trust of even a single client, we'd lose the entire community's faith - it is simply not worth it. We seek to change the future of finance and are driven by this mission.

Sharing

QuantConnect provides support with a specific support agent, not an anonymous team. When submitting a support ticket, you explicitly grant that team member access to your project. You can remove the support agent from your project collaboration at any time.

Security

We take a multi-level approach to security, from physical security to digital and information systems, internal processes, and testing.

Physical Security of Servers

Physical access to our servers is limited to a few dedicated team members who QuantConnect has vetted. Only those credentialed team members can access the physical servers, and we schedule all work in advance. Work on the servers is always done in pairs to prevent single rouge actors from accessing the servers. We host our servers in a world-class security facility (Equinix) with security staff 24/7.

Information and Digital Security

We use all good common sense information security processes: passwordless servers, encryption in the database and backups, encrypted traffic, and network monitoring. We keep most of our servers off the internet and only available on private networks for the smallest possible surface area. We have regular network and code penetration testing. All code is containerized and isolated in services so that root network access would provide little-to-no benefit.

Beyond these basics, we've built active monitoring technology which proactively detects and blocks threats. We have human detection services to reduce the chances of brute-force attacks. We have documented processes for client notifications in the event of strange network activity.

Processes

Deployment environments are automated and enforce code peer-review to be deployed, reducing the chances of a rouge internal agent.

We limit staff access to the physical servers, restricting core database access to only a handful of senior staff. Database credentials are carefully restricted in scope, access locations and frequently rotated.

Privacy

Protecting users' private and intellectual property is of utmost importantance to us. QuantConnect complies with GDPR and all relevant privacy laws. We will never sell or publish your email address. We request knowledge of your real identity to ensure compliance with our data licenses but accept using an alias on public profiles for privacy. For more information about what data we collect, which tracking technologies we use, and how we use and share your data, see our Privacy Policy .

4 Organizations

An organization is a collection of members that share hardware resources, share access to datasets, and collaborate together to develop projects. Hardware resources are used to run backtests, launch research notebooks, deploy live trading algorithms, and store project data. You can create new organizations and join existing ones. You can be a member in any number of organizations. We offer several organization tiers so you can tailor your team's subscriptions as you grow over time. For the times when you need access to a QuantConnect engineer to help solve development issues, assign support seats among your team. There are several tiers of support seats to match the level of support your team requires.

Getting Started

Learn the basics

Tier Features

Tiers to serve all team sizes

Resources

Share hardware nodes to research and trade

Data Storage

ObjectStore and projects take space

Support

Connect with QC experts

Members

Effectively manage your team

Administration

Manage your organization

Billing

Configure your QC services

Credit

Gift to others or spend on QC services

Training

Get team members up to speed

See Also

Collaboration
Learning Center

4.1 Getting Started

Introduction

An organization is a collection of members that share hardware resources, share access to datasets, and collaborate together to develop projects. Hardware resources are used to run backtests, launch research notebooks, deploy live trading algorithms, and store project data. You can create new organizations and join existing ones. You can be a member in any number of organizations. We offer several organization tiers so you can tailor your team's subscriptions as you grow over time. For the times when you need access to a QuantConnect engineer to help solve development issues, assign support seats among your team. There are several tiers of support seats to match the level of support your team requires.

Add Organizations

Follow these steps to add new organizations to your profile:

1. Log in to the Algorithm Lab.
2. In the top navigation bar, click Connected as: organizationName .
3. In the Switch Organization panel, click Create Organization .
4. Enter the organization name and then click Add .

   The organization name must be unique. "Created Successfully" displays.

Switch Organizations

Follow these steps to switch organizations:

1. Log in to the Algorithm Lab.
2. In the top navigation bar, click Connected as: organizationName .
3. In the Switch Organization panel, click the name of the organization for which you want to connect.

   The top navigation bar displays the new organization name.

Rename Organizations

Follow these steps to change your organization name:

1. Open the organization homepage .
2. Hover over the organization name and then click the pencil icon that appears.

3. Enter the new organization name and then click Save Changes .

   "Organization Name Updated Successfully" displays.

4.2 Tier Features

Introduction

An organization is a collection of members that share hardware resources, share access to datasets, and collaborate together to develop projects. Hardware resources are used to run backtests, launch research notebooks, deploy live trading algorithms, and store project data. You can create new organizations and join existing ones. You can be a member in any number of organizations. We offer several organization tiers so you can tailor your team's subscriptions as you grow over time. For the times when you need access to a QuantConnect engineer to help solve development issues, assign support seats among your team. There are several tiers of support seats to match the level of support your team requires.

Organizations let you coordinate resources and teamwork on QuantConnect Cloud. There are 5 tiers of organizations and each tier has its own set of features. Each account starts with a personal organization on the Free tier with access to one free backtest node and one free research node. However, to accommodate the growth of your trading skills and business, you can adjust the tier of your organization at any time. Higher tiers offer more live nodes to run more live algorithms, more backtesting nodes for faster concurrent backtesting, and many other features.

Free Tier

The Free tier provides cloud access to datasets for all of the asset classes in our Datasets Market . The free data ranges from minute to daily resolutions and can be used to either run backtests or perform analysis in the Research Environment . When backtesting, Free organizations have access to our built-in auto-complete and debugging features in the web IDE. After a successful backtest, Free organizations can use our report generator to create professional-grade reports that reflect their backtest performance. Free organizations have access to our online documentation, community forum, YouTube channel, and Learning Center.

Quant Researcher Tier

The Quant Researcher tier is designed for self-directed investors, students, academics, and independent traders seeking to manage their personal portfolio. We recommend the Quant Researcher Pack to make the most of QuantConnect.

The Quant Researcher tier builds on the features included in the Free tier. Organizations on the Quant Researcher tier have access to the QuantConnect API and can use the CLI to run Lean locally. When members in these organizations need assistance from a QuantConnect engineer, support seats are available to request private support . Members within Quant Researcher organizations that have the required permissions can adjust the resources within the organization.

In Quant Researcher organizations, members can use second and tick resolution data from the Datasets Market. There is no limit on the number of projects these organizations can hold. They can produce up to 100KB of logs/backtest, 3MB of logs/day. 10 million orders/backtest, and can have up to two backtesting nodes to run up to two concurrent backtests. Members in these organizations can have up to two active coding sessions in the organization. After a successful backtest, members in these organizations can use parameter optimization tools to improve the performance of their backtest. When the members are ready to deploy strategies live, Quant Researcher organizations can subscribe to up to 2 live trading nodes to unlock live trading with real or paper money. Each live algorithm in a Quant Research organization can send up to 20 Telegram, Email, or Webhook notifications per hour for free. SMS notifications and additional Telegram, Email, or Webhook notifications require QuantConnect Credit (QCC).

Team Tier

The Team tier is designed for sophisticated individuals and teams of quant collaborators such as Quant Start Ups, Fintech Companies, and Emerging Managers. We recommend the Team Pack to make the most of QuantConnect.

The Team tier expands on the features included in the Quant Researcher tier. Organizations on the Team tier can have up to 10 members and the members can collaborate on projects together. These organizations can produce 1MB of logs/backtest, 10MB of logs/day, and there is no limit on the number of orders that they can place in backtests. Organizations on the Team tier can have up to 10 backtesting nodes, 10 research nodes, and 10 live trading nodes. Members in these organizations can have up to four active coding sessions in the organization.

To accommodate a large number of projects in Team organizations, these organizations can expand the capacity of their ObjectStore up to 10GB. Annual contracts for onboarding services are available on request to get teams operational in the shortest amount of time. When live trading, Team organizations have more options than the lower tiers because both the Trading Technologies brokerage and our live Futures data feed are available. Each live algorithm in a Team organization can send up to 60 Telegram, Email, or Webhook notifications per hour for free.

Trading Firm Tier

The Trading Firm tier is designed for growing quantitative firms, prop desks, hedge funds, ETF companies, professional teams of quants, and sophisticated independent investors. It has special features for collaborating with consultants to protect the investor IP. If you are a company on QuantConnect, we recommend the Trading Firm Pack to make the most of QuantConnect.

The Trading Firm tier builds on the features included in the Team tier. Organizations on the Trading Firm tier can have an unlimited number of members and an unlimited number of collaborators simultaneously working on individual projects. The IP ownership of all the projects in these organizations remains within the organization. There is no limit on the number of backtesting, research, and live trading nodes these organizations can rent. They can produce 5MB of logs/backtest and 50MB of logs/day. Members in these organizations can have up to eight active coding sessions in the organization. Each live algorithm in a Trading Firm organization can send up to 240 Telegram, Email, or Webhook notifications per hour for free.

The owner of a Trading Firm organization can grant various permissions to the organizationâ€™s members, including designating a member to manage the organization's billing. These organizations have access to custom lean builds, so they can use feature branches or historical master branches to run their strategies. An example of this could be granting only a few members of your team live trading deployment access.

In addition to the brokerages and data feeds available to Team organizations, Trading Firm organizations can use Interactive Brokers Financial Advisor accounts to manage sub-accounts for clients.

Institution Tier

The Institution tier is designed for established larger funds, large prop desks, hedge funds, banks, ETF vehicles, and professional teams of quants. It is "unlocked", so you can run it on premise to serve your internal teams. If this sounds interesting, reach out and we'd be happy to arrange a demonstration for your department.

The Institution tier builds on the features included in the Trading Firm tier. Organizations on the Institution tier have no limit on the amount of backtest logs that they can produce and each member in the organization can have up to 16 active coding sessions. These organizations can use the Terminal Link CLI tool to live trade Equities, Futures, and Options via the Bloomberg EMSX. They can also request custom libraries and frameworks to use in the QuantConnect web IDE and receive instant messaging support from a QuantConnect engineer. Each live algorithm in an Institution organization can send up to 3,600 Telegram, Email, or Webhook notifications per hour for free.

4.3 Resources

Introduction

Organizations can subscribe to hardware resources to run backtests, launch research notebooks, and deploy live trading algorithms to co-located servers. Organizations also have access to storage resources via the ObjectStore to store data between backtests or live trading deployments. To promote efficiency, all of these resources within your organization are shared among all of the members within the organization. A team of several quants can all share one backtest, research, and live trading node.

Backtesting Nodes

Backtesting nodes enable you to run backtests. The more backtesting nodes your organization has, the more concurrent backtests that you can run. Several models of backtesting nodes are available. Backtesting nodes that are more powerful can run faster backtests and backtest nodes with more RAM can handle more memory-intensive operations like training machine learning models, processing Options data, and managing large universes. The following table shows the specifications of the backtesting node models:

Name	Number of Cores	Processing Speed (GHz)	RAM (GB)	GPU
B-MICRO	2	3	8	0
B2-8	2	4.9	8	0
B4-12	4	4.9	12	0
B4-16-GPU	4	3	16	1/3
B8-16	8	4.9	16	0

Refer to the Pricing page to see the price of each backtesting node model. You get one free B-MICRO backtesting node in your first organization. This node incurs a 20-second delay when you launch backtests, but the delay is removed and the node is replaced when you subscribe to a new backtesting node in the organization.

GPU nodes perform best on repetitive and highly-parallel tasks like training machine learning models. It takes time to transfer the data to the GPU for computation, so if your algorithm doesn't train machine learning models, the extra time it takes to transfer the data can make it appear that GPU nodes run slower than CPU nodes.

You can't use backtesting nodes for optimizations. The CPU nodes are available on a fair usage basis. The GPU nodes can be shared with a maximum of three members. Depending on the server load, you may use the all of the GPU's processing power.

Research Nodes

Research nodes enable you to use the Jupyter Research Environment. Several models of research nodes are available. More powerful research nodes allow you to handle more data and run faster computations in your notebooks. The following table shows the specifications of the research node models:

Name	Number of Cores	Processing Speed (GHz)	RAM (GB)	GPU
R1-4	1	2.4	4	0
R2-8	2	2.4	8	0
R4-12	4	2.4	12	0
R4-16-GPU	4	3	16	1/3
R8-16	8	2.4	16	0

Refer to the Pricing page to see the price of each research node model. You get one free R1-4 research node in your first organization, but the node is replaced when you subscribe to a new research node in the organization.

The CPU nodes are available on a fair usage basis. The GPU nodes can be shared with a maximum of three members. Depending on the server load, you may use all of the GPU's processing power.

Live Trading Nodes

Live trading nodes enable you to deploy live algorithms to our professionally-managed, co-located servers. Several models of live trading nodes are available. More powerful live trading nodes allow you to run algorithms with larger universes and give you more time for machine learning training. Each security subscription requires about 5MB of RAM. The following table shows the specifications of the live trading node models:

Name	Number of Cores	Processing Speed (GHz)	RAM (GB)	GPU
L-MICRO	1	2.6	0.5	0
L1-1	1	2.6	1	0
L1-2	1	2.6	2	0
L2-4	2	2.6	4	0

Refer to the Pricing page to see the price of each live trading node model.

GPU nodes perform best on repetitive and highly-parallel tasks like training machine learning models. It takes time to transfer the data to the GPU for computation, so if your algorithm doesn't train machine learning models, the extra time it takes to transfer the data can make it appear that GPU nodes run slower than CPU nodes.

Sharing Resources

Your organization's nodes are shared among all of the organization's members to reduce the amount of time that nodes idle. In the Algorithm Lab, you can see which nodes are available within your organization. By default, the best-performing resource is selected when you run a backtest or launch a research notebook, but you can select a specific resource to use.

Node Quotas

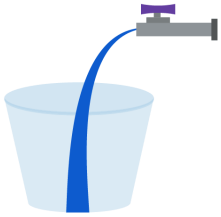
The following table shows the number of nodes each organization tier can have:

Tier	Backtest	Research	Live Trading
Free	1	1	0
Quant Researcher	2	1	2
Team	10	10	10
Trading Firm	Inf.	Inf.	Inf.
Institution	Inf.	Inf.	Inf.

Training Quotas

Algorithms normally must return from the `OnData` method within 10 minutes, but the `Train` method lets you increase this amount of time. Training resources are allocated with a leaky bucket algorithm where you can use a maximum of n-minutes in a single training session and the number of minutes available refills over time. This gives you a reservoir of training time when you need it and recharges the reservoir to prepare for the next training session. The reservoir only starts draining after you exceed the standard 10 minutes of training time.

The following animation demonstrates the leaky bucket algorithm. The tap continuously adds water to the bucket. When the bucket is full, water spills over the rim of the bucket. The water represents your training resources. When your algorithm exceeds the 10 minutes of training time, holes open at the bottom of the bucket and water begins to drain out. When your algorithm stops training, the holes close and the bucket fills up with water.



The following table shows the amount of extra time that each backtesting and live trading node can spend training machine learning models:

Model	Capacity (min)	Refill Rate (min/day)
B-MICRO	20	1
B2-8	30	5
B4-12	60	10
B8-16	90	15
L-MICRO	30	5
L1-1	60	10
L1-2	90	15
L1-4	120	20

The refill rate in the table above is based on the real-world clock time, not the backtest clock time. In backtests, the `Train` method is synchronous, so it will block your algorithm from executing while the model is trained. In live trading, the method runs asynchronously, so ensure your model is ready to use before you continue executing the algorithm. Training occurs on a separate thread, so use a semaphore to track the model state.

Log Quotas

Per our Terms and Conditions, you may not use the logs to export dataset information. The following table shows the amount of logs each organization tier can produce:

Tier	Logs Per Backtest	Logs Per Day
Free	10KB	3MB
Quant Researcher	100KB	3MB
Team	1MB	10MB
Trading Firm	5MB	50MB
Institution	Inf.	Inf.

If you delete a backtest or project that produced logs, your quotas aren't restored. Additionally, daily log quotas aren't fully restored at midnight. They are restored according to a 24-hour following window.

The log files of each live trading project can store up to 100,000 lines for up to one year. If you log more than 100,000 lines or some lines become older than one year, we remove the oldest lines in the files so your project stays within the quota.

To avoid reaching the limits, we recommend logging sparsely, focusing on the change events instead of logging every time loop. You can use the debugger to inspect objects during runtime. If you use the debugger, you should rarely reach the log limits.

Coding Session Quotas

If you have a project open, it uses a coding session. Paid organizations can have multiple active coding sessions, but free users can only have one coding session open at a time. The following table shows how many active coding sessions you can have on each organization tier:

Tier	Initial Coding Session Quota
Quant Researcher	2
Team	4
Trading Firm	8
Institution	16

If the organization you're in has more live trading nodes than your initial coding session quota, then your coding session quota increases to the number of live trading nodes you have in the organization so you can view all your live strategies.

The quota for free organizations is a global quota, so you can have one active coding session across all of your free organizations. The quotas for paid organizations are at the organization level. Therefore, if you are in two Quant Researcher organizations, you can have two active coding sessions in one of those organizations and another two active sessions in the other organization. These paid tier quotas are for each account, not for the organization as a whole. For instance, a Trading Firm organization can have more than eight members and all of the members can simultaneously work on projects within the organization.

File Size Quotas

The maximum file size you can have in a project depends on your organization's tier. The following table shows the quota of each tier:

Tier	Max File Size (KB)
Free	32
Quant Researcher	64
Team	128
Trading Firm	256
Institution	256

Live Trading Notification Quotas

The number of Telegram, email, or webhook notifications you can send in each live algorithm for free depends on the teir of your organization. The following table shows the hourly quotas:

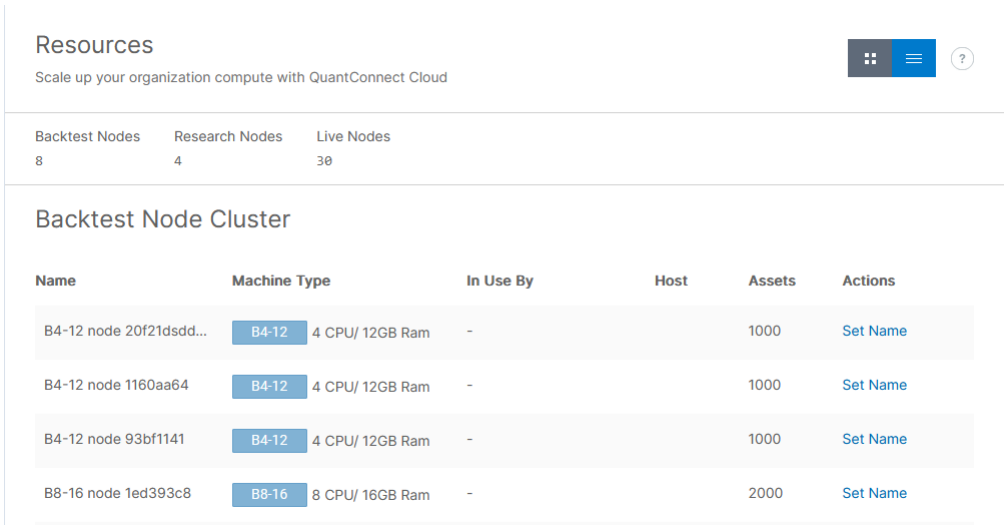
Teir	Number of Notifications Per Hour
Free	N/A
Quant Researcher	20
Team	60
Trading Firm	240
Institution	3,600

If you exceed the hourly quota, each additional Telegram, email, or webhook notification costs 1 QuantConnect Credit (QCC).

Each SMS notification you send to a US or Canadian phone number costs 1 QCC. Each SMS notification you send to an international phone number costs 10 QCC.

View All Nodes

The Resources page displays your backtesting, research, and live trading node clusters. To view the page, log in to the Algorithm Lab and then, in the left navigation bar, click Organization > Resources .



To toggle the format of the page, click the buttons in the top-right. If the page is in table view, each cluster section includes a table with the following columns:

Column	Description
Name	Name of the node
Machine Type	The node model and specifications
In Use By	The name of the member using the node
Host	The live trading server name
Assets	The recommended maximum number of assets to avoid RAM errors.
Actions	A list of possible actions

Add Nodes

You need billing permissions in the organization to add nodes.

Follow these steps to add nodes to your organization:

1. Open the Resources page.
2. Click Add nodeType Node for the type of node you want to add.
3. Select the node specifications.
4. Click Add Node .

The Resources page displays the new node.

Remove Nodes

You need billing permissions in the organization to remove nodes. If you remove nodes during your billing period, your organization will receive a pro-rated credit on your account, which is applied to future invoices.

Follow these steps to remove nodes from your organization:

1. Log in to the Algorithm Lab.
2. In the left navigation bar, click Organization > Home .
3. On the organization homepage, click Edit Plan .
4. Click the Customize Plan > Build Your Own Pack > Compute Nodes tab.
5. Click the minus sign next to the node model you want to remove.
6. Click Proceed to Checkout .

Rename Nodes

We assign a default name to hardware nodes that includes the model name and an arbitrary string of characters, but you can follow these steps to rename the nodes in your organization:

1. Open the Resources page.
2. Click Set Name on the node that you want to rename.
3. Enter the new node name and then click Save Changes .

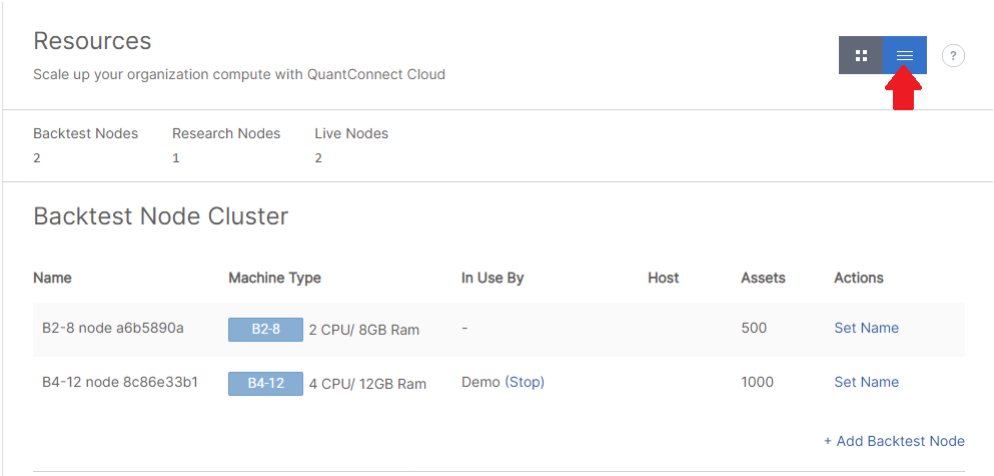
The Resources page displays the new node name.

Stop Nodes

You need stop node permissions in the organization to stop nodes other members are using. If you stop a node, it terminates the running backtest, research, or live trading sessions. When you stop a live trading node, the portfolio holdings don't change but the algorithm stops executing.

Follow these steps to stop nodes that are running in your organization:

1. Open the Resources page.
2. Click the icon with the three horizontal lines icon in the top-right corner to format the page into table view.



3. Click Stop in the row with the node that you want to stop.

4.4 Data Storage

Introduction

The ObjectStore is an organization-specific key-value storage location to save and retrieve data in QuantConnect's cache. Similar to a dictionary or hash table, a key-value store is a storage system that saves and retrieves objects by using keys. A key is a unique string that is associated with a single record in the key-value store and a value is an object being stored. Some common use cases of the ObjectStore include the following:

- Transporting data between the backtesting environment and the research environment.
- Training machine learning models in the research environment before deploying them to live trading.

The ObjectStore is shared across the entire organization. Using the same key, you can access data across all projects in an organization.

Supported Types

The ObjectStore has helper methods to store strings, JSON objects, XML objects, and bytes.

The ObjectStore has helper methods to store strings and bytes.

```
ObjectStore.Save(stringKey, stringValue);
ObjectStore.SaveJson<T>(jsonKey, jsonValue);
ObjectStore.SaveXml<T>(xmlKey, xmlValue);
ObjectStore.SaveBytes(bytesKey, bytesValue);
```

```
self.ObjectStore.Save(string_key, string_value)
self.ObjectStore.SaveBytes(bytes_key, bytes_value)
```

To store an object that is in a different format, you need to encode it to one of the supported data types. For instance, if you train a machine learning model and it is in binary format, encode it into base 64 before saving it.

The ObjectStore also has helper methods to retrieve the stored objects.

```
var stringValue = ObjectStore.Read(stringKey);
var jsonValue = ObjectStore.SaveJson<T>(jsonKey);
var xmlValue = ObjectStore.SaveXml<T>(xmlKey);
var bytesValue = ObjectStore.SaveBytes(bytesKey);
```

```
string_value = self.ObjectStore.Read(string_key)
bytes_value = self.ObjectStore.ReadBytes(bytes_key)
```

For complete examples of using the ObjectStore, see Object Store .

Storage Sizes

All organizations get 50 MB of free storage in the ObjectStore. Paid organizations can subscribe to more storage space. The following table shows the cost of the supported storage sizes:

Storage Size (GB)	Monthly Cost ($)
0.05	0
2	10
5	20
10	50
50	100

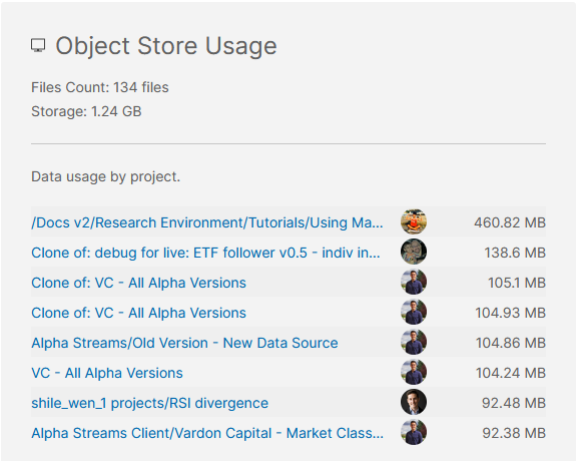
Research to Live Considerations

When you deploy a live algorithm, you can access the data within minutes of modifying the ObjectStore. Ensure your algorithm is able to handle a changing dataset.

The live environment's access to the ObjectStore is much slower than in research and backtesting. Limit the individual objects to less than 50 MB to prevent live trading access issues.

Monitor Usage

The Resources page shows the total storage used in your organization and the storage used by individual projects so that you can easily manage your storage space. To view the page, log in to the Algorithm Lab and then, in the left navigation bar, click Organization > Resources.



Edit Storage Plan

You need storage billing permissions and a paid organization to edit the size of the organization's ObjectStore.

Follow these steps to edit the amount of storage available in your organization's ObjectStore:

1. Log in to the Algorithm Lab.
2. In the left navigation bar, click Organization > Resources .
3. On the Resources page, scroll down to the Storage Resources and then click Add Object Store Capacity .
4. On the Pricing page, select a storage plan.
5. Click Proceed to Checkout .

Delete Storage

To free up storage space, delete the key-value pairs in the ObjectStore by calling the `Delete` method with a key.

```
ObjectStore.Delete(key);
```

```
self.ObjectStore.Delete(key)
```

4.5 Support

Introduction

The community is a great resource for support for developing algorithms. However, for more personalized assistance and privacy, you can submit support tickets to request assistance from a QuantConnect engineer. Premium support allows you to share your algorithms with IP protection and enables the Support Team to address issues with live algorithms. In order to submit a support ticket, you must have a support seat in your organization.

There are three tiers of support seats and each tier provides different services. You can file support tickets to get private assistance with issues, but support tickets should not replace your efforts of performing your own research and reading through the documentation. If you need further assistance than what our Support Team offers, consider hiring an Integration Partner .

Features

The services our Support Team provides depend on the tier of your support seat.

Bug Reports

The Lean trading engine is under constant development, so you may occasionally experience bugs when using it. If you think you have found a bug, share a simple backtest with us that reproduces the issue. You can contact us either through email , Discord , or the forum . We will review your submission. If we confirm you've found a bug, we will create a GitHub Issue to have it resolved. Subscribe to the GitHub Issue to track our progress in fixing the bug.

Feature	Community	Bronze	Silver	Gold
Bug Reports	✓	✓	✓	✓

Email Support

Our support ticket system enables you to privately email with our Support Team. We address support tickets in a first-in, first-out order, but we give priority to tickets opened by members with higher support seats. The following table shows our response time for each of the support tiers:

Tier	Response Time (hours)
Gold	24
Silver	48
Bronze	72 or Best Effort

Feature	Community	Bronze	Silver	Gold
Email Support	-	✓	✓	✓

IP Protection

If you attach a backtest or live trading deployment when you open a support ticket , the intellectual property of your project is protected. We have a restricted subset of the Support Team who can access private support tickets. Using paid support plans ensures only a limited subset of the QuantConnect team can access the algorithms you attach to support tickets. These team members are carefully selected, have been at QuantConnect for at least 2 years, and have passed a background check. In contrast, if you share a backtest or live trading deployment to the forum for assistance, your project becomes part of the public domain.

Feature	Community	Bronze	Silver	Gold
IP Protection	-	✓	✓	✓

Live Trading Debugging

Paid support plans have access to our live deployment debugging service. If you experience an issue with a live trading deployment, open a support ticket. We can assist with uncovering the issue, fixing the issue, and getting you ready to redeploy the algorithm. We can't assist with live trading issues in the community forum or Discord.

Feature	Community	Bronze	Silver	Gold
Live Trading Debugging	-	✓	✓	✓

Algorithm Design Suggestions

If you have a silver or gold support seat, we can offer suggestions on the design of your algorithms. Our Support Team members are experts on the inner workings of Lean, so we can guide you on improving the efficiency of your algorithms by following our common design patterns. We can usually reduce the size of your project's code files and increase the speed of your project backtesting.

Feature	Community	Bronze	Silver	Gold
Algorithm Design Suggestions	-	-	✓	✓

Private Chatroom

When you require instant access to our Support team, we can open a private chatroom in Discord. In our private chatroom, you can ask our Support Team questions at any time and we focus on responding as quickly as we can. Request a private chatroom to avoid waiting for email responses on your support tickets.

Feature	Community	Bronze	Silver	Gold
Private Chatroom	-	-	-	✓

Phone Call Consultations

We offer phone support to members with gold support seats. You can take advantage of our phone support for up to 1 hour per month. During phone calls, feel free to ask about anything related to QuantConnect, Lean, or quant trading. We recommend planning your questions before you call in order to best utilize the time available.

Feature	Community	Bronze	Silver	Gold
Phone Call Consultations	-	-	-	✓

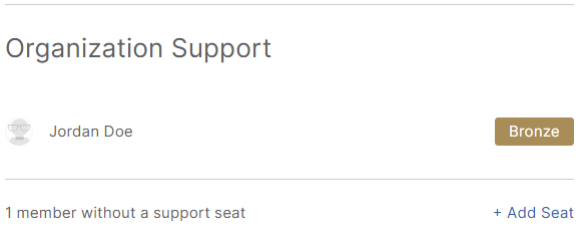
Summary

The following table shows the features available in each tier:

Feature	Community	Bronze	Silver	Gold
Bug Reports	✓	✓	✓	✓
Email Support	-	✓	✓	✓
IP Protection	-	✓	✓	✓
Live Trading Debugging	-	✓	✓	✓
Algorithm Design Suggestions	-	-	✓	✓
Private Chatroom	-	-	-	✓
Phone Call Consultations	-	-	-	✓

View All Seats

The Team Management page displays the support seat assignments within your organization. To view the page, log in to the Algorithm lab, and then in the left navigation bar, click Organization > Members .



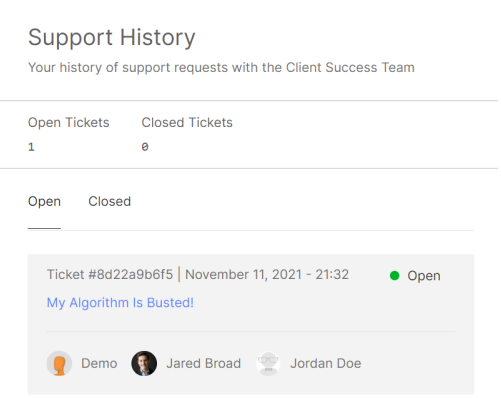
Add Seats

Follow these steps to assign support seats to members in your organization:

1. Open the Team Management page.
2. Scroll down to the Organization Support section and then click Add Seat .
3. On the Support page, click Select under the tier of seat you want to assign.
4. In the Assign seatTier Seat section, click the team member field and then select the team member you want to assign the seat to from the drop-down menu.
5. Click Add Seat +$ price .

View All Tickets

The Support History page displays all of your organization's support tickets. To view the page, log in to the Algorithm Lab and then, in the left navigation bar, click Support > Organization Tickets .



To see all closed tickets, click Closed . To view the conversation history with our Support Team regarding a ticket, click a ticket.

Open New Tickets

Follow these steps to open support tickets:

1. Log in to the Algorithm Lab.
2. In the left navigation bar, click Support .
3. On the Support page, enter a subject and message.
4. If you want to attach a live deployment or backtest to the support ticket, follow these steps:
    1. Click Add Project , select a project, and then click OK .
    2. (Optional) Click Add Live Deployment , select a live deployment, and then click OK .
    3. (Optional) Click Add Backtest , select a backtest, and then click OK .
5. Click the By submitting this request, I give QuantConnect Support Staff permission to view my project check box.
6. Click Send .

Comment on Tickets

Follow these steps to comment on support tickets:

1. Open the Support History page.
2. If the ticket you want to comment on is closed, click Closed .
3. Click the ticket on which you want to comment.
4. Enter your comment and then click Send .

Close Tickets

Follow these steps to close support tickets:

1. Open the Support History page.
2. Click the ticket you want to close.
3. (Optional) Enter a reason for closing the ticket.
4. Click Send and Close Ticket .

Open Closed Tickets

Follow these steps to open closed support tickets:

1. Open the Support History page.
2. Click Closed .
3. Click the ticket that you want to open.
4. (Optional) Enter a reason for opening the ticket.
5. Click Open Ticket and Send .

Ticket Quotas

The following table shows the number of tickets each seat tier can have open at one time:

Tier	Open Ticket Quota
Gold	16
Silver	8
Bronze	4

4.6 Members

Introduction

Organizations consist of members and members can be part of multiple organizations. The number of members your organization can have depends on the tier of your organization. All members in your organization can access the resources within the organization. If you are the manager of a Trading Firm or Institution organization, you can grant additional permissions to your team members.

View All Seats

You need a seat for each member in your organization.

To see the number of team seats you have in your organization, open the Billing page and then scroll down to the Products section.

Add Seats

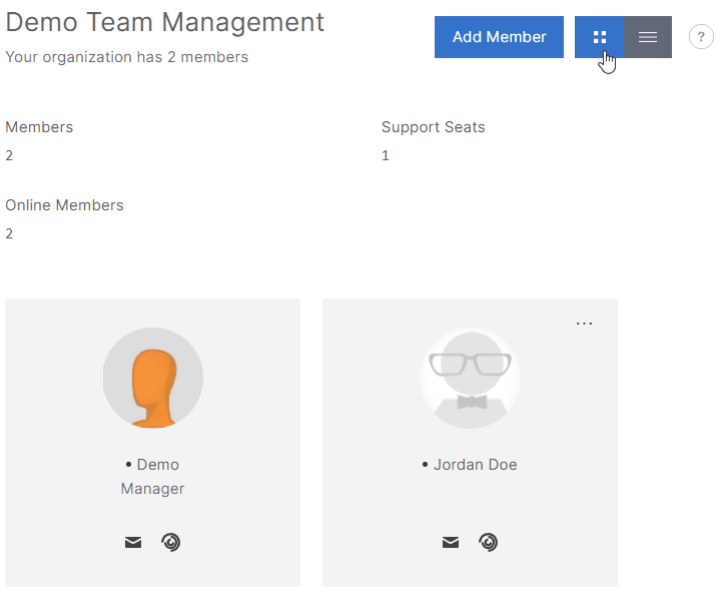
You need billing permissions within an organization to add team seats.

Follow these steps to add team seats to your organization:

1. Open the organization homepage .
2. Click Edit Plan .
3. Click the Customize Plan > Build Your Own Pack > Organization Seats tab.
4. In the organizationTier Seats section, click the plus icon.
5. Click Proceed to Checkout .

View All Members

The Team Management page displays all of the members in your organization. To view the page, log in to the Algorithm Lab and then, in the left navigation bar, click Organization > Members .



To toggle the format of the page, click the buttons in the top-right.

Membership Quotas

The number of members your organization can have depends on the organization's tier . In general, higher organization tiers can have more members within the organization to share resources. This design enables you to upgrade your organization as your trading business grows over time. The following table shows the number of members each organization tier can have:

Tier	Minimum Members	Maximum Members
Free	1	1
Quant Researcher	1	1
Team	2	10
Trading Firm	2	Unlimited
Institution	2	Unlimited

Add Members

You need vacant team seats and team add permissions within an organization to add team members.

Follow these steps to add team members to your organization:

1. Open the Team Management page.
2. Click Add Member .
3. Enter the email address of the new team member and then click Add Member .

   The email you enter should be the email the new member uses to log in to their QuantConnect account.

4. If you have team edit permissions, select the permissions for the new member and then click Save Changes . Otherwise, click Cancel .

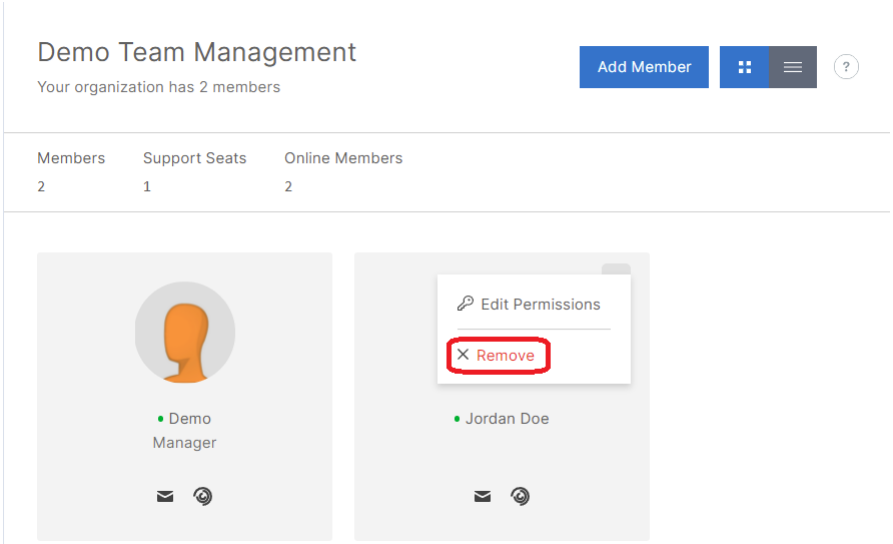
   The Team Management page displays and the new member is included.

Remove Members

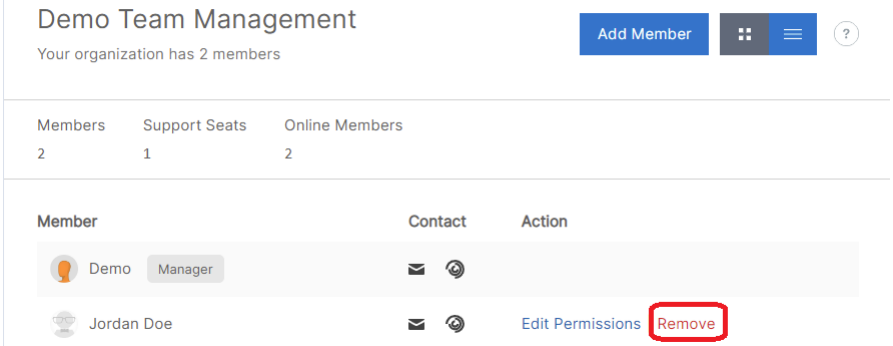
You need team removal permission to remove members from your organization. Before removing members, stop all nodes they are using. Removing members is not reversible and the members will lose access to the organization's projects. On the Team tier, projects that the members created in the organization will remain with the members. On the Trading Firm and Institution tiers, projects that the members created in the organization will be transferred to the organization manager.

Follow these steps to remove members from your organization:

1. Open the Team Management page.
2. If the Team Management page is in tile view, follow these steps:
    1. Click the three dots icon in the top-right corner of the member you want to remove and then click Remove .



2. Click Remove .
3. If the Team Management page is in table view, follow these steps:
   1. Click Remove next to the member you want to remove from the organization.



   2. Click Remove .

Permissions

Each member of your organization has access to the resources within the organization. If you are the manager of a Trading Firm or Institution organization, you can grant additional permissions to your team members. There are several categories of permissions.

Billing Permissions

The following table shows the supported billing permissions:

Permission	Description
Update	Permission to change the organization's subscriptions and billing details.

Stop Node Permissions

The following table shows the supported node permissions:

Permission	Description
Backtest	Permission to stop backtesting nodes .
Research	Permission to stop research nodes .
Live	Permission to stop live trading nodes .

To stop active nodes, see Stop Nodes .

Team Permissions

The following table shows the supported team permissions:

Permission	Description
Add	Permission to add new members .
Remove	Permission to remove existing members .
Edit	Permission to change the permissions of other members .

Storage Permissions

The following table shows the supported team permissions:

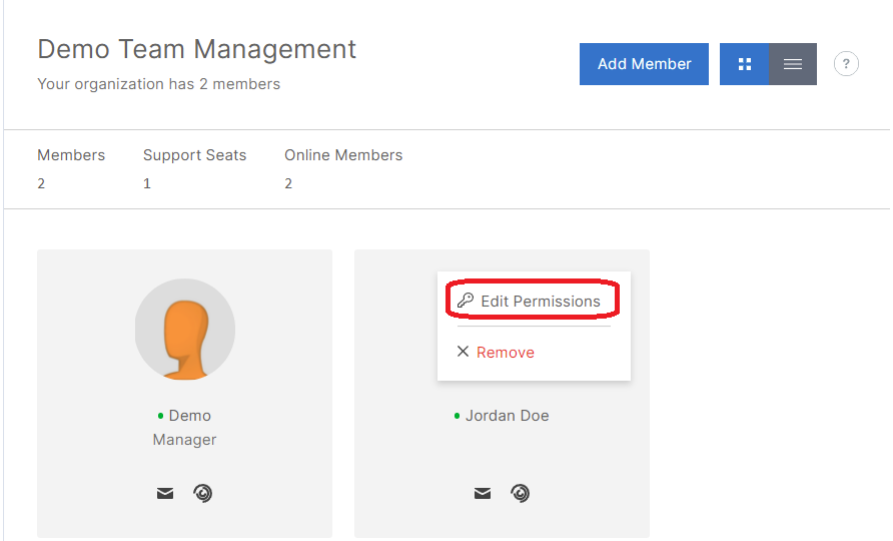
Permission	Description
Create	Permission to write to the Object Store .
Delete	Permission to delete data in the Object Store .
Billing	Permission to subscribe to more space in the Object Store .

Edit Permissions

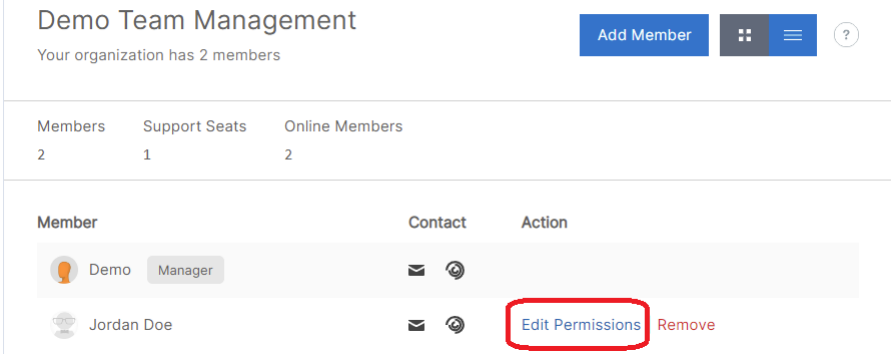
You need team edit permission to edit the permissions of team members.

Follow these steps to edit team permissions:

1. Open the Team Management page.
2. If the Team Management page is in tile view, follow these steps:
    1. On the Team Management page, click the three dots in the top-right corner of the member you want to edit and then click Edit Permissions .



    2. Select and deselect permissions as desired and then click Save Changes .
3. If the Team Management page is in table view, follow these steps:
    1. On the Team Management page, click Edit Permissions next to the member whose permissions you want to edit.



    2. Select and deselect permissions as desired and then click Save Changes .

4.7 Administration

Introduction

You can view and manage your organizations from the Algorithm Lab. The algorithms you store in the Algorithm Lab are secure and you maintain their intellectual property.

Intellectual Property

All individuals on QuantConnect own their intellectual property (IP) on our platform. Your code is private and only accessible by people you share the project with and with support-engineers when you submit a support ticket. At no point does QuantConnect ever claim ownership of user IP. The only case where algorithm code becomes public domain is when they are shared to the forum. In this case, algorithms need to be made public domain to allow the sharing of the algorithm code.

It is common when companies hire engineers to write software, they require their employees to sign an agreement that gives the company IP ownership of any code written for work. They need this because they're paying you to write software, and the company needs to sell that software to turn a profit. Similarly, the Organizations feature allows you to control who holds IP ownership over a project. Each type of organization has its own mechanisms for handling project IP ownership.

Individual Organizations

The Free and the Quant Researcher tiers only allow single-member organizations. This means you can't collaborate with anyone else inside the QuantConnect platform. Simply put, you own the IP for any projects you work on since you are the sole collaborator.

Team Organizations

For organizations that allow multiple users to collaborate on projects, the user who created the project owns it; this can be you or one of your teammates. If you add a teammate/collaborate, they can clone it, but the original project belongs to the person who first created it.

Trading Firm & Institution Organizations

For Trading Firm and Institution organizations, which are generally used by companies and funds, the firm owns all employee projects. This is made to suit firms that wish to hire consultants and need to ensure the code remains with the company when the consultant work is finished. You have to explicitly create a project in an organization for it to be created on the organization's account.

Corporate Branding

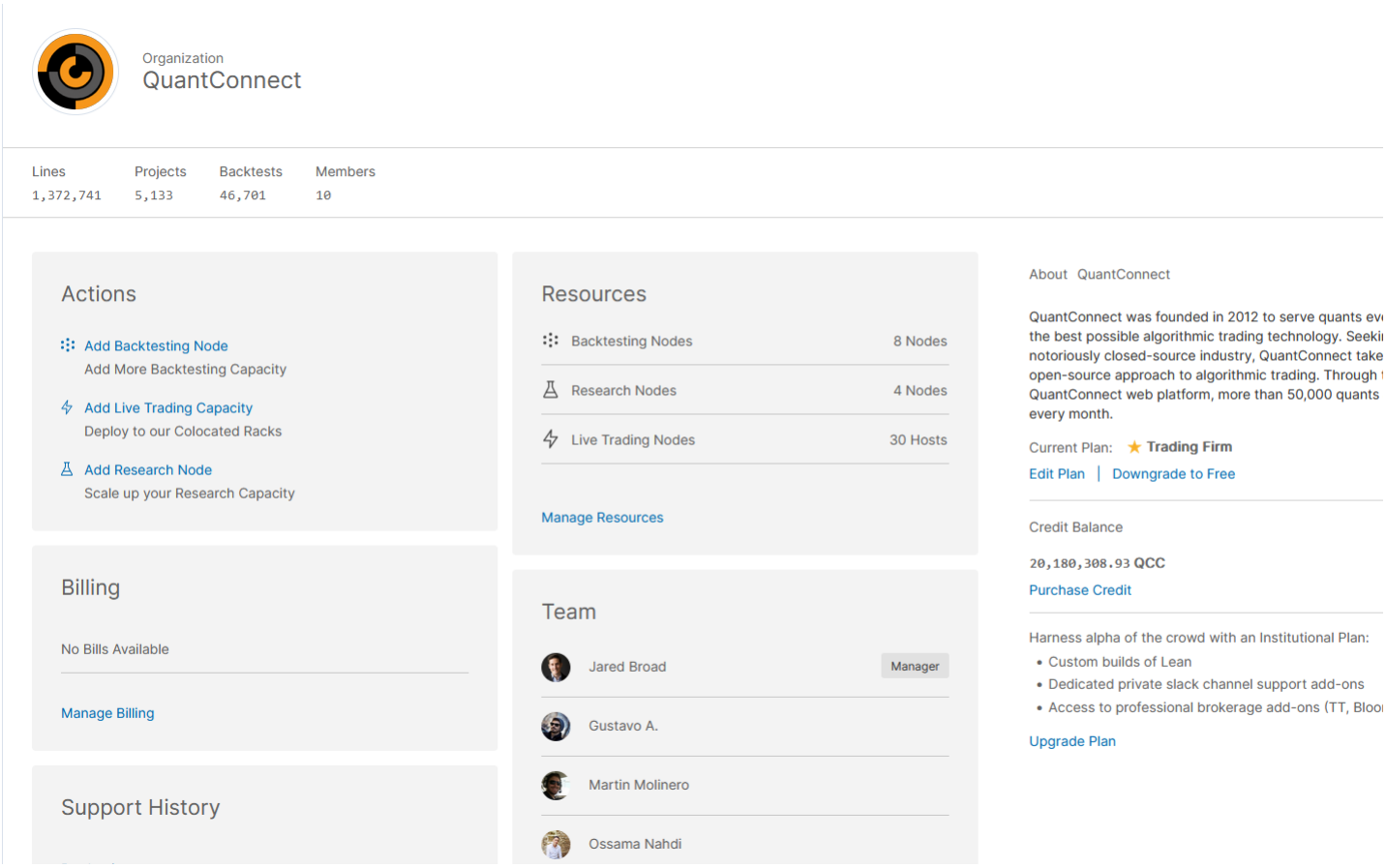
You can customize your organizationâ€™s image, name, and description in the Algorithm Lab to match your branding. If you have a Trading Firm or Institution organization, you can integrate the Algorithm Lab into your website so that your company logo is in the navigation bar and the color matches your website's theme.

Migrating Projects

If you are the administrator of an organization, you can migrate a project out of the organization to another organization in which youâ€™re a member. When your project is migrated, the project files are copied but the content stored in the ObjectStore is not retained.

View the Organization Homepage

The organization homepage displays a summary of your organization. To view the page, log in to the Algorithm Lab and then, in the left navigation bar, click Organization > Home .



The organization homepage displays your organization's brand and statistics at the top of the page. The following table describes the remaining sections of the page:

Section	Description
Actions	Add nodes
Resources	View and manage nodes
Billing	View and manage bills
Team	View and manage team members
Support History	View support tickets
Plan	View and change the organization tier
Credit Balance	View and purchase QuantConnect Credit

Edit the Organization Branding

You can edit your organization's image and name.

Image

Follow these steps to change your organization image:

1. Open the organization homepage .
2. Click the organization image.
3. Click Choose file , select a file from your device, and then click Open .

   Your organization image must be in gif , jpg , or png format and less than 1MB in size.

4. Click Save .

   "Photo Uploaded" displays.

Name

Follow these steps to change your organization name:

1. Open the organization homepage .
2. Hover over the organization name and then click the pencil icon that appears.

3. Enter the new organization name and then click Save Changes .

   "Organization Name Updated Successfully" displays.

View All Organizations

To view all of the organizations for which you're a member, log in to the Algorithm lab and then, in the top navigation bar, click Connected as: organizationName .

Organization	Type	Owner	Members	Preferred
✓ Demo	Team	You	2	◉
MyOrganization	Free	You	1	○

Add Organizations

Follow these steps to add new organizations to your profile:

1. Log in to the Algorithm Lab.
2. In the top navigation bar, click Connected as: organizationName .
3. In the Switch Organization panel, click Create Organization .
4. Enter the organization name and then click Add .

   The organization name must be unique. "Created Successfully" displays.

Switch Organizations

Follow these steps to switch organizations:

1. Log in to the Algorithm Lab.
2. In the top navigation bar, click Connected as: organizationName .
3. In the Switch Organization panel, click the name of the organization for which you want to connect.

   The top navigation bar displays the new organization name.

Set a Preferred Organization

Follow these steps to set your preferred organization:

1. Log in to the Algorithm Lab.
2. In the top navigation bar, click Connected as: organizationName .
3. In the Switch Organization panel, select the radio button under the Preferred column that corresponds to the organization that you want to set as the preferred organization.

   "Preferred organization selected" displays. Refresh the page to connect as your preferred organization.

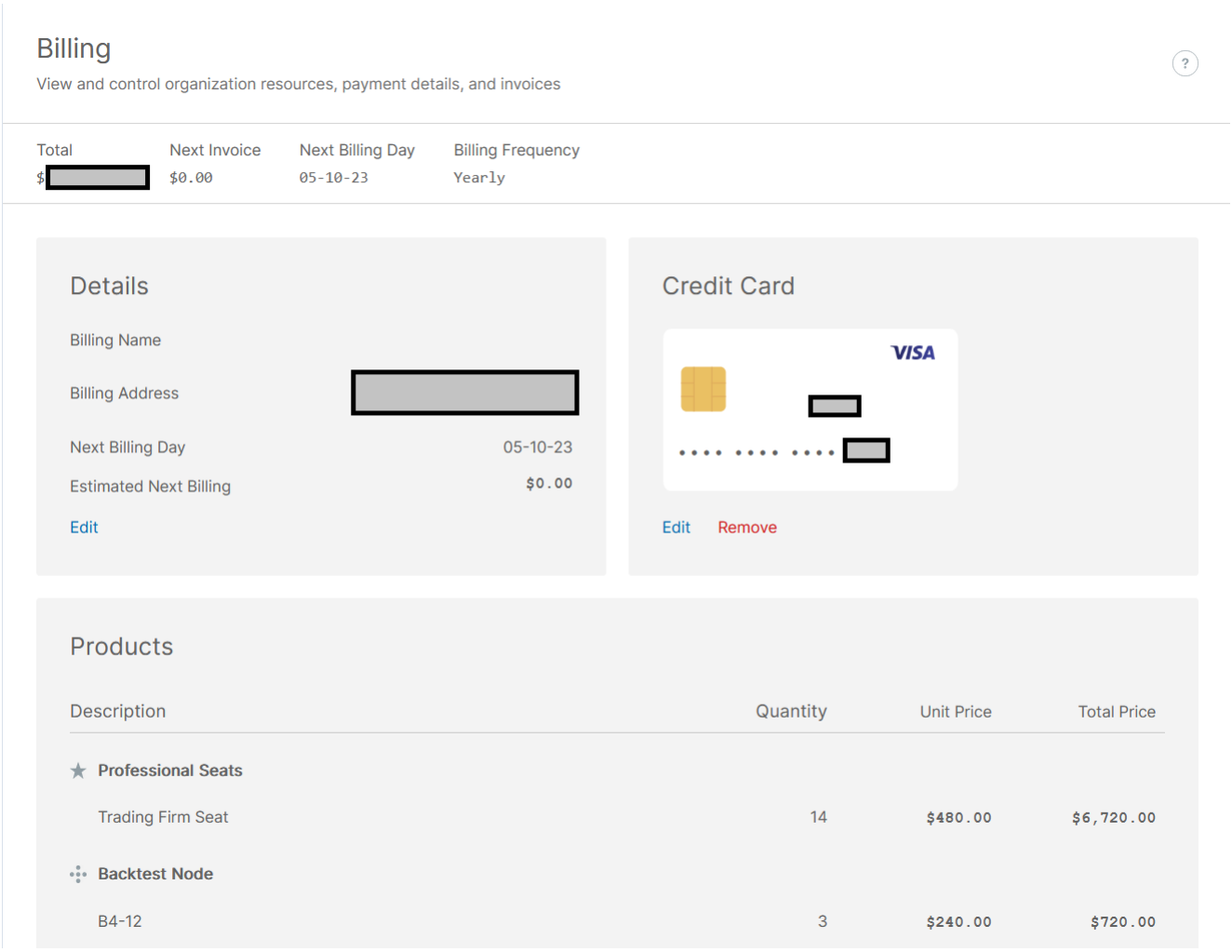
4.8 Billing

Introduction

The owner of an organization is responsible for the billing, but the responsibility can be delegated in Trading Firm or Institution organizations. Your organization's billing information is never saved by QuantConnect. Itâ€™s passed to the Stripe billing system. If you cancel your subscription, your live trading nodes stop running. So, for user safety, your subscriptions automatically renew each month.

View Billing Information

The Billing page displays your organization's billing details. To view the page, log into the Algorithm Lab and then, in the left navigation bar, click Organization > Billing .



The Billing page displays the billing cost, date, and frequency at the top of the page. The following table describes the remaining sections of the page:

Section	Description
Details	The billing name and address associated with the credit card.
Credit Card	The credit card used to pay the bill.
Products	A breakdown of the organization's subscriptions.
Invoices	All the organization's invoices.
Organization Credit (QCC)	All QCC purchases and expenses.

Edit Billing Details

Follow these steps to edit your organization's billing details:

1. Open the Billing page.
2. In the Details section, click Edit .
3. Enter the new billing details and then click Save .

   The Details section displays the new billing details.

Edit the Credit Card

You can add, replace, and remove the organization's credit card.

Add

Follow these steps to add a credit card:

1. Open the Billing page.
2. Click Add Card .
3. Enter the credit card details and then click Save .

   The Credit Card section displays the last 4 digits of your credit card.

Replace

Follow these steps to change the credit card:

1. Open the Billing page.
2. In the Credit Card section, click Edit .
3. Enter the new credit card details and then click Save .

   The Credit Card section displays the last 4 digits of your new credit card.

Remove

Follow these steps to remove the credit card:

1. Open the Billing page.
2. Click Remove .
3. Click OK .

   The Credit Card section displays "No entries found".

Download Invoices and Receipts

Follow these steps to download invoices and receipts:

1. Open the Billing page.
2. Scroll down to the Invoices section and then click Download as PDF next to the invoice or receipt that you want to download.
3. Click Download invoice or Download receipt .

   A prompt to download the file to your local machine displays.

Change Organization Tiers

You can change your organization to any of the paid tiers or the Free tier.

Paid Tiers

Follow these steps to change to a paid organization tier:

1. Log in to the Algorithm Lab.
2. In the left navigation bar, click Organization > Home .
3. On the organization homepage, click Edit Plan .
4. Click the Choose a Plan tab.
5. Click CHOOSE TIER under the organization tier you want.
6. Select a tier pack.

   The following table describes the type of packs we have available:

Pack Type	Description
Suggested Packs	Packs with pre-selected team seats, support seats, and nodes.
Build Your Own Pack	Packs with custom selections for team seats, support seats, add-ons, and market subscriptions.

7. Select monthly or annual billing.
8. (Optional) Click + Add Coupon and then enter your coupon code.
9. If your organization doesn't have a credit card added, click Proceed to Checkout and then enter your credit card details.
10. Click Update Subscriptions or Subscribe Now .

Free Tier

Follow these steps to downgrade to the Free tier:

1. Log in to the Algorithm Lab.
2. In the left navigation bar, click Organization > Home .
3. On the organization homepage, click Downgrade to Free .
4. Select the I understand that my subscription will be terminated immediately check box and then click Continue .
5. Select the reason for downgrading and then click Downgrade .

4.9 Credit

Introduction

We created QuantConnect Credit (QCC) to enable micropayments on QuantConnect. You can use QCC to optimize parameters, download datasets, gift to members in the forum, and apply to your organization's monthly invoice. You can purchase QCC in the Algorithm Lab at a rate of 1 QCC = $0.01 USD. Since QCC is owned by organizations instead of members, all of the members within your organization have the ability to spend the QCC balance.

Optimizing Parameters

You need QCC in your organization to unlock parameter optimization. Parameter optimization jobs use optimization nodes, which are rented on a time basis. Therefore, the longer it takes to run all of the backtests in your optimization job, the more QCC it costs. Before you run optimizations, we estimate on how much QCC it will cost to run the job, but the final cost can differ from our estimates because we can't know exactly how long all of the backtests will take to run ahead of time. For instructions on optimizing parameters, see Launch Optimization Jobs.

Downloading Datasets

You can spend some QCC to download datasets from the Dataset Market to your local machine. The cost of downloading depends on the dataset and it's calculated on a per-file or per-day basis. For instructions on downloading datasets, see Licensing.

Giving to Others

To show your appreciation for contributions in the forum, give some QuantConnect Credit (QCC) rewards. The following table shows the available QCC rewards:

Award	Description	Cost (QCC)
Silver Award	A simple token of recognition from one quant to another. Keep up the great work.	80
Gold Award	This is some great work! Gold star!	600
Platinum Award	Highly resistant to oxidation, this award is for those contributions which will stand the test of time. Strong, classic, and useful for most high technology products.	1,200
Medal for Excellence	The QuantConnect Medal for Excellence is awarded by a member of the QuantConnect staff for exceptional contributions to the QuantConnect community.	3,000
Plutonium Award	Nuclear hot! This post is incredible and deserves recognition as such. Show the author your appreciation for their work.	2,000
Docs Shakespeare	You've left a mark by a contribution to the documentation for the community. Your edits and examples will be followed for generations to come.	500
Nobel Laureate	Bestowed in recognition of quantitative advances.	80
Spaghetti Code Award	Following the intricate flows of code noodle, this code compiles and runs.	300
Jedi Quant	Quant promise you show. Your code channel the force into. Hmmmmmm.	200
Live Trader	That looks like a wild ride.	50
Today I Learned	Thank you for upgrading my brain.	50
Machine Unlearning	I plan on letting the computers do all the work for me.	150
Totally Overfit	I see parameters everywhere.	80
Mind Blown Award	Something incredibly amazing, mind-boggling, and you're shocked senseless.	80
Research Rembrandt	A Jupyter notebook work of art, pulling together all the right hues and plots to be a true masterpiece.	100
Cayman Island Award	Let's get a boat and start a fund together. Did I mention the boat?	800
Printing Money Award	Awarded to profitable algorithms.	100
Stronger Together Award	We're stronger together. Let's make this happen!	120
Appreciate the Support Award	Quant trading is a hard road, we could all use a hand.	250
	You're a master craftsman, taking raw materials and molding them into works of art for the good of the world.	600

Award	Description	Cost (QCC)
Master Craftsman		

Purchase QCC

Follow these steps to purchase QuantConnect Credit (QCC):

1. Open the Billing page.
2. Scroll down to the Organization Credit (QCC) section and then click Purchase Credit .
3. Select a credit pack and then click Continue .
4. If your preferred organization has a credit card and you want to charge that card, click Purchase .
5. If your preferred organization doesn't have a credit card, enter the credit card details and then click Purchase .

   Only the preferred organization can be charged when purchasing QCC. If you want to charge a different organization you are a member of, set it as your preferred organization .

Apply QCC to Invoices

You can apply QCC to your organization's invoices to pay for your subscriptions, but you must enable invoice payments before your invoice is generated to pay it with QCC.

Follow these steps to enable invoice payment with QCC:

1. Open the Billing page.
2. In the Organization Credit (QCC) section, select the Automatically apply excess monthly QCC credit to my QuantConnect invoices check box.

   "Credit Preferences Updated Successfully" displays.

4.10 Training

Introduction

Onboard new team members to your organization through the content in the Learning Center . The Learning Center enables you to systematically track and monitor the progress of your team members on the courses you purchase or create. If you purchase or create a course, you can access it from the Organization > Resources page in the Algorithm Lab.

Paid Courses

Currently, there are only free tutorials in the Learning Center, but we expect to expand the paid course offerings through 2023. We charge for paid courses based on the number of members in your organization. For example, if there are 5 team members in your organization, multiply the listed price of a course by 5. When you purchase a course, the team members in your organization have lifetime access to the course.

Private Courses

Private courses are not for sale in the Learning Center. You can upload private courses to your Organization > Resources page, but they will not be available for other organizations to purchase. Create private courses to help onboard your team members while using the familiar Learning Center environment. Since the Learning Center has built-in tools to help you monitor your team members, you can track the progress of your team members as they work on your internal private courses.

This feature is designed for Institutional clients with large teams of quants on QuantConnect

5 Learning Center

The Learning Center is a coding environment to learn quantitative trading using LEAN. The Learning Center features a collection of courses from educators from the QuantConnect team and the community. The goal of the Learning Center is to give you an understanding of robust algorithm design and the tools you need to implement your own trading strategies. As you work through the courses, youâ€™ll manage a portfolio, use indicators in technical trading strategies, trade on universes of assets, automate trades based on market behavior, and understand how data moves through your algorithm. After you've completed a course, you can keep the code to perform further research and deploy to live trading. Start learning today because you need to complete 30% of the Bootcamp lessons to post in the community forum.

Training

Get team members up to speed

Educators

Add courses to the Learning Center

Course Structure

Structured into digestible portions

See Also

Available Courses
Learn Programming
Algorithm Engine

5.1 Training

Introduction

The Learning Center is a coding environment to learn quantitative trading using LEAN. The Learning Center features a collection of courses from educators from the QuantConnect team and the community. The goal of the Learning Center is to give you an understanding of robust algorithm design and the tools you need to implement your own trading strategies. As you work through the courses, youâ€™ll manage a portfolio, use indicators in technical trading strategies, trade on universes of assets, automate trades based on market behavior, and understand how data moves through your algorithm. After you've completed a course, you can keep the code to perform further research and deploy to live trading. Start learning today because you need to complete 30% of the Bootcamp lessons to post in the community forum.

Onboard new team members to your organization through the content in the Learning Center. The Learning Center enables you to systematically track and monitor the progress of your team members on the courses that you purchase or create. If you purchase or create a course, you can access it from the Organization > Resources page in the Algorithm Lab.

Articles

QuantConnect maintains collections of related tutorials we call a Learning Series . We have tutorial series covering the topics below - each with a set of articles or tutorials:

Investment Strategy Library

The Strategy Library is a collection of tutorials written by the QuantConnect team and community members. Review these tutorials to learn about trading strategies found in the academic literature and how to implement them with QuantConnect/LEAN.

Introduction to Financial Python

Introduces basic Python functionality in the context of quantitative finance.

Introduction to Options

Introduces Options to those who are Option novices and have basic knowledge of applied mathematics, statistics, and financial markets.

Applied Options

Simple Options trading algorithms on QuantConnect for those who already have basic knowledge of Options markets.

Paid Courses

Currently, there are only free tutorials in the Learning Center, but we expect to expand the paid course offerings through 2023. We charge for paid courses based on the number of members in your organization. For example, if there are 5 team members in your organization, multiply the listed price of a course by 5. When you purchase a course, the team members in your organization have lifetime access to the course.

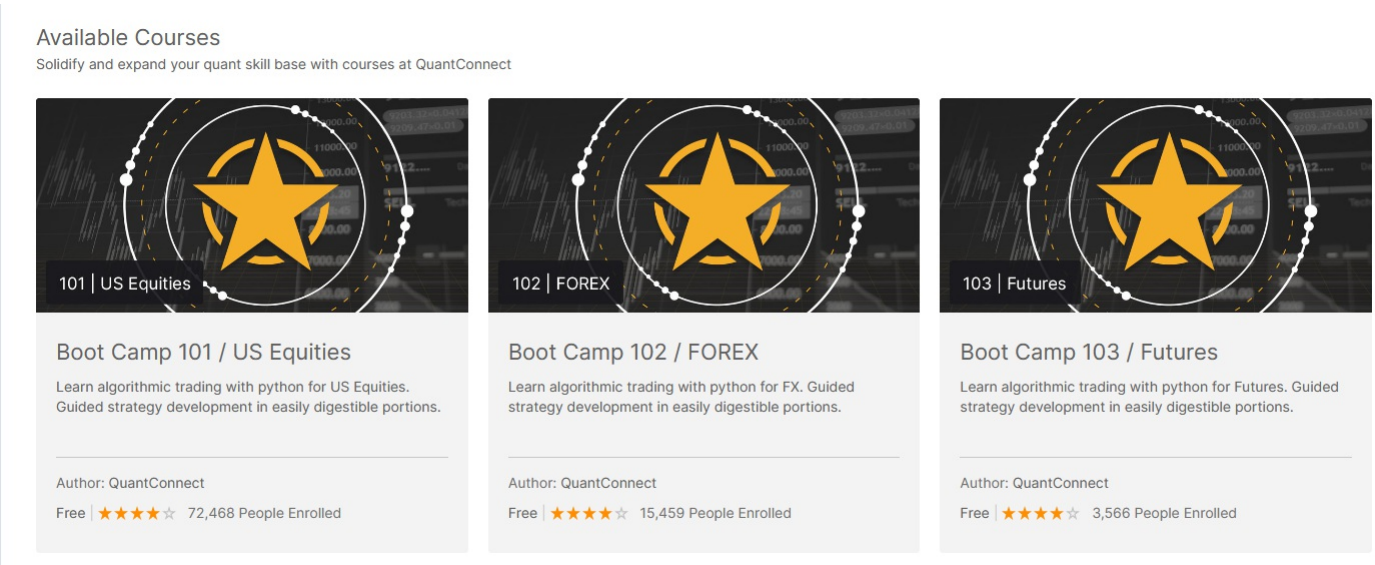
Private Courses

Private courses are not for sale in the Learning Center. You can upload private courses to your Organization > Resources page, but they will not be available for other organizations to purchase. Create private courses to help onboard your team members while using the familiar Learning Center environment. Since the Learning Center has built-in tools to help you monitor your team members, you can track the progress of your team members as they work on your internal private courses.

This feature is designed for Institutional clients with large teams of quants on QuantConnect

View All Courses

The Available Courses page displays all of the courses in the Learning Center. To view the page, open the Algorithm Lab and then, in the left navigation bar, click Learning Center > All Courses .



Each course displays the following information:

- Name
- Description
- Author
- Price
- Review summary
- The number of students

Click a course to learn more about it, including the following:

- Instructor biography
- Requirements
- Syllabus
- Reviews

Enroll in Courses

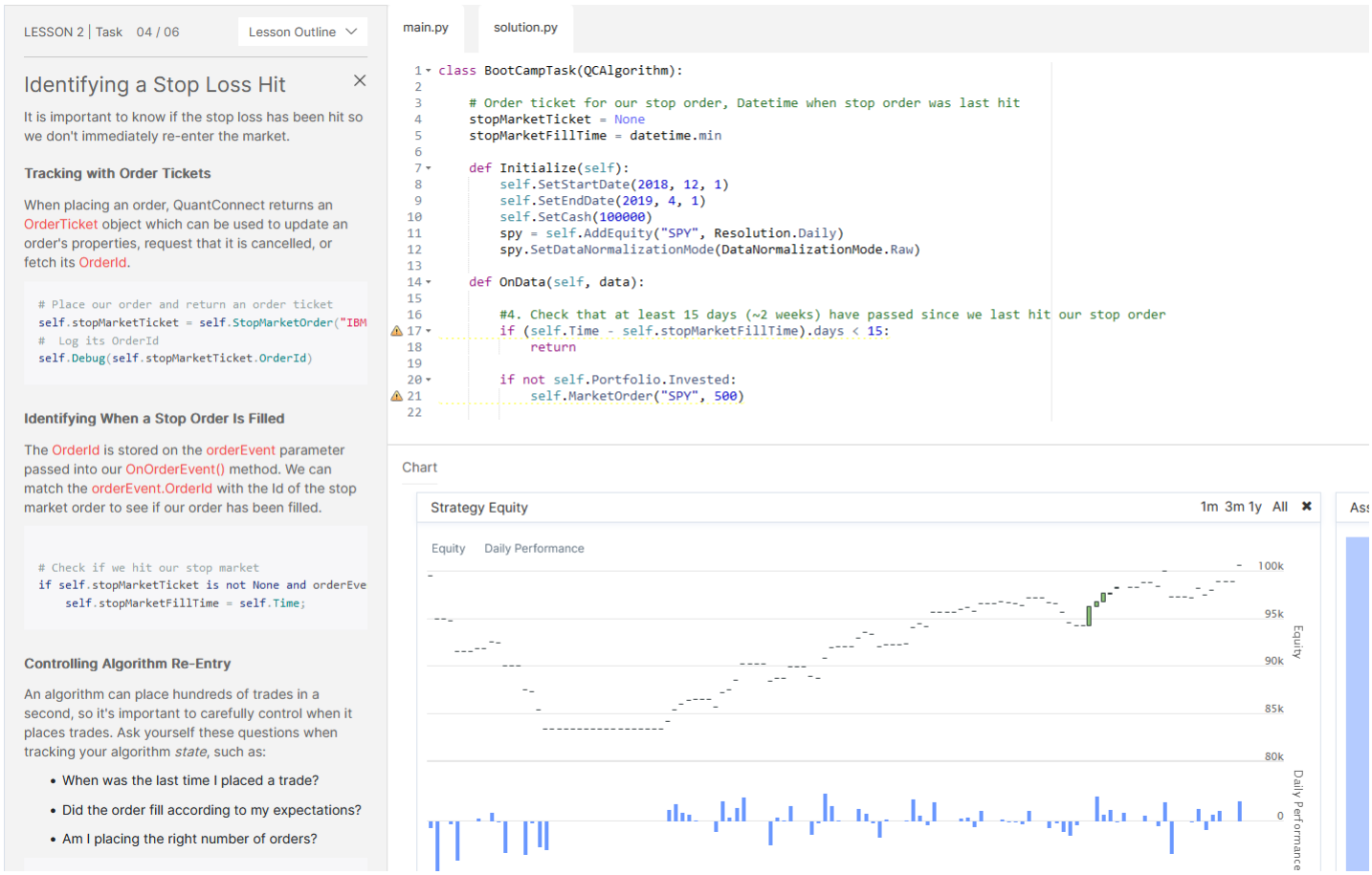
Follow these steps to enroll in Learning Center courses:

1. Log in to the Algorithm Lab.
2. In the left navigation bar, click Learning Center > All Courses .
3. On the Available Courses page, click the course in which you want to enroll.
4. Click Enroll .

   The Learning Center environment displays.

Navigate the Course IDE

The course IDE automatically displays when you enroll in a course.



Follow these steps to navigate the course IDE:

1. Read the instructions in the left panel.
2. Update the main.py file with your answer.
3. (Optional) Scroll down to the bottom of the instruction panel and click Show Hint to show a hint.

   A hint displays at the bottom of the instruction panel.

4. (Optional) Scroll down to the bottom of the instruction panel and click Solution to show the solution file.

   A solution.py file displays.

5. (Optional) Click Reset to reset the main.py file.
6. Click Submit to check your answer.

   The Chart panel displays your backtest results.

7. If an error message displays, restart from step 2.
8. Click Continue .

View Course Progress

Follow these steps to view your course progress:

1. Log in to the Algorithm Lab.
2. In the left navigation bar, click Learning Center > All Courses .
3. On the Available Courses page, click the course for which you want to see your progress.

   On the course page, the About this Course section displays your progress.

View Completed Courses

Log in to the Algorithm Lab and then, in the left navigation bar, click Learning Center > Completed to view your completed courses.

Submit Reviews

You need to complete a course before you can submit a review on it.

Follow these steps to review courses:

1. Open the Completed Courses page.
2. Click the course for which you want to submit a review.
3. Scroll down to the User Reviews section and then click the number of stars you want to give the course in your review.
4. Write your review.
5. Click Submit Review .

   The User Reviews section displays your review.

Report Errors

To report errors you experience with courses, email us an explanation of the error and the task URL where the error occurred.

5.2 Educators

Introduction

Educators are QuantConnect experts who contribute courses to the Learning Center. We are always looking for experts to become Educators and share their insight with the community. As an Educator, you create the description, requirements, and lessons of each course you contribute. You also provide the images for the course listing in the Learning Center. As students complete your course, you'll receive course reviews so you can improve your course material.

Registration

Anyone can register to become an Educator. We are looking for Educators that specialize in the following areas:

- Quant finance
- HTML, CSS, and Javascript
- C# and Python
- Jupyter

If you have the skills listed above, contact us to start the registration process.

Compensation

Educators are compensated in knowledge, commissions, and exposure. You earn a $5,000 commission per accepted course and a 70% revenue share for your courses. When your courses are listed in the Learning Center, your name and social media accounts are public, giving you exposure to the QuantConnect community.

5.3 Course Structure

Introduction

The courses in the Learning Center are structured in a way so you can complete courses at your own pace. The course structure enables you to improve your skills in finance, statistics, and software development while learning the QuantConnect API in easily digestible portions. The idea behind the course lessons is to focus on implementing individual strategies rather than learning just the theory.

Lessons

Courses are broken up into multiple lessons. Lessons are made up of videos, readings, and coding exercises. Lessons break up the process of learning into digestible tasks. Each lesson builds on an understanding of the API from the lesson before it, so we recommend completing the lessons in order. The introductory video of each lesson demonstrates the process of completing the tasks in the lesson before you implement them yourself.

Tasks

Lessons are broken up into multiple tasks to test your understanding of the course topic. Each task is accompanied by text instruction to guide you to complete the task. The Learning Center environment, where you complete each task, is similar to the regular web IDE used for backtesting and live trading. After you read the task instructions and run your solution algorithm, you are informed if you completed the task. If you need assistance, you can view a hint or the full solution file .

Results

To check if you have completed a task correctly, backtest your algorithm in the Learning Center environment and then the result window displays your results. If you receive an error message, update the code and then run the backtest again. If you pass the task, you'll be prompted to proceed to the next task in the course. If you're having trouble completing the task, you can copy the task solution file.

Solutions

A solution file for each task is available in the Learning Center environment. You may use it, but we recommend you try to solve the problem before you check the solution file. You'll learn best by solving the problem on your own. However, errors can occur when running backtests, so you may use the solution file to ensure the environment is running without error.

Errors

If you run into an error when you are working on a task, compare your code to the solution file. The error may be caused by either an error in your submission or an error in the backend of the Learning Center environment. In cases when errors occur in the backtest of the Learning Center environment, email us the following information:

- The URL of the task.
- An explanation of the issue.
- A code snippet to reproduce the error.

6 Projects

Projects contain files to run backtests, launch research notebooks, perform parameter optimizations, and deploy live trading strategies. You need to create projects in order to create strategies and share your work with other members. Projects enable you to generate institutional-grade reports on the performance of your backtests. You can create your projects from scratch or you can utilize pre-built libraries and third-party libraries to expedite your development process.

Getting Started

Learn the basics

Structure

How projects are made

Files

Where code lives

IDE

A browser coding experience

Debugging

Solve those coding errors

Collaboration

Work with your team members

Code Sessions

Connect to the remote IDE

Shared Libraries

Share code across projects

Package Environments

Bundled python libraries

LEAN Engine Versions

Customized LEAN engine versions

See Also

Backtesting
Sharing Backtests
Report

6.1 Getting Started

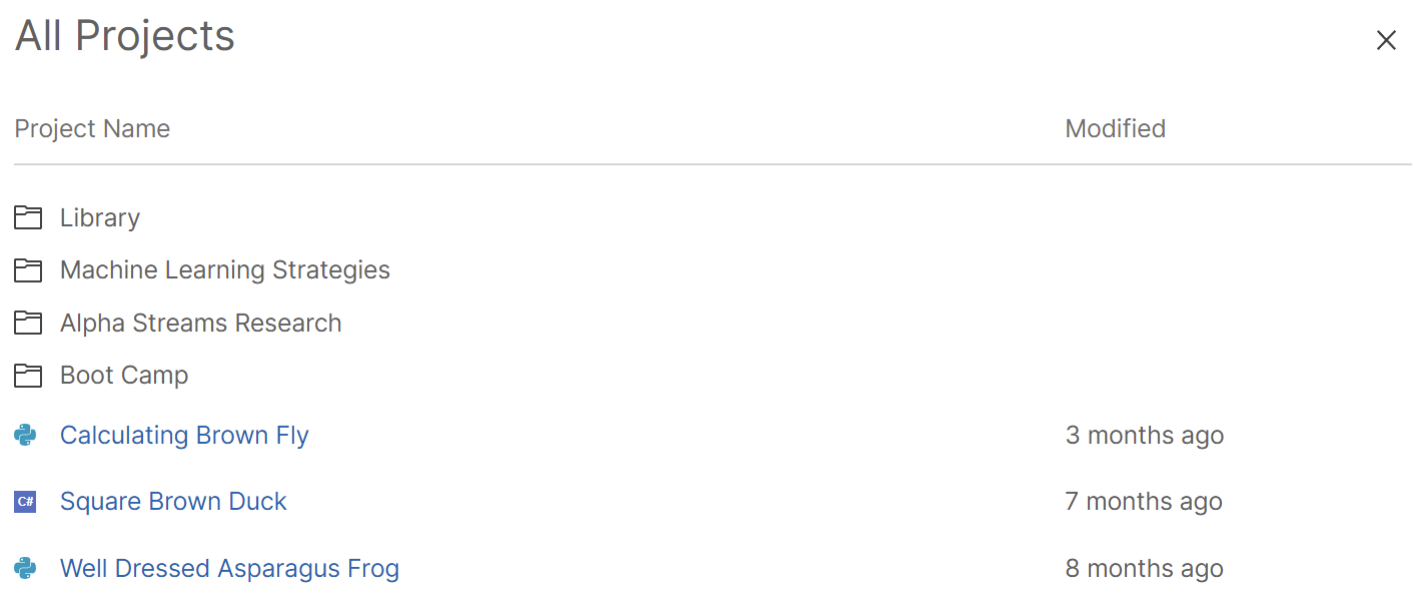
Introduction

Projects contain files to run backtests, launch research notebooks, perform parameter optimizations, and deploy live trading strategies. You need to create projects in order to create strategies and share your work with other members. Projects enable you to generate institutional-grade reports on the performance of your backtests. You can create your projects from scratch or you can utilize pre-built libraries and third-party packages to expedite your development process.

The Algorithm lab enables you to create, store, and manage your projects in the cloud. You can only access your own projects unless you share them with others, or add collaborators.

View All Projects

The All Projects page displays all of your QuantConnect projects in the organization, including libraries and Boot Camp lessons.



Click a project or directory on the page to open it.

Follow these steps to view the page:

1. Log in to the Algorithm Lab.
2. In the left navigation bar, click Projects .
3. Click Open Project .

Create Projects

Follow these steps to create new projects:

1. Log in to the Algorithm Lab.
2. In the left navigation bar, click Projects .
3. On the Projects page, click Create New Algorithm .

   The web IDE displays an empty project.

Close Projects

In the Project panel, click Close to close projects.

Clone Projects

Clone a project to create a new copy of the project and save it within the same organization. When you clone a project, the project files are duplicated but the backtest results are not retained. Cloning enables you to test small changes in your projects before merging the changes back into the original project.

To clone a project, open the project and then, in the Project panel, click Clone .

Migrate Projects

Migrating moves a project from one organization to another. You must be the organization administrator to migrate projects out of the organization. Migrate a project to run the project using resources from a different organization and to collaborate on the project with members from a different organization. When you migrate projects, the project files are copied but the content stored in the ObjectStore is not retained.

To migrate a project, open the project and then, in the Project panel, click Migrate .

Rename Projects

Follow these steps to rename a project:

1. Open the project .
2. In the Project panel, hover over the project name and then click the pencil icon that appears.

3. In the Name field, enter the new project name and then click Save Changes .

The project name must only contain - , _ , letters, numbers, and spaces. The project name can't start with a space or be any of the following reserved names: CON, PRN, AUX, NUL, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, or LPT9.

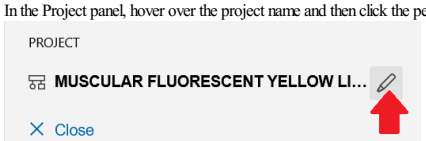
Create Project Directories

Set the name of a project to directoryName / projectName to create a project directory.

Set Descriptions

Follow these steps to set the project description:

1. Open the project .
2. In the Project panel, hover over the project name and then click the pencil icon that appears.



3. In the Description field, enter the new project description and then click Save Changes .

Edit Parameters

Algorithm parameters are hard-coded values for variables in your project that are set outside of the code files. Add parameters to your projects to remove hard-coded values from your code files and to perform parameter optimizations. You can add parameters, set default parameter values, and remove parameters from your projects.

Add Parameters

Follow these steps to add an algorithm parameter to a project:

1. Open the project .
2. In the Project panel, click Add New Parameter .
3. Enter the parameter name.

   The parameter name must be unique in the project.

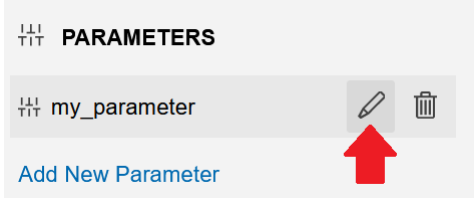
4. Enter the default value.
5. Click Create Parameter .

To get the parameter values into your algorithm, see Get Parameters .

Set Default Parameter Values

Follow these steps to set the default value of an algorithm parameter in a project:

1. Open the project .
2. In the Project panel, hover over the algorithm parameter and then click the pencil icon that appears.



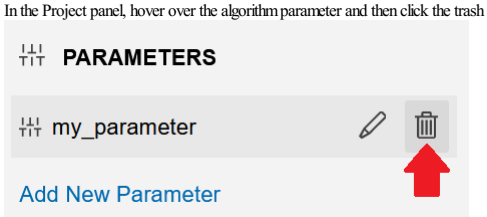
3. Enter a default value for the parameter and then click Save .

   The Project panel displays the default parameter value next to the parameter name.

Delete Parameters

Follow these steps to delete an algorithm parameter in a project:

1. Open the project .
2. In the Project panel, hover over the algorithm parameter and then click the trash can icon that appears.



3. Remove the `GetParameter` calls that were associated with the parameter from your code files.

Delete Projects

You can delete a project when it is open or closed.

Delete Open Projects

In the Project panel, click Delete , and then click Yes to delete the project.

Delete Closed Projects

Follow these steps to delete the project:

1. Open the My Projects page.
2. If the project is in a directory, click the directory files to navigate to the project file.
3. Hover over the project file and then click the trash can icon that appears.

"Project deleted" displays.

6.2 Structure

Introduction

Projects organize your algorithm data. They have settings, files, results, and attached libraries.

Your account has a directory to organize the projects that you have access to in each of your organizations. If you switch the organization that you are connected as , your directory of projects is updated to reflect the projects that you have access to within the new organization.

Files

New projects contain code files (.py or .cs) and notebook files (.ipynb). Run backtests with code files and launch the Research Environment with notebook files. Code files must stay within your size quotas . To keep files small, files can import code from other code files. To aid navigation, you can rename, move, and delete files in the web IDE. Notebook files save the input cells, but not the output cells.

Directories

Your directory of projects can contain nested directories of projects to make navigation easier. Similarly, the code and notebook files in your projects can contain nested directories of files. For example, if you have multiple Alpha models in your strategy, you can create an alphas directory in your project to hold a file for each Alpha model.

Description

You can give a project a description to provide a high-level overview of the project and its functionality. Descriptions make it easier to return to old projects and understand what is going on at a high level without having to look at the code. The project description is also displayed at the top of backtest reports , which you can create after your backtest completes.

Libraries

Libraries are reusable code files that you can import into any project for use in backtesting, research, and live trading. Use libraries to increase your development speed and save yourself from copy-pasting between projects. You can create libraries and add them to your projects using the web IDE. Your libraries are saved under the Library directory in the Algorithm Lab.

Parameters

Algorithm parameters are hard-coded values for variables in your project that are set outside of the code files. Add parameters to your projects to remove hard-coded values from your code files and to perform parameter optimizations . To get the parameter values into your algorithm, see Get Parameters . The parameter values are sent to your algorithm when you deploy the algorithm, so it's not possible to change the parameter values while the algorithm runs.

6.3 Files

Introduction

The files in your projects enable you to implement trading algorithms, perform research, and store important information. Python projects start with a main.py and a research.ipynb file. C# projects start with a Main.cs and a Research.ipynb file. Use the main.py or Main.cs file to implement trading algorithms and use the ipynb file to access the Research Environment.

Supported File Types

The IDE supports .cs , .ipynb , and .py files.

Add Files

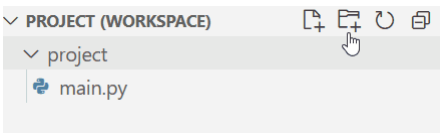
Follow these steps to add a file to a project:

1. Open the project .
2. In the right navigation menu, click the Explorer icon.
3. In the Explorer panel, expand the Workspace (Workspace) section.
4. Click the New File icon.
5. Enter a file name and extension.
6. Press Enter .

Add Directories

Follow these steps to add a directory to a project:

1. Open the project .
2. In the right navigation menu, click the Explorer icon.
3. In the Explorer panel, expand the Workspace (Workspace) section.
4. Click the New Directory icon.
5. Enter a directory name and then press Enter .



Open Files

Follow these steps to open a file in a project:

1. Open the project .
2. In the right navigation menu, click the Explorer icon.
3. In the Explorer panel, click the file you want to open.

Close Files

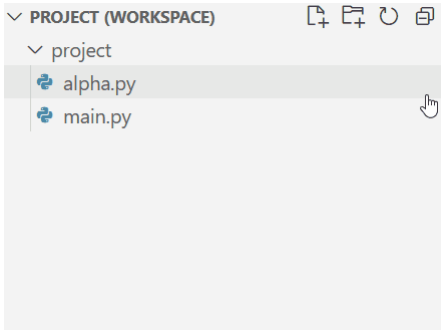
To close a file, at the top of the IDE, click the x button on the file tab you want to close.

To close all of the files in a project, at the top of the IDE, right-click one of the file tabs and then click Close All .

Rename Files and Directories

Follow these steps to rename a file or directory in a project:

1. Open the project .
2. In the right navigation menu, click the Explorer icon.
3. In the Explorer panel, right-click the file or directory you want to rename and then click Rename .
4. Enter the new name and then press Enter .



Delete Files and Directories

Follow these steps to delete a file or directory in a project:

1. Open the project .
2. In the right navigation menu, click the Explorer icon.
3. In the Explorer panel, right-click the file or directory you want to delete and then click Delete Permanently .
4. Click Delete .

Size Quotas

The maximum file size you can have in a project depends on your organization's tier. The following table shows the quota of each tier:

Tier	Max File Size (KB)
Free	32
Quant Researcher	64
Team	128
Trading Firm	256
Institution	256

6.4 IDE

Introduction

The web Integrated Development Environment (IDE) lets you work on research notebooks and develop algorithms for backtesting and live trading. When you open a project , the IDE automatically displays. You can access your trading algorithms from anywhere in the world with just an internet connection and a browser. If you prefer to use a different IDE, the CLI allows you to develop locally in your preferred IDE.

Supported Languages

The Lean engine supports C# and Python. Python is less verbose, has more third-party libraries, and is more popular among the QuantConnect community than C#. C# is faster than Python and it's easier to contribute to Lean if you have features written in C# modules. Python is also the native language for the research notebooks, so it's easier to use in the Research Environment.

The programming language that you have set on your account determines how autocomplete and IntelliSense are verified and determines the types of files that are included in your new projects. If you have Python set as your programming language, new projects will have .py files. If you have C# set as your programming language, new projects will have .cs files.

Change Languages

Follow these steps to select a programming language:

1. Log in to the Algorithm Lab.
2. In the top navigation bar, click yourUsername > My Account .
3. On your Account page, in the Account Settings section, click C# or Python .

   "Preferred language setting has been updated" displays.

Autocomplete and Intellisense

Intellisense is a GUI tool in your code files that shows auto-completion options and presents the members that are accessible from the current object. The tool works by searching for the statement that you're typing, given the context. You can use Intellisense to auto-complete method names and object attributes. When you use it, a pop-up displays in the IDE with the following information:

- Member type
- Member description
- The parameters that the method accepts (if the member is a method)

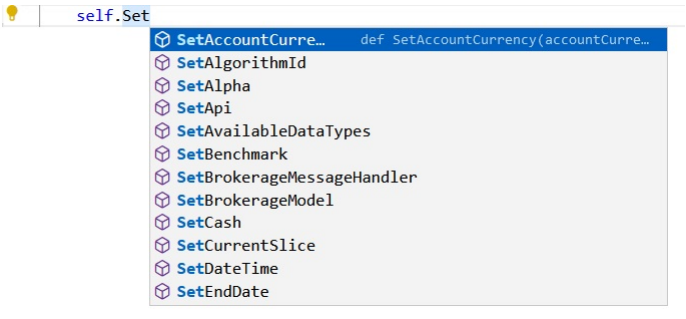
Use Intellisense to speed up your algorithm development. It works with all of the default class members in Lean, but it doesn't currently support class names or user-defined objects.

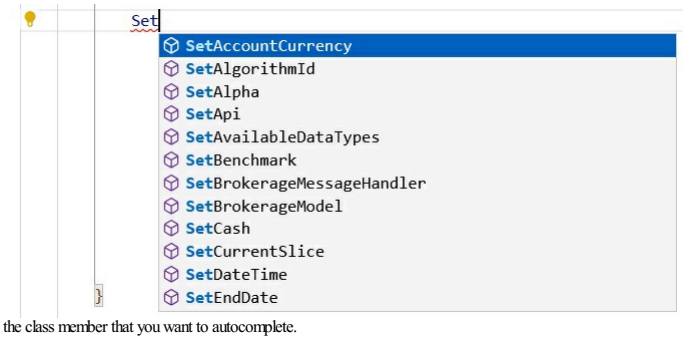
Use Autocomplete

Follow these steps to use autocomplete:

1. Open a project .
2. Type the first few characters of a variable, function, class, or class member that you want to autocomplete (for example, `self.Set Set` or `SimpleMovingAverage.Upda`).
3. Press CTRL+Space .

   If there are class members that match the characters you provided, a list of class members displays.



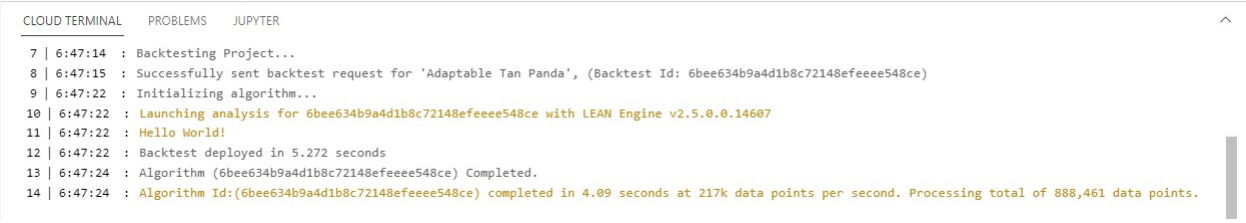


4. Select the class member that you want to autocomplete.

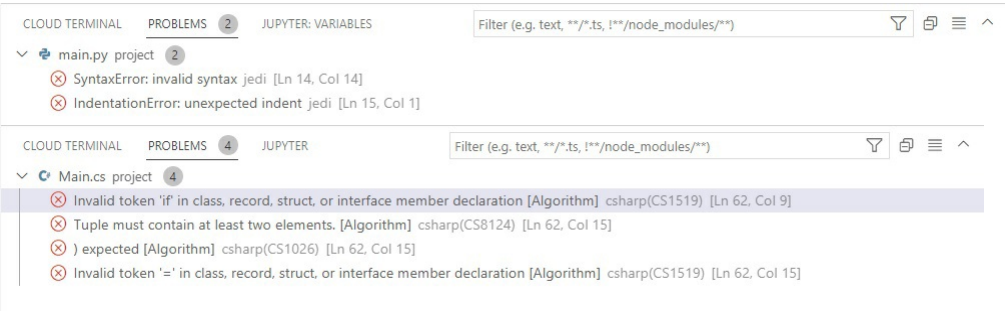
   The rest of the class member name is automatically written in the code file.

Cloud Terminal

The terminal panel at the bottom of the IDE shows API messages, errors, and the logs from your algorithms.



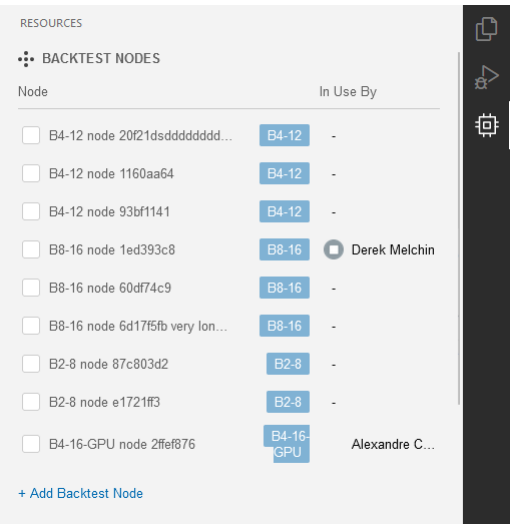
The Problems tab of the panel highlights the coding errors in your algorithms.



Manage Nodes

The Resources panel shows the cloud backtesting, research, and live trading nodes within your organization.

To view the Resources panel, open a project and then, in the right navigation menu, click the Resources icon.



The panel displays the following information for each node:

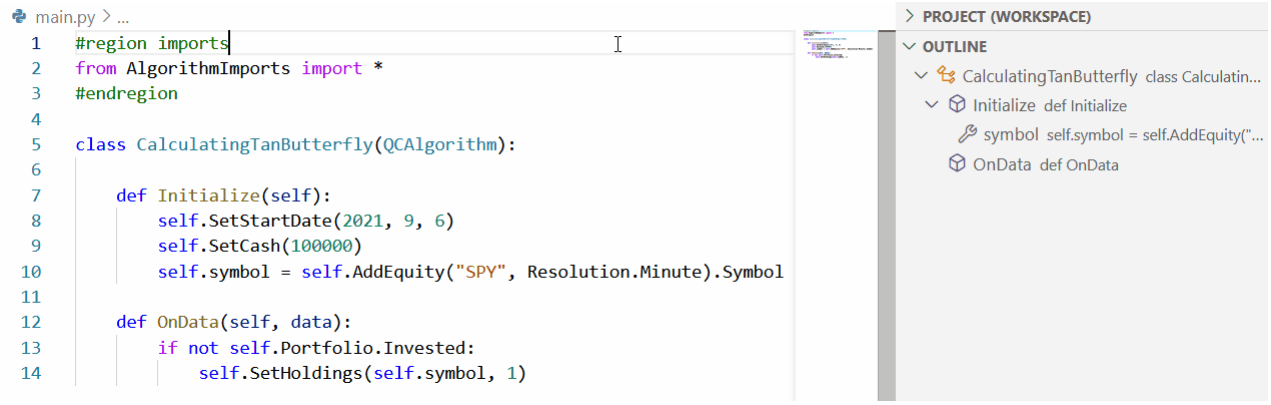
Column	Description
Node	The node name and model.
In Use By	The owner and name of the project using the node.

To stop a running node, click the stop button next to it. You can stop nodes that you are using, but you need stop node permissions to stop nodes other members are using.

By default, we select the best node available in your clusters when you launch a backtest or research notebook. To use a specific node, click the check box next to a node in the panel.

Navigate the File Outline

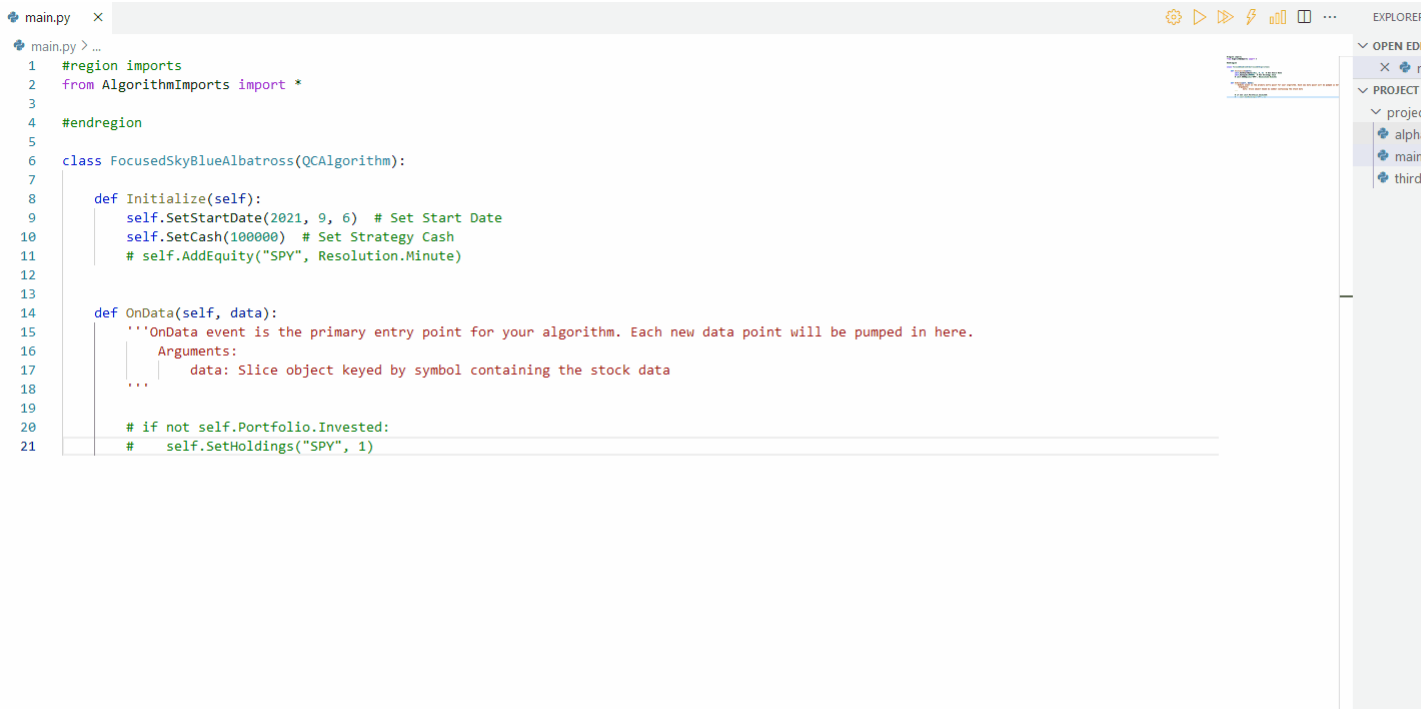
The Outline section in the Explorer panel is an easy way to navigate your files. The section shows the name of classes, members, and functions defined throughout the file. Click one of the names to jump your cursor to the respective definition in the file. To view the Outline , open a project and then, in the right navigation menu, click the Explorer icon.



Split the Editor

The editor can split horizontally and vertically to display multiple files at once. Follow these steps to split the editor:

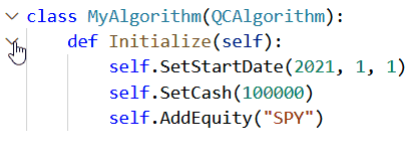
1. Open a project .
2. In the right navigation bar, click the Explorer icon.
3. In the QC (Workspace) section, drag and drop the files you want to open.



Use this feature instead of opening multiple browser tabs for a single project. If you open open multiple browser tabs, two code sessions will be updating the same project, which will cause the code sessions to fall out of sync.

Show and Hide Code Blocks

The editor can hide and show code blocks to make navigating files easier. To hide and show code blocks, open a project and then click the arrow icon next to a line number.



Keyboard Shortcuts

Keyboard shortcuts are combinations of keys that you can issue to manipulate the IDE. They can speed up your workflow because they remove the need for you to reach for your mouse.

Follow these steps to view the keyboard shortcuts of your account:

1. Open a project .
2. Press F1 .
3. Enter "Preferences: Open Keyboard Shortcuts".
4. Click Preferences: Open Keyboard Shortcuts .

To set a key binding for a command, click the pencil icon in the left column of the keyboard shortcuts table, enter the key combination, and then press Enter .

Themes

The Algorithm Lab offers light and dark themes. Follow these steps to change themes:

1. Log in to your account.
2. In the top navigation bar, click yourUsername > My Account .
3. On your Account page, in the Account Settings section, click Light Theme or Dark Theme

   Your Account page refreshes and displays the new theme.

Supported Browsers

The IDE works with Chrome, Edge, Firefox, and Safari. For more information about browser support, see Browser Support in the Visual Studio Code documentation.

Cookies

You need third-party cookies enabled on your browser to use the IDE. To enable third-party cookies, see the support page of the following browsers:

- Brave
- Chrome
- Edge
- Firefox
- Safari

Troubleshooting

If you experience issues trying to load the IDE, follow these steps:

1. Check if you're using one of the supported browsers .
2. Test a different supported browser.
3. Enable cookies .
4. Disable your browser add-ons.
5. Check your anti-virus settings.

   The internet protection of some anti-virus products block the "Service Workers" that the IDE needs to operate. Kaspersky and Avast are the two products that commonly block the IDE from using Service Workers. These are background threads that improve the IDE's experience.

6. Configure your network settings to use Google Public DNS .

   The DNS settings of some ISPs block Microsoft DNS, so some panels don't load and display the following message: "Server IP address could not be found".

   The following resources explain how to change your networks settings:

- Firefox DNS-over-HTTPS
- DNS Encryption with DNS over HTTPs (DoH) on Chrome
- Microsoft Edge to have DNS over HTTPS (DoH) as the default DNS settings
- Set up 1.1.1.1 - macOS

- Check your internet connection and speed.
- Try to load the IDE with a different computer, tablet, or cell phone.
- Clear your browser cache, especially if you created your project before the new IDE (March 2022).

Your browser may cache data from the old IDE.

6.5 Debugging

Introduction

Debugging is the process of systematically using a tool to find and fix errors in software. Errors can cause unintended trades, unexpected algorithm crashes, and faulty risk management logic. We use a debugging tool step through code line-by-line, and inspect the variables to understand the internal state of the program. You have many tools to debug your algorithm, including our built-in debugger, logging statements, and charting.

Coding Errors

Coding errors are errors that cause your projects to not build, throw exceptions, or behave unexpectedly. There are generally 3 types of coding errors: build, runtime, and logic errors. Each type of error occurs for different reasons.

Build Errors

Build errors occur when the interpreter's syntax check fails. An example code snippet that produces a build error is the following:

```
var a = 1;
if (a = 2) {}
```

```
a = 1
if a = 2:
    pass
```

If build errors occur in your project, you can not use the debugger, logging statements, or custom charts to debug the issue. You're notified of build errors in the following ways:

- The line where the error occurs is underlined in red.
- The Problems panel at the bottom of the IDE displays the errors.
- The Explorer panel highlights the editors, files, and outlines in red where the error occurs.

Runtime Errors

Runtime Errors, also called exceptions, occur when the interpreterâ€™s syntax checks pass but an error occurs during execution. An example code snippet that produces a runtime error is the following:

```
var a = new List<int>() {1};
var b = a[1];
```

```
a = [1]
a[1]
```

If runtime errors occur in your project, a stack trace of the error is added to the Cloud Terminal and the log file. For example, the snippet above produces the following error message:

Runtime Error: IndexError : list index out of range
at OnData
a[1]
═══
at Python.Runtime.PyObject.Invoke(PyTuple args in main.py: line 17
(Open Stack Trace)
Runtime Error: Index was out of range. Must be non-negative and less than the size of the collection. (Parameter 'index') in Main.cs:line 24 (Open Stack Trace)

The stack trace from the build error identifies the line of code where the error occurs. If the error doesn't reference your project files, it's an issue with Lean or another library. To view more information about the error, click (Open Stack Trace) .

Logic Errors

Logic errors occur when your algorithm behaves in an unexpected or unintended manner. These types of errors don't halt the program execution, so they are difficult to diagnose. An example code snippet that produces a logic error is the following:

```
var average = x + y / 2.0; // instead of (x + y) / 2.0
```

```
average = x + y / 2 # instead of (x + y) / 2
```

To resolve logic errors, carefully trace your algorithm. You may use the `Log` and `Debug` methods or the built-in debugger.

Debugger

The debugger is a built-in tool to help you debug coding errors while backtesting. The debugger enables you to slow down the code execution, step through the program line-by-line, and inspect the variables to understand the internal state of the program. For more information about the backtesting debugger, see Backtest Debugging .

Logging Statements

Algorithms can record string messages ('log statements') to a file for analysis after a backtest is complete, or as a live algorithm is running. These records can assist in debugging logical flow errors in the project code. Consider adding them in the code block of an `if` statement to signify an error has been caught.

It's good practice to add logging statements to live algorithms so you can understand its behavior and keep records to compare against backtest results. If you don't add logging statements to a live algorithm and the algorithm doesn't trade as you expect, it's difficult to evaluate the underlying problem.

Log

Log statements are added to the log file while your algorithm continues executing. Logging dataset information is not permitted. Use `Log` statements to debug your backtests and live trading algorithms.

Log length is capped by organization tier . If your organization hits the daily limit, contact us .

If you log the same content multiple times, only the first instance is added to the log file. To bypass this rate-limit, add a timestamp to your log messages.

For live trading, the log files of each cloud project can store up to 100,000 lines for up to one year. If you log more than 100,000 lines or some lines become older than one year, we remove the oldest lines in the files so your project stays within the quota.

To record the algorithm state when the algorithm stops executing, add log statements to the `OnEndOfAlgorithm` event handler.

```
Log("My log message");
```

```
self.Log("My log message")
```

Debug

Debug statements are the same as log statements, but `Debug` statements are orange in the Cloud Terminal. Use these statements when you want to give more attention to a message in the Cloud Terminal. Debug messages can be up to 200 characters in length. If you send multiple debug statements within 1 second, your messages are rate-limited to avoid crashing your browser.

```
Debug("My debug message");
```

```
self.Debug("My debug message")
```

Error

Error statements are the same as log statements, but `Error` statements are displayed in red text in the Cloud Terminal. Use these statements when you want to give the most attention to a message in the Cloud Terminal. Error statements are rate-limited like debug statements.

```
Error("My error message");
```

```
self.Error("My error message")
```

Quit

Quit statements cause your project to stop running and may log some data to the log file and Cloud Terminal. These statements are orange in the Cloud Terminal. When you call the `Quit` method, the program continues executing until the

end of the method definition. If you want to quit execution immediately, `return` after you call `Quit` .

```
Quit("My quit message");
```

```
self.Quit("My quit message")
```

Charting

You can use the IDE charting capabilities to plot values over time when debugging. To add data points to a custom chart, call the `Plot` method with a chart name, series name, and value. For a full example, see Charting .

```
Plot(chart, series, value);
```

```
self.Plot(chart, series, value)
```

If you run your algorithm in QuantConnect Cloud, we limit the number of points a chart can have to 4,000 because intensive charting generates hundreds of megabytes (200MB) of data, which is too much to stream online or display in a web browser. If you exceed the limit, the following error message is thrown:

Exceeded maximum data points per series, chart update skipped.

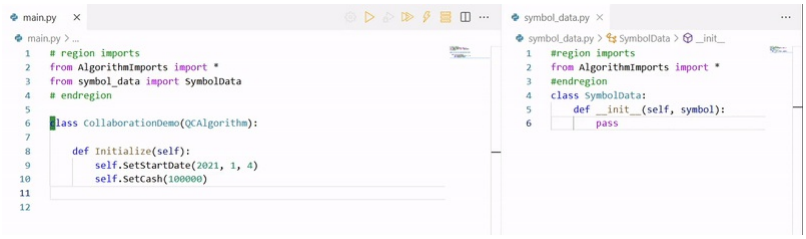
6.6 Collaboration

Introduction

Project collaboration is a real-time coding experience with other members of your team. Collaborating can speed up your development time. By working with other members in an organization, members within the organization can specialize in different parts of the project.

Video Demo

When there are multiple people working on the same project, the cursor of each member is visible in the IDE and all file changes occur in real-time for everyone. The following video demonstrates the collaboration feature:



Add Team Members

You need to own the project to add team members to it.

Follow these steps to add team members to a project:

1. Open the project .
2. In the Collaborate section of the Project panel, click Add Collaborator .
3. Click the Select User... field and then click a member from the drop-down menu.
4. If you want to give the member live control of the project, select the Live Control check box.
5. Click Add User .

   The member you add receives an email with a link to the project.

If the project has a shared library , the collaborator can access the project, but not the library. To grant them access to the library, add them as a collaborator to the library project.

Collaborator Quotas

The number of members you can add to a project depends on your organization's tier . The following table shows the number of collaborators each tier can have per project:

Tier	Collaborators per Project
Free	Unsupported
Quant Researcher	Unsupported
Team	10
Trading Firm	Unlimited
Institution	Unlimited

Toggle Live Control

You need to have added a member to the project to toggle their live control of the project.

Follow these steps to enable and disable live control for a team member:

1. Open the project .
2. In the Collaborate section of the Project panel, click the profile image of the team member.
3. Click the Live Control check box.
4. Click Save Changes .

Remove Team Members

Follow these steps to remove a team member from a project you own:

1. Open the project .
2. In the Collaborate section of the Project panel, click the profile image of the team member.
3. Click Remove User .

To remove yourself as a collaborator from a project you don't own, delete the project .

Intellectual Property

All individuals on QuantConnect own their intellectual property (IP) on our platform. Your code is private and only accessible by people you share the project with and with support-engineers when you submit a support ticket. At no point does QuantConnect ever claim ownership of user IP. The only case where algorithm code becomes public domain is when they are shared to the forum. In this case, algorithms need to be made public domain to allow the sharing of the algorithm code.

It is common when companies hire engineers to write software, they require their employees to sign an agreement that gives the company IP ownership of any code written for work. They need this because they're paying you to write software, and the company needs to sell that software to turn a profit. Similarly, the Organizations feature allows you to control who holds IP ownership over a project. Each type of organization has its own mechanisms for handling project IP ownership.

Individual Organizations

The Free and the Quant Researcher tiers only allow single-member organizations. This means you can't collaborate with anyone else inside the QuantConnect platform. Simply put, you own the IP for any projects you work on since you are the sole collaborator.

Team Organizations

For organizations that allow multiple users to collaborate on projects, the user who created the project owns it; this can be you or one of your teammates. If you add a teammate/collaborate, they can clone it, but the original project belongs to the person who first created it.

Trading Firm & Institution Organizations

For Trading Firm and Institution organizations, which are generally used by companies and funds, the firm owns all employee projects. This is made to suit firms that wish to hire consultants and need to ensure the code remains with the company when the consultant work is finished. You have to explicitly create a project in an organization for it to be created on the organization's account.

Other Collaboration Methods

Additional methods of collaboration include cloning, sharing, and migrating projects.

Clone Projects

Clone a project to create a new copy of the project and save it within the same organization. When you clone a project, the project files are duplicated but the backtest results are not retained. Cloning enables you to test small changes in your projects before merging the changes back into the original project.

To clone projects, open the project you want to clone and then, in the Project panel, click Clone . "Project cloned successfully" displays.

Share Projects

Run a backtest and then make the backtest results public to share a project. Once a backtest is made public, a link is generated for you that opens the backtest results and the project files. You can directly give the link to others, attach the backtest to a forum discussion, or embed the backtest into a website. However, note that when you make a backtest public, the project files are accessible to anyone who visits the link, even after you delete the project. As a result, we don't recommend collaborating on projects by making backtests public and sharing the link with your collaborators. Instead, add team members to your project since it protects your intellectual property.

You can share a backtest at any time when it's executing. Although, if you generate a link to share the backtest before the backtest completes, the link that's generated will not contain all of the backtest results. Some reasons to share your project include the following:

- Attach the project to the forum to ask for help, gather feedback, or report an issue.
- Attach the project to a data issue to reduce the amount of time it takes to fix the data issue.
- Share a link to the project with others to give them a copy of the project files and the backtest results.

To share a research notebook, save the notebook and run a backtest.

Migrate Projects

Migrating moves a project from one organization to another. You must be the organization administrator to migrate projects out of the organization. Migrate a project to run the project using resources from a different organization and to collaborate on the project with members from a different organization. When you migrate projects, the project files are copied but the content stored in the ObjectStore is not retained.

Follow these steps to migrate projects:

1. Open the project you want to migrate.
2. In the Project panel, click Migrate .
3. Click the name of the organization to which you want to migrate the project and then click Migrate .

   The top navigation bar displays Connected as: theOrganizationYouMigratedTo .

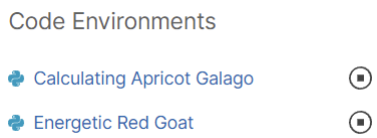
6.7 Code Sessions

Introduction

Code sessions let you access a cloud hosted IDE to research and develop trading algorithms. When you open a project, a new code session starts running with the latest master branch of the LEAN trading engine. The session is ready to go with access to the full QuantConnect data library and the cloud resources of the QuantConnect technology stack.

View Code Sessions

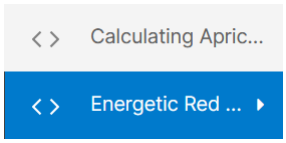
The Projects page displays all of your running code sessions in your current organization. To view the page, log in to the Algorithm Lab and then, in the left navigation menu, click Projects .



To open one of the code sessions, click the project name.

To stop the code sessions, click the stop icon next to a project name. If you log out, the code sessions don't automatically stop.

The left navigation bar of the Algorithm Lab also shows the running code sessions underneath Projects . The blue code session represents the session that's currently open. The gray code sessions represent the sessions that are currently minimized.



Code Session Quotas

If you have a project open, it uses a coding session. Paid organizations can have multiple active coding sessions, but free users can only have one coding session open at a time. The following table shows how many active coding sessions you can have on each organization tier:

Tier	Initial Coding Session Quota
Quant Researcher	2
Team	4
Trading Firm	8
Institution	16

If the organization you're in has more live trading nodes than your initial coding session quota, then your coding session quota increases to the number of live trading nodes you have in the organization so you can view all your live strategies.

The quota for free organizations is a global quota, so you can have one active coding session across all of your free organizations. The quotas for paid organizations are at the organization level. Therefore, if you are in two Quant Researcher organizations, you can have two active coding sessions in one of those organizations and another two active sessions in the other organization. These paid tier quotas are for each account, not for the organization as a whole. For instance, a Trading Firm organization can have more than eight members and all of the members can simultaneously work on projects within the organization.

6.8 Shared Libraries

Introduction

Project libraries are QuantConnect projects you can merge into your project to avoid duplicating code files. If you have tools that you use across several projects, create a library.

Create Libraries

Follow these steps to create a library:

1. Create a new project .
2. In the project panel, click Add Library .
3. Click Create New .
4. In the Input Library Name field, enter a name for the library (e.g. Calculators).
5. Click Create Library .

   The template library files are added to your project. View the files in the Explorer panel.

6. In the right navigation menu, click the Explorer icon.
7. In Explorer panel, open the Library.py Library.cs file, rename it to reflect its purpose (e.g.: TaxesCalculator.py TaxesCalculator.cs), and implement your library.

Add Libraries

Follow these steps to add a library to your project:

1. Open the project .
2. In the Project panel, click Add Library .
3. Click the Choose a library... field and then click a library from the drop-down menu.
4. Click Add Library (e.g. Calculators).

   The library files are added to your project. To view the files, in the right navigation menu, click the Explorer icon.

5. Import the library into your project to use the library.

```
using Calculators;
namespace QuantConnect.Algorithm.CSharp
{
    public class AddLibraryAlgorithm : QCAlgorithm
    {
        private TaxesCalculator _taxesCalculator = new();
    }
}
```

```
from Calculators.TaxesCalculator import TaxesCalculator
class AddLibraryAlgorithm(QCAlgorithm):
    taxes_calculator = TaxesCalculator()
```

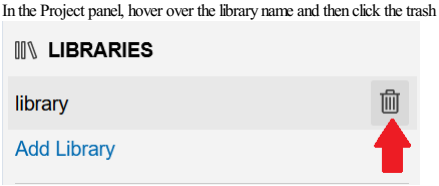
Rename Libraries

To rename a library, open the library project file and then rename the project .

Remove Libraries

Follow these steps to remove a library from your project:

1. Open the project that contains the library you want to remove.
2. In the Project panel, hover over the library name and then click the trash can icon that appears.



   The library files are removed from your project.

Delete Libraries

To delete a library, delete the library project file .

6.9 Package Environments

Introduction

Libraries (or packages) are third-party software that you can use in your projects. You can use many of the available open-source libraries to complement the classes and methods that you create. Libraries reduce your development time because it's faster to use a pre-built, open-source library than to write the functionality. Libraries can be used in backtesting, research, and live trading. The environments support various libraries for machine learning, plotting, and data processing. As members often request new libraries, we frequently add new libraries to the underlying docker image that runs the Lean engine.

This feature is primarily for Python algorithms as not all Python libraries are compatible with each other. We've bundled together different sets of libraries into distinct environments. To use the libraries of an environment, set the environment in your project and add the relevant `using import` statement of a library at the top of your file.

Set Environment

Follow these steps to set the library environment:

1. Open a project .
2. In the Project panel, click the Python Foundation field and then select an environment from the drop-down menu.

Default Environment

The default environment supports the following libraries:

# Name	Version
absl-py	1.4.0
adagio	0.2.4
aesara	2.8.12
aiodns	3.0.0
aiohttp	3.8.4
aiohttp-cors	0.7.0
aiosignal	1.3.1
alabaster	0.7.13
ale-py	0.7.4
alembic	1.10.3
alpaca-trade-api	0.26
alphalens-reloaded	0.4.3
altair	4.2.2
ansi2html	1.8.0
antlr4-python3-runtime	4.11.1
anyio	3.6.2
appdirs	1.4.4
arch	5.3.1
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arviz	0.15.1
astropy	5.2.1
asttokens	2.2.1
astunparse	1.6.3
async-timeout	4.0.2
asyncio-nats-client	0.11.5
attrs	23.1.0
autograd	1.5
autokeras	1.1.0
autoray	0.6.3
AutoROM	0.4.2
AutoROM.accept-rom-license	0.6.1
ax-platform	0.3.0
Babel	2.12.1
backcall	0.2.0
bayesian-optimization	1.4.2
beautifulsoup4	4.11.2
bleach	6.0.0
blessed	1.20.0
blis	0.7.9
blosc2	2.0.0
bokeh	2.4.3
boltons	23.0.0
botorch	0.8.2
Bottleneck	1.3.7
brotlipy	0.7.0
cachetools	5.3.0
captum	0.6.0
catalogue	2.0.8
catboost	1.1.1
category-encoders	2.6.0
ccxt	3.0.72
certifi	2022.12.7
cffi	1.15.1
chardet	5.1.0
charset-normalizer	2.0.4
check-shapes	1.0.0
click	7.1.2
clikit	0.6.2
cloudpickle	2.2.1
cmaes	0.9.1
cmdstanpy	1.1.0
colorama	0.4.6
colorcet	3.0.1
colorful	0.5.5
colorlog	6.7.0
colorlover	0.3.0
colour	0.1.5
comm	0.1.3
commonmark	0.9.1
conda	23.3.1
conda-content-trust	0.1.3
conda-package-handling	2.0.2
conda_package_streaming	0.7.0
confection	0.0.4
cons	0.4.5
contourpy	1.0.7
convertdate	2.4.0
copulae	0.7.7
copulas	0.8.0
cramjam	2.6.2
crashtest	0.3.1
creme	0.6.1
cryptography	38.0.4
cufflinks	0.17.3
cvxopt	1.3.0
cvxpy	1.3.0
cycler	0.11.0
cymem	2.0.7
Cython	0.29.33
darts	0.23.1
dash	2.9.3
dash-core-components	2.0.0
dash-cytoscape	0.3.0
dash-html-components	2.0.0
dash-table	5.0.0
dask	2023.4.0
deap	1.3.3
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
Deprecated	1.2.13
dgl	1.0.1

dill	0.3.6
dimod	0.12.3
diskcache	5.6.1
distlib	0.3.6
distributed	2021.4.1
dm-tree	0.1.8
docutils	0.19
DoubleML	0.5.2
dtreeviz	2.2.1
dtw-python	1.3.0
dwave-cloud-client	0.10.3
dwave-drivers	0.4.4
dwave-greedy	0.3.0
dwave-hybrid	0.6.9
dwave-inspector	0.3.0
dwave-inspectorapp	0.3.0
dwave-neal	0.6.0
dwave-networkx	0.8.12
dwave-ocean-sdk	6.1.1
dwave-preprocessing	0.5.3
dwave-samplers	1.0.0
dwave-system	1.18.0
dwave-tabu	0.5.0
dwavebinarycsp	0.2.0
dx	0.1.22
ecos	2.0.12
EMD-signal	1.4.0
empyrical	0.5.5
empyrical-reloaded	0.5.9
en-core-web-md	3.5.0
en-core-web-sm	3.5.0
entrypoints	0.4
ephem	4.1.4
et-xmlfile	1.1.0
etuples	0.3.8
exceptiongroup	1.1.1
exchange-calendars	4.2.6
executing	1.2.0
ezyrb	1.3.0.post2304
fastai	2.7.11
fastai2	0.0.30
fastcore	1.5.29
fastdownload	0.0.7
fasteners	0.18
fastjsonschema	2.16.3
fastparquet	2023.2.0
fastprogress	1.0.3
fasttext	0.9.2
featuretools	0.23.1
filelock	3.12.0
findiff	0.9.2
finrl	0.3.1
FixedEffectModel	0.0.5
Flask	2.0.3
flatbuffers	23.3.3
fonttools	4.39.3
fracdiff	0.9.0
frozendict	2.3.7
frozenlist	1.3.3
fs	2.4.16
fsspec	2023.4.0
fst-pso	1.8.1
fugue	0.8.3
fugue-sql-antlr	0.1.6
future	0.18.3
fuzzy-c-means	1.6.3
FuzzyTM	2.0.5
gast	0.4.0
gensim	4.3.0
gevent	22.10.2
gluonts	0.12.3
google-api-core	2.11.0
google-auth	2.17.3
google-auth-oauthlib	0.4.6
google-pasta	0.2.0
googleapis-common-protos	1.59.0
gpflow	2.7.0
gplearn	0.4.2
gpustat	1.1
GPUtil	1.4.0
gpytorch	1.9.1
graphviz	0.20.1
greenlet	2.0.2
grpcio	1.54.0
gym	0.21.0
Gymnasium	0.26.3
gymnasium-notices	0.0.1
h2o	3.40.0.1
h5netcdf	1.1.0
h5py	3.8.0
hijri-converter	2.2.4
hmmlearn	0.2.8
holidays	0.23
holoviews	1.15.4
homebase	1.0.1
hopcroftkarp	1.2.5
html5lib	1.1
httpstan	4.9.1
hurst	0.0.5
hvplot	0.8.2
hyperopt	0.2.7
idna	3.4
iisignature	0.24
ijson	3.2.0.post0
imageio	2.27.0
imagesize	1.4.1
imbalanced-learn	0.10.1
importlib-metadata	4.13.0
importlib-resources	5.12.0
iniconfig	2.0.0
injector	0.20.1
interpret	0.3.0
interpret-core	0.3.2
ipykernel	6.22.0
ipython	8.12.0
ipython-genutils	0.2.0
ipywidgets	8.0.4
itsdangerous	2.1.2
jax	0.4.4
jaxlib	0.4.4
jedi	0.18.2
Jinja2	3.1.2
joblib	1.2.0
jqdatasdk	1.8.11
json5	0.9.11
jsonpatch	1.32
jsonpointer	2.0
jsonschema	4.17.3
jupyter	1.0.0
jupyter-bokeh	3.0.5

jupyter-console	6.6.3
jupyter-dash	0.4.2
jupyter-resource-usage	0.7.2
jupyter-server	1.24.0
jupyter_client	8.2.0
jupyter_core	5.3.0
jupyterlab	3.4.4
jupyterlab-pygments	0.2.2
jupyterlab-widgets	3.0.7
jupyterlab_server	2.22.1
keract	4.5.1
keras	2.11.0
keras-nlp	0.4.1
keras-rl	0.4.2
keras-tuner	1.3.5
kiwisolver	1.4.4
kmapper	2.0.1
korean-lunar-calendar	0.3.1
kt-legacy	1.0.5
langcodes	3.3.0
lark	1.1.5
lazy_loader	0.2
lazypredict	0.2.12
libclang	16.0.0
lightgbm	3.3.5
lime	0.2.0.1
line-profiler	4.0.2
linear-operator	0.3.0
littleutils	0.2.2
livelossplot	0.5.5
llvmlite	0.39.1
locket	1.0.0
logical-unification	0.4.5
LunarCalendar	0.0.9
lxml	4.9.2
lz4	4.3.2
Mako	1.2.4
Markdown	3.4.3
MarkupSafe	2.1.2
marshmallow	3.19.0
matplotlib	3.7.0
matplotlib-inline	0.1.6
miniful	0.0.6
miniKanren	1.0.3
minorminer	0.2.9
mistune	2.0.5
mljar-supervised	0.11.5
mlxtend	0.21.0
mmh3	2.5.1
modin	0.18.1
mpi4py	3.1.4
mplfinance	0.12.9b7
mpmath	1.2.1
msgpack	1.0.4
mthree	2.4.0
multidict	6.0.4
multipledispatch	0.6.0
multiprocess	0.70.14
multitasking	0.0.11
murmurhash	1.0.9
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.1
nbeats-keras	1.8.0
nbeats-pytorch	1.8.0
nbformat	5.8.0
nest-asyncio	1.5.6
networkx	2.8.8
neural-tangents	0.6.2
neuralprophet	0.5.0
nfoursid	1.0.1
nltk	3.8.1
nose	1.3.7
notebook	6.5.4
notebook_shim	0.2.2
ntlm-auth	1.5.0
numba	0.56.4
numerapi	2.13.2
numexpr	2.8.4
numpy	1.23.5
numpydoc	1.5.0
nvidia-cublas-cu11	11.10.3.66
nvidia-cuda-nvrtc-cu11	11.7.99
nvidia-cuda-runtime-cu11	11.7.99
nvidia-cudnn-cu11	8.5.0.96
nvidia-ml-py	11.525.112
oauthlib	3.2.2
openai	0.26.5
opencensus	0.11.2
opencensus-context	0.1.3
opencv-python	4.7.0.72
openpyxl	3.1.1
opt-einsum	3.3.0
optuna	3.1.0
orjson	3.8.10
ortools	9.4.1874
osqp	0.6.2.post9
outdated	0.2.2
packaging	23.1
pandas	1.5.3
pandas-datareader	0.10.0
pandas-flavor	0.5.0
pandas-market-calendars	4.1.4
pandas-ta	0.3.14b0
pandocfilters	1.5.0
panel	0.14.3
param	1.13.0
parso	0.8.3
partd	1.4.0
pastel	0.2.1
pathos	0.3.0
pathy	0.10.1
patsy	0.5.3
pbr	5.11.1
penaltymodel	1.0.2
PennyLane	0.29.0
PennyLane-Lightning	0.29.0
PennyLane-qiskit	0.29.0
persim	0.3.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
pingouin	0.5.3
pip	22.3.1
pkgutil_resolve_name	1.3.10
platformdirs	3.2.0
plotly	5.13.1
plotly-resampler	0.8.3.2
plucky	0.4.3
pluggy	1.0.0

ply	3.11
pmdarima	2.0.2
polars	0.16.9
pox	0.3.2
ppft	1.7.6.6
pprofile	2.1.0
ppscore	1.2.0
preshed	3.0.8
prometheus-client	0.16.0
prompt-toolkit	3.0.38
property-cached	1.6.4
prophet	1.1.2
protobuf	3.19.6
psutil	5.9.5
psycopg2-binary	2.9.6
ptvsd	4.3.2
ptyprocess	0.7.0
PuLP	2.7.0
pure-eval	0.2.2
py-cpuinfo	9.0.0
py-heat	0.0.6
py-heat-magic	0.0.2
py-lets-be-rational	1.0.1
py-spy	0.3.14
py-vollib	1.0.1
py4j	0.10.9.7
pyaml	21.10.1
pyarrow	11.0.0
pyasn1	0.4.8
pyasn1-modules	0.2.8
pybind11	2.10.4
pycares	4.3.0
pycosat	0.6.4
pycparser	2.21
pyct	0.5.0
pydantic	1.10.7
pyDeprecate	0.3.2
pydevd-pycharm	201.8538.36
pydmd	0.4.0.post2301
pyerfa	2.0.0.3
pyfolio	0.9.2
pyfolio-reloaded	0.9.5
pyFUME	0.2.25
Pygments	2.15.1
pykalman	0.9.5
pylev	1.4.0
pyluach	2.2.0
pymannkendall	1.4.3
pymc	5.0.2
pymdptoolbox	4.0b3
PyMeeus	0.5.12
PyMySQL	1.0.3
pynndescent	0.5.9
pyod	1.0.9
Pyomo	6.5.0
pyOpenSSL	22.0.0
pyparsing	3.0.9
pyportfolioopt	1.5.4
pyqubo	1.3.1
pyrb	1.0.1
pyro-api	0.1.2
pyro-ppl	1.8.4
pyrsistent	0.19.3
pysimdjson	5.0.2
PySocks	1.7.1
pystan	3.6.0
pytensor	2.9.1
pytest	7.3.1
python-dateutil	2.8.2
python-statemachine	1.0.3
pytorch-ignite	0.4.11
pytorch-lightning	1.7.4
pytz	2023.3
pyvinecopulib	0.6.2
pyviz-comms	2.2.1
PyWavelets	1.4.1
PyYAML	6.0
pyzmq	25.0.2
qdldl	0.1.7
qiskit	0.41.1
qiskit-aer	0.11.2
qiskit-ibmq-provider	0.20.1
qiskit-terra	0.23.2
qpd	0.4.1
qtconsole	5.4.2
QtPy	2.3.1
quadprog	0.1.11
quantecon	0.6.0
QuantLib	1.29
QuantStats	0.0.59
rauth	0.7.3
ray	2.3.0
rectangle-packer	2.0.1
regex	2023.3.23
requests	2.28.1
requests-ntlm	1.1.0
requests-oauthlib	1.3.1
retrying	1.3.4
retworkx	0.12.1
rich	12.4.4
ripser	0.6.4
Riskfolio-Lib	4.0.3
riskparityportfolio	0.4
river	0.14.0
rsa	4.9
ruamel.yaml	0.17.21
ruamel.yaml.clib	0.2.6
ruptures	1.1.7
rustworkx	0.12.1
SALib	1.4.7
scikeras	0.10.0
scikit-image	0.19.3
scikit-learn	1.2.1
scikit-learn-extra	0.2.0
scikit-multiflow	0.5.3
scikit-optimize	0.9.0
scikit-plot	0.3.7
scikit-tda	1.0.0
scipy	1.10.1
scs	3.2.3
sdeint	0.3.0
seaborn	0.12.2
semantic-version	2.10.0
Send2Trash	1.8.0
setuptools	64.0.2
setuptools-scm	7.1.0
shap	0.41.0
simpful	2.10.0
simplejson	3.19.1
simpy	4.0.1

```
six                             1.16.0
sklearn-json                    0.1.0
skope-rules                     1.0.1
sktime                          0.16.1
slicer                          0.0.7
smart-open                      6.3.0
sniffio                         1.3.0
snowballstemmer                 2.2.0
sortedcontainers                2.4.0
soupsieve                       2.4.1
spacy                           3.5.0
spacy-legacy                    3.0.12
spacy-loggers                   1.0.4
Sphinx                          6.1.3
sphinxcontrib-applehelp         1.0.4
sphinxcontrib-devhelp           1.0.2
sphinxcontrib-htmlhelp          2.0.1
sphinxcontrib-jsmath            1.0.1
sphinxcontrib-qthelp            1.0.3
sphinxcontrib-serializinghtml 1.1.5
SQLAlchemy                      1.4.47
sqlglot                         11.5.5
srsly                           2.4.6
ssm                             0.0.1
stable-baselines3               1.7.0
stack-data                      0.6.2
statsforecast                   1.5.0
statsmodels                     0.13.5
stellargraph                    1.2.1
stevedore                       5.0.0
stochastic                      0.7.0
stockstats                      0.5.2
stumpy                          1.11.1
symengine                       0.10.0
sympy                           1.11.1
ta                              0.10.2
TA-Lib                          0.4.18
tables                          3.8.0
tabulate                        0.8.10
tadasets                        0.0.4
tbats                           1.1.3
tblib                           1.7.0
tenacity                        8.2.2
tensorboard                     2.11.2
tensorboard-data-server         0.6.1
tensorboard-plugin-wit          1.8.1
tensorboardX                    2.6
tensorflow                      2.11.0
tensorflow-addons               0.19.0
tensorflow-decision-forests   1.2.0
tensorflow-estimator            2.11.0
tensorflow-hub                  0.13.0
tensorflow-io-gcs-filesystem  0.32.0
tensorflow-probability          0.19.0
tensorflow-text                 2.11.0
tensorly                        0.8.0
tensortrade                     1.0.3
termcolor                       2.2.0
terminado                       0.17.1
tf2jax                          0.3.3
thinc                           8.1.9
threadpoolctl                   3.1.0
thriftpy2                       0.4.16
thundergbm                      0.3.17
tick                            0.7.0.1
tifffile                        2023.4.12
tinycss2                        1.2.1
toml                            0.10.2
tomli                           2.0.1
toolz                           0.12.0
torch                           1.13.1
torch-cluster                   1.6.0
torch-geometric                 2.2.0
torch-scatter                   2.1.0
torch-sparse                    0.6.16
torch-spline-conv               1.2.1
torchmetrics                    0.9.3
torchvision                     0.14.1
tornado                         6.3
tqdm                            4.64.1
trace-updater                   0.0.9.1
trading-calendars               2.1.1
traitlets                       5.9.0
treeinterpreter                 0.2.3
triad                           0.8.6
tsfresh                         0.20.0
tslearn                         0.5.3.2
tweepy                          4.10.0
typeguard                       3.0.2
typer                           0.3.2
typing_extensions               4.5.0
umap-learn                      0.5.3
urllib3                         1.26.14
virtualenv                      20.21.0
wasabi                          1.1.1
wcwidth                         0.2.6
webargs                         8.2.0
webencodings                    0.5.1
websocket-client                1.5.1
websockets                      10.4
Werkzeug                        2.2.3
wheel                           0.37.1
widgetsnbextension              4.0.7
wordcloud                       1.8.2.2
wrapt                           1.14.1
wrds                            3.1.6
wurlitzer                       3.0.3
xarray                          2023.1.0
xarray-einstats                 0.5.1
xgboost                         1.7.4
xlrd                            2.0.1
XlsxWriter                      3.1.0
yarl                            1.8.2
yellowbrick                     1.5
yfinance                        0.2.18
zict                            3.0.0
zipp                            3.15.0
zope.event                      4.6
zope.interface                  6.0
zstandard                       0.18.0
```

```
# Name                                Version
Accord                                3.6.0
Accord.Fuzzy                          3.6.0
Accord.MachineLearning                3.6.0
Accord.Math                           3.6.0
Accord.Statistics                     3.6.0
CloneExtensions                       1.3.0
Common.Logging                        3.4.1
Common.Logging.Core                   3.4.1
CsvHelper                             19.0.0
```

Deedle	2.1.0
DotNetZip	1.16.0
DynamicInterop	0.9.1
fasterflect	3.0.0
FSharp.Core	4.5.2
MathNet.Numerics	4.15.0
McMaster.Extensions.CommandLineUtils	2.6.0
Microsoft.IO.RecyclableMemoryStream	2.3.1
Microsoft.NET.Test.Sdk	16.9.4
Microsoft.TestPlatform.ObjectModel	16.9.4
Moq	4.16.1
NetMQ	4.0.1.6
Newtonsoft.Json	13.0.2
NodaTime	3.0.5
NUnit	3.13.3
NUnit3TestAdapter	4.2.1
protobuf-net	3.1.33
QLNet	1.12.0
QuantConnect.pythonnet	2.0.18
RestSharp	106.12.0
SharpZipLib	1.3.3
System.ComponentModel.Composition	6.0.0
Accord.Audio	3.6.0
Accord.Genetic	3.6.0
Accord.MachineLearning.GPL	3.6.0
Catalyst	1.0.31087
Catalyst.Models.English	1.0.30952
CNTK.CPUOnly	2.8.0-rc0.dev20200201
Google.OrTools	9.4.1874
LibTopoART	0.96.0
MathNet.Filtering	0.7.0
MathNet.Filtering.Kalman	0.7.0
MathNet.Spatial	0.6.0
Microsoft.Data.Analysis	0.19.1
Microsoft.ML	1.7.1
Microsoft.ML.CpuMath	1.7.1
Microsoft.ML.DataView	1.7.1
Microsoft.ML.Ensemble	0.19.1
Microsoft.ML.FastTree	1.7.1
Microsoft.ML.LightGbm	1.7.1
Microsoft.ML.Mkl.Components	1.7.1
Microsoft.ML.OnnxRuntime	1.12.1
Microsoft.ML.TensorFlow	1.7.1
Microsoft.ML.TimeSeries	1.7.1
NumSharp	0.30.0
Plotly.NET	3.0.1
Plotly.NET.Interactive	3.0.2
SharpLearning.AdaBoost	0.31.8
SharpLearning.Common.Interfaces	0.31.8
SharpLearning.Containers	0.31.8
SharpLearning.CrossValidation	0.31.8
SharpLearning.DecisionTrees	0.31.8
SharpLearning.Ensemble	0.31.8
SharpLearning.FeatureTransformations	0.31.8
SharpLearning.GradientBoost	0.31.8
SharpLearning.InputOutput	0.31.8
SharpLearning.Metrics	0.31.8
SharpLearning.Neural	0.31.8
SharpLearning.Optimization	0.31.8
SharpLearning.RandomForest	0.31.8
SharpLearning.XGBoost	0.31.8
SharpNeatLib	2.4.4
TensorFlow.Keras	0.7.0
TensorFlow.NET	0.70.1

Pomegranate Environment

If you use C#, this environment isn't available.

The Pomegranate environment supports the following libraries:

# Name	Version
absl-py	1.4.0
adagio	0.2.4
aesara	2.8.12
aiodns	3.0.0
aiohttp	3.8.4
aiohttp-cors	0.7.0
aiosignal	1.3.1
alabaster	0.7.13
ale-py	0.7.4
alembic	1.10.3
alpaca-trade-api	0.26
alphalens-reloaded	0.4.3
altair	4.2.2
ansi2html	1.8.0
antlr4-python3-runtime	4.11.1
anyio	3.6.2
appdirs	1.4.4
arch	5.3.1
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
arviz	0.15.1
astropy	5.2.1
asttokens	2.2.1
astunparse	1.6.3
async-timeout	4.0.2
asyncio-nats-client	0.11.5
attrs	23.1.0
autograd	1.5
autokeras	1.1.0
autoray	0.6.3
AutoROM	0.4.2
AutoROM.accept-rom-license	0.6.1
ax-platform	0.3.0
Babel	2.12.1
backcall	0.2.0
bayesian-optimization	1.4.2
beautifulsoup4	4.11.2
bleach	6.0.0
blessed	1.20.0
blis	0.7.9
blosc2	2.0.0
bokeh	2.4.3
boltons	23.0.0
botorch	0.8.2
Bottleneck	1.3.7
brotlipy	0.7.0
cachetools	5.3.0
captum	0.6.0
catalogue	2.0.8
catboost	1.1.1
category-encoders	2.6.0
ccxt	3.0.72
certifi	2022.12.7
cffi	1.15.1
chardet	5.1.0
charset-normalizer	2.0.4
check-shapes	1.0.0

click	7.1.2
clikit	0.6.2
cloudpickle	2.2.1
cmaes	0.9.1
cmdstanpy	1.1.0
colorama	0.4.6
colorcet	3.0.1
colorful	0.5.5
colorlog	6.7.0
colorlover	0.3.0
colour	0.1.5
comm	0.1.3
commonmark	0.9.1
conda	23.3.1
conda-content-trust	0.1.3
conda-package-handling	2.0.2
conda_package_streaming	0.7.0
confection	0.0.4
cons	0.4.5
contourpy	1.0.7
convertdate	2.4.0
copulae	0.7.7
copulas	0.8.0
cramjam	2.6.2
crashtest	0.3.1
creme	0.6.1
cryptography	38.0.4
cufflinks	0.17.3
cvxopt	1.3.0
cvxpy	1.3.0
cycler	0.11.0
cymem	2.0.7
Cython	0.29.33
darts	0.23.1
dash	2.9.3
dash-core-components	2.0.0
dash-cytoscape	0.3.0
dash-html-components	2.0.0
dash-table	5.0.0
dask	2023.4.0
deap	1.3.3
debugpy	1.5.1
decorator	5.1.1
defusedxml	0.7.1
Deprecated	1.2.13
dgl	1.0.1
dill	0.3.6
dimod	0.12.3
diskcache	5.6.1
distlib	0.3.6
distributed	2021.4.1
dm-tree	0.1.8
docutils	0.19
DoubleML	0.5.2
dtreeviz	2.2.1
dtw-python	1.3.0
dwave-cloud-client	0.10.3
dwave-drivers	0.4.4
dwave-greedy	0.3.0
dwave-hybrid	0.6.9
dwave-inspector	0.3.0
dwave-inspectorapp	0.3.0
dwave-neal	0.6.0
dwave-networkx	0.8.12
dwave-ocean-sdk	6.1.1
dwave-preprocessing	0.5.3
dwave-samplers	1.0.0
dwave-system	1.18.0
dwave-tabu	0.5.0
dwavebinarycsp	0.2.0
dx	0.1.22
ecos	2.0.12
EMD-signal	1.4.0
empyrical	0.5.5
empyrical-reloaded	0.5.9
en-core-web-md	3.5.0
en-core-web-sm	3.5.0
entrypoints	0.4
ephem	4.1.4
et-xmlfile	1.1.0
etuples	0.3.8
exceptiongroup	1.1.1
exchange-calendars	4.2.6
executing	1.2.0
ezyrb	1.3.0.post2304
fastai	2.7.11
fastai2	0.0.30
fastcore	1.5.29
fastdownload	0.0.7
fasteners	0.18
fastjsonschema	2.16.3
fastparquet	2023.2.0
fastprogress	1.0.3
fasttext	0.9.2
featuretools	0.23.1
filelock	3.12.0
findiff	0.9.2
finrl	0.3.1
FixedEffectModel	0.0.5
Flask	2.0.3
flatbuffers	23.3.3
fonttools	4.39.3
fracdiff	0.9.0
frozendict	2.3.7
frozenlist	1.3.3
fs	2.4.16
fsspec	2023.4.0
fst-pso	1.8.1
fugue	0.8.3
fugue-sql-antlr	0.1.6
future	0.18.3
fuzzy-c-means	1.6.3
FuzzyTM	2.0.5
gast	0.4.0
gensim	4.3.0
gevent	22.10.2
gluonts	0.12.3
google-api-core	2.11.0
google-auth	2.17.3
google-auth-oauthlib	0.4.6
google-pasta	0.2.0
googleapis-common-protos	1.59.0
gpflow	2.7.0
gplearn	0.4.2
gpustat	1.1
GPUtil	1.4.0
gpytorch	1.9.1
graphviz	0.8.4
greenlet	2.0.2
grpcio	1.54.0

gym	0.21.0
Gymnasium	0.26.3
gymnasium-notices	0.0.1
h2o	3.40.0.1
h5netcdf	1.1.0
h5py	3.8.0
hijri-converter	2.2.4
hmmlearn	0.2.8
holidays	0.23
holoviews	1.15.4
homebase	1.0.1
hopcroftkarp	1.2.5
html5lib	1.1
httpstan	4.9.1
hurst	0.0.5
hvplot	0.8.2
hyperopt	0.2.7
idna	3.4
iisignature	0.24
ijson	3.2.0.post0
imageio	2.27.0
imagesize	1.4.1
imbalanced-learn	0.10.1
importlib-metadata	4.13.0
importlib-resources	5.12.0
iniconfig	2.0.0
injector	0.20.1
interpret	0.3.0
interpret-core	0.3.2
ipykernel	6.22.0
ipython	8.12.0
ipython-genutils	0.2.0
ipywidgets	8.0.4
itsdangerous	2.1.2
jax	0.4.4
jaxlib	0.4.4
jedi	0.18.2
Jinja2	3.1.2
joblib	1.2.0
jqdatasdk	1.8.11
json5	0.9.11
jsonpatch	1.32
jsonpointer	2.0
jsonschema	4.17.3
jupyter	1.0.0
jupyter-bokeh	3.0.5
jupyter-console	6.6.3
jupyter-dash	0.4.2
jupyter-resource-usage	0.7.2
jupyter-server	1.24.0
jupyter_client	8.2.0
jupyter_core	5.3.0
jupyterlab	3.4.4
jupyterlab-pygments	0.2.2
jupyterlab-widgets	3.0.7
jupyterlab_server	2.22.1
keract	4.5.1
keras	2.11.0
keras-nlp	0.4.1
keras-rl	0.4.2
keras-tuner	1.3.5
kiwisolver	1.4.4
kmapper	2.0.1
korean-lunar-calendar	0.3.1
kt-legacy	1.0.5
langcodes	3.3.0
lark	1.1.5
lazy_loader	0.2
lazypredict	0.2.12
libclang	16.0.0
lightgbm	3.3.5
lime	0.2.0.1
line-profiler	4.0.2
linear-operator	0.3.0
littleutils	0.2.2
livelossplot	0.5.5
llvmlite	0.38.1
locket	1.0.0
logical-unification	0.4.5
LunarCalendar	0.0.9
lxml	4.9.2
lz4	4.3.2
Mako	1.2.4
Markdown	3.4.3
MarkupSafe	2.1.2
marshmallow	3.19.0
matplotlib	3.7.0
matplotlib-inline	0.1.6
miniful	0.0.6
miniKanren	1.0.3
minorminer	0.2.9
mistune	2.0.5
mljar-supervised	0.11.5
mlxtend	0.21.0
mmh3	2.5.1
modin	0.18.1
mpi4py	3.1.4
mplfinance	0.12.9b7
mpmath	1.2.1
msgpack	1.0.4
mthree	2.4.0
multidict	6.0.4
multipledispatch	0.6.0
multiprocess	0.70.14
multitasking	0.0.11
murmurhash	1.0.9
mxnet	1.9.1
nbclassic	0.5.5
nbclient	0.7.3
nbconvert	7.3.1
nbeats-keras	1.8.0
nbeats-pytorch	1.8.0
nbformat	5.8.0
nest-asyncio	1.5.6
networkx	2.8.8
neural-tangents	0.6.2
neuralprophet	0.5.0
nfoursid	1.0.1
nltk	3.8.1
nose	1.3.7
notebook	6.5.4
notebook_shim	0.2.2
ntlm-auth	1.5.0
numba	0.55.1
numerapi	2.13.2
numexpr	2.8.4
numpy	1.21.5
numpydoc	1.5.0
nvidia-cublas-cu11	11.10.3.66
nvidia-cuda-nvrtc-cu11	11.7.99

nvidia-cuda-runtime-cu11	11.7.99
nvidia-cudnn-cu11	8.5.0.96
nvidia-ml-py	11.525.112
oauthlib	3.2.2
openai	0.26.5
opencensus	0.11.2
opencensus-context	0.1.3
opencv-python	4.7.0.72
openpyxl	3.1.1
opt-einsum	3.3.0
optuna	3.1.0
orjson	3.8.10
ortools	9.4.1874
osqp	0.6.2.post9
outdated	0.2.2
packaging	23.1
pandas	1.5.3
pandas-datareader	0.10.0
pandas-flavor	0.5.0
pandas-market-calendars	4.1.4
pandas-ta	0.3.14b0
pandocfilters	1.5.0
panel	0.14.3
param	1.13.0
parso	0.8.3
partd	1.4.0
pastel	0.2.1
pathos	0.3.0
pathy	0.10.1
patsy	0.5.3
pbr	5.11.1
penaltymodel	1.0.2
PennyLane	0.29.0
PennyLane-Lightning	0.29.0
PennyLane-qiskit	0.29.0
persim	0.3.1
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.5.0
pingouin	0.5.3
pip	22.0.4
pkgutil_resolve_name	1.3.10
platformdirs	3.2.0
plotly	5.13.1
plotly-resampler	0.8.3.2
plucky	0.4.3
pluggy	1.0.0
ply	3.11
pmdarima	2.0.2
polars	0.16.9
pomegranate	0.14.8
pox	0.3.2
ppft	1.7.6.6
pprofile	2.1.0
ppscore	1.2.0
preshed	3.0.8
prometheus-client	0.16.0
prompt-toolkit	3.0.38
property-cached	1.6.4
prophet	1.1.2
protobuf	3.19.6
psutil	5.9.5
psycopg2-binary	2.9.6
ptvsd	4.3.2
ptyprocess	0.7.0
PuLP	2.7.0
pure-eval	0.2.2
py-cpuinfo	9.0.0
py-heat	0.0.6
py-heat-magic	0.0.2
py-lets-be-rational	1.0.1
py-spy	0.3.14
py-vollib	1.0.1
py4j	0.10.9.7
pyaml	21.10.1
pyarrow	11.0.0
pyasn1	0.4.8
pyasn1-modules	0.2.8
pybind11	2.10.4
pycares	4.3.0
pycosat	0.6.4
pycparser	2.21
pyct	0.5.0
pydantic	1.10.7
pyDeprecate	0.3.2
pydevd-pycharm	201.8538.36
pydmd	0.4.0.post2301
pyerfa	2.0.0.3
pyfolio	0.9.2
pyfolio-reloaded	0.9.5
pyFUME	0.2.25
Pygments	2.15.1
pykalman	0.9.5
pylev	1.4.0
pyluach	2.2.0
pymannkendall	1.4.3
pymc	5.0.2
pymdptoolbox	4.0b3
PyMeeus	0.5.12
PyMySQL	1.0.3
pynndescent	0.5.9
pyod	1.0.9
Pyomo	6.5.0
pyOpenSSL	22.0.0
pyparsing	3.0.9
pyportfolioopt	1.5.4
pyqubo	1.3.1
pyrb	1.0.1
pyro-api	0.1.2
pyro-ppl	1.8.4
pyrsistent	0.19.3
pysimdjson	5.0.2
PySocks	1.7.1
pystan	3.6.0
pytensor	2.9.1
pytest	7.3.1
python-dateutil	2.8.2
python-statemachine	1.0.3
pytorch-ignite	0.4.11
pytorch-lightning	1.7.4
pytz	2023.3
pyvinecopulib	0.6.2
pyviz-comms	2.2.1
PyWavelets	1.4.1
PyYAML	6.0
pyzmq	25.0.2
qdldl	0.1.7
qiskit	0.41.1
qiskit-aer	0.11.2
qiskit-ibmq-provider	0.20.1

qiskit-terra	0.23.2
qpd	0.4.1
qtconsole	5.4.2
QtPy	2.3.1
quadprog	0.1.11
quantecon	0.6.0
QuantLib	1.29
QuantStats	0.0.59
rauth	0.7.3
ray	2.3.0
rectangle-packer	2.0.1
regex	2023.3.23
requests	2.28.1
requests-ntlm	1.1.0
requests-oauthlib	1.3.1
retrying	1.3.4
networkx	0.12.1
rich	12.4.4
ripser	0.6.4
Riskfolio-Lib	4.0.3
riskparityportfolio	0.4
river	0.14.0
rsa	4.9
ruamel.yaml	0.17.21
ruamel.yaml.clib	0.2.6
ruptures	1.1.7
rustworkx	0.12.1
SALib	1.4.7
scikeras	0.10.0
scikit-image	0.19.3
scikit-learn	1.2.1
scikit-learn-extra	0.2.0
scikit-multiflow	0.5.3
scikit-optimize	0.9.0
scikit-plot	0.3.7
scikit-tda	1.0.0
scipy	1.8.0
scs	3.2.3
sdeint	0.3.0
seaborn	0.12.2
semantic-version	2.10.0
Send2Trash	1.8.0
setuptools	56.0.0
setuptools-scm	7.1.0
shap	0.41.0
simpful	2.10.0
simplejson	3.19.1
simpy	4.0.1
six	1.16.0
sklearn-json	0.1.0
skope-rules	1.0.1
sktime	0.16.1
slicer	0.0.7
smart-open	6.3.0
sniffio	1.3.0
snowballstemmer	2.2.0
sortedcontainers	2.4.0
soupsieve	2.4.1
spacy	3.5.0
spacy-legacy	3.0.12
spacy-loggers	1.0.4
Sphinx	6.1.3
sphinxcontrib-applehelp	1.0.4
sphinxcontrib-devhelp	1.0.2
sphinxcontrib-htmlhelp	2.0.1
sphinxcontrib-jsmath	1.0.1
sphinxcontrib-qthelp	1.0.3
sphinxcontrib-serializinghtml	1.1.5
SQLAlchemy	1.4.47
sqlglot	11.5.5
srsly	2.4.6
ssm	0.0.1
stable-baselines3	1.7.0
stack-data	0.6.2
statsforecast	1.5.0
statsmodels	0.13.5
stellargraph	1.2.1
stevedore	5.0.0
stochastic	0.7.0
stockstats	0.5.2
stumpy	1.11.1
symengine	0.10.0
sympy	1.11.1
ta	0.10.2
TA-Lib	0.4.18
tables	3.8.0
tabulate	0.8.10
tadasets	0.0.4
tbats	1.1.3
tblib	1.7.0
tenacity	8.2.2
tensorboard	2.11.2
tensorboard-data-server	0.6.1
tensorboard-plugin-wit	1.8.1
tensorboardX	2.6
tensorflow	2.11.0
tensorflow-addons	0.19.0
tensorflow-decision-forests	1.2.0
tensorflow-estimator	2.11.0
tensorflow-hub	0.13.0
tensorflow-io-gcs-filesystem	0.32.0
tensorflow-probability	0.19.0
tensorflow-text	2.11.0
tensorly	0.8.0
tensortrade	1.0.3
termcolor	2.2.0
terminado	0.17.1
tf2jax	0.3.3
thinc	8.1.9
threadpoolctl	3.1.0
thriftpy2	0.4.16
thundergbm	0.3.17
tick	0.7.0.1
tifffile	2023.4.12
tigramite	5.1.0.3
tinycss2	1.2.1
toml	0.10.2
tomli	2.0.1
toolz	0.12.0
torch	1.13.1
torch-cluster	1.6.0
torch-geometric	2.2.0
torch-scatter	2.1.0
torch-sparse	0.6.16
torch-spline-conv	1.2.1
torchmetrics	0.9.3
torchvision	0.14.1
tornado	6.3
tqdm	4.64.1
trace-updater	0.0.9.1

trading-calendars	2.1.1
traitlets	5.9.0
treeinterpreter	0.2.3
triad	0.8.6
tsfresh	0.20.0
tslearn	0.5.3.2
tweepy	4.10.0
typeguard	3.0.2
typer	0.3.2
typing_extensions	4.5.0
umap-learn	0.5.3
urllib3	1.26.14
virtualenv	20.21.0
wasabi	1.1.1
wcwidth	0.2.6
webargs	8.2.0
webencodings	0.5.1
websocket-client	1.5.1
websockets	10.4
Werkzeug	2.2.3
wheel	0.37.1
widgetsnbextension	4.0.7
wordcloud	1.8.2.2
wrapt	1.14.1
wrds	3.1.6
wurlitzer	3.0.3
xarray	2023.1.0
xarray-einstats	0.5.1
xgboost	1.7.4
xlrd	2.0.1
XlsxWriter	3.1.0
yarl	1.8.2
yellowbrick	1.5
yfinance	0.2.18
zict	3.0.0
zipp	3.15.0
zope.event	4.6
zope.interface	6.0
zstandard	0.18.0

Hudson & Thames Environment

If you use C#, this environment isn't available.

The Hudson & Thames environment supports the following libraries:

# Name	Version
absl-py	1.3.0
analytics-python	1.4.0
ansi2html	1.8.0
anyio	3.6.2
appdirs	1.4.4
arch	4.16.1
argon2-cffi	21.3.0
argon2-cffi-bindings	21.2.0
asttokens	2.2.1
astunparse	1.6.3
attrs	22.2.0
Babel	2.11.0
backcall	0.2.0
backoff	1.10.0
beautifulsoup4	4.11.1
bleach	5.0.1
Brotli	1.0.9
bs4	0.0.1
cachetools	5.2.0
certifi	2022.12.7
cffi	1.15.1
charset-normalizer	2.1.1
click	8.1.3
cloudpickle	2.2.0
clr-loader	0.1.6
comm	0.1.2
cramjam	2.6.2
cssselect	1.2.0
cvxpy	1.2.0
cycler	0.11.0
Cython	0.29.28
dash	1.21.0
dash-bootstrap-components	0.13.1
dash-core-components	1.17.1
dash-cytoscape	0.2.0
dash-html-components	1.1.4
dash-table	4.12.0
dask	2022.12.1
debugpy	1.6.4
decorator	4.4.2
defusedxml	0.7.1
distributed	2022.12.1
ecos	2.0.12
entrypoints	0.4
executing	1.2.0
fake-useragent	1.1.1
fastjsonschema	2.16.2
fastparquet	0.8.1
Flask	2.1.3
Flask-Compress	1.13
flatbuffers	22.12.6
fsspec	2022.11.0
future	0.18.2
gast	0.5.3
getmac	0.8.3
google-auth	2.15.0
google-auth-oauthlib	0.4.6
google-pasta	0.2.0
grpcio	1.51.1
h5py	3.7.0
HeapDict	1.0.1
ht-auth	0.1.0
idna	3.4
importlib-metadata	5.2.0
importlib-resources	5.10.1
ipykernel	6.19.4
ipython	8.7.0
ipython-genutils	0.2.0
itsdangerous	2.1.2
jedi	0.18.2
Jinja2	3.1.2
joblib	1.2.0
json5	0.9.10
jsonschema	4.17.3
jupyter_client	7.4.8
jupyter_core	5.1.1
jupyter-dash	0.4.2
jupyter-server	1.23.4
jupyterlab	3.4.4
jupyterlab-pygments	0.2.2
jupyterlab_server	2.17.0
keras	2.8.0

Keras-Preprocessing	1.1.2
kiwisolver	1.4.4
libclang	14.0.6
llvmlite	0.39.1
locket	1.0.0
lxml	4.6.2
Markdown	3.4.1
MarkupSafe	2.1.1
matplotlib	3.4.3
matplotlib-inline	0.1.6
matrixprofile	1.1.10
mistune	2.0.4
mlfinlab	1.6.0
monotonic	1.6
mpmath	1.2.1
msgpack	1.0.4
multitasking	0.0.11
nbclassic	0.4.8
nbclient	0.7.2
nbconvert	7.2.7
nbformat	5.7.1
nest-asyncio	1.5.6
networkx	2.5.1
notebook	6.5.2
notebook_shim	0.2.2
numba	0.56.4
numpy	1.20.1
oauthlib	3.2.2
opt-einsum	3.3.0
osqp	0.6.2.post8
packaging	22.0
pandas	1.1.5
pandocfilters	1.5.0
parse	1.19.0
parso	0.8.3
partd	1.3.0
patsy	0.5.3
pexpect	4.8.0
pickleshare	0.7.5
Pillow	9.3.0
pip	22.0.4
pkgutil_resolve_name	1.3.10
platformdirs	2.6.1
plotly	5.11.0
pmdarima	1.8.5
POT	0.8.2
prometheus-client	0.15.0
prompt-toolkit	3.0.36
property-cached	1.6.4
protobuf	3.11.2
psutil	5.9.4
ptyprocess	0.7.0
pure-eval	0.2.2
pyasn1	0.4.8
pyasn1-modules	0.2.8
pybind11	2.10.1
pycparser	2.21
pyee	8.2.2
Pygments	2.13.0
pykalman	0.9.5
pyparsing	3.0.9
pyppeteer	1.0.2
pyquery	2.0.0
pyrsistent	0.19.2
python-dateutil	2.8.2
pytz	2022.7
pyvinecopulib	0.5.5
PyYAML	6.0
pyzmq	24.0.1
qdldl	0.1.5.post2
requests	2.28.1
requests-html	0.10.0
requests-oauthlib	1.3.1
retrying	1.3.4
rsa	4.9
scikit-learn	0.24.2
scipy	1.9.3
scs	3.2.0
seaborn	0.11.1
Send2Trash	1.8.0
setuptools	56.0.0
setuptools-scm	7.1.0
six	1.16.0
sniffio	1.3.0
sortedcontainers	2.4.0
soupsieve	2.3.2.post1
stack-data	0.6.2
statsmodels	0.13.5
stumpy	1.11.1
TA-Lib	0.4.18
tblib	1.7.0
tenacity	8.1.0
tensorboard	2.8.0
tensorboard-data-server	0.6.1
tensorboard-plugin-wit	1.8.1
tensorflow	2.8.0
tensorflow-io-gcs-filesystem	0.29.0
termcolor	2.1.1
terminado	0.17.1
tf-estimator-nightly	2.8.0.dev2021122109
threadpoolctl	3.1.0
tinycss2	1.2.1
tomli	2.0.1
toolz	0.12.0
tornado	6.2
tqdm	4.64.1
traitlets	5.8.0
tsfresh	0.19.0
typing_extensions	4.4.0
urllib3	1.26.13
w3lib	2.1.1
wcwidth	0.2.5
webencodings	0.5.1
websocket-client	1.4.2
websockets	10.4
Werkzeug	2.0.0
wheel	0.38.4
wrapt	1.14.1
yahoo-fin	0.8.6
yfinance	0.1.63
zict	2.2.0
zipp	3.11.0

Request New Libraries

To request a new library, contact us . We will add the library to the queue for review and deployment. Since the libraries run on our servers, we need to ensure they are secure and won't cause harm. The process of adding new libraries takes 2-4 weeks to complete. View the list of libraries currently under review on the Issues list of the Lean GitHub repository .

6.10 LEAN Engine Versions

Introduction

The latest master branch on the LEAN GitHub repository is the default engine branch that runs the backtesting, research, and live trading nodes in QuantConnect Cloud. The latest version of LEAN is generally the safest as it includes all bug fixes.

Trading Firm or Institution tier users concerned for stability can elect to use older or custom versions of LEAN in the IDE. These are powered by the QuantConnect/LEAN Github Branches . We use a continuous deployment process to ship custom branches to production for trading. To create a custom version of LEAN, make a pull request to LEAN which will be reviewed by our team.

Change Branches

Follow these steps to change the LEAN engine branch that runs your backtests and live trading algorithms:

1. Open a project .
2. In the Project panel, click the LEAN Engine field and then click a branch from the drop-down menu.
3. (Optional) Click About Version to display the branch description.
4. If you want to always use the master branch, select the Always use Master Branch check box.
5. Click Select .

Changing the Lean engine branch only affects the current project. If you create a new project , the new project will use the master branch by default.

Request New Branches

Before starting a pull-request to create a new branch, contact us to discuss the goals of the project.

7 Research

The Research Environment is a Jupyter notebook -based environment where you can access our data through the `QuantBook` class instead of through the `QCAlgorithm` class in a backtest. The environment supports both Python and C#. If you use Python, you can import code from the code files in your project into the Research Environment to aid development.

Getting Started

Learn the basics

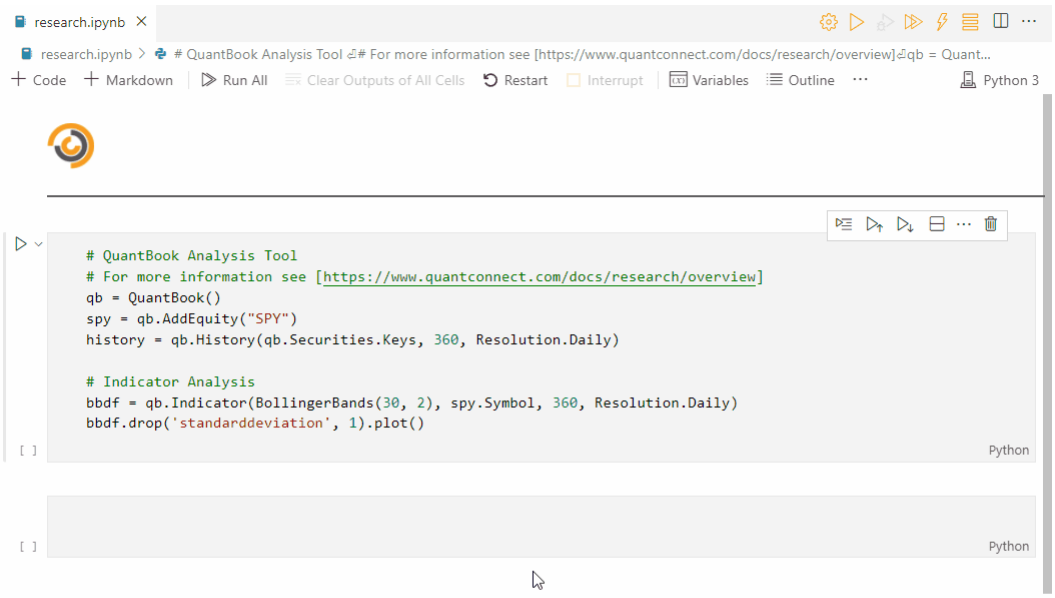
Deployment

Spin up a notebook

See Also

Research Environment Product
Charting
Object Store

7.1 Getting Started



Introduction

The Research Environment is a Jupyter notebook -based, interactive commandline environment where you can access our data through the `QuantBook` class. The environment supports both Python and C#. If you use Python, you can import code from the code files in your project into the Research Environment to aid development.

Before you run backtests, we recommend testing your hypothesis in the Research Environment. It's easier to perform data analysis and produce plots in the Research Environment than in a backtest.

Before backtesting or live trading with machine learning models, you may find it beneficial to train them in the Research Environment, save them in the ObjectStore, and then load them from the ObjectStore into the backtesting and live trading environment

In the Research Environment, you can also use the QuantConnect API to import your backtest results for further analysis.

Note: This chapter is an introduction to the Research Environment for the Algorithm Lab. For more comprehensive information on using research notebooks, see our dedicated Research Environment documentation.

Example

The following snippet demonstrates how to use the Research Environment to plot the price and Bollinger Bands of the S&P 500 index ETF, SPY:

The following snippet demonstrates how to use the Research Environment to print the price of the S&P 500 index ETF, SPY:

```
// Load the required assembly files and data types
#load "../Initialize.csx"
#load "../QuantConnect.csx"
using QuantConnect;
using QuantConnect.Data;
using QuantConnect.Algorithm;
using QuantConnect.Research;

// Create a QuantBook
var qb = new QuantBook();

// Create a security subscription
var symbol = qb.AddEquity("SPY").Symbol;

// Request some historical data
var history = qb.History(symbol, 70, Resolution.Daily);

foreach (var tradeBar in history)
{
    Console.WriteLine($"{tradeBar.EndTime} :: {tradeBar.ToString()}");
}
```

```
# Create a QuantBook
qb = QuantBook()

# Create a security subscription
spy = qb.AddEquity("SPY")

# Request some historical data
history = qb.History(qb.Securities.Keys, 360, Resolution.Daily)

# Calculate the Bollinger Bands
bbdf = qb.Indicator(BollingerBands(30, 2), spy.Symbol, 360, Resolution.Daily)

# Plot the data
bbdf.drop('standarddeviation', 1).plot()
```

Open Notebooks

Each new project you create contains a notebook file by default. Follow these steps to open the notebook:

1. Open the project .
2. In the right navigation menu, click the Explorer icon.
3. In the Explorer panel, expand the Workspace (Workspace) section.
4. Click the Research.ipynb research.ipynb file.

Run Notebook Cells

Notebooks are a collection of cells where you can execute code snippets or write MarkDown.

The following describes some helpful keyboard shortcuts to speed up your research:

Keyboard Shortcut	Description
Shift+Enter	Run the selected cell.
a	Insert a cell above the selected cell.
b	Insert a cell below the selected cell.
x	Cut the selected cell.
v	Paste the copied or cut cell.
z	Undo cell actions.

Stop Nodes

You need stop node permissions to stop research nodes in the cloud.

Follow these steps to stop a research node:

1. Open the project .
2. In the right navigation menu, click the Resources icon.
3. Click the stop button next to the research node you want to stop.

Add Notebooks

Follow these steps to add notebook files to a project:

1. Open the project .
2. In the right navigation menu, click the Explorer icon.
3. In the Explorer panel, expand the Workspace (Workspace) section.
4. Click the New File icon.
5. Enter fileName .ipynb .
6. Press Enter .

Rename Notebooks

Follow these steps to rename a notebook in a project:

1. Open the project .
2. In the right navigation menu, click the Explorer icon.
3. In the Explorer panel, right-click the notebook you want to rename and then click Rename .
4. Enter the new name and then press Enter .

Delete Notebooks

Follow these steps to delete a notebook in a project:

1. Open the project .
2. In the right navigation menu, click the Explorer icon.
3. In the Explorer panel, right-click the notebook you want to delete and then click Delete Permanently .
4. Click Delete .

Learn Jupyter

The following table lists some helpful resources to learn Jupyter:

Type	Name	Producer
Text	Jupyter Tutorial	tutorialspoint
Text	Jupyter Notebook Tutorial: The Definitive Guide	DataCamp
Text	An Introduction to DataFrame	Microsoft Developer Blogs

7.2 Deployment

Introduction

This page is an introduction to the Research Environment for the Algorithm Lab. For more comprehensive information on using research notebooks, see the Research Environment documentation product.

Resources

Research nodes enable you to spin up an interactive, command-line, Jupyter notebook environments. Several models of research nodes are available. More powerful research nodes allow you to handle more data and run faster computations in your notebooks. The following table shows the specifications of the research node models:

Name	Number of Cores	Processing Speed (GHz)	RAM (GB)	GPU
R1-4	1	2.4	4	0
R2-8	2	2.4	8	0
R4-12	4	2.4	12	0
R4-16-GPU	4	3	16	1/3
R8-16	8	2.4	16	0

Refer to the Pricing page to see the price of each research node model. You get one free R1-4 research node in your first organization, but the node is replaced when you subscribe to a paid research node in the organization.

You need an idle research node in your organization to launch the Research Environment. You can view all of your organization's nodes from the Resources page or the Resources panel in the IDE. The Resources panel lets you select a specific research node to use. The CPU nodes are available on a fair usage basis. The GPU nodes can be shared with a maximum of three members. Depending on the server load, you may use all of the GPU's processing power.

Sharing

The Research Environment does not currently support simultaneous coding between two peers. You can view the notebook of your colleague, but you can't edit the notebook at the same time. We recommend that only one person opens a notebook because your work may not save if several people open the same notebook. To enable your team members to see your notebook, add them to the project.

Sharing your notebooks is helpful when publishing your findings online, getting feedback on your research process, and explaining your methodology to others. To share a notebook with members not in your organization, run a backtest, generate a link to share the backtest, and then give the backtest link to the other person. They will be able to clone the project and launch the Research Environment using the notebook files.

Historical Data

On our platform, you can access historical data for all of the asset classes that we support. We have datasets for Equities, Options, Futures, Crypto, Forex, CFD, Indices, and alternative data. View the Dataset Market to see all of the datasets that we support. To import custom data, see Custom Data.

Look-Ahead Bias

Look-ahead bias occurs when an algorithm makes decisions using data that would not have yet been available. As you work in the Research Environment, take measures to reduce look-ahead bias in your workflow. If look-ahead bias seeps into your research, you may find profitable results on historical data, however, you will not be able to apply your findings in a live trading algorithm because the data does not exist in real-time.

An example of look-ahead bias is using the closing price to make trading decisions and filling your orders at the same closing price. This can happen in the Research Environment because all of the data is available at the same time. This type of look-ahead bias cannot occur when backtesting with Lean because the data is fed into your algorithm in discrete slices of time and an order is filled using a slice that occurs after the order was placed.

A second example of look-ahead bias occurs when researchers test a model on the same dataset that was used to train the model. In this situation, the model is trained using data that would not have been available when the model was created in practice. A third example of look-ahead bias is when researchers select a universe to apply their trading strategy to based on the past performance of the universe.

8 Backtesting

Backtesting is the process of simulating a trading algorithm on historical data. By running a backtest, you can measure how the algorithm would have performed in the past. Although past performance doesn't guarantee future results, an algorithm that has a proven track record can provide investors with more confidence when deploying to live trading than an algorithm that hasn't performed favorably in the past. Use the QuantConnect platform to run your backtests because we have institutional-grade datasets, an open-source backtesting engine that's constantly being improved, cloud servers to execute the backtests, and the backtesting hardware is maintained 24/7 by QuantConnect engineers.

Getting Started

Learn the basics

Research Guide

Backtesting 101

Deployment

Cloud backtests with institutional-grade data

Results

Gathering historical results

Debugging

Solve coding errors

Report

Measuring algorithm performance

Engine Performance

Track LEAN performance metrics

See Also

Writing Algorithms
Optimization
Resources

8.1 Getting Started

Introduction

Backtesting is the process of simulating a trading algorithm on historical data. By running a backtest, you can measure how the algorithm would have performed in the past. Although past performance doesn't guarantee future results, an algorithm that has a proven track record can provide investors with more confidence when deploying to live trading than an algorithm that hasn't performed favorably in the past. Use the QuantConnect platform to run your backtests because we have institutional-grade datasets, an open-source backtesting engine that's constantly being improved, cloud servers to execute the backtests, and the backtesting hardware is maintained 24/7 by QuantConnect engineers.

Run Backtests

To run a backtest, open a project and then click the Backtest icon. If the project successfully builds, "Received backtest backtestName request" displays. If the backtest successfully launches, the IDE displays the backtest results page in a new tab. If the backtest fails to launch due to coding errors, the new tab displays the error. As the backtest executes, you can refresh or close the IDE without interfering with the backtest because it runs on our cloud servers.

View All Backtests

Follow these steps to view all of the backtests of a project:

1. Open the project that contains the backtests you want to view.
2. In the top-right corner of the IDE, click the Backtest Results icon.

   A table containing all of the backtest results for the project is displayed. If there is a play icon to the left of the name, it's a backtest result . If there is a fast-forward icon next to the name, it's an optimization result .

3. (Optional) In the top-right corner, select the Show field and then select one of the options from the drop-down menu to filter the table by backtest or optimization results.
4. (Optional) In the bottom-right corner, click the Hide Error check box to remove backtest and optimization results from the table that had a runtime error.
5. (Optional) Use the pagination tools at the bottom to change the page.
6. (Optional) Click a column name to sort the table by that column.
7. Click a row in the table to open the results page of that backtest or optimization.

Rename Backtests

We give an arbitrary name (for example, "Smooth Apricot Chicken") to your backtest result files, but you can follow these steps to rename them:

1. Open the backtest history of the project.
2. Hover over the backtest you want to rename and then click the pencil icon that appears.
3. Enter the new backtest name and then click OK .

8.2 Research Guide

Introduction

QuantConnect aims to teach and inspire our community to create high-performing algorithmic trading strategies. We measure our success by the profits created by the community through their live trading. As such, we try to build the best quantitative research techniques possible into the product to encourage a robust research process.

Hypothesis-Driven Research

We recommend you develop an algorithmic trading strategy based on a central hypothesis. You should develop an algorithm hypothesis at the start of your research and spend the remaining time exploring how to test your theory. If you find yourself deviating from your core theory or introducing code that isn't based around that hypothesis, you should stop and go back to thesis development.

Wang et al. (2014) illustrate the danger of creating your hypothesis based on test results. In their research, they examined the earnings yield factor in the technology sector over time. During 1998-1999, before the tech bubble burst, the factor was unprofitable. If you saw the results and then decided to bet against the factor during 2000-2002, you would have lost a lot of money because the factor performed extremely well during that time.

Hypothesis development is somewhat of an art and requires creativity and great observation skills. It is one of the most powerful skills a quant can learn. We recommend that an algorithm hypothesis follow the pattern of cause and effect. Your aim should be to express your strategy in the following sentence:

> *A change in {cause} leads to an {effect}.*

To search for inspiration, consider causes from your own experience, intuition, or the media. Generally, causes of financial market movements fall into the following categories:

- Human psychology
- Real-world events/fundamentals
- Invisible financial actions

Consider the following examples:

Cause	*leads to*	Effect
Share class stocks are the same company, so any price divergence is irrational...		A perfect pairs trade. Since they are the same company, the price will revert.
New stock addition to the S&P500 Index causes fund managers to buy up stock...		An increase in the price of the new asset in the universe from buying pressure.
Increase in sunshine-hours increases the production of oranges...		An increase in the supply of oranges, decreasing the price of Orange Juice Futures.
Allegations of fraud by the CEO causes investor faith in the stock to fall...		A collapse of stock prices for the company as people panic.
FDA approval of a new drug opens up new markets for the pharmaceutical company...		A jump in stock prices for the company.
Increasing federal interest rates restrict lending from banks, raising interest rates...		Restricted REIT leverage and lower REIT ETF returns.

There are millions of potential alpha strategies to explore, each of them a candidate for an algorithm. Once you have chosen a strategy, we recommend exploring it for no more than 8-32 hours, depending on your coding ability.

Research Panel

We launched the Research Guide in 2019 to inform you about common quantitative research pitfalls. It displays a power gauge for the number of backtests performed, the number of parameters used, and the time invested in the strategy. These measures can give a ballpark estimate of the overfitting risk of the project. Generally, as a strategy becomes more overfit on historical data, it is less likely to perform well in live trading. These properties were selected based on the recommended best practices of the global quantitative research community.

Research guide

Restricting Backtests

According to current research, the number of backtests performed on an idea should be limited to prevent overfitting. In theory, each backtest performed on an idea moves it one step closer to being overfitted as you are testing and selecting for strategies written into your code instead of being based on a central thesis. For more information, see the paper Probability of Backtest Overfitting (Bailey, Borwein, Jim Zho, & LÃ³pez de Prado, 2015).

QuantConnect does not restrict the number of backtests performed on a project, but we have implemented the counter as a guide for your reference. Your coding skills are a factor in how many backtests constitute overfitting, so if you are a new programmer you can increase these targets.

Backtest Count Overfit Reference

0-30: **Likely Not Overfit** 30-70: **Possibly Overfitting** 70+ **Probably Overfitting**

Reducing Strategy Parameters

With just a handful of parameters, it is possible to create an algorithm that perfectly models historical markets. Current research suggests keeping your parameter count to a minimum to decrease the risk of overfitting.

Parameter Overfit Reference

0-10: **Likely Not Overfit** 10-20: **Possibly Overfitting** 20+ **Probably Overfitting**

Limiting Research Time Invested

As you spend more time on one algorithm, research suggests you are more likely to be overfitting the strategy to the data. It is common to become attached to an idea and spend weeks or months to perform well in a backtest. Assuming you are a proficient coder who fully understands the QuantConnect API, we recommend no more than 16 hours of work per experiment. In theory, within two full working days, you should be able to test a single hypothesis thoroughly.

Research Time Overfitting Reference

0-8 Hours: **Likely Not Overfit** 8-16 Hours: **Possibly Overfitting** 16 Hours+ **Probably Overfitting**

Parameter Detection

Using parameters is almost unavoidable, but a strategy trends toward being overfitted as more parameters get added or fine-tuned. Adding or optimizing parameters should only be done by a robust methodology such as walk-forward optimization. The parameter detection system is a general guide to inform you of how many parameters are present in the algorithm. It looks for criteria to warn that code is potentially a parameter. The following table shows the criteria for parameters:

Parameter Types	Example Instances
Numeric Comparison	Numeric operators used to compare numeric arguments: <= < > >=
Time Span	Setting the interval of `TimeSpan` or `timedelta`
Order Event	Inputting numeric arguments when placing orders
Scheduled Event	Inputting numeric arguments when scheduling an algorithm event to occur
Variable Assignment	Assigning numeric values to variables
Mathematical Operation	Any mathematical operation involving explicit numbers
Lean API	Numeric arguments passed to Indicators, Consolidators, Rolling Windows, etc.

The following table shows common expressions that are not parameters:

Non-Parameter Types	Example Instances
Common APIs	`SetStartDate` , `SetEndDate` , `SetCash` , etc.
Boolean Comparison	Testing for True or False conditions
String Numbers	Numbers formatted as part of `Log` or `Debug` statements
Variable Names	Any variable names that use numbers as part of the name (for example, `smaIndicator200`)
Common Functions	Rounding, array indexing, boolean comparison using 1/0 for True/False, etc.

Overfitting

Overfitting occurs when you fine-tune the parameters of an algorithm to fit the detail and noise of backtesting data to the extent that it negatively impacts the performance of the algorithm on new data. The problem is that the parameters don't necessarily apply to new data and thus negatively impact the algorithm's ability to generalize and trade well in all market conditions. The following table shows ways that overfitting can manifest itself:

Data Practice	Description
Data Dredging	Performing many statistical tests on data and only paying attention to those that come back with significant results.
Hyper-Tuning Parameters	Manually changing algorithm parameters to produce better results without altering the test data.
Overfit Regression Models	Regression, machine learning, or other statistical models with too many variables will likely introduce overfitting to an algorithm.
Stale Testing Data	Not changing the backtesting data set when testing the algorithm. Any improvements might not be able to be generalized to different datasets.

An algorithm that is dynamic and generalizes to new data is more valuable to funds and individual investors. It is more likely to survive across different market conditions and apply to new asset classes and markets.

8.3 Deployment

Introduction

Deploy a backtest to simulate your trading algorithm on our cloud servers. Since the same Lean engine is used to run backtests and live trading algorithms, it's easy to transition from backtesting to live trading once you are satisfied with the historical performance of your algorithm. If you find any issues with Lean, the historical data, or the website when running backtests, we'll resolve the issue.

Nodes

You need an idle backtesting node in your organization to deploy a backtest. You can view the status of all of your organization's nodes and select a specific node to use in the Resources panel of the IDE. Backtesting nodes that are more powerful can run faster backtests and backtest nodes with more RAM can handle more memory-intensive operations like training machine learning models, processing Options data, and managing large universes. The following table shows the specifications of the backtesting node models:

Name	Number of Cores	Processing Speed (GHz)	RAM (GB)	GPU
B-MICRO	2	3	8	0
B2-8	2	4.9	8	0
B4-12	4	4.9	12	0
B4-16-GPU	4	3	16	1/3
B8-16	8	4.9	16	0

Refer to the Pricing page to see the price of each backtesting node model. The first organization on each account is given one free B-MICRO backtesting node. B-MICRO nodes have a 20-second delay when you launch backtests, but the delay is removed and the node is replaced when you upgrade the tier of your organization and add a new backtesting node .

The CPU nodes are available on a fair usage basis. The GPU nodes can be shared with a maximum of three members. Depending on the server load, you may use all of the GPU's processing power.

Concurrent Backtesting

Concurrent backtesting is the process of running multiple backtests at the same time within a single organization. Concurrent backtesting speeds up your strategy development because you don't have to wait while a single backtest finishes executing. You need multiple backtesting nodes in your organization to run concurrent backtests. The more backtesting nodes that your organization has, the more concurrent backtests you can execute. If you are trying to fine-tune your parameters , consider running a parameter optimization . If you run a parameter optimization job, you can run multiple backtests concurrently without having multiple backtest nodes.

The number of backtesting nodes that you can have in your organization depends on the tier of your organization. The following table shows the backtesting node quotas:

Tier	Node Quota
Free	1
Quant Researcher	2
Team	10
Trading Firm	Unlimited
Institution	Unlimited

Logs

The amount of logs you can generate per backtest and per day depends on the tier of your organization. The following table shows the amount of logs you can generate on each tier:

Tier	Logs Per Backtest	Logs Per Day
Free	10KB	3MB
Quant Researcher	100KB	3MB
Team	1MB	10MB
Trading Firm	5MB	50MB
Institution	Inf.	Inf.

Orders

The number of orders you can place in a single backtest depends on the tier of your organization. The following table shows the number of orders you can place on each tier:

Tier	Orders Quota
Free	10K
Quant Researcher	10M
Team	Unlimited
Trading Firm	Unlimited
Institution	Unlimited

Security

Your code is stored in a database, isolated from the internet. When the code leaves the database, it is compiled and obfuscated before being deployed to the cloud. If the cloud servers were compromised, this process makes it difficult to read your strategy.

As we've seen over recent years, there can never be any guarantee of security with online websites. However, we deploy all modern and common security procedures. We deploy nightly software updates to keep the server up to date with the latest security patches. We also use SSH key login to avoid reliance on passwords. Internally, we use processes to ensure only a handful of people have access to the database and we always restrict logins to never use root credentials.

See our Security and IP documentation for more information.

Build Projects

If the compiler finds errors during the build process, the IDE highlights the lines of code that caused the errors in red. Your projects will automatically build after each keystroke. To manually build a project, open the project and then click the ▫ Build icon. If the build process is unresponsive, try refreshing the page and building again. If the build process continues to be unresponsive, check our Status page or contact us .

Run Backtests

To run a backtest, open a project and then click the ▫ Backtest icon. If the project successfully builds, "Received backtest backtestName request" displays. If the backtest successfully launches, the IDE displays the backtest results page in a new tab. If the backtest fails to launch due to coding errors, the new tab displays the error. As the backtest executes, you can refresh or close the IDE without interfering with the backtest because it runs on our cloud servers.

Stop Backtests

To stop a running backtest, open the Resources panel , and then click the stop icon next to the backtest node. You can stop nodes that you are using, but you need stop node permissions to stop nodes other members are using.

8.4 Results

Introduction

The backtest results page shows your algorithm's performance. Review the results page to see how your algorithm has performed during the backtest and to investigate how you might improve your algorithm before live trading.

View Backtest Results

The backtest results page automatically displays when you deploy a backtest . The backtest results page presents the equity curve, trades, logs, performance statistics, and much more information.

The content in the backtest results page updates as your backtest executes. You can close or refresh the window without interrupting the backtest because the backtesting node processes on our servers. If you close the page, you can view all of the project's backtests to open the page again. Unless you explicitly make the backtest public, only you can view its results. If you delete a backtest result or you are inactive for 12 months, we archive your backtest results.

Runtime Statistics

The banner at the top of the backtest results page displays the runtime statistics of your backtest.

The following table describes the default runtime statistics:

Statistic	Description
Capacity	The maximum amount of money an algorithm can trade before its performance degrades from market impact.
Equity	The total portfolio value if all of the holdings were sold at current market rates.
Fees	The total quantity of fees paid for all the transactions.
Holdings	The absolute sum of the items in the portfolio.
Net Profit	The dollar-value return across the entire trading period.
PSR	The probability that the estimated Sharpe ratio of an algorithm is greater than a benchmark (1).
Return	The rate of return across the entire trading period.
Unrealized	The amount of profit a portfolio would capture if it liquidated all open positions and paid the fees for transacting and crossing the spread.
Volume	The total value of assets traded for all of an algorithm's transactions.

To add a custom runtime statistic, call the `SetRuntimeStatistic` method with a `name` and `value` . The `value` argument can be a `string` or a number.

```
SetRuntimeStatistic(name, value);
```

```
self.SetRuntimeStatistic(name, value)
```

Built-in Charts

The backtest results page displays a set of built-in charts to help you analyze the performance of your algorithm. The following table describes the charts displayed on the page:

Chart	Description
Strategy Equity	Time series of equity and daily performance
Capacity	Time series of strategy capacity snapshots
Drawdown	Time series of equity peak-to-trough value
Benchmark	Time series of the benchmark closing price (SPY, by default)
Exposure	Time series of long and short exposure ratios
Assets Sales Volume	Chart showing the proportion of total volume for each traded security
Portfolio Turnover	Time series of the portfolio turnover rate

Custom Charts

The results page shows the custom charts that you create.

Supported Chart Types

We support the following types of charts:

If you use `SeriesType.Candle` and plot enough values, the plot displays candlesticks. However, the `Plot` method only accepts one numerical value per time step, so you can't plot candles that represent the open, high, low, and close values of each bar in your algorithm. The charting software automatically groups the data points you provide to create the candlesticks, so you can't control the period of time that each candlestick represents.

To create other types of charts, save the plot data in the ObjectStore and then load it into the Research Environment. In the Research Environment, you can create other types of charts with third-party charting packages .

Supported Markers

When you create scatter plots, you can set a marker symbol. We support the following marker symbols:

Chart Quotas

Custom charts are limited to 4,000 data points. Intensive charting requires hundreds of megabytes of data, which is too much to stream online or display in a web browser. If you exceed the quota, the Cloud Terminal displays the following message:

Exceeded maximum points per chart, data skipped

You can create up to 10 custom chart series per algorithm. If you exceed the quota, your algorithm stops executing and the Cloud Terminal displays the following message:

Exceeded maximum series count: Each backtest can have up to 10 series in total.

Demonstration

For more information about creating custom charts, see Charting .

Adjust Charts

You can manipulate the charts displayed on the backtest results page.

Toggle Charts

To display and hide a chart on the backtest results page, in the Select Chart section, click the name of a chart.

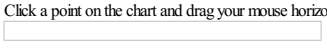
Toggle Chart Series

To display and hide a series on a chart on the backtest results page, click the name of a series at the top of a chart.

Adjust the Display Period

To zoom in and out of a time series chart on the backtest results page, perform either of the following actions:

- Click the 1m , 3m , 1y , or All period in the top-right corner of the chart.
- Click a point on the chart and drag your mouse horizontally to highlight a specific period of time in the chart.



If you adjust the zoom on a chart, it affects all of the charts.

After you zoom in on a chart, slide the horizontal bar at the bottom of the chart to adjust the time frame that displays.

Resize Charts

To resize a chart on the backtest results page, hover over the bottom-right corner of the chart. When the resize cursor appears, hold the left mouse button and then drag to the desired size.

Move Charts

To move a chart on the backtest results page, click, hold, and drag the chart title.

Refresh Charts

Refreshing the charts on the backtest results page resets the zoom level on all the charts. If you refresh the charts while your algorithm is executing, only the data that was seen by the Lean engine after you refreshed the charts is displayed. To refresh the charts, in the Select Chart section, click the reset icon.

Storage

You can store data in the ObjectStore during a backtest and then load the data into the Research Environment for further analysis. The Research Environment is a Jupyter notebook-based environment with access to many third-party libraries , so you can produce plots and display DataFrames in many more ways than the backtest results page supports.

Key Statistics

The backtest results page displays many key statistics to help you analyze the performance of your algorithm.

Overall Statistics

The Overview tab on the backtest results page displays tables for Overall Statistics and Rolling Statistics. The Overall Statistics table displays the following statistics:

- Probabilistic Sharpe Ratio (PSR)
- Total Trades
- Average Loss
- Drawdown
- Net Profit
- Loss Rate
- Profit-Loss Ratio
- Beta
- Annual Variance
- Tracking Error
- Total Fees
- Lowest Capacity Asset
- Sharpe Ratio
- Average Win
- Compounding Annual Return
- Expectancy
- Win Rate
- Alpha
- Annual Standard Deviation
- Information Ratio
- Treynor Ratio
- Estimated Strategy Capacity

Some of the statistics above are sampled throughout the backtest to produce a time series of rolling statistics. The time series are displayed in the Rolling Statistics table. You can download the data in the Overall Statistics and Rolling Statistics tables for further analysis.

Ranking

The backtest results page displays an Ranking section that shows the PSR and rank (percentile) of your algorithm.

The rank of your algorithm is calculated as

$$ CDF\left(\frac{PSR_{algo} - \overline{PSR}}{\sigma_{PSR}}\right) $$

where CDF is the normal cumulative distribution function and PSR_{algo} is your algorithm's PSR. $\overline{PSR}$ and σ_{PSR} are the mean PSR and the standard deviation of PSR values, respectively, calculated from all of the backtests that have the following attributes:

- Occurred in the last 30 days
- Had more than 90 tradable days
- Had a PSR value in the interval (0, 100)

Research Guide

For information about the Research Guide, see Research Guide .

Reports

The backtest report provides a summary of your algorithm's performance in PDF format. Follow these steps to generate one:

1. Open the backtest results page of the backtest for which you want to generate a report.
2. Click the Report tab.
3. If the project doesn't have a description, enter one and then click Save .
4. Click Download Report .

   The report may take a minute to generate.

5. If the browser says that the report is being generated, repeat step 4.

Orders

The backtest results page displays the orders of your algorithm and you can download them to your local machine.

View in the GUI

To see the orders that your algorithm created, open the backtest results page and then click the Orders tab. If there are more than 10 orders, use the pagination tools at the bottom of the Orders Summary table to see all of the orders. Click on an individual order in the Orders Summary table to reveal additional information regarding the following:

- Submissions
- Fills
- Partial fills
- Updates
- Cancellations
- Option contract exercises and expiration

The timestamps in the Order Summary table are based in Eastern Time (ET).

Download CSV

To download the orders in CSV format, open the backtest results page, click the Orders tab, and then click Download Orders . The content of the CSV file is the content displayed in the Orders Summary table when the table rows are collapsed. To retrieve all of the content in the Orders Summary table, download the backtest results JSON file .The timestamps in the CSV and JSON files are based in Coordinated Universal Time (UTC).

Insights

The backtest results page displays the insights of your algorithm and you can download them to your local machine.

View in the GUI

To see the insights your algorithm has emitted, open the backtest result page and then click the Insights tab. If there are more than 10 insights, use the pagination tools at the bottom of the Insights Summary table to see all of the insights. The timestamps in the Insights Summary table are based in Eastern Time (ET).

Download JSON

To download the insights in JSON format, open the backtest result page, click the Insights tab, and then click Download Insights . The timestamps in the CSV file are based in Coordinated Universal Time (UTC).

Logs

The backtest results page displays the logs of your backtest and you can download them to your local machine. The timestamps of the statements in the log file are based in your algorithm time zone .

View in the GUI

To see the log file that was created throughout a backtest, open the backtest result page and then click the Logs tab.

Download Log File

To download the log file that was created throughout a backtest, open the backtest result page, click the Logs tab, and then click Download Logs .

Project Files

The backtest results page displays the project files used to run the backtest. To view the files, click the Code tab. By default, the main.py or Main.cs file displays. To view other files in the project, click the file name and then select a different file from the drop-down menu.

Share Results

The backtest results page enables you to share your backtest results. You need to make a backtest public in order to share it. To make a backtest public, on the backtest results page, click the Share tab and then click Make Public . Once you make a backtest public, the Share tab displays a link to the backtest report , a link to an embedded backtest result, and a script to embed the embedded backtest result into a website.

The following widget is an example of an embedded backtest result:

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	1
Average Win	0%
Average Loss	0%
Compounding Annual Return	7.027%
Drawdown	55.500%
Expectancy	0
Net Profit	347.980%
Sharpe Ratio	0.383
Probabilistic Sharpe Ratio	0.020%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0
Beta	1.007
Annual Standard Deviation	0.161
Annual Variance	0.026
Information Ratio	0.022
Tracking Error	0.002
Treynor Ratio	0.061
Total Fees	$5.10
Estimated Strategy Capacity	$45000000.00
Lowest Capacity Asset	SPY R735QTJ8XC9X

```
class MyAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2000, 1, 1)
        self.SetCash(100000)
        self.symbol = self.AddEquity("SPY", Resolution.Daily).Symbol

    def OnData(self, data):
        if not self.Portfolio.Invested:
            self.SetHoldings(self.symbol, 1)
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	1
Average Win	0%
Average Loss	0%
Compounding Annual Return	7.027%
Drawdown	55.500%
Expectancy	0
Net Profit	347.980%
Sharpe Ratio	0.383
Probabilistic Sharpe Ratio	0.020%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0
Beta	1.007
Annual Standard Deviation	0.161
Annual Variance	0.026
Information Ratio	0.022
Tracking Error	0.002
Treynor Ratio	0.061
Total Fees	$5.10
Estimated Strategy Capacity	$45000000.00
Lowest Capacity Asset	SPY R735QTJ8XC9X

```
namespace QuantConnect.Algorithm.CSharp
{
    public class MyAlgorithm : QCAlgorithm
    {
  private Symbol _symbol;

        public override void Initialize()
        {
            SetStartDate(2000, 1, 1);
            SetCash(100000);
```

To attach the embedded backtest result to a forum discussion, see Create Discussions or Post Comments .

After you've made your backtest results public, the results are always stored, anyone with the link can access the results, and the results can't be made private again because someone may have already cloned your project.

Download Results

You can download the following information from the backtest results page:

- Runtime statistics
- Charts
- The data in the Overview tab
- The data in the Orders tab

To download the preceding information, open the backtest results page, click the Overview tab, and then click Download Results .

View All Backtests

Follow these steps to view all of the backtests of a project:

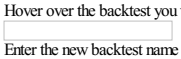
1. Open the project that contains the backtests you want to view.
2. In the top-right corner of the IDE, click the Backtest Results icon.

   A table containing all of the backtest results for the project is displayed. If there is a play icon to the left of the name, it's a backtest result . If there is a fast-forward icon next to the name, it's an optimization result .

3. (Optional) In the top-right corner, select the Show field and then select one of the options from the drop-down menu to filter the table by backtest or optimization results.
4. (Optional) In the bottom-right corner, click the Hide Error check box to remove backtest and optimization results from the table that had a runtime error.
5. (Optional) Use the pagination tools at the bottom to change the page.
6. (Optional) Click a column name to sort the table by that column.
7. Click a row in the table to open the results page of that backtest or optimization.

Rename Backtests

We give an arbitrary name (for example, "Smooth Apricot Chicken") to your backtest result files, but you can follow these steps to rename them:

1. Hover over the backtest you want to rename and then click the pencil icon that appears.

2. Enter the new backtest name and then click OK .

Clone Backtests

Hover over the backtest you want to clone, and then click the clone icon that appears to clone the backtest.

A new project is created with the backtest code files.

Delete Backtests

Hover over the backtest you want to delete, and then click the trash can icon that appears to delete the backtest.

Errors

If a backtest produces more than 700 MB of data, then LEAN can't upload the results and the backtest results page appears empty.

8.5 Debugging

Introduction

The debugger is a built-in tool to help you debug coding errors while backtesting. The debugger enables you to slow down the code execution, step through the program line-by-line, and inspect the variables to understand the internal state of the program.

Breakpoints

Breakpoints are lines in your algorithm where execution pauses. You need at least one breakpoint in your code files to start the debugger. Open a project to start adjusting its breakpoints.

Add Breakpoints

Click to the left of a line number to add a breakpoint on that line.

Edit Breakpoint Conditions

Follow these steps to customize what happens when a breakpoint is hit:

1. Right-click the breakpoint and then click Edit Breakpoint... .
2. Click one of the options in the following table:

Option	Additional Steps	Description
Expression	Enter an expression and then press Enter .	The breakpoint only pauses the algorithm when the expression is true.
Hit Count	Enter an integer and then press Enter .	The breakpoint doesn't pause the algorithm until its hit the number of times you specify.

Enable and Disable Breakpoints

To enable a breakpoint, right-click it and then click Enable Breakpoint .

To disable a breakpoint, right-click it and then click Disable Breakpoint .

Follow these steps to enable and disable all breakpoints:

1. In the right navigation menu, click the Run and Debug icon.
2. In the Run and Debug panel, hover over the Breakpoints section and then click the Toggle Active Breakpoints icon.

Remove Breakpoints

To remove a breakpoint, right-click it and then click Remove Breakpoint .

Follow these steps to remove all breakpoints:

1. In the right navigation menu, click the Run and Debug icon.
2. In the Run and Debug panel, hover over the Breakpoints section and then click the Remove All Breakpoints icon.

Launch Debugger

Follow these steps to launch the debugger:

1. Open the project you want to debug.
2. In your project's code files, add at least one breakpoint.
3. Click the Debug icon.

If the Run and Debug panel is not open, it opens when the first breakpoint is hit.

Control Debugger

After you launch the debugger, you can use the following buttons to control it:

Button	Name	Default Keyboard Shortcut	Description

Button	Name	Default Keyboard Shortcut	Description
	Continue		Continue execution until the next breakpoint
	Step Over	Alt+F10	Step to the next line of code in the current or parent scope
	Step Into	Alt+F11	Step into the definition of the function call on the current line
	Restart	Shift+F11	Restart the debugger
	Disconnect	Shift+F5	Exit the debugger

Inspect Variables

After you launch the debugger, you can inspect the state of your algorithm as it executes each line of code. You can inspect local variables or custom expressions. The values of variables in your algorithm are formatted in the IDE to improve readability. For example, if you inspect a variable that references a DataFrame, the debugger represents the variable value as the following:

Local Variables

The Variables section of the Run and Debug panel shows the local variables at the current breakpoint. If a variable in the panel is an object, click it to see its members. The panel updates as the algorithm runs.

Follow these steps to update the value of a variable:

1. In the Run and Debug panel, right-click a variable and then click Set Value .
2. Enter the new value and then press Enter .

Custom Expressions

The Watch section of the Run and Debug panel shows any custom expressions you add. For example, you can add an expression to show the current date in the backtest.

Follow these steps to add a custom expression:

1. Hover over the Watch section and then click the plus icon that appears.
2. Enter an expression and then press Enter .

8.6 Report

Introduction

Reports provide a summary of your algorithm's performance. They outline key statistics, returns, and performance during various market crises. You can generate a performance report after your backtest completes and download the report as a PDF.

Key Statistics

The top of the backtest report displays statistics to summarize your algorithm's performance. The following table describes the key statistics in the report:

Statistic	Description
Runtime Days	The number of days in the backtest or live trading period.
Turnover	The percentage of the algorithm's portfolio that was replaced in a given year.
CAGR	The annual percentage return that would be required to grow a portfolio from its starting value to its ending value.
Markets	The asset classes that the algorithm trades.
Trades per day	The total number of trades during the backtest divided by the number of days in the backtest. Trades per day is an approximation of the algorithm's trading frequency.
Drawdown	The largest peak to trough decline in an algorithm's equity curve.
Probabilistic SR	The probability that the estimated Sharpe ratio of an algorithm is greater than a benchmark (1).
Sharpe Ratio	A measure of the risk-adjusted return, developed by William Sharpe.
Information Ratio	The amount of excess return from the risk-free rate per unit of systematic risk.
Strategy Capacity	The maximum amount of money an algorithm can trade before its performance degrades from market impact.

Returns

The backtest report displays charts to show the algorithm's returns per trade, per day, per month, per year, and the cumulative returns over the backtest.

Returns per Trade

This chart displays a histogram that shows the distribution of returns per trade over the backtesting period.

Daily Returns

This chart displays the returns of each day. Blue bars represent profitable days and gray bars represent unprofitable days.

Monthly Returns

This chart displays the return of each month. We convert the original equity curve series into a monthly series and calculate the returns of each month. Green cells represent months with a positive return and red cells represent months with a negative return. Months that have a greater magnitude of returns are represented with darker cells. Yellow cells represent months with a relatively small gain or loss. White rectangles represent months that are not included in the backtest period. The values in the cells are percentages.

Annual Returns

This chart displays the return of each year. We calculate the total return within each year and represent each year with a blue bar. The red dotted line represents the average of the annual returns.

Cumulative Returns

This chart displays the cumulative returns of your algorithm. The blue line represents your algorithm and the gray line represents the benchmark.

Asset Allocation

This chart displays a time-weighted average of the absolute holdings value for each asset that entered your portfolio during the backtest. When an asset has a percentage that is too small to be shown in the pie chart, it is incorporated into an "Others" category.

Drawdown

This chart displays the peak-to-trough drawdown of your portfolio's equity throughout the backtest period. The drawdown of each day is defined as the percentage loss since the maximum equity value before the current day. The drawdowns are calculated based on daily data. The top 5 drawdown periods are marked in the chart with different colors.

Rolling Statistics

The backtest report displays time series for your portfolio's rolling beta and Sharpe ratio .

Rolling Portfolio Beta

This chart displays the rolling portfolio beta over trailing 6 and 12 month periods. The light blue line represents the 6 month period and the dark blue line represents the 12 month period.

Rolling Sharpe Ratio

This chart displays the rolling portfolio Sharpe ratio over trailing 6 and 12 month periods. The light blue line represents the 6 month period and the dark blue line represents the 12 month period.

Exposure

The backtest report displays time series for your portfolio's overall leverage and your portfolio's long-short exposure by asset class.

Leverage

This chart displays your algorithm's utilization of leverage over time.

Long-Short Exposure By Asset Class

This chart displays your algorithm's long-short exposure by asset class over time.

Crisis Events

This set of charts displays the cumulative returns of your algorithm and the benchmark during various historical periods. The blue line represents the cumulative returns of your algorithm and the grey line represents the cumulative return of the benchmark. The report only contains the crisis event that occurred during your algorithm's backtest period. The following table shows the crisis events that may be included in your backtest report:

Crisis Name	Start Date	End Date
DotCom Bubble 2000	2/26/2000	9/10/2000
September 11, 2001	9/5/2001	10/10/2001
U.S. Housing Bubble 2003	1/1/2003	2/20/2003
Global Financial Crisis 2007	10/1/2007	12/1/2011
Flash Crash 2010	5/1/2010	5/22/2010
Fukushima Meltdown 2011	3/1/2011	4/22/2011
U.S. Credit Downgrade 2011	8/5/2011	9/1/2011
ECB IR Event 2012	9/5/2012	10/12/2012
European Debt Crisis 2014	10/1/2014	10/29/2014
Market Sell-Off 2015	8/10/2015	10/10/2015
Recovery 2010-2012	1/1/2010	10/1/2012
New Normal 2014-2019	1/1/2014	1/1/2019
COVID-19 Pandemic 2020	2/10/2020	9/20/2020

8.7 Engine Performance

Introduction

A set of benchmark algorithms are periodically run to test the status and speed of the Lean master branch. View the Lean Performance Benchmark page to see the results. The chart at the top of the page shows the data points per second for each of the benchmark algorithms. The table at the bottom of the page shows the benchmark algorithms that are run to produce the results.

9 Datasets

Datasets are a stream of data points you use in your algorithms to make trading decisions and fill orders. Our Dataset Market is a portal where we aggregate datasets for you to use in your algorithms. Our Dataset Market includes price, fundamental, and alternative datasets. Consider using fundamental and alternative datasets to incorporate more information in your trading decisions. Fundamental and alternative datasets contain information that is not present in the price. Price data is commonly researched for trading ideas, so you may find it easier to discover alpha in other types of datasets.

The Dataset Market enables you to easily load datasets into your trading algorithms for use in the cloud or locally. The datasets come configured ready to integrate into your research and backtesting without any need for cleaning. The datasets in our market are vetted by the QuantConnect team to be high-quality, contain actionable information, and be free of survivorship-bias. Our Dataset Market is growing quickly. New datasets are added frequently.

Navigating the Market

The lay of the land

Categories

Diverse types of datasets available

Data Issues

Data isn't always perfect

Misconceptions

It might not be a data issue

Licensing

Ways to access the data

Vendors

People who provide datasets

See Also

Dataset Market
Purchasing Datasets
Contributing Datasets

9.1 Navigating the Market

Introduction

Datasets are a stream of data points you use in your algorithms to make trading decisions and fill orders. Our Dataset Market is a portal where we curate datasets available for use in your algorithms. It includes price, fundamental, and alternative datasets. Consider using fundamental and alternative datasets to incorporate more information in your trading decisions. Fundamental and alternative datasets contain information that is not present in the price. Price data is commonly researched for trading ideas, so you may find it easier to discover alpha in other types of datasets.

The Dataset Market enables you to easily load datasets into your trading algorithms for use in the cloud or locally. The datasets come configured ready to integrate into your research and backtesting without any need for cleaning. The datasets in our market are vetted by the QuantConnect team to be high-quality, contain actionable information, and be free of survivorship-bias. Our Dataset Market is growing quickly, as new datasets are added frequently.

The Dataset Market is a place where you can view, subscribe to, and download datasets. We provide an example algorithm for each dataset that you can clone to easily get started using the new dataset. We also provide an example research notebook for most datasets to demonstrate how to use the dataset in the Research Environment. You can always view the dataset reviews to learn about the experience other members have had using the dataset. This page explains the layout of the Dataset Market to help you navigate the marketplace.

View All Listings

The Dataset Market displays all our supported datasets. To view the page, in the top navigation bar, click Data.

Each dataset displays the name, description, coverage, start date, and price of the dataset. Coverage is the number of assets, securities, contracts, currency pairs, or articles that are included in the dataset. To view the listing page of a dataset, click the dataset.

You can search the market by applying filters or searching for keywords.

Filter Listings by Category

Click the Category: All field and then select a category from the drop-down menu to only display datasets in that category.

Filter Listings by Delivery Options

Click the Delivery Options: All field and then select an option from the drop-down menu to only display datasets with that delivery option.

Search for Keywords

Enter keywords in the search bar to only display datasets that contain those keywords.

Homepage

The homepage of a dataset listing displays everything that you need to get started using the dataset. The following table describes the tabs on the homepage:

Tab	Description
About	High-level overview of the dataset and the data provider
Documentation	Instructions on using the dataset in backtests and the Research Environment
Research	A demonstration research notebook of analyzing the dataset
Examples	Full example algorithms that use the dataset
Licenses	A list of licenses available for the dataset
CLI	Command generator to download the dataset with the CLI
Pricing	The price to access the dataset in the cloud or on your local machine
Data Explorer	A table to inspect the dataset files and report data issues

The following table describes the sections displayed under the About tab for most datasets:

Tab	Description
Introduction	High-level overview of what the dataset includes, who it's created by, and how it's created.
About the Provider	Description of the data provider and a link to their website.
Getting Started	The line(s) of code that you need to use the dataset in algorithms.
Data Summary	A table that displays the dataset's start date, coverage, resolution, density, and timezone.
Example Applications	A list of ideas on using the dataset in your algorithm.
Data Point Attributes	A set of widgets that display the factors in the dataset, the class members of objects that you use when accessing the dataset, and enumeration values that you can use to customize the dataset.
Pricing	The price to access the dataset in the cloud or on your local machine.
Reviews	Reviews from members who have purchased the dataset.

Sidebar

The sidebar of the dataset listing provides a brief summary of the dataset. The following table describes the summary content:

Tab	Description
Pricing	The number of licensing options available
Start Date	The date of the first data point.
Coverage	The number of assets, securities, contracts, currency pairs, or articles that the dataset includes.
Delivery Methods	The various delivery methods the dataset supports.
About the Provider	A link to the data provider's website.

Documentation

The Documentation tab on a dataset listing demonstrates how to use the dataset. The documentation covers requesting the data, accessing the data in your algorithm, and performing history requests. We provide documentation in C# and Python so you can easily integrate the dataset into your algorithms, regardless of the programming language you use.

The Documentation tab also has a Data Point Attributes section to show the dataset's attributes. If an attribute has a custom data type, you can click the attribute to view the attributes of the custom data type.

Factor Research

Some dataset listings have a Research tab that displays an analysis of the data point attributes in the dataset. Follow these steps to clone the example research notebook of a dataset:

1. Log in to the Algorithm Lab.
2. Open a dataset listing page .
3. Click the Research tab, if available.
4. Click Clone This Notebook .
5. Click Clone Algorithm .

Examples

The Examples tab on a dataset listing shows how to use the dataset in a trading algorithm. We provide examples in C# and Python for both of the classic and framework algorithm designs. Copy-paste these example algorithms to jumpstart your strategy research. Consider adjusting the strategy to make it your own or using the parameter optimization feature to improve the performance of the algorithms.

Licenses

The Licensing tab shows the available licenses for the dataset. Each dataset comes with its own licensing requirements, depending on the data vendor. For more information about licensing types, see Licensing .

Reviews

The bottom of the dataset listing page shows reviews published by QuantConnect members. You can sort, filter, and write dataset reviews.

Sort Reviews

Open a dataset listing page , click the Most Recent field, and then select a metric from the drop-down menu to sort the reviews by that metric.

Filter Reviews

Open a dataset listing page , click the All field, and then select a number of stars from the drop-down menu to only display the reviews with that rating.

Write Reviews

Follow these steps to write a review:

1. Log in to the Algorithm Lab.
2. Open a dataset listing page .
3. At the bottom of the page, select a number of stars to give your review.
4. Write your review.
5. Click Submit Review .

9.2 Categories

Introduction

Dataset categories are a way to identify different types of datasets in our Dataset Market. We provide many price, fundamental, and alternative datasets for you to use in your research and trading. Datasets that include factors outside of the security price are less researched, so they may have more alpha to discover. Incorporate alternative datasets into your algorithms so that you can make more informed trading decisions.

Geospatial

Geospatial data is data related to objects that have a position in the world.

Commerce

Commerce data is data on customer and business behavior.

Financial Market

Financial market data is data on the trading activity on exchanges.

Consumer

Consumer data is data on all aspects of consumers, including online shopping behaviors, consumer demographics, and consumer attitudes.

B2B

Business-to-business (B2B) data is data on businesses that sell goods and services to other businesses.

Transport and Logistics

Transport and logistics data is data on the transportation of goods and the logistics of the transportation.

Environmental

Environmental data is data on the state of the environment, including meteorological data, biodiversity data, and pollution data.

Credit Rating

Credit rating data is data on the financial position of individuals and businesses.

Real Estate

Real estate data is data on residential and commercial real estate, including ownership data, real estate listing data, and real estate demographic data.

Web

Web data is data on the behavior of internet users.

Legal

Legal data is data on the law, including new regulations, government website data, and litigation history.

Health Care

Healthcare data is data on patient-doctor visits, including claims data, fitness wearables data, and health record data.

Entertainment

Entertainment data is data on the media consumption preferences and behaviors of consumers.

Energy

Energy data is data on energy production, distribution, and consumption.

Industry

Industry data is data on various groupings in the economy.

Political

Political data is data that's collected on political activity, including election votes and political party policies.

News and Events

News and events data is data that's collected from news providers regarding current events.

9.3 Data Issues

Introduction

Data issues are incorrect or missing values in a dataset. These issues are generally a result of human error or from mistakes in the data collection process. Data issues can be reported by any QuantConnect member. When data issues are reported and verified, our Data Team works to quickly resolve them. Thanks to the communal efforts of the QuantConnect members, the QuantConnect data is reviewed and fixed by over 235,000 people, giving you a very high-quality source of data.

Common Issues

Data issues can occur in both historical and live data feeds. Some common examples of data issues include the following:

- Missing or incorrect values
- Splits and dividends
- Listings and delistings
- Ticker changes

View Current Issues

You can view a list of all the current data issues under the Data Issues tab in the community forum . Before you report a new issue, review the list of current issues to ensure that the issue is not already reported. The number of open data issues can sometimes be large, but our Data Team works on resolving them as quickly as possible while prioritizing the most important ones.

Report New Issues

When all of the QuantConnect members report the data issues that they find, we can ensure the datasets are high quality for everyone. The easier it is for our Data Team to detect and reproduce the issues you report, the faster we can resolve them. If you encounter an issue with live data, email us a description of the issue. If you find an issue with the historical data of a dataset, follow these steps to report it:

1. Log in to the Algorithm Lab.
2. Open the Data Explorer Issues page.
3. On the Data Explorer page, fill out the form.
4. Follow these steps to attach a backtest or notebook that demonstrates the issue:
    1. Click Attach Backtest .
    2. Click the Project field and then select the project from the drop-down menu.
    3. Click the Backtest field and then select the backtest from the drop-down menu.
5. Click Publish .

9.4 Misconceptions

Introduction

Some data issues are reported that aren't actually data issues. Instead, they are from a misunderstanding of how the data is collected, timestamped, formatted, and normalized. These misunderstandings are caused by assumptions that the data should be the same across different platforms, should have the same timezones, should be timestamped a certain way, and should be normalized the same as other data sources.

Cross-Platform Discrepancies

You may find our data can sometimes be slightly different from the data that's displayed on other platforms. Most of the differences occur because our data is institutional quality while a lot of the other platforms use a cheaper alternative. We use the Consolidated Tape Association (CTA) and Unlisted Trading Privileges (UTP) tick feeds, which cover the entire US tick feed. In contrast, most charting websites use the Better Alternative Trading System (BATS), which has very permissive display policies but only covers about 6-7% of the total market volume . Our tick feed doesn't include over-the-counter (OTC) trades, but the data on other platforms like Yahoo Finance include OTC trades.

Timezone Differences

Datasets all have different timezones. Most price datasets are timestamped in Eastern Time (ET). However, Future markets have more exotic timezones, depending on where the Future contract is trading. QuantConnect allows the raw data to be in different timezones. For US Equities, the timezone is ET. For Oanda Forex prices, the timezone is Coordinated Universal Time (UTC). In contrast, other charting platforms may display Oanda data with ET timestamps. Oanda Forex uses UTC, but Oanda CFD uses timezones relative to each of the CFD products that Oanda lists. QuantConnect accurately reflects all of these timezones from the relative markets that theyâ€™re trading.

Misaligned Timestamps

Every piece of data has a period. Some data is near-instantaneous, like tick data. Other data has a longer period, like second, minute, hour, and daily bars. QuantConnect delivers this data to your algorithms at the end of the period to ensure that lookahead bias doesnâ€™t occur. When you look at the `Time` property of your algorithm, the period has already ended, so it looks as if the data is offset by one period. To compare the timestamps of our data to other data, use the `Time` property of the current bar. The `Time` property of the bar is the start of the bar and the `EndTime` property is the end of the bar. If you use Python and request historical data, the `time` index in the DataFrame that's returned maps to the `EndTime` of the respective bar. For more information about timestamps, see Time Modeling .

Data Normalization

The data normalization mode defines how historical data is adjusted to accommodate for splits, dividends, and continuous Future contract roll overs. When you compare the data in the Dataset Market to data that's hosted on other platforms, the data may have different values because a different data normalization mode is being used to adjust the data. Ensure datasets are using the same normalization mode before reporting data issues. The most common way to recognize this bug is by comparing the two price series and seeing them significantly deviate in the past. The following data normalization modes are available:

Adjusted Prices

By default, LEAN adjusts US Equity data for splits and dividends to produce a smooth price curve. We use the entire split and dividend history to adjust historical prices. This process ensures you get the same adjusted prices, regardless of the backtest end date.

Backtest differences occur when you run backtests before a split or dividend occurs in live trading and then run the same backtest after it occurs. The second time you run the backtest, the adjusted prices will be different so it can cause different backtest results. The difference can be significant in large universes because of multiple corporate actions and the cummulative effect of orders with a small difference.

Opening and Closing Auctions

The opening and closing price of the day is set by very specific opening and closing auction ticks. When a stock like Apple is listed, itâ€™s listed on Nasdaq. The open auction tick on Nasdaq is the price thatâ€™s used as the official open of the day. NYSE, BATS, and other exchanges also have opening auctions, but the only official opening price for Apple is the opening auction on the exchange where it was listed.

We set the opening and closing prices of the first and last bars of the day to the official auction prices. This process is used for second, minute, hour, and daily bars for the 9:30 AM and 4:30 PM Eastern Time (ET) prices. In contrast, other platforms might not be using the correct opening and closing prices.

The official auction prices are usually emitted 2-30 seconds after the market open and close. We do our best to use the official opening and closing prices in the bars we build, but the delay can be so large that there isn't enough time to update the opening and closing price of the bar before it's injected into your algorithms. For example, if you subscribe to second resolution data, we wait until the end of the second for the opening price but most second resolution data wonâ€™t get the official opening price. If you subscribe to minute resolution data, we wait until the end of the minute for the opening auction price. Most of the time, youâ€™ll get the actual opening auction price with minute resolution data, but there are always exceptions. Nasdaq and NYSE can have delays in publishing the opening auction price, but we donâ€™t have control over those issues and we have to emit the data on time so that you get the bar you are expecting.

Live and Backtesting Differences

In live trading, bars are built using the exchange timestamps with microsecond accuracy. This microsecond-by-microsecond processing of the ticks can mean that the individual bars between live trading and backtesting can have slightly different ticks. As a result, it's possible for a tick to be counted in different bars between backtesting and live trading, which can lead to bars having slightly different open, high, low, close, and volume values.

There is a delay in when new live data is available for backtesting. It's normally available after 24-48 hours. If you need to closely monitor new data, use live paper trading .

9.5 Licensing

Introduction

You can license datasets in the Dataset Market to use in the cloud for live trading and research or to download locally. We have contracts with the data providers in the Dataset Market that define the costs of each license. All of the datasets can be used in QuantConnect Cloud. There are some free licenses, but we can't freely redistribute most of the datasets.

Free

We strive to make as many datasets available for free to use in the cloud and to download locally as possible. We also list proprietary datasets that are available for license using our cloud or download paid licensing. Most price data is free for use in the cloud.

Cloud

If you have a Cloud license for a dataset, you can access the dataset in the Algorithm Lab for research, backtests, and live trading. The cost of the license is added to your monthly bill, which you can pay with your organization's credit card or QuantConnect Credit balance. With one Cloud license for a dataset, all of the members in your organization can access the dataset in the cloud.

Add Cloud Access

You need an organization above the Free tier to purchase cloud access to datasets.

Follow these steps to add cloud access to datasets:

1. Log in to the Algorithm Lab.
2. Open the listing page of a dataset for which you want to gain cloud access.
3. On the dataset listing page, click the Pricing tab.
4. Under the Cloud Access section, click SUBSCRIBE .
5. On the Pricing page, click Proceed to Checkout .

Remove Cloud Access

Follow these steps to remove cloud access to datasets:

1. Open the organization homepage .
2. Click Edit Plan .
3. On the pricing page, click the Customize Plan > Build Your Own Pack > Data tab.
4. In the Datasets Subscriptions section, next to the name of the dataset you want to remove, click Added .
5. Click Proceed to Checkout .

Download

If you have a Download license, you can store datasets on your local machine. This download is for the licensed organization's internal LEAN use only and cannot be redistributed or converted in any format. If you study the data and produce some charts, you may share images of the charts online if the original data can't be to reconstructed from the image. The cost of the license depends on the dataset and it's calculated on a per-file or per-day basis. For more information about downloading datasets, see Downloading Data .

9.6 Vendors

Introduction

We welcome submissions of new datasets by data companies. Review our submission process to learn how to get your dataset listed on QuantConnect.

Submission Criteria

Datasets must meet the following criteria to be considered for the Datasets Market:

- A well-defined dataset with a clear and static vision for the data to minimize churn or changes.
- Robust ticker and security links to ensure the tickers are tracked well through time. ISIN, FIGI, or point-in-time tickers are supported.
- Sufficient organizational funding to ensure at least 1 year of operation.
- Reliable API with no dead links or 502 errors.
- Consistent delivery schedule with data delivered on time and in time for trading.
- Consistent data format with notifications and lead time on data format updates.
- At least 1 year of historical data.
- Free of survivorship bias.
- Good documentation of the dataset.

If the dataset is alternative data, in addition to the criteria above, the dataset must practice the Alternative Investment Standards defined by the non-profit Investment Data Standards Organization (IDSO). The Alternative Investment Standards outline the rules and best practices for collecting and distributing alternative datasets. For example, the IDSO Web Crawling Best Practices publication states that â€œa data collector should assess a website according to the terms of its robots.txtâ€.

Review Process

The dataset review process is in place to ensure that your dataset matches the submission criteria. The review process can take several weeks. If your dataset is accepted, we'll begin integrating it into the Datasets Market. If your dataset is rejected, we'll provide feedback to help you get the dataset accepted.

Contributing Datasets

We encourage you to integrate your own datasets. To integrate your dataset, see the Contributing Datasets tutorial. The integration process only takes about 1 day of engineering.

Give Free Trials

Follow these steps to give a free trial of your dataset to a QuantConnect organization:

1. Log in to the Algorithm Lab.
2. Open the Dataset Market .
3. On the Datasets page, click the dataset that you want to give as a trial.
4. In the right sidebar of the dataset listing, click Dashboard .
5. On the dataset dashboard page, click give trial .
6. In the Give Trial to Organization window, enter the expriation date of the trial and then click OK .
7. In the Organization Owners Email window, enter the email address of the member who owns the organization that you want to grant the trial.
8. If the email address you entered owns mulitple organizations, in the Select target organization window, select an organization from the drop-down menu and then click OK .
9. Click OK .

Contacting Our Team

If you want to discuss integrating your dataset into the Datasets Market, contact us . We look forward to working with you so that we can provide QuantConnect members with access to more high-quality datasets.

10 Live Trading

A live algorithm is an algorithm that trades in real-time with real market data. QuantConnect enables you to run your algorithms in live mode with real-time market data. Deploy your algorithms using QuantConnect because our infrastructure is battle-tested. We have successfully hosted more than 200,000 live algorithms and have had more than $15B in volume traded on our servers since 2015. The algorithms that our members create are run on co-located servers and the trading infrastructure is maintained at all times by our team of engineers. It's common for members to achieve 6-months of uptime with no interruptions.

Getting Started

Learn the basics

Brokerages

All the supported brokerages

Data Feeds

Stream data into your algorithm

Deployment

Run on co-located servers

Notifications

Stay informed on algorithm decisions

Results

Intervene in your live algorithms

Algorithm Control

Differences between backtesting and live trading

Reconciliation

What could happen

Risks

See Also

Adding Notifications
Set Up Paper Trading

10.1 Getting Started

Introduction

A live algorithm is an algorithm that trades in real-time with real market data. QuantConnect enables you to run your algorithms in live mode with real-time market data. Deploy your algorithms using QuantConnect because our infrastructure is battle-tested. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. The algorithms that our members create are run on co-located servers and the trading infrastructure is maintained at all times by our team of engineers. It's common for members to achieve 6-months of uptime with no interruptions.

Deploy Live Algorithms

The following video demonstrates how to deploy live paper trading algorithms:

You must have an available live trading node for each live trading algorithm you deploy.

Follow these steps to deploy a live paper trading algorithm:

1. Open the project that you want to deploy.
2. Click the Deploy Live icon.
3. On the Deploy Live page, click the Brokerage field and then click Paper Trading from the drop-down menu.
4. Click the Node field and then click the live trading node that you want to use from the drop-down menu.
5. (Optional) Follow these steps to start the algorithm with existing cash holdings (see video):
    1. In the Algorithm Cash State section, click Show .
    2. Click Add Currency .
    3. Enter the currency ticker (for example, USD or BTC) and a quantity.
6. (Optional) Follow these steps to start the algorithm with existing position holdings (see video):
    1. In the Algorithm Holdings State section, click Show .
    2. Click Add Holding .
    3. Enter the symbol ID, symbol, quantity, and average price.
7. (Optional) Set up notifications .
8. Configure the Automatically restart algorithm setting.

    By enabling automatic restarts, the algorithm will use best efforts to restart the algorithm if it fails due to a runtime error. This can help improve the algorithm's resilience to temporary outages such as a brokerage API disconnection.

9. Click Deploy .

The deployment process can take up to 5 minutes. When the algorithm deploys, the live results page displays.

To deploy a live algorithm with a different brokerage, see the Deploy Live Algorithms section of the brokerage integration documentation .

Stop Live Algorithms

The live trading results page has a Stop button to immediately stop your algorithm from executing. When you stop a live algorithm, your portfolio holdings are retained. Stop your algorithm if you want to perform any of the following actions:

- Update your project's code files
- Upgrade the live trading node
- Update the settings you entered into the deployment wizard
- Place manual orders through your brokerage account instead of the web IDE

Furthermore, if you receive new securities in your portfolio because of a reverse merger, you also need to stop and redeploy the algorithm.

LEAN actively terminates live algorithms when it detects interference outside of the algorithm's control to avoid conflicting race conditions between the owner of the account and the algorithm, so avoid manipulating your brokerage account and placing manual orders on your brokerage account while your algorithm is running. If you need to adjust your brokerage account holdings, stop the algorithm, manually place your trades, and then redeploy the algorithm.

Follow these steps to stop your algorithm:

1. Open your algorithm's live results page .
2. Click Stop .
3. Click Stop again.

Liquidate Live Algorithms

The live results page has a Liquidate button that acts as a "kill switch" to sell all of your portfolio holdings. If your algorithm has a bug in it that caused it to purchase a lot of securities that you didn't want, this button let's you easily liquidate your portfolio instead of placing many manual trades. When you click the Liquidate button, if the market is open for an asset you hold, the algorithm liquidates it with market orders. If the market is not open, the algorithm places market

on open orders. After the algorithm submits the liquidation orders, it stops executing.

Follow these steps to liquidate your positions:

1. Open your algorithm's live results page .
2. Click Liquidate .
3. Click Liquidate again.

10.2 Brokerages

Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

Binance

Crypto & Crypto Futures

Bitfinex

Crypto

Coinbase

Crypto

Interactive Brokers

US Equities, Equity Options, Forex, Futures, Future Options, Index, & Index Options

Kraken

Crypto

Oanda

Forex & CFD

Prime Brokerages

US Equities, Equity Options, & Futures

QuantConnect Paper Trading

US Equities, Forex, CFD, Crypto, Futures, & Future Options

Samco

India Equities, India Equity Options, & India Futures

TD Ameritrade

US Equities

Tradier

US Equities & Equity Options

Trading Technologies

Futures

Wolverine

US Equities

Zerodha

India Equities

FIX Connections

Financial Information eXchange

Unsupported Brokerages

Request New Additions

See Also

Adding Notifications
Set Up Paper Trading

10.2.1 Binance

Introduction

QuantConnect enables you to run your algorithms in live mode with real-time market data. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

Binance was founded by Changpeng Zhao in 2017 with the goal to "increase the freedom of money globally". Binance provides access to trading Crypto through spot markets and perpetual Futures. They serve clients with no minimum deposit when depositing Crypto. Binance also provides an NFT marketplace, a mining pool, and services to deposit Crypto coins in liquidity pools to earn rewards.

To view the implementation of the Binance brokerage integration, see the Lean.Brokerages.Binance repository .

Account Types

Binance supports cash and margin accounts for spot trades, but only supports margin accounts for Futures trades. Binance US only supports cash accounts. To set the account type in an algorithm, see the Binance brokerage model documentation .

Create an Account

Follow the account creation wizard on the Binance.com or Binance.us website to create a Binance account.

You will need API credentials to deploy live algorithms with your brokerage account. After you open your account, create API credentials and store them somewhere safe. As you create credentials, make the following configurations:

- Select the Restrict access to trusted IPs only check box and then enter our IP address, 15.235.119.112.
- If you are going to trade Crypto Futures, select the Enable Futures check box.

Paper Trading

Binance supports paper trading through the Binance Spot Test Network. You don't need a Binance account to create API credentials for the Spot Test Network.

Follow these steps to set up paper trading with the Binance Spot Test Network:

1. Log in to the Binance Spot Test Network with your GitHub credentials.
2. In the API Keys section, click Generate HMAC_SHA256 Key .
3. Enter a description and then click Generate .
4. Store your API key and API key secret somewhere safe.

Paper trading Binance Crypto Futures or with Binance US isn't currently available.

Sub-Accounts

Our Binance and Binance US integrations don't support trading with sub-accounts. You must use your main account.

Asset Classes

Our Binance integration supports trading Crypto and Crypto Futures .

```
AddCrypto("BTCUSDT", Resolution.Minute, Market.Binance);
AddCryptoFuture("BTCUSD", Resolution.Minute, Market.Binance);
AddCrypto("BTCUSDT", Resolution.Minute, Market.BinanceUS);
```

```
self.AddCrypto("BTCUSDT", Resolution.Minute, Market.Binance)
self.AddCryptoFuture("BTCUSD", Resolution.Minute, Market.Binance)
```

Our Binance US integration supports trading Crypto .

```
AddCrypto("BTCUSDT", Resolution.Minute, Market.BinanceUS);
```

```
self.AddCrypto("BTCUSDT", Resolution.Minute, Market.BinanceUS)
```

If you call the `SetBrokerageModel` method with the correct `BrokerageName` , then you don't need to pass a `Market` argument to the `AddCrypto` method because the brokerage model has a default market .

Data Feeds

Our Crypto data feed provides Crypto data during live trading.

Orders

We model the Binance and Binance US APIs by supporting several order types, supporting order properties, and not supporting order updates. When you deploy live algorithms, you can place manual orders through the IDE.

Order Types

The following table describes the available order types for each asset class that our Binance and Binance US integrations support:

Order Type	Crypto	Crypto Futures
MarketOrder	✓	✓
LimitOrder	✓	✓
StopLimitOrder	✓	

Order Properties

We model custom order properties from the Binance and Binance US APIs. The following table describes the members of the `BinanceOrderProperties` object that you can set to customize order execution:

Property	Description
`TimeInForce`	A TimeInForce instruction to apply to the order. The following instructions are supported: • `Day` • `GoodTilCanceled` • `GoodTilDate`
`PostOnly`	A flag to signal that the order must only add liquidity to the order book and not take liquidity from the order book. If part of the order results in taking liquidity rather than providing liquidity, the order is rejected without any part of it being filled.

Updates

We model the Binance and Binance US APIs by not supporting order updates, but you can cancel an existing order and then create a new order with the desired arguments. For more information about this workaround, see the Workaround for Brokerages That Donâ€™t Support Updates .

Fees

To view the Binance or Binance US trading fees, see the Trading Fees page on the Binance.com website or the Fee Structure page on the Binance.us website. To view how we model their fees, see Fees . The Binance Spot Test Network does not charge order fees.

Margin

We model buying power and margin calls to ensure your algorithm stays within the margin requirements. If you trade Crypto Perpetual Futures, we model the margin cost and payments of your Crypto Future holdings by directly adjusting your portfolio cash. For more information about Futures margin interest modeling, see the Binance Futures Model .

Slippage

Orders through Binance and Binance US do not experience slippage in backtests. In live trading, your orders may experience slippage.

To view how we model Binance and Binance US slippage, see Slippage .

Fills

We fill market orders immediately and completely in backtests. In live trading, if the quantity of your market orders exceeds the quantity available at the top of the order book, your orders are filled according to what is available in the order book.

To view how we model Binance and Binance US order fills, see Fills .

Settlements

Trades settle immediately after the transaction

To view how we model settlement for Binance and Binance US trades, see Settlement .

Security and Stability

When you deploy live algorithms with Binance or Binance US, we don't save your brokerage account credentials.

Deposits and Withdraws

You can deposit and withdraw cash from your brokerage account while you run an algorithm that's connected to the account. We sync the algorithm's cash holdings with the cash holdings in your brokerage account every day at 7:45 AM Eastern Time (ET).

Demo Algorithm

The following algorithm demonstrates the functionality of the Binance and Binance US brokerages:

Binance

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0.01%
Average Loss	0%
Compounding Annual Return	5.955%
Drawdown	8.400%
Expectancy	0
Net Profit	5.520%
Sharpe Ratio	0.578
Probabilistic Sharpe Ratio	30.832%
Loss Rate	0%
Win Rate	100%
Profit-Loss Ratio	0
Alpha	-0.04
Beta	0.096
Annual Standard Deviation	0.074
Annual Variance	0.005
Information Ratio	-1.299
Tracking Error	0.628
Treynor Ratio	0.444
Total Fees	$44.24
Estimated Strategy Capacity	$520000.00
Lowest Capacity Asset	BTCUSDT 18N

```
class BinanceBrokerageExampleAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2021, 1, 1)
        self.SetCash('USDT', 100000)

        self.SetBrokerageModel(BrokerageName.Binance, AccountType.Cash)

        self.symbol = self.AddCrypto("BTCUSDT", Resolution.Minute).Symbol
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0.01%
Average Loss	0%
Compounding Annual Return	5.955%
Drawdown	8.400%
Expectancy	0
Net Profit	5.520%
Sharpe Ratio	0.578
Probabilistic Sharpe Ratio	30.832%
Loss Rate	0%
Win Rate	100%
Profit-Loss Ratio	0
Alpha	-0.04
Beta	0.096
Annual Standard Deviation	0.074
Annual Variance	0.005
Information Ratio	-1.299
Tracking Error	0.628
Treynor Ratio	0.444
Total Fees	$44.24
Estimated Strategy Capacity	$520000.00
Lowest Capacity Asset	BTCUSDT 18N

```
namespace QuantConnect.Algorithm.CSharp
{
    public class BinanceBrokerageExampleAlgorithm : QCAlgorithm
    {
        private Symbol _symbol;

        public override void Initialize()
        {
            SetStartDate(2021, 1, 1);
            SetCash("USDT", 100000);
```

Binance US

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0.02%
Average Loss	0%
Compounding Annual Return	-2.885%
Drawdown	21.000%
Expectancy	0
Net Profit	-4.270%
Sharpe Ratio	-0.095
Probabilistic Sharpe Ratio	5.379%
Loss Rate	0%
Win Rate	100%
Profit-Loss Ratio	0
Alpha	0.003
Beta	0.192
Annual Standard Deviation	0.129
Annual Variance	0.017
Information Ratio	0.13
Tracking Error	0.513
Treynor Ratio	-0.064
Total Fees	$43.99
Estimated Strategy Capacity	$77000.00
Lowest Capacity Asset	BTCUSD 2S7

```
# QUANTCONNECT.COM - Democratizing Finance, Empowering Individuals.
# Lean Algorithmic Trading Engine v2.0. Copyright 2014 QuantConnect Corporation.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0.02%
Average Loss	0%
Compounding Annual Return	-2.885%
Drawdown	21.000%
Expectancy	0
Net Profit	-4.270%
Sharpe Ratio	-0.095
Probabilistic Sharpe Ratio	5.379%
Loss Rate	0%
Win Rate	100%
Profit-Loss Ratio	0
Alpha	0.003
Beta	0.192
Annual Standard Deviation	0.129
Annual Variance	0.017
Information Ratio	0.13
Tracking Error	0.513
Treynor Ratio	-0.064
Total Fees	$43.99
Estimated Strategy Capacity	$77000.00
Lowest Capacity Asset	BTCUSD 2S7

```
/*
 * QUANTCONNECT.COM - Democratizing Finance, Empowering Individuals.
 * Lean Algorithmic Trading Engine v2.0. Copyright 2014 QuantConnect Corporation.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS.
```

Virtual Pairs

All fiat and Crypto currencies are individual assets. When you buy a pair like BTCUSD, you trade USD for BTC. In this case, LEAN removes some USD from your portfolio cashbook and adds some BTC. The virtual pair BTCUSD represents your position in that trade, but the virtual pair doesn't actually exist. It simply represents an open trade. When you deploy a live algorithm, LEAN populates your cashbook with the quantity of each currency, but it can't get your position of each virtual pair.

Deploy Live Algorithms

You must have an available live trading node for each live trading algorithm you deploy.

Follow these steps to deploy a live algorithm:

1. Open the project you want to deploy.
2. Click the Deploy Live icon.
3. On the Deploy Live page, click the Brokerage field and then click one of the Binance exchanges from the drop-down menu.
4. Enter your API key and secret.

   To generate your API credentials, see Account Types . Your account details are not saved on QuantConnect.

5. Click on the Environment field and then click one of the environments.

   The following table shows the supported environments:

Environment	Description
Real	Trade with real money
Demo	Trade with paper money through the Binance Global brokerage

6. Click the Node field and then click the live trading node that you want to use from the drop-down menu.
7. (Optional) Set up notifications .
8. Configure the Automatically restart algorithm setting.

   By enabling automatic restarts, the algorithm will use best efforts to restart the algorithm if it fails due to a runtime error. This can help improve the algorithm's resilience to temporary outages such as a brokerage API disconnection.

9. Click Deploy .

The deployment process can take up to 5 minutes. When the algorithm deploys, the live results page displays. If you know your brokerage positions before you deployed, you can verify they have been loaded properly by checking your equity value in the runtime statistics, your cashbook holdings, and your position holdings.

10.2.2 Bitfinex

Introduction

QuantConnect enables you to run your algorithms in live mode with real-time market data. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

Bitfinex was founded by Giancarlo Devasini and Raphael Nicolle in 2012 with the goal to "give our users the ultimate cryptocurrency trading experience". Bitfinex provides access to trading Crypto for clients outside prohibited jurisdictions with no minimum deposit to set up an account. If you fund your account with fiat currency, they enforce a 10,000 minimum for USD, EUR, and GBP. However, if you fund your account with Crypto, they do not enforce a minimum deposit. Bitfinex also provides Crypto staking, a mobile app, and an unrealized profit leaderboard for the traders on the platform. Bitfinex has always been at the forefront of technological innovation in digital asset trading.

To view the implementation of the Bitfinex brokerage integration, see the Lean.Brokerages.Bitfinex repository.

Account Types

Bitfinex supports cash and margin accounts. To set the account type in an algorithm, see the Bitfinex brokerage model documentation.

Use `AccountType.Cash` to connect to your Exchange wallet or `AccountType.Margin` to connect to your Margin wallet. You can not connect to your Funding or Capital Raise wallet. If you provide the wrong `AccountType` to the `SetBrokerageModel` method, you may connect to an empty wallet, causing Lean to throw a warning. If you have a currency in your wallet that ends with "F0", it will not load into your `CashBook`.

Create an Account

Follow the account creation wizard on the Bitfinex website to create a Bitfinex account.

You will need API credentials to deploy live algorithms. After you have an account, create API credentials and store them somewhere safe.

Paper Trading

Bitfinex supports paper trading with only the TESTBTCTESTUSD and TESTBTCTESTUSDT securities. Follow these steps to paper trade with Bitfinex:

1. Create a paper trading sub-account and refill the paper balance For instructions, see Paper Trading at Bitfinex on the Bitfinex website.
2. Create an API key for your sub-account. For instructions, see How to create and revoke a Bitfinex API Key on the Bitfinex website.
3. Use `AccountType.Cash` in your algorithms.

To paper trade securities other than TESTBTCTESTUSD and TESTBTCTESTUSDT, follow these steps to simulate paper trading with the QuantConnect Paper Trading brokerage:

1. In the `Initialize` method of your algorithm, add one of the preceding `SetBrokerageModel` method calls.
2. Deploy your algorithm with the QuantConnect Paper Trading brokerage.

Asset Classes

Our Bitfinex integration supports trading Crypto.

```
AddCrypto("BTCUSDT", Resolution.Minute, Market.Bitfinex);
```

```
self.AddCrypto("BTCUSDT", Resolution.Minute, Market.Bitfinex)
```

If you call the `SetBrokerageModel` method with the correct `BrokerageName`, then you don't need to pass a `Market` argument to the `AddCrypto` method because the brokerage model has a default market.

Data Feeds

Our Crypto data feed provides Crypto data during live trading.

Orders

We model the Bitfinex API by supporting several order types, order properties, and order updates. When you deploy live algorithms, you can place manual orders through the IDE.

Order Types

The following table describes the available order types for each asset class that our Bitfinex integration supports:

Order Type	Crypto
MarketOrder	✓
LimitOrder	✓
LimitIfTouchedOrder	✓
StopMarketOrder	✓
StopLimitOrder	✓
MarketOnOpenOrder	✓
MarketOnCloseOrder	✓

Order Properties

We model custom order properties from the Bitfinex API. The following table describes the members of the `BitfinexOrderProperties` object that you can set to customize order execution:

Property	Description
`TimeInForce`	A TimeInForce instruction to apply to the order. The following instructions are supported: • `Day` • `GoodTilCanceled` • `GoodTilDate`
`Hidden`	A flag to signal that the order should be hidden. Hidden orders do not appear in the order book, so they do not influence other market participants. Hidden orders incur the taker fee.
`PostOnly`	A flag to signal that the order must only add liquidity to the order book and not take liquidity from the order book. If part of the order results in taking liquidity rather than providing liquidity, the order is rejected without any part of it being filled.

Updates

We model the Bitfinex API by supporting order updates.

Fees

To view the Bitfinex trading fees, see the Fees Schedule page on the Bitfinex website. To view how we model their fees, see Fees.

To use the Bitfinex brokerage in a live algorithm, the following table shows the fee settings you need on your Account > Fees page on the Bitfinex website:

Fee Setting	Value
Default currency for fees	USD
Fee type for Exchange orders	Currency Exchange Fee

Margin

We model buying power and margin calls to ensure your algorithm stays within the margin requirements.

Slippage

Orders through Bitfinex do not experience slippage in backtests. In paper trading and live trading, your orders may experience slippage.

To view how we model Bitfinex slippage, see Slippage .

Fills

We fill market orders immediately and completely in backtests. In live trading, if the quantity of your market orders exceeds the quantity available at the top of the order book, your orders are filled according to what is available in the order book.

To view how we model Bitfinex order fills, see Fills .

Settlements

Trades settle immediately after the transaction

To view how we model settlement for Bitfinex trades, see Settlement .

Security and Stability

When you deploy live algorithms with Bitfinex, we don't save your brokerage account credentials.

We call the Bitfinex API to place live trades. Sometimes the API may be down. Check the Bitfinex status page to see if the API is currently working.

Deposits and Withdraws

You can deposit and withdraw cash from your brokerage account while you run an algorithm that's connected to the account. We sync the algorithm's cash holdings with the cash holdings in your brokerage account every day at 7:45 AM Eastern Time (ET).

Demo Algorithm

The following algorithm demonstrates the functionality of the Bitfinex brokerage:

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0.35%
Average Loss	0%
Compounding Annual Return	10.689%
Drawdown	17.200%
Expectancy	0
Net Profit	10.653%
Sharpe Ratio	0.528
Probabilistic Sharpe Ratio	27.765%
Loss Rate	0%
Win Rate	100%
Profit-Loss Ratio	0
Alpha	-0.086
Beta	0.238
Annual Standard Deviation	0.164
Annual Variance	0.027
Information Ratio	-1.235
Tracking Error	0.517
Treynor Ratio	0.364
Total Fees	$82.67
Estimated Strategy Capacity	$4500000.00
Lowest Capacity Asset	BTCUSD E3

```
class BitfinexBrokerageExampleAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2021, 1, 1)
        self.SetCash(100000)

        self.SetBrokerageModel(BrokerageName.Bitfinex, AccountType.Margin)

        self.symbol = self.AddCrypto("BTCUSD", Resolution.Minute).Symbol
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0.35%
Average Loss	0%
Compounding Annual Return	10.689%
Drawdown	17.200%
Expectancy	0
Net Profit	10.653%
Sharpe Ratio	0.528
Probabilistic Sharpe Ratio	27.765%
Loss Rate	0%
Win Rate	100%
Profit-Loss Ratio	0
Alpha	-0.086
Beta	0.238
Annual Standard Deviation	0.164
Annual Variance	0.027
Information Ratio	-1.235
Tracking Error	0.517
Treynor Ratio	0.364
Total Fees	$82.67
Estimated Strategy Capacity	$4500000.00
Lowest Capacity Asset	BTCUSD E3

```
namespace QuantConnect.Algorithm.CSharp
{
    public class BitfinexBrokerageExampleAlgorithm : QCAlgorithm
    {
        private Symbol _symbol;

        public override void Initialize()
        {
            SetStartDate(2021, 1, 1);
            SetCash(100000);
```

Virtual Pairs

All fiat and Crypto currencies are individual assets. When you buy a pair like BTCUSD, you trade USD for BTC. In this case, LEAN removes some USD from your portfolio cashbook and adds some BTC. The virtual pair BTCUSD represents your position in that trade, but the virtual pair doesn't actually exist. It simply represents an open trade. When you deploy a live algorithm, LEAN populates your cashbook with the quantity of each currency, but it can't get your position of each virtual pair.

Deploy Live Algorithms

You must have an available live trading node for each live trading algorithm you deploy.

Follow these steps to deploy a live algorithm:

1. Open the project that you want to deploy.

   If you are deploying a paper trading algorithm without the QuantConnect Paper Trading brokerage, include the following lines of code in the `Initialize` method of your algorithm:

   ```
   self.SetAccountCurrency("TESTUSD") # or "TESTUSDT"
   self.SetBrokerageModel(BrokerageName.Bitfinex, AccountType.Cash)
   self.SetBenchmark(lambda x: 0) # or the Symbol of the TESTBTCTESTUSD/TESTBTCTESTUSDT securities
   ```

   ```
   SetAccountCurrency("TESTUSD"); // or "TESTUSDT"
   SetBrokerageModel(BrokerageName.Bitfinex, AccountType.Cash);
   SetBenchmark(x => 0); // or the Symbol of the TESTBTCTESTUSD/TESTBTCTESTUSDT securities
   ```

2. Click the Deploy Live icon.
3. On the Deploy Live page, click the Brokerage field and then click Bitfinex Exchange from the drop-down menu.
4. Enter your API key and secret.

   To generate your API credentials, see Account Types . Your account details are not saved on QuantConnect.

5. Click the Node field and then click the live trading node that you want to use from the drop-down menu.
6. (Optional) Set up notifications .
7. Configure the Automatically restart algorithm setting.

   By enabling automatic restarts, the algorithm will use best efforts to restart the algorithm if it fails due to a runtime error. This can help improve the algorithm's resilience to temporary outages such as a brokerage API disconnection.

8. Click Deploy .

The deployment process can take up to 5 minutes. When the algorithm deploys, the live results page displays. If you know your brokerage positions before you deployed, you can verify they have been loaded properly by checking your equity value in the runtime statistics, your cashbook holdings, and your position holdings.

10.2.3 Coinbase

Introduction

QuantConnect enables you to run your algorithms in live mode with real-time market data. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

Coinbase was founded by Brian Armstrong and Fred Ehrsam in 2012 with the goal to "increase economic freedom in the world". Coinbase provides access to trading Crypto for clients in over 100 countries with no minimum deposit. Coinbase also provides a self-hosted Crypto wallet, a Visa debit rewards card, and Bitcoin collateral-backed lines of credit.

To view the implementation of the Coinbase brokerage integration, see the Lean.Brokerages.CoinbasePro repository.

Account Types

Coinbase supports cash accounts. To set the account type in an algorithm, see the Coinbase brokerage model documentation.

Create an Account

Follow the Create a Coinbase account tutorial on the Coinbase website to create an account.

You will need API credentials to deploy live algorithms. After you have an account, create API credentials and store them somewhere safe. As you create credentials, enable View and Trade permissions.

Paper Trading

Coinbase supports paper trading through the Coinbase Sandbox. You need a Coinbase account to paper trade in the Sandbox.

To create API credentials, log in to your Sandbox account and then follow the instructions for creating credentials for Coinbase. As you create credentials, enable View and Trade permissions.

After you create API credentials for your Sandbox account, follow these steps to add capital to your account:

1. In the top navigation bar of the Sandbox, click Portfolios.
2. On the Portfolios page, click Deposit.
3. In the Deposit window, click the asset that you want to deposit into your account.
4. Click Coinbase.com.
5. In the Amount field, enter the quantity of the asset to deposit.
6. Click Deposit.
7. Click Done.

Asset Classes

Our Coinbase integration supports trading Crypto.

```
AddCrypto("BTCUSD", Resolution.Minute, Market.GDAX);
```

```
self.AddCrypto("BTCUSD", Resolution.Minute, Market.GDAX)
```

If you call the `SetBrokerageModel` method with the correct `BrokerageName`, then you don't need to pass a `Market` argument to the `AddCrypto` method because the brokerage model has a default market.

Data Feeds

Our Crypto data feed provides Crypto data during live trading.

Orders

We model the Coinbase API by supporting several order types, supporting order properties, and not supporting order updates. When you deploy live algorithms, you can place manual orders through the IDE.

Order Types

The following table describes the available order types for each asset class that our Coinbase integration supports:

Order Type	Crypto
MarketOrder	✓
LimitOrder	✓
StopMarketOrder	✓
StopLimitOrder	✓

Order Properties

We model custom order properties from the Coinbase API. The following table describes the members of the `GDAXOrderProperties` object that you can set to customize order execution:

Property	Description
`TimeInForce`	A TimeInForce instruction to apply to the order. The `GoodTilCanceled TimeInForce` is supported.
`PostOnly`	A flag that signals the order must only add liquidity to the order book and not take liquidity from the order book. If part of the order results in taking liquidity rather than providing liquidity, the order is rejected without any part of it being filled.

Updates

We model the Coinbase API by not supporting order updates, but you can cancel an existing order and then create a new order with the desired arguments. For more information about this workaround, see the Workaround for Brokerages That Donâ€™t Support Updates.

Fees

To view the Coinbase trading fees, see the What are the fees on Coinbase? page on the Coinbase website. To view how we model their fees, see Fees.

Margin

Coinbase doesn't support margin trading.

Slippage

Orders through Coinbase do not experience slippage in backtests. In paper trading and live trading, your orders may experience slippage.

To view how we model Coinbase slippage, see Slippage.

Fills

We fill market orders immediately and completely in backtests. In live trading, if the quantity of your market orders exceeds the quantity available at the top of the order book, your orders are filled according to what is available in the order book.

To view how we model Coinbase order fills, see Fills.

Settlements

Trades settle immediately after the transaction

To view how we model settlement for Coinbase trades, see Settlement .

Security and Stability

When you deploy live algorithms with Coinbase, we don't save your brokerage account credentials.

We call the Coinbase API to place live trades. Sometimes the API may be down. Check the Coinbase status page to see if the API is currently working.

Deposits and Withdraws

You can deposit and withdraw cash from your brokerage account while you run an algorithm that's connected to the account. We sync the algorithm's cash holdings with the cash holdings in your brokerage account every day at 7:45 AM Eastern Time (ET).

Demo Algorithm

The following algorithm demonstrates the functionality of the Coinbase brokerage:

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

Overall Statistics	
Total Trades	2
Average Win	0.35%
Average Loss	0%
Compounding Annual Return	11.075%
Drawdown	15.100%
Expectancy	0
Net Profit	10.249%
Sharpe Ratio	0.606
Probabilistic Sharpe Ratio	31.462%
Loss Rate	0%
Win Rate	100%
Profit-Loss Ratio	0
Alpha	-0.083
Beta	0.196
Annual Standard Deviation	0.142
Annual Variance	0.02
Information Ratio	-1.382
Tracking Error	0.561
Treynor Ratio	0.44
Total Fees	$222.14
Estimated Strategy Capacity	$1000000.00
Lowest Capacity Asset	BTCUSD XJ

QUANTCONNECT

```
class CoinbaseProBrokerageExampleAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2021, 1, 1)
        self.SetCash(100000)

        self.SetBrokerageModel(BrokerageName.GDAX, AccountType.Cash)

        self.symbol = self.AddCrypto("BTCUSD", Resolution.Minute).Symbol
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

Overall Statistics	
Total Trades	2
Average Win	0.35%
Average Loss	0%
Compounding Annual Return	11.187%
Drawdown	15.100%
Expectancy	0
Net Profit	10.373%
Sharpe Ratio	0.614
Probabilistic Sharpe Ratio	31.734%
Loss Rate	0%
Win Rate	100%
Profit-Loss Ratio	0
Alpha	-0.087
Beta	0.202
Annual Standard Deviation	0.142
Annual Variance	0.02
Information Ratio	-1.404
Tracking Error	0.549
Treynor Ratio	0.43
Total Fees	$222.14
Estimated Strategy Capacity	$1000000.00
Lowest Capacity Asset	BTCUSD XJ

QUANTCONNECT

```
namespace QuantConnect.Algorithm.CSharp
{
    public class CoinbaseProBrokerageExampleAlgorithm : QCAlgorithm
    {
        private Symbol _symbol;

        public override void Initialize()
        {
            SetStartDate(2021, 1, 1);
            SetCash(100000);
```

Virtual Pairs

All fiat and Crypto currencies are individual assets. When you buy a pair like BTCUSD, you trade USD for BTC. In this case, LEAN removes some USD from your portfolio cashbook and adds some BTC. The virtual pair BTCUSD represents your position in that trade, but the virtual pair doesn't actually exist. It simply represents an open trade. When you deploy a live algorithm, LEAN populates your cashbook with the quantity of each currency, but it can't get your position of each virtual pair.

Deploy Live Algorithms

You must have an available live trading node for each live trading algorithm you deploy.

Follow these steps to deploy a live algorithm:

1. Open the project you want to deploy.
2. Click the Deploy Live icon.
3. On the Deploy Live page, click the Brokerage field and then click Coinbase from the drop-down menu.
4. Enter your Coinbase API key, API secret, and passphrase.

   To generate your API credentials, see the Create an Account section in the Account Types documentation. Your account details are not saved on QuantConnect.

5. Click on the Environment field and then click one of the environments.

   The following table shows the supported environments:

Environment	Description
Live	Trade with real money
Paper	Trade with paper money

6. Click the Node field and then click the live trading node that you want to use from the drop-down menu.
7. (Optional) Set up notifications .
8. Configure the Automatically restart algorithm setting.

   By enabling automatic restarts, the algorithm will use best efforts to restart the algorithm if it fails due to a runtime error. This can help improve the algorithm's resilience to temporary outages such as a brokerage API disconnection.

9. Click Deploy .

The deployment process can take up to 5 minutes. When the algorithm deploys, the live results page displays. If you know your brokerage positions before you deployed, you can verify they have been loaded properly by checking your equity value in the runtime statistics, your cashbook holdings, and your position holdings.

10.2.4 Interactive Brokers

Introduction

QuantConnect enables you to run your algorithms in live mode with real-time market data. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

Interactive Brokers (IB) was founded by Thomas Peterffy in 1993 with the goal to "create technology to provide liquidity on better terms. Compete on price, speed, size, diversity of global products and advanced trading tools". IB provides access to trading Equities, ETFs, Options, Futures, Future Options, Forex, Gold, Warrants, Bonds, and Mutual Funds for clients in over 200 countries and territories with no minimum deposit. IB also provides paper trading, a trading platform, and educational services.

To view the implementation of the IB brokerage integration, see the Lean.Brokerages.InteractiveBrokers repository.

Account Types

The IB API does not support the IBKR LITE plan. You need an IBKR PRO plan. Individual and Financial Advisor (FA) accounts are available.

Individual Accounts

IB supports cash and margin accounts. To set the account type in an algorithm, see the IB brokerage model documentation.

FA Accounts

IB supports FA accounts for Trading Firm and Institution organizations. FA accounts enable certified professionals to use a single trading algorithm to manage several client accounts. For more information about FA accounts, see Financial Advisors.

Create an Account

You need to open an IBKR Pro account to deploy algorithms with IB. The IB API does not support IBKR Lite accounts. To create an IB account, see the Open an Account page on the IB website.

You need to activate IBKR Mobile Authentication (IB Key) to deploy live algorithms with your brokerage account. After you open your account, follow the installation and activation instructions on the IB website.

Paper Trading

IB supports paper trading. Follow the Opening a Paper Trading Account page in the IB documentation to set up your paper trading account.

If you want to use IB data feeds and trade with your paper trading account, follow these steps:

1. Log in to the IB Client Portal.
2. In the top-right corner, click the person icon and then click Settings.
3. In the Account Configuration section, click Paper Trading Account.
4. Click Yes.
5. Click Save.

The IB paper trading environment simulates most aspects of a production Trader Workstation account, but you may encounter some differences due to its construction as a simulator with no execution or clearing abilities.

Asset Classes

Our Interactive Brokers integration supports the following asset classes:

- US Equities
- Equity Options
- Forex
- Futures
- Future Options
- Indices
- Index Options

You may not be able to trade all assets with IB. For example, if you live in the EU, you can't trade US ETFs. Check with your local regulators to know which assets you are allowed to trade. You may need to adjust settings in your brokerage account to live trade some assets.

Data Feeds

You might need to purchase an IB data feed subscription for your trading. For more information about live data feeds, see Data Feeds.

Orders

We model the IB API by supporting several order types, order properties, and order updates. When you deploy live algorithms, you can place manual orders through the IDE.

Order Types

The following table describes the order types that our IB integration supports: supports. For specific details about each order type, refer to the IB documentation.

Order Type	IB Documentation Page
MarketOrder	Market Orders
LimitOrder	Limit Orders
LimitIfTouchedOrder	Limit if Touched Orders
StopMarketOrder	Stop Orders
StopLimitOrder	Stop-Limit Orders
MarketOnOpenOrder	Market-on-Open (MOO) Orders
MarketOnCloseOrder	Market-on-Close (MOC) Orders
ComboMarketOrder	Spread Orders
ComboLimitOrder	Spread Orders
ComboLegLimitOrder	Spread Orders
ExerciseOption	Options Exercise

The following table describes the available order types for each asset class that IB supports:

Order Type	US Equity	Equity Options	Forex	Futures	Futures Options	Index Options
MarketOrder	✓	✓	✓	✓	✓	✓
LimitOrder	✓	✓	✓	✓	✓	✓
LimitIfTouchedOrder	✓	✓	✓	✓	✓	✓
StopMarketOrder	✓	✓	✓	✓	✓	✓
StopLimitOrder	✓	✓	✓	✓	✓	✓
MarketOnOpenOrder	✓	✓	✓			✓
MarketOnCloseOrder	✓	✓	✓	✓	✓	✓
ComboMarketOrder		✓			✓	✓
ComboLimitOrder		✓			✓	✓
ComboLegLimitOrder		✓			✓	✓
ExerciseOption		✓ Not supported for cash-settled Options			✓	

Order Properties

We model custom order properties from the IB API. The following table describes the members of the `InteractiveBrokersOrderProperties` object that you can set to customize order execution. The table does not include the preceding methods for FA accounts.

Property	Description
`TimeInForce`	A TimeInForce instruction to apply to the order. The following instructions are supported: • `Day` • `GoodTilCanceled` • `GoodTilDate`
`OutsideRegularTradingHours`	A flag to signal that the order may be triggered and filled outside of regular trading hours.

Updates

We model the IB API by supporting order updates .

Financial Advisor Group Orders

To place FA group orders, see Financial Advisors .

Fractional Trading

The IB API and FIX/CTCI don't support fractional trading .

Handling Splits

If you're using raw data normalization and you have active orders with a limit, stop, or trigger price in the market for a US Equity when a stock split occurs, the following properties of your orders automatically adjust to reflect the stock split:

- Quantity
- Limit price
- Stop price
- Trigger price

Fill Time

IB has a 400 millisecond fill time for live orders.

Brokerage Liquidations

When IB liquidates part of your position, you receive an order event that contains the `Brokerage Liquidation` message.

Fees

To view the IB trading fees, see the Commissions page on the IB website. To view how we model their fees, see Fees .

Margin

We model buying power and margin calls to ensure your algorithm stays within the margin requirements. If you have more than $25,000 in your brokerage account, you can use the `PatternDayTradingMarginModel` to make use of the 4x intraday leverage and 2x overnight leverage available on most brokerages from the PDT rule .

Slippage

Orders through IB do not experience slippage in backtests. In paper trading and live trading, your orders may experience slippage.

To view how we model IB slippage, see Slippage .

Fills

We fill market orders immediately and completely in backtests. In live trading, if the quantity of your market orders exceeds the quantity available at the top of the order book, your orders are filled according to what is available in the order book.

To view how we model IB order fills, see Fills .

Settlements

If you trade with a margin account, trades settle immediately

To view how we model settlement for IB trades, see Settlement .

Security and Stability

When you deploy live algorithms with IB, we don't save your brokerage account credentials.

We call the IB API to place live trades. Sometimes the API may be down. Check the IB status page to see if the API is currently working.

Connections

By default, IB only supports one connection at a time to your account. If you interfere with your brokerage account while an algorithm is connected to it, the algorithm may stop executing. If you deploy a live running algorithm with your IB account and want to open Trader Workstation (TWS) with the same IB account, create a second user on your IB account and log in to TWS with the new user credentials. To run more than one algorithm with IB, open an IB subaccount for each additional algorithm.

If you can't log in to TWS with your credentials, contact IB. If you can log in to TWS but can't log in to the deployment wizard, contact us and provide the algorithm ID and deployment ID.

SMS 2FA

Our IB integration doesn't support Two-Factor Authentication (2FA) via SMS or the Online Security Code Card. Use the IB Key Security via IBKR Mobile instead.

System Resets

If your IB account has 2FA enabled, you receive a notification on your IB Key device every Sunday to re-authenticate the connection between IB and your live algorithm. When you deploy your algorithm , you can select a time on Sunday to receive the notification. If you don't re-authenticate before the timeout period, your algorithm quits executing. Ensure your IB Key device has sufficient battery for the time you expect to receive the notification. If you don't receive a notification, see I am not receiving IBKR Mobile notifications on the IB website.

Deposits and Withdraws

You can deposit and withdraw cash from your brokerage account while you run an algorithm that's connected to the account. We sync the algorithm's cash holdings with the cash holdings in your brokerage account every day at 7:45 AM Eastern Time (ET).

Demo Algorithm

The following algorithm demonstrates the functionality of the IB brokerage:

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

Overall Statistics

QUANTCONNECT

```
class InteractiveBrokersBrokerageExampleAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2021, 1, 1)
        self.SetCash(100000)

        self.SetBrokerageModel(BrokerageName.InteractiveBrokersBrokerage, AccountType.Margin)

        self.symbol = self.AddEquity("SPY", Resolution.Minute, extendedMarketHours = True).Symbol

        # Set default order properites
        self.DefaultOrderProperties = InteractiveBrokersOrderProperties()
        self.DefaultOrderProperties.Account = "TestAccount1"   # Only for FA accounts
        self.DefaultOrderProperties.FaProfile = "TestProfile1" # Only for FA accounts
        self.DefaultOrderProperties.FaGroup = "TestGroup1"     # Only for FA accounts
        self.DefaultOrderProperties.FaMethod = "EqualQuantity" # Only for FA accounts
        self.DefaultOrderProperties.TimeInForce = TimeInForce.GoodTilCanceled
        self.DefaultOrderProperties.OutsideRegularTradingHours = False


    def OnData(self, data):
        if self.Portfolio.Invested:
            return

        # Place an order with the default order properties
        self.LimitOrder(self.symbol, 10, data[self.symbol].Price + 10)

        # Place an order and with new order properties
        orderProperties = InteractiveBrokersOrderProperties()
        orderProperties.Account = "TestAccount2"   # Only for FA accounts
        orderProperties.FaProfile = "TestProfile2" # Only for FA accounts
        orderProperties.FaGroup = "TestGroup2"     # Only for FA accounts
        orderProperties.FaMethod = "NetLiq"        # Only for FA accounts
        orderProperties.OutsideRegularTradingHours = True
        ticket = self.LimitOrder(self.symbol, 10, round(data[self.symbol].Price * .9, 2), orderProperties = orderProperties)

        # Update the order
        update_fields = UpdateOrderFields()
        update_fields.Quantity = 8
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics

Total Trades	2
Average Win	0%
Average Loss	0%
Compounding Annual Return	1.674%
Drawdown	0.500%
Expectancy	0
Net Profit	1.557%
Sharpe Ratio	1.406
Probabilistic Sharpe Ratio	67.414%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	0
Beta	0
Annual Standard Deviation	0.008
Annual Variance	0
Information Ratio	1.406
Tracking Error	0.008
Treynor Ratio	0
Total Fees	$2.00
Estimated Strategy Capacity	$2600000.00
Lowest Capacity Asset	SPY R735QTJ8XC9X

```
namespace QuantConnect.Algorithm.CSharp
{
    public class InteractiveBrokersBrokerageExampleAlgorithm : QCAlgorithm
    {
        private Symbol _symbol;

        public override void Initialize()
        {
            SetStartDate(2021, 1, 1);
            SetCash(100000);
```

Deploy Live Algorithms

You must have an available live trading node for each live trading algorithm you deploy.

Follow these steps to deploy a live algorithm:

1. Open the project that you want to deploy.
2. Click the Deploy Live icon.
3. On the Deploy Live page, click the Brokerage field and then click Interactive Brokers from the drop-down menu.
4. Enter your IB user name, ID, and password.

   Your account details are not saved on QuantConnect.

5. In the Weekly Restart UTC field, enter the Coordinated Universal Time (UTC) time of when you want to receive notifications on Sundays to re-authenticate your account connection.

   For example, 4 PM UTC is equivalent to 11 AM Eastern Standard Time, 12 PM Eastern Daylight Time, 8 AM Pacific Standard Time, and 9 AM Pacific Daylight Time. To convert from UTC to a different time zone, see the UTC Time Zone Converter on the UTC Time website.

   If your IB account has 2FA enabled, you receive a notification on your IB Key device every Sunday to re-authenticate the connection between IB and your live algorithm. If you don't re-authenticate before the timeout period, your algorithm quits executing.

6. Click the Data Provider field and then click one of the data feeds from the drop-down menu.

   The following table describes the available data feeds:

Data Feed	Description
QuantConnect	Use data collected across all of the exchanges. For more details about this data feed, see Data Feeds .
IB	Use data sourced directly from IB. For more details about this data feed, see the IB data feed guide.
QuantConnect + IB	Use a combination of the QuantConnect and IB data feeds. For more details about this option, see Hybrid QuantConnect Data Feed .

7. If you have subscriptions to IB data feeds and want to use them, click the Data Provider field and then click Interactive Brokers from the drop-down menu.

   If you use IB data feeds and trade with a paper trading account, you need to share the data feed with your paper trading account. For instructions on sharing data feeds, see Account Types .

8. Click the Node field and then click the live trading node that you want to use from the drop-down menu.
9. (Optional) Set up notifications .
10. Configure the Automatically restart algorithm setting.

    By enabling automatic restarts, the algorithm will use best efforts to restart the algorithm if it fails due to a runtime error. This can help improve the algorithm's resilience to temporary outages such as a brokerage API disconnection.

11. Click Deploy .

The deployment process can take up to 5 minutes. When the algorithm deploys, the live results page displays. If you know your brokerage positions before you deployed, you can verify they have been loaded properly by checking your equity value in the runtime statistics, your cashbook holdings, and your position holdings.

10.2.5 Kraken

Introduction

QuantConnect enables you to run your algorithms in live mode with real-time market data. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

Kraken was founded by Jesse Powell in 2011 with the goal to "accelerate the adoption of cryptocurrency so that you and the rest of the world can achieve financial freedom and inclusion". Kraken provides access to trading Crypto through spot and Futures markets for clients with a minimum deposit of around $0-$150 USD for currency and Crypto deposits. Kraken also provides staking services, educational content, and a developer grant program.

To view the implementation of the Kraken brokerage integration, see the Lean.Brokerages.Kraken repository.

Account Types

Kraken supports cash and margin accounts. To set the account type in an algorithm, see the Kraken brokerage model documentation.

Create an Account

Follow the account creation wizard on the Kraken website to create a Kraken account.

You will need API credentials to deploy live algorithms with your brokerage account. After you open your account, create API credentials and store them somewhere safe.

Paper Trading

The Kraken brokerage doesn't support paper trading, but you can follow these steps to simulate it:

1. In the `Initialize` method of your algorithm, add one of the preceding `SetBrokerageModel` method calls.
2. Deploy your algorithm with the QuantConnect Paper Trading brokerage.

Asset Classes

Our Kraken integration supports trading Crypto.

```
AddCrypto("BTCUSDT", Resolution.Minute, Market.Kraken);
```

```
self.AddCrypto("BTCUSDT", Resolution.Minute, Market.Kraken)
```

If you call the `SetBrokerageModel` method with the correct `BrokerageName`, then you don't need to pass a `Market` argument to the `AddCrypto` method because the brokerage model has a default market.

Data Feeds

Our Crypto data feed provides Crypto data during live trading.

Orders

We model the Kraken API by supporting several order types, supporting order properties, and not supporting order updates. When you deploy live algorithms, you can place manual orders through the IDE.

Order Types

The following table describes the available order types for each asset class that our Kraken integration supports:

Order Type	Crypto
MarketOrder	✓
LimitOrder	✓
LimitIfTouchedOrder	✓
StopMarketOrder	✓
StopLimitOrder	✓

Order Properties

We model custom order properties from the Kraken API. The following table describes the members of the `KrakenOrderProperties` object that you can set to customize order execution:

Property	Description
`TimeInForce`	A TimeInForce instruction to apply to the order. The following instructions are supported: • `Day` • `GoodTilCanceled` • `GoodTilDate`
`PostOnly`	A flag to signal that the order must only add liquidity to the order book and not take liquidity from the order book. If part of the order results in taking liquidity rather than providing liquidity, the order is rejected without any part of it being filled.
`FeeInBase`	A flag to signal that the order fees should be paid in the base currency, which is the default behavior when selling. This flag must be the opposite of the `FeeInQuote` flag.
`FeeInQuote`	A flag to signal that the order fees should be paid in the quote currency, which is the default behavior when buying. This flag must be the opposite of the `FeeInBase` flag.
`NoMarketPriceProtection`	A flag to signal that no Market Price Protection should be used.
`ConditionalOrder`	An `Order` that's submitted when the primary order is executed. The `ConditionalOrder` quantity must match the primary order quantity and the `ConditionalOrder` direction must be the opposite of the primary order direction. This order property is only available for live algorithms.

Updates

We model the Kraken API by not supporting order updates, but you can cancel an existing order and then create a new order with the desired arguments. For more information about this workaround, see the Workaround for Brokerages That Donâ€™t Support Updates.

Fees

To view the Kraken trading fees, see the Fee Schedule page on the Kraken website. To view how we model their fees, see Fees.

Margin

We model buying power and margin calls to ensure your algorithm stays within the margin requirements.

Slippage

Orders through Kraken do not experience slippage in backtests. In live trading, your orders may experience slippage.

To view how we model Kraken slippage, see Slippage.

Fills

We fill market orders immediately and completely in backtests. In live trading, if the quantity of your market orders exceeds the quantity available at the top of the order book, your orders are filled according to what is available in the order book.

To view how we model Kraken order fills, see Fills.

Settlements

Trades settle immediately after the transaction

To view how we model settlement for Kraken trades, see Settlement .

Security and Stability

When you deploy live algorithms with Kraken, we don't save your brokerage account credentials.

We call the Kraken API to place live trades. Sometimes the API may be down. Check the Kraken status page to see if the API is currently working.

Deposits and Withdraws

You can deposit and withdraw cash from your brokerage account while you run an algorithm that's connected to the account. We sync the algorithm's cash holdings with the cash holdings in your brokerage account every day at 7:45 AM Eastern Time (ET).

Demo Algorithm

The following algorithm demonstrates the functionality of the Kraken brokerage:

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0%
Average Loss	0%
Compounding Annual Return	44.096%
Drawdown	41.700%
Expectancy	0
Net Profit	40.493%
Sharpe Ratio	0.923
Probabilistic Sharpe Ratio	39.440%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0.145
Beta	0.661
Annual Standard Deviation	0.461
Annual Variance	0.213
Information Ratio	-1.815
Tracking Error	0.241
Treynor Ratio	0.644
Total Fees	$122.73
Estimated Strategy Capacity	$290000.00
Lowest Capacity Asset	BTCUSD 10B

```
class KrakenBrokerageExampleAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2021, 1, 1)
        self.SetCash(100000)

        self.SetBrokerageModel(BrokerageName.Kraken, AccountType.Cash)

        self.symbol = self.AddCrypto("BTCUSD", Resolution.Minute).Symbol
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0%
Average Loss	0%
Compounding Annual Return	44.096%
Drawdown	41.700%
Expectancy	0
Net Profit	40.493%
Sharpe Ratio	0.923
Probabilistic Sharpe Ratio	39.440%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0.145
Beta	0.661
Annual Standard Deviation	0.461
Annual Variance	0.213
Information Ratio	-1.815
Tracking Error	0.241
Treynor Ratio	0.644
Total Fees	$122.73
Estimated Strategy Capacity	$290000.00
Lowest Capacity Asset	BTCUSD 10B

```
namespace QuantConnect.Algorithm.CSharp
{
    public class KrakenBrokerageExampleAlgorithm : QCAlgorithm
    {
        private Symbol _symbol;
        private bool _ordered = false;

        public override void Initialize()
        {
            SetStartDate(2021, 1, 1);
```

Virtual Pairs

All fiat and Crypto currencies are individual assets. When you buy a pair like BTCUSD, you trade USD for BTC. In this case, LEAN removes some USD from your portfolio cashbook and adds some BTC. The virtual pair BTCUSD represents your position in that trade, but the virtual pair doesn't actually exist. It simply represents an open trade. When you deploy a live algorithm, LEAN populates your cashbook with the quantity of each currency, but it can't get your position of each virtual pair.

Deploy Live Algorithms

You must have an available live trading node for each live trading algorithm you deploy.

Follow these steps to deploy a live algorithm:

1. Open the project that you want to deploy.
2. Click the Deploy Live icon.
3. On the Deploy Live page, click the Brokerage field and then click Kraken Exchange from the drop-down menu.
4. Enter your Kraken API secret and key.

   Gather your API credentials from the API Management Settings page on the Kraken website. Your account details are not saved on QuantConnect.

5. Click the Verification Tier field and then click your verification tier from the drop-down menu.

   For more information about verification tiers, see Verification levels explained on the Kraken website.

6. Click the Node field and then click the live trading node that you want to use from the drop-down menu.
7. (Optional) Set up notifications .
8. Configure the Automatically restart algorithm setting.

   By enabling automatic restarts, the algorithm will use best efforts to restart the algorithm if it fails due to a runtime error. This can help improve the algorithm's resilience to temporary outages such as a brokerage API disconnection.

9. Click Deploy .

The deployment process can take up to 5 minutes. When the algorithm deploys, the live results page displays. If you know your brokerage positions before you deployed, you can verify they have been loaded properly by checking your equity value in the runtime statistics, your cashbook holdings, and your position holdings.

10.2.6 Oanda

Introduction

QuantConnect enables you to run your algorithms in live mode with real-time market data. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

Oanda was founded by Dr. Michael Stumm and Dr. Richard Olsen in 1995 with the goal to "transform all aspects of how the world interacts with currencies, whether that be trading or utilizing currency data and information". Oanda provides access to trading Forex and CFDs for clients in over 240 countries and territories with no minimum deposit . Oanda also provides demo accounts, advanced charting tools, and educational content

To view the implementation of the Oanda brokerage integration, see the Lean.Brokerages.OANDA repository .

Account Types

Oanda supports margin accounts. To set the account type in an algorithm, see the Oanda brokerage model documentation .

Create an Account

Follow the How to open an account page on the Oanda website to open an Oanda account.

You will need your account number and access token to deploy live algorithms. To get your account number, open the Account Statement page on the Oanda website. Your account number is formatted as ###-###-#######-### . To get your access token, open the Manage API Access on the Oanda website.

Paper Trading

Oanda supports paper trading. Follow these steps to set up an Oanda paper trading account:

1. Create an Oanda demo account .
2. Log in to your demo account .
3. On the Account page, in the My Services section, click Manage API Access .
4. On the Your key to OANDA's API page, click Generate .

   Your access token displays. Store it somewhere safe. You need your access token to deploy an algorithm with your paper trading account.

5. In the top navigation bar, click My Account .
6. On the Account page, in the Manage Funds section, click View .
7. On the My Funds page, in the Account Summary section, note your v20 Account Number.

   You need your v20 Account Number to deploy an algorithm with your paper trading account.

Asset Classes

Our Oanda integration supports trading Forex and CFDs .

Data Feeds

Our Forex and CFD data feeds provide trading data during live trading.

Orders

We model the Oanda API by supporting several order types, a `TimeInForce` order instruction, and order updates. When you deploy live algorithms, you can place manual orders through the IDE.

Order Types

The following table describes the available order types for each asset class that our Oanda integration supports:

Order Type	Forex	CFD
MarketOrder	✓	✓
LimitOrder	✓	✓
StopMarketOrder	✓	✓

Time In Force

We model the `GoodTilCanceled` TimeInForce from the Oanda API.

Updates

We model the Oanda API by supporting order updates .

Fees

To view the Oanda trading fees, see the Our Charges and Fees page on the Oanda website. To view how we model their fees, see Fees .

Margin

We model buying power and margin calls to ensure your algorithm stays within the margin requirements.

Slippage

Orders through Oanda do not experience slippage in backtests. In paper trading and live trading, your orders may experience slippage.

To view how we model Oanda slippage, see Slippage .

Fills

We fill market orders immediately and completely in backtests. In live trading, if the quantity of your market orders exceeds the quantity available at the top of the order book, your orders are filled according to what is available in the order book.

To view how we model Oanda order fills, see Fills .

Settlements

Trades settle immediately after the transaction

To view how we model settlement for Oanda trades, see Settlement .

Security and Stability

When you deploy live algorithms with Oanda, we don't save your brokerage account credentials.

We call the Oanda API to place live trades. Sometimes the API may be down. Check the Oanda status page to see if the API is currently working.

Deposits and Withdraws

You can deposit and withdraw cash from your brokerage account while you run an algorithm that's connected to the account. We sync the algorithm's cash holdings with the cash holdings in your brokerage account every day at 7:45 AM Eastern Time (ET).

Demo Algorithm

The following algorithm demonstrates the functionality of the Oanda brokerage:

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

Overall Statistics	
Total Trades	1
Average Win	0%
Average Loss	0%
Compounding Annual Return	0.000%
Drawdown	0.000%
Expectancy	0
Net Profit	0.000%
Sharpe Ratio	-1.282
Probabilistic Sharpe Ratio	0.377%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0
Beta	0
Annual Standard Deviation	0
Annual Variance	0
Information Ratio	1.196
Tracking Error	0.047
Treynor Ratio	-0.06
Total Fees	$0.00
Estimated Strategy Capacity	$520000000000.00
Lowest Capacity Asset	EURUSD 8G

QUANTCONNECT

```
class OandaBrokerageExampleAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2021, 1, 1)
        self.SetCash(100000)

        self.SetBrokerageModel(BrokerageName.OandaBrokerage, AccountType.Margin)

        self.symbol = self.AddForex("EURUSD", Resolution.Minute).Symbol
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

Overall Statistics	
Total Trades	1
Average Win	0%
Average Loss	0%
Compounding Annual Return	0.000%
Drawdown	0.000%
Expectancy	0
Net Profit	0.000%
Sharpe Ratio	-1.281
Probabilistic Sharpe Ratio	0.379%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0
Beta	0
Annual Standard Deviation	0
Annual Variance	0
Information Ratio	1.196
Tracking Error	0.047
Treynor Ratio	-0.06
Total Fees	$0.00
Estimated Strategy Capacity	$520000000000.00
Lowest Capacity Asset	EURUSD 8G

QUANTCONNECT

```
namespace QuantConnect.Algorithm.CSharp
{
    public class OandaBrokerageExampleAlgorithm : QCAlgorithm
    {
        private Symbol _symbol;

        public override void Initialize()
        {
            SetStartDate(2021, 1, 1);
            SetCash(100000);
```

Deploy Live Algorithms

You must have an available live trading node for each live trading algorithm you deploy.

Follow these steps to deploy a live algorithm:

1. Open the project you want to deploy.
2. Click the Deploy Live icon.
3. On the Deploy Live page, click the Brokerage field and then click Oanda from the drop-down menu.
4. Enter your Oanda account Id and access token.

   To get your account ID and access token, see the Create an Account section in the Account Types documentation. Your account details are not saved on QuantConnect.

5. Click the Environment field and then click one of the environments.

   The following table shows the supported environments:

Environment	Description
Real	Trade real money with fxTrade
Demo	Trade paper money with fxTrade Practice

6. Click the Node field and then click the live trading node that you want to use from the drop-down menu.
7. (Optional) Set up notifications .
8. Configure the Automatically restart algorithm setting.

   By enabling automatic restarts, the algorithm will use best efforts to restart the algorithm if it fails due to a runtime error. This can help improve the algorithm's resilience to temporary outages such as a brokerage API disconnection.

9. Click Deploy .

The deployment process can take up to 5 minutes. When the algorithm deploys, the live results page displays. If you know your brokerage positions before you deployed, you can verify they have been loaded properly by checking your equity value in the runtime statistics, your cashbook holdings, and your position holdings.

10.2.7 Prime Brokerages

Introduction

QuantConnect enables you to run your algorithms in live mode with real-time market data. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

On the QuantConnect Local Platform, LEAN can connect to the Bloombergâ„¢ Desktop API (DAPI) through our Terminal Link plug-in. This product is in no way affiliated with or endorsed by Bloombergâ„¢; it is simply an add-on. Add Terminal link to your organization to access the 1,300+ prime brokerages in the Bloomberg Execution Management System network.

For more infomation about Terminal Link, refer to the CLI documentation .

10.2.8 QuantConnect Paper Trading

Introduction

QuantConnect enables you to run your algorithms in live mode with real-time market data. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

QuantConnect Paper Trading lets you run live, real-time data feeds into your algorithm but execute trades using fictional capital. Instead of your orders being routed to an exchange when you're paper trading, your order fills are simulated. Use paper trading to test your algorithm without risking real money and to ensure your backtest wasn't overfit before deploying with real money. You can use the paper trading brokerage without needing to sign up for a real brokerage account. If you don't set a brokerage model in your algorithm with the `SetBrokerageModel` method, the paper trading brokerage uses the `DefaultBrokerageModel` to simulate trades.

To view the implementation of the QuantConnect Paper Trading brokerage, see PaperBrokerage.cs in the LEAN GitHub repository. To view the implementation of the backtesting brokerage, see BacktestingBrokerage.cs in the LEAN GitHub repository.

Account Types

The QuantConnect Paper Trading brokerage supports cash and margin accounts. To set the account type in an algorithm, see the paper trading brokerage model documentation.

If you pass a different `BrokerageName` to the `SetBrokerageModel` method, the new brokerage model defines the account types that are available.

Asset Classes

The QuantConnect Paper Trading brokerage supports the following asset classes:

- US Equities
- Crypto
- Forex
- CFD
- Futures
- Future Options

If you set the brokerage model to a model other than the DefaultBrokerageModel, the new brokerage model defines the asset classes you can trade.

Data Feeds

We can only provide paper trading on the assets for which we have a live data feed.

Orders

The following sections describe how the `DefaultBrokerageModel` handles orders. If you set the brokerage model to a different model, the new brokerage model defines how orders are handled.

Order Types

The following table describes the available order types for each asset class that the `DefaultBrokerageModel` supports:

Order Type	US Equity	Crypto	Crypto Futures	Forex	CFD	Futures	Futures Options
MarketOrder	✓	✓	✓	✓	✓	✓	✓
LimitOrder	✓	✓	✓	✓	✓	✓	✓
LimitIfTouchedOrder	✓	✓	✓	✓	✓	✓	✓
StopMarketOrder	✓	✓	✓	✓	✓	✓	✓
StopLimitOrder	✓	✓	✓	✓	✓	✓	✓
MarketOnOpenOrder	✓	✓		✓	✓		
MarketOnCloseOrder	✓	✓		✓	✓	✓	✓
ComboMarketOrder							✓
ComboLimitOrder							✓
ComboLegLimitOrder							✓
ExerciseOption							✓

Time In Force

The `DefaultBrokerageModel` supports the following TimeInForce instructions:

- `Day`
- `GoodTilCanceled`
- `GoodTilDate`

Updates

The `DefaultBrokerageModel` supports order updates.

Handling Splits

If you're using raw data normalization and you have active orders with a limit, stop, or trigger price in the market for a US Equity when a stock split occurs, the following properties of your orders automatically adjust to reflect the stock split:

- Quantity
- Limit price
- Stop price
- Trigger price

Fees

The following table shows the fees that the `DefaultBrokerageModel` charges for each of the supported asset classes:

Asset Class	Fee
Equities	$0.005/share with a $1 minimum fee
Crypto	$0
Forex	$0
CFDs	$0
Futures	$1.85/contract
Future Options	$1.85/contract

There is no fee to exercise Option contracts.

If you set the brokerage model to a model other than the `DefaultBrokerageModel`, the new brokerage model defines the order fees.

To see the fee models that the `DefaultBrokerageModel` uses, see Fees.

Margin

We model buying power and margin calls to ensure your algorithm stays within the margin requirements. If you set the brokerage model to a different model, the new brokerage model defines how margin is modeled. If you have more than $25,000 in your brokerage account, you can use the `PatternDayTradingMarginModel` to make use of the 4x intraday leverage and 2x overnight leverage available on most brokerages from the PDT rule.

Slippage

Orders through the `DefaultBrokerageModel` do not experience slippage in backtests or paper trading. For more information about the slippage model the `DefaultBrokerageModel` uses, see Slippage .

Fills

The `DefaultBrokerageModel` fills market orders immediately and completely. When available, bid and ask spread will be used for the fill prices.

To view how we model realistic order fills, see Fills .

Settlements

If you trade with a margin account, trades settle immediately

To view how we model settlement for paper trades, see Settlement .

Brokerage Models

The QuantConnect Paper Trading brokerage uses the `DefaultBrokerageModel` by default, but you can use any of the brokerage models .

Deposits and Withdraws

The QuantConnect Paper Trading brokerage supports deposits and withdraws.

```
Porfolio.CashBook.Add(AccountCurrency, 100);
Porfolio.CashBook.Add("ETH", -1);
```

```
self.Porfolio.CashBook.Add(self.AccountCurrency, 100)
self.Porfolio.CashBook.Add("ETH", -1)
```

Demo Algorithm

The following algorithm demonstrates the functionality of the `DefaultBrokerageModel` :

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0%
Average Loss	0%
Compounding Annual Return	0.279%
Drawdown	0.100%
Expectancy	0
Net Profit	0.260%
Sharpe Ratio	1.466
Probabilistic Sharpe Ratio	70.005%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0
Beta	0.012
Annual Standard Deviation	0.001
Annual Variance	0
Information Ratio	-1.698
Tracking Error	0.104
Treynor Ratio	0.156
Total Fees	$2.00
Estimated Strategy Capacity	$2700000000.00
Lowest Capacity Asset	SPY R735QTJ8XC9X

```
class QCPaperTradingBrokerageExampleAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2021, 1, 1)
        self.SetCash(100000)

        self.SetBrokerageModel(BrokerageName.QuantConnectBrokerage, AccountType.Margin)

        self.symbol = self.AddEquity("SPY", Resolution.Minute).Symbol
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0%
Average Loss	0%
Compounding Annual Return	0.279%
Drawdown	0.100%
Expectancy	0
Net Profit	0.260%
Sharpe Ratio	1.466
Probabilistic Sharpe Ratio	70.005%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0
Beta	0.012
Annual Standard Deviation	0.001
Annual Variance	0
Information Ratio	-1.698
Tracking Error	0.104
Treynor Ratio	0.156
Total Fees	$2.00
Estimated Strategy Capacity	$2700000000.00
Lowest Capacity Asset	SPY R735QTJ8XC9X

```
namespace QuantConnect.Algorithm.CSharp
{
    public class QCPaperTradingBrokerageExampleAlgorithm : QCAlgorithm
    {
        private Symbol _symbol;

        public override void Initialize()
        {
            SetStartDate(2021, 1, 1);
            SetCash(100000);
```

Deploy Live Algorithms

You must have an available live trading node for each live trading algorithm you deploy.

Follow these steps to deploy a live paper trading algorithm:

1. Open the project that you want to deploy.
2. Click the Deploy Live icon.
3. On the Deploy Live page, click the Brokerage field and then click Paper Trading from the drop-down menu.
4. Click the Node field and then click the live trading node that you want to use from the drop-down menu.
5. (Optional) Follow these steps to start the algorithm with existing cash holdings (see video):
   1. In the Algorithm Cash State section, click Show .
   2. Click Add Currency .
   3. Enter the currency ticker (for example, USD or BTC) and a quantity.
6. (Optional) Follow these steps to start the algorithm with existing position holdings (see video):
   1. In the Algorithm Holdings State section, click Show .
   2. Click Add Holding .
   3. Enter the symbol ID, symbol, quantity, and average price.
7. (Optional) Set up notifications .
8. Configure the Automatically restart algorithm setting.

   By enabling automatic restarts, the algorithm will use best efforts to restart the algorithm if it fails due to a runtime error. This can help improve the algorithm's resilience to temporary outages such as a brokerage API disconnection.

9. Click Deploy .

The deployment process can take up to 5 minutes. When the algorithm deploys, the live results page displays.

10.2.9 Samco

Introduction

QuantConnect enables you to run your algorithms in live mode with real-time market data. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

Samco was founded by Jimeet Modi in2015 with a mission of providing retail investors access to sophisticated financial technology that can assist retail investors in creating wealth at a low cost. Samco provides access to India Equities for clients in India with no minimum balance. Samco also provides stock ratings, mutual funds, and a mini-portfolio investment platform.

To view the implementation of the Samco brokerage integration, see the Lean.Brokerages.Samco repository .

Account Types

Samco supports cash and margin accounts. To set the account type in an algorithm, see the Samco brokerage model documentation .

Samco only supports trading in Indian Rupees, so set the account currency of your algorithm to INR.

Create an Account

Follow the account creation wizard on the Samco website to create a Samco account.

Paper Trading

Samco doesn't support paper trading.

Asset Classes

Our Samco integration supports trading the following asset classes:

- Indian Equities
- India Equity Options
- India Futures

Data Feeds

The Samco data feed provides India Equities data during live trading.

Orders

We model the Samco API by supporting several order types, supporting order properties, and order updates. When you deploy live algorithms, you can place manual orders through the IDE.

Order Types

The following table describes the available order types for each asset class that our Samco integration supports:

Order Type	India Equity
MarketOrder	✓
LimitOrder	✓
StopMarketOrder	✓

Order Properties

We model custom order properties from the Samco API. The following table describes the members of the `IndiaOrderProperties` object that you can set to customize order execution:

Property	Description
`Exchange`	Select the exchange for sending the order to. The following instructions are available: • `NSE` • `BSE`
`ProductType`	A `ProductType` instruction to apply to the order. The `IndiaProductType` enumeration has the following members:
`TimeInForce`	A TimeInForce instruction to apply to the order. The following instructions are available: • `Day` • `GoodTilCanceled` • `GoodTilDate`

Updates

We model the Samco API by supporting order updates .

Handling Splits

If you're using raw data normalization and you have active orders with a limit, stop, or trigger price in the market for a US Equity when a stock split occurs, the following properties of your orders automatically adjust to reflect the stock split:

- Quantity
- Limit price
- Stop price
- Trigger price

Fees

To view the Samco trading fees, see the Regulatory and Exchanges Charges page on the Samco website. To view how we model their fees, see Fees .

Margin

We model buying power and margin calls to ensure your algorithm stays within the margin requirements.

Slippage

Orders through Samco do not experience slippage in backtests. In live trading, your orders may experience slippage.

To view how we model Samco slippage, see Slippage .

Fills

We fill market orders immediately and completely in backtests. In live trading, if the quantity of your market orders exceeds the quantity available at the top of the order book, your orders are filled according to what is available in the order book.

To view how we model Samco order fills, see Fills .

Settlements

If you trade with a margin account, trades settle immediately

To view how we model settlement for Samco trades, see Settlement .

Security and Stability

When you deploy live algorithms with Samco, we don't save your brokerage account credentials.

Deposits and Withdraws

You can deposit and withdraw cash from your brokerage account while you run an algorithm that's connected to the account. We sync the algorithm's cash holdings with the cash holdings in your brokerage account every day at 7:45 AM Eastern Time (ET).

Demo Algorithm

The following algorithm demonstrates the functionality of the Samco brokerage:

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

Overall Statistics	
Total Trades	2
Average Win	0%
Average Loss	0%
Compounding Annual Return	-0.007%
Drawdown	0.000%
Expectancy	0
Net Profit	-0.008%
Sharpe Ratio	-0.561
Probabilistic Sharpe Ratio	2.648%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0
Beta	0
Annual Standard Deviation	0
Annual Variance	0
Information Ratio	-1.437
Tracking Error	0.117
Treynor Ratio	-0.187
Total Fees	$0.02
Estimated Strategy Capacity	$160000000000.00
Lowest Capacity Asset	YESBANK TA8Y7QR36M31

QUANTCONNECT

```
class SamcoBrokerageExampleAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2021, 1, 1)
        self.SetAccountCurrency("INR")
        self.SetCash(100000)

        self.SetBrokerageModel(BrokerageName.Samco, AccountType.Margin)

        self.symbol = self.AddEquity("YESBANK", Resolution.Minute, Market.India).Symbol
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

Overall Statistics	
Total Trades	2
Average Win	0%
Average Loss	0%
Compounding Annual Return	-0.007%
Drawdown	0.000%
Expectancy	0
Net Profit	-0.008%
Sharpe Ratio	-0.561
Probabilistic Sharpe Ratio	2.648%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0
Beta	0
Annual Standard Deviation	0
Annual Variance	0
Information Ratio	-1.437
Tracking Error	0.117
Treynor Ratio	-0.187
Total Fees	$0.02
Estimated Strategy Capacity	$160000000000.00
Lowest Capacity Asset	YESBANK TA8Y7QR36M31

QUANTCONNECT

```
using QuantConnect.Orders;

namespace QuantConnect.Algorithm.CSharp
{
    public class SamcoBrokerageExampleAlgorithm : QCAlgorithm
    {
        private Symbol _symbol;
        private bool _ordered = false;

        public override void Initialize()
```

Deploy Live Algorithms

You must have an available live trading node for each live trading algorithm you deploy.

Follow these steps to deploy a live trading algorithm:

1. Open the project that you want to deploy.
2. Click the Deploy Live icon.
3. On the Deploy Live page, click the Brokerage field and then click Samco from the drop-down menu.
4. Enter your client ID, password, and date of birth.

   Your account details aren't saved on QuantConnect.

5. Click the Product Type field and then click one of the following options from the drop-down menu:

Product Type	Description
MIS	Intraday products
CNC	Delivery products
NRML	Carry forward products

6. Click the Trading Segment field and then click one of the following options from the drop-down menu:

Trading Segment	Description
EQUITY	For trading Equities on the National Stock Exchange of India (NSE) or the Bombay Stock Exchange (BSE)
COMMODITY	For trading commodities on the Multi Commodity Exchange of India (MCX)

7. Click the Node field and then click the live trading node that you want to use from the drop-down menu.
8. (Optional) Set up notifications .
9. Configure the Automatically restart algorithm setting.

   By enabling automatic restarts, the algorithm will use best efforts to restart the algorithm if it fails due to a runtime error. This can help improve the algorithm's resilience to temporary outages such as a brokerage API disconnection.

10. Click Deploy .

The deployment process can take up to 5 minutes. When the algorithm deploys, the live results page displays.

10.2.10 TD Ameritrade

Introduction

QuantConnect enables you to run your algorithms in live mode with real-time market data. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

TD Ameritrade was founded by Joe Ricketts in 1971 with the goal to make "smart investors smarter through award-winning trading technology, education, and service". TD Ameritrade provides access to trading Equities, Options, Futures, Forex, Crypto, and other assets for clients with no minimum deposit. TD Ameritrade also provides a collateral lending program, education services, and a desktop trading platform.

To view the implementation of the TD Ameritrade brokerage integration, see the Lean.Brokerages.TDAmeritrade repository.

Account Types

TD Ameritrade supports cash and margin accounts. To set the account type in an algorithm, see the TD Ameritrade brokerage model documentation.

Create an Account

Follow the account creation wizard on the TD Ameritrade website to create a TD Ameritrade account.

You will need API credentials to deploy live algorithms with your brokerage account. You cannot create new API credentials due to the Trader API Schwab Integration.

If you have the API key, follow these steps to retrieve the access token:

- Use the API key to create the link to generate the access token request:
  https://auth.tdameritrade.com/oauth?client_id= **your-api-key** %40AMER.OAUTHAP&response_type=code&redirect_uri=http%3A%2F%2Flocalhost
- Paste the link into your browser and click Go or hit Enter. It will take you to a "Secure Log-In" page. You need to log in with a TD Ameritrade **client** account.
- Once you have logged in click the "Allow" button. It will take you to a "This site can't be reached error" page. You will find the access token in the page URL:
  http://localhost/?code= **this-blob-is-your-access-code**

The redirect URI must use `http`, as `https` is not supported.

See the following resources for additional information:

- Authentication FAQ on the TD Ameritrade website
- TD Ameritrade API Access - 2019 Guide on r/algotrading

  In this tutorial, the access token is decoded. QuantConnect's implementation required the encoded access token (see **this-blob-is-your-access-code** above).

Paper Trading

The TD Ameritrade API doesn't support paper trading, but you can follow these steps to simulate it:

1. In the `Initialize` method of your algorithm, add one of the preceding `SetBrokerageModel` method calls.
2. Deploy your algorithm with the QuantConnect Paper Trading brokerage.

Asset Classes

Our TD Ameritrade integration supports trading US Equities.

You may not be able to trade all assets with TD Ameritrade. For example, if you live in the EU, you can't trade US ETFs. Check with your local regulators to know which assets you are allowed to trade. You may need to adjust settings in your brokerage account to live trade some assets.

Data Feeds

Our US Equities data feed provides US Equity data during live trading.

Orders

We model the TD Ameritrade API by supporting several order types, the `TimeInForce` order property, and order updates. When you deploy live algorithms, you can place manual orders through the IDE.

Order Types

The following table describes the available order types for each asset class that our TD Ameritrade integration supports:

Order Type	Equity
MarketOrder	✓
LimitOrder	✓
StopMarketOrder	✓
StopLimitOrder	✓

Time In Force

We model the TD Ameritrade API by supporting the following TimeInForce instructions:

- `Day`
- `GoodTilCanceled`
- `GoodTilDate`

Updates

We model the TD Ameritrade API by supporting order updates.

Handling Splits

If you're using raw data normalization and you have active orders with a limit, stop, or trigger price in the market for a US Equity when a stock split occurs, the following properties of your orders automatically adjust to reflect the stock split:

- Quantity
- Limit price
- Stop price
- Trigger price

Fees

To view the TD Ameritrade trading fees, see the Pricing page on the TD Ameritrade website. To view how we model their fees, see Fees.

Margin

We model buying power and margin calls to ensure your algorithm stays within the margin requirements. If you have more than $25,000 in your brokerage account, you can use the `PatternDayTradingMarginModel` to make use of the 4x intraday leverage and 2x overnight leverage available on most brokerages from the PDT rule.

Slippage

Orders through TD Ameritrade do not experience slippage in backtests. In paper trading and live trading, your orders may experience slippage.

To view how we model TD Ameritrade slippage, see Slippage .

Fills

We fill market orders immediately and completely in backtests. In live trading, if the quantity of your market orders exceeds the quantity available at the top of the order book, your orders are filled according to what is available in the order book.

To view how we model TD Ameritrade order fills, see Fills .

Settlements

If you trade with a margin account, trades settle immediately

To view how we model settlement for TD Ameritrade trades, see Settlement .

Security and Stability

When you deploy live algorithms with TD Ameritrade, we don't save your brokerage account credentials.

Deposits and Withdraws

You can deposit and withdraw cash from your brokerage account while you run an algorithm that's connected to the account. We sync the algorithm's cash holdings with the cash holdings in your brokerage account every day at 7:45 AM Eastern Time (ET).

Demo Algorithm

The following algorithm demonstrates the functionality of the TD Ameritrade brokerage:

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0%
Average Loss	0%
Compounding Annual Return	0.307%
Drawdown	2.200%
Expectancy	0
Net Profit	0.600%
Sharpe Ratio	0.187
Probabilistic Sharpe Ratio	8.627%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0.001
Beta	0.072
Annual Standard Deviation	0.012
Annual Variance	0
Information Ratio	-0.294
Tracking Error	0.15
Treynor Ratio	0.03
Total Fees	$0.00
Estimated Strategy Capacity	$53000000.00
Lowest Capacity Asset	SPY R735QTJ8XC9X

```
# region imports
from AlgorithmImports import *
# endregion

class TDAmeritradeBrokerageExampleAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2021, 1, 1)
        self.SetCash(100000)
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

Overall Statistics

QUANTCONNECT

```
#region imports
    using System;
    using System.Collections;
    using System.Collections.Generic;
    using System.Linq;
    using System.Globalization;
    using System.Drawing;
    using QuantConnect;
    using QuantConnect.Algorithm.Framework;
    using QuantConnect.Algorithm.Framework.Selection;
    using QuantConnect.Algorithm.Framework.Alphas;
    using QuantConnect.Algorithm.Framework.Portfolio;
    using QuantConnect.Algorithm.Framework.Execution;
    using QuantConnect.Algorithm.Framework.Risk;
    using QuantConnect.Parameters;
    using QuantConnect.Benchmarks;
    using QuantConnect.Brokerages;
    using QuantConnect.Util;
    using QuantConnect.Interfaces;
    using QuantConnect.Algorithm;
    using QuantConnect.Indicators;
    using QuantConnect.Data;
    using QuantConnect.Data.Consolidators;
    using QuantConnect.Data.Custom;
    using QuantConnect.DataSource;
    using QuantConnect.Data.Fundamental;
    using QuantConnect.Data.Market;
    using QuantConnect.Data.UniverseSelection;
    using QuantConnect.Notifications;
    using QuantConnect.Orders;
    using QuantConnect.Orders.Fees;
    using QuantConnect.Orders.Fills;
    using QuantConnect.Orders.Slippage;
    using QuantConnect.Scheduling;
    using QuantConnect.Securities;
    using QuantConnect.Securities.Equity;
    using QuantConnect.Securities.Future;
    using QuantConnect.Securities.Option;
    using QuantConnect.Securities.Forex;
```

Deploy Live Algorithms

You must have an available live trading node for each live trading algorithm you deploy.

Follow these steps to deploy a live algorithm:

1. Open the project that you want to deploy.
2. Click the Deploy Live icon.
3. On the Deploy Live page, click the Brokerage field and then click TD Ameritrade from the drop-down menu.
4. Enter your TD Ameritrade account ID, key, and token.

   To get your account credentials, see Account Types . Your account details are not saved on QuantConnect.

5. Click the Node field and then click the live trading node that you want to use from the drop-down menu.
6. (Optional) Set up notifications .
7. Configure the Automatically restart algorithm setting.

   By enabling automatic restarts, the algorithm will use best efforts to restart the algorithm if it fails due to a runtime error. This can help improve the algorithm's resilience to temporary outages such as a brokerage API disconnection.

8. Click Deploy .

The deployment process can take up to 5 minutes. When the algorithm deploys, the live results page displays. If you know your brokerage positions before you deployed, you can verify they have been loaded properly by checking your equity value in the runtime statistics, your cashbook holdings, and your position holdings.

10.2.11 Tradier

Introduction

QuantConnect enables you to run your algorithms in live mode with real-time market data. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

Tradier was founded by Dan Raju, Peter Laptewicz, Jason Barry, Jeyashree Chidambaram, and Steve Agalloco in 2012 with the goal to "deliver a choice of low-cost, high-value brokerage services to traders". Tradier provides access to trading Equities and Options for clients in over 250 countries and territories with no minimum deposit for cash accounts . Tradier also delivers custody, clearing, execution, and billing on behalf of registered advisors.

To view the implementation of the Tradier brokerage integration, see the Lean.Brokerages.Tradier repository .

Account Types

Tradier supports cash and margin accounts. To set the account type in an algorithm, see the Tradier brokerage model documentation .

Create an Account

Follow the account creation wizard on the Tradier website to create a Tradier account.

You will need your account ID and access token to deploy live algorithms. After you have an account, get your account ID and token from the Settings > API Access page on the Tradier website. Your account ID is the alpha-numeric code in a drop-down field on the page.

Paper Trading

Tradier supports paper trading, but with the following caveats:

- Account activity is unavailable since this information is populated from Tradier's clearing firm.
- Streaming Tradier market data is unavailable due to exchange restrictions related to delayed data.

To get your paper trading account number and access token, open the API Access page on the Tradier website and then scroll down to the Sandbox Account Access (Paper Trading) section.

If you trade Equities, you can use the QuantConnect data feed to get real-time data. If you trade Options, you must use delayed data from the Tradier data feed . If you trade Equities and Options, use the Tradier data feed. We don't currently have a hybrid QuantConnect-Tradier data feed. If you trade with the demo environment, Tradier doesn't offer streaming market data due to exchange restrictions related to delayed data, so you must use our data feed.

Asset Classes

Our Tradier integration supports trading US Equities and Equity Options .

You may not be able to trade all assets with Tradier. For example, if you live in the EU, you can't trade US ETFs. Check with your local regulators to know which assets you are allowed to trade. You may need to adjust settings in your brokerage account to live trade some assets.

Data Feeds

You might need to purchase a Tradier data feed subscription for your trading. For more information about live data feeds, see Data Feeds .

Orders

We model the Tradier API by supporting several order types and the `TimeInForce` order property. Tradier partially supports order updates, but does not support trading during extended market hours. When you deploy live algorithms, you can place manual orders through the IDE.

Order Types

The following table describes the available order types for each asset class that our Tradier integration supports:

Order Type	Equity	Equity Options
MarketOrder	✓	✓
LimitOrder	✓	✓
StopMarketOrder	✓	✓
StopLimitOrder	✓	✓

Time In Force

We model the Tradier API by supporting the following TimeInForce instructions:

- `Day`
- `GoodTilCanceled` (not available for short selling)
- `GoodTilDate`

Updates

We model the Tradier API by supporting most order updates . To update the quantity of an order, cancel the order and then submit a new order with the desired quantity. For more information about this workaround, see the Workaround for Brokerages That Donâ€™t Support Updates .

Extended Market Hours

Tradier doesn't support extended market hours trading. If you place an order outside of regular trading hours, the order will be processed at market open.

Automatic Cancellations

If you have open orders for a security when it performs a reverse split, Tradier automatically cancels your orders.

Errors

To view the order-related error codes from Tradier, see Error Responses in their documentation.

Fees

To view the Tradier trading fees, see the Pricing page on the Tradier website. To view how we model their fees, see Fees .

Margin

We model buying power and margin calls to ensure your algorithm stays within the margin requirements. If you have more than $25,000 in your brokerage account, you can use the `PatternDayTradingMarginModel` to make use of the 4x intraday leverage and 2x overnight leverage available on most brokerages from the PDT rule .

Slippage

Orders through Tradier do not experience slippage in backtests. In live trading, your orders may experience slippage.

To view how we model Tradier slippage, see Slippage .

Fills

We fill market orders immediately and completely in backtests. In live trading, if the quantity of your market orders exceeds the quantity available at the top of the order book, your orders are filled according to what is available in the

order book.

To view how we model Tradier order fills, see Fills .

Settlements

If you trade with a margin account, trades settle immediately

To view how we model settlement for Tradier trades, see Settlement .

Security and Stability

When you deploy live algorithms with Tradier, we don't save your brokerage account credentials.

We call the Tradier API to place live trades. Sometimes the API may be down. Check the Tradier status page to see if the API is currently working.

Deposits and Withdraws

You can deposit and withdraw cash from your brokerage account while you run an algorithm that's connected to the account. We sync the algorithm's cash holdings with the cash holdings in your brokerage account every day at 7:45 AM Eastern Time (ET).

Demo Algorithm

The following algorithm demonstrates the functionality of the Tradier brokerage:

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0%
Average Loss	0%
Compounding Annual Return	0.281%
Drawdown	0.100%
Expectancy	0
Net Profit	0.262%
Sharpe Ratio	1.48
Probabilistic Sharpe Ratio	70.552%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0
Beta	0.012
Annual Standard Deviation	0.001
Annual Variance	0
Information Ratio	-1.698
Tracking Error	0.104
Treynor Ratio	0.157
Total Fees	$0.00
Estimated Strategy Capacity	$2700000000.00
Lowest Capacity Asset	SPY R735QTJ8XC9X

```
class TradierBrokerageExampleAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2021, 1, 1)
        self.SetCash(100000)

        self.SetBrokerageModel(BrokerageName.TradierBrokerage, AccountType.Margin)

        self.symbol = self.AddEquity("SPY", Resolution.Minute).Symbol
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0%
Average Loss	0%
Compounding Annual Return	0.319%
Drawdown	0.100%
Expectancy	0
Net Profit	0.318%
Sharpe Ratio	1.621
Probabilistic Sharpe Ratio	76.746%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0
Beta	0.013
Annual Standard Deviation	0.001
Annual Variance	0
Information Ratio	-1.855
Tracking Error	0.107
Treynor Ratio	0.176
Total Fees	$0.00
Estimated Strategy Capacity	$2600000000.00
Lowest Capacity Asset	SPY R735QTJ8XC9X

```
namespace QuantConnect.Algorithm.CSharp
{
    public class TradierBrokerageExampleAlgorithm : QCAlgorithm
    {
        private Symbol _symbol;

        public override void Initialize()
        {
            SetStartDate(2021, 1, 1);
            SetCash(100000);
```

Deploy Live Algorithms

You must have an available live trading node for each live trading algorithm you deploy.

Follow these steps to deploy a live algorithm:

1. Open the project that you want to deploy.
2. Click the Deploy Live icon.
3. On the Deploy Live page, click the Brokerage field and then click Tradier from the drop-down menu.
4. Enter your Tradier account Id and token.

   To get your account ID and token, see the Create an Account section in the Account Types documentation. Your account details are not saved on QuantConnect.

5. Click the Environment field and then click one of the environments from the drop-down menu.

   The following table shows the supported environments:

Environment	Description
Real	Trade with real money
Demo	Trade with paper money

6. Click the Data Provider field and then click one of the data feeds from the drop-down menu.

   The following table describes the available data feeds:

Data Feed	Description
QuantConnect	Use data collected across all of the exchanges. For more details about this data feed, see Data Feeds .
Tradier	Use data sourced directly from Tradier. This data feed isn't available for the demo environment. For more details about this data feed, see the Tradier data feed guide.

7. Click the Node field and then click the live trading node that you want to use from the drop-down menu.
8. (Optional) Set up notifications .
9. Configure the Automatically restart algorithm setting.

   By enabling automatic restarts, the algorithm will use best efforts to restart the algorithm if it fails due to a runtime error. This can help improve the algorithm's resilience to temporary outages such as a brokerage API disconnection.

10. Click Deploy .

The deployment process can take up to 5 minutes. When the algorithm deploys, the live results page displays. If you know your brokerage positions before you deployed, you can verify they have been loaded properly by checking your equity value in the runtime statistics, your cashbook holdings, and your position holdings.

10.2.12 Trading Technologies

Introduction

QuantConnect enables you to run your algorithms in live mode with real-time market data. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

Trading Technologies (TT) was founded by Gary Kemp in 1994 with the goal to create professional trading software, infrastructure, and data solutions for a wide variety of users. TT provides access to trading Futures, Options, and Crypto. TT also provides a charting platform, infrastructure services, and risk management tools. TT is not actually a brokerage. The firm is a brokerage router with access to more than 30 execution destinations.

To view the implementation of the TT integration, see the Lean.Brokerages.TradingTechnologies repository .

Modeling

The `TradingTechnologiesBrokerageModel` does not have specific modeling for fees and slippage because TT is an order router and can execute on many exchanges and brokerages. To set the brokerage model and account type in an algorithm, see the TT brokerage model documentation . In live trading, TT reports the total fees of your orders after each order fill. Pass a different `BrokerageName` to `SetBrokerageModel` to backtest your algorithm with fee and slippage modeling. The brokerage model you set should support the asset classes and orders in your algorithm.

Create an Account

Follow the account creation wizard on the TT website to create a TT account.

Paper Trading

Our TT integration does not support paper trading through the TT Simulation environment, but you can follow these steps to simulate it:

1. In the `Initialize` method of your algorithm, add one of the preceding `SetBrokerageModel` method calls.
2. Deploy your algorithm with the QuantConnect Paper Trading brokerage .

Asset Classes

Our TT integration supports trading Futures .

Data Feeds

Our Futures data feed provides Futures data during live trading.

Orders

We model the TT API by supporting several order types, the `TimeInForce` order property, and order updates. When you deploy live algorithms, you can place manual orders through the IDE.

Order Types

The following table describes the available order types for each asset class that our TT integration supports:

Order Type	Futures
MarketOrder	✓
LimitOrder	✓
StopMarketOrder	✓
StopLimitOrder	✓

TT enforces the following order rules:

- If you are buying (selling) with a `StopMarketOrder` or a `StopLimitOrder` , the stop price of the order must be greater (less) than the current security price.
- If you are buying (selling) with a `StopLimitOrder` , the limit price of the order must be greater (less) than the stop price.

Time In Force

We model the TT API by supporting the `Day` and `GoodTilCanceled` TimeInForce order properties.

Updates

We model the TT API by supporting order updates .

Fees

To view the TT trading fees, see the Pricing page on the TT website. To view how we model their fees, see Fees .

Margin

We model buying power and margin calls to ensure your algorithm stays within the margin requirements.

Slippage

Orders through TT do not experience slippage in backtests. In live trading, your orders may experience slippage.

To view how we model TT slippage, see Slippage .

Fills

We fill market orders immediately and completely in backtests. In live trading, if the quantity of your market orders exceeds the quantity available at the top of the order book, your orders are filled according to what is available in the order book.

To view how we model TT order fills, see Fills .

Security and Stability

When you deploy live algorithms with TT, we don't save your brokerage account credentials.

We call the TT API to place live trades. Sometimes the API may be down. Check the TT status page to see if the API is currently working.

Deposits and Withdraws

You can deposit and withdraw cash from your brokerage account while you run an algorithm that's connected to the account. We sync the algorithm's cash holdings with the cash holdings in your brokerage account every day at 7:45 AM Eastern Time (ET).

Demo Algorithm

The following algorithm demonstrates the functionality of the TT brokerage:

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	8
Average Win	28.58%
Average Loss	0%
Compounding Annual Return	139.661%
Drawdown	24.600%
Expectancy	0
Net Profit	125.612%
Sharpe Ratio	2.247
Probabilistic Sharpe Ratio	73.150%
Loss Rate	0%
Win Rate	100%
Profit-Loss Ratio	0
Alpha	0
Beta	0
Annual Standard Deviation	0.461
Annual Variance	0.213
Information Ratio	2.247
Tracking Error	0.461
Treynor Ratio	0
Total Fees	$0.00
Estimated Strategy Capacity	$0
Lowest Capacity Asset	ES XUERCWA6EWAP

```
class TradingTechnologiesBrokerageExampleAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2021, 1, 1)
        self.SetCash(100000)

        self.SetBrokerageModel(BrokerageName.TradingTechnologies, AccountType.Margin)

        self.continuous_contract = self.AddFuture(Futures.Indices.SP500EMini, Resolution.Minute
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	8
Average Win	28.58%
Average Loss	0%
Compounding Annual Return	139.659%
Drawdown	24.600%
Expectancy	0
Net Profit	125.612%
Sharpe Ratio	2.247
Probabilistic Sharpe Ratio	73.150%
Loss Rate	0%
Win Rate	100%
Profit-Loss Ratio	0
Alpha	0
Beta	0
Annual Standard Deviation	0.461
Annual Variance	0.213
Information Ratio	2.247
Tracking Error	0.461
Treynor Ratio	0
Total Fees	$0.00
Estimated Strategy Capacity	$0
Lowest Capacity Asset	ES XUERCWA6EWAP

```
namespace QuantConnect.Algorithm.CSharp
{
    public class TradingTechnologiesBrokerageExampleAlgorithm : QCAlgorithm
    {
        private Future _continuousContract;
        private Security _currentContract;

        public override void Initialize()
        {
            SetStartDate(2021, 1, 1);
```

Deploy Live Algorithms

You must have an available live trading node for each live trading algorithm you deploy.

Follow these steps to deploy a live algorithm:

1. Open the project that you want to deploy.
2. Click the Deploy Live icon.
3. On the Deploy Live page, click the Brokerage field and then click Trading Technologies from the drop-down menu.
4. Enter your TT user name, account name, routing sender, session password, app key, and app secret.

   Our TT integration routes orders via the TT FIX 4.4 Connection. Contact your TT representative to set the exchange where you would like your orders sent. Your account details are not saved on QuantConnect.

   Our integration fetches your positions using the REST endpoint, so the app key and app secret are your REST App credentials.

5. Click the Environment field and then click one of the environments from the drop-down menu.

   The following table shows the supported environments:

Environment	Description
Live	Trade in the production environment
UAT	Trade in the User Acceptance Testing environment

6. Click the Node field and then click the live trading node that you want to use from the drop-down menu.
7. If your brokerage account has existing cash holdings, follow these steps (see video):

    1. In the Algorithm Cash State section, click Show .
    2. Click Add Currency .
    3. Enter the currency ticker (for example, USD or CAD) and a quantity.
8. (Optional) Set up notifications .
9. Configure the Automatically restart algorithm setting.

   By enabling automatic restarts, the algorithm will use best efforts to restart the algorithm if it fails due to a runtime error. This can help improve the algorithm's resilience to temporary outages such as a brokerage API disconnection.

10. Click Deploy .

The deployment process can take up to 5 minutes. When the algorithm deploys, the live results page displays. If you know your brokerage positions before you deployed, you can verify they have been loaded properly by checking your equity value in the runtime statistics, your cashbook holdings, and your position holdings.

10.2.13 Wolverine

Introduction

QuantConnect enables you to run your algorithms in live mode with real-time market data. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

Wolverine Execution Services is a diversified financial institution specializing in proprietary trading, asset management, order execution services, and technology solutions. They are recognized as a market leader in derivatives valuation, trading, and value-added order execution across global Equity, Options, and Futures markets. Their focus on innovation, achievement, and integrity serves the interests of their clients and colleagues. Wolverine Execution Services is headquartered in Chicago, with branch offices in New York, San Francisco, and London. They serve funds that have at least $5M assets under management.

To view the implementation of the Wolverine Execution Services brokerage integration, see the Lean.Brokerages.Wolverine repository .

Account Types

Wolverine Execution Services supports cash and margin accounts. To set the account type in an algorithm, see the Wolverine brokerage model documentation .

Create an Account

To create a Wolverine Execution Services account, contact their staff through the TradeWex website.

Paper Trading

Wolverine Execution Services doesn't support paper trading, but you can follow these steps to simulate it:

1. In the `Initialize` method of your algorithm, add one of the preceding `SetBrokerageModel` method calls.
2. Deploy your algorithm with the QuantConnect Paper Trading brokerage .

Asset Classes

Our Wolverine Execution Services integration supports trading US Equities .

You may not be able to trade all assets with Wolverine. For example, if you live in the EU, you can't trade US ETFs. Check with your local regulators to know which assets you are allowed to trade. You may need to adjust settings in your brokerage account to live trade some assets.

Data Feeds

Our US Equities data feed provides US Equities data during live trading.

Orders

We model the Wolverine Execution Services API by supporting order types, but not order updates or extended market hours trading. When you deploy live algorithms, you can place manual orders through the IDE.

Order Types

Our Wolverine Execution Services integration supports market orders .

Updates

We model the Wolverine Execution Services API by not supporting order updates.

Extended Market Hours

Wolverine Execution Services doesn't support extended market hours trading. If you place an order outside of regular trading hours, the order is invalid.

Fees

Wolverine Execution Services charge $0.005 per share you trade. To view how we model their fees, see Fees .

Margin

We model buying power and margin calls to ensure your algorithm stays within the margin requirements. If you have more than $25,000 in your brokerage account, you can use the `PatternDayTradingMarginModel` to make use of the 4x intraday leverage and 2x overnight leverage available on most brokerages from the PDT rule .

Slippage

Orders through Wolverine Execution Services do not experience slippage in backtests. In live trading, your orders may experience slippage.

To view how we model Wolverine slippage, see Slippage .

Fills

We fill market orders immediately and completely in backtests. In live trading, if the quantity of your market orders exceeds the quantity available at the top of the order book, your orders are filled according to what is available in the order book.

To view how we model Wolverine Execution Services order fills, see Fills .

Settlements

If you trade with a margin account, trades settle immediately

To view how we model settlement for Wolverine trades, see Settlement .

Security and Stability

When you deploy live algorithms with Wolverine Execution Services, we don't save your brokerage account credentials.

Deposits and Withdraws

You can deposit and withdraw cash from your brokerage account while you run an algorithm that's connected to the account. We sync the algorithm's cash holdings with the cash holdings in your brokerage account every day at 7:45 AM Eastern Time (ET).

Demo Algorithm

The following algorithm demonstrates the functionality of the Wolverine Execution Services brokerage:

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	1
Average Win	0%
Average Loss	0%
Compounding Annual Return	1.533%
Drawdown	11.200%
Expectancy	0
Net Profit	2.869%
Sharpe Ratio	0.199
Probabilistic Sharpe Ratio	9.459%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0.004
Beta	0.389
Annual Standard Deviation	0.063
Annual Variance	0.004
Information Ratio	-0.311
Tracking Error	0.099
Treynor Ratio	0.032
Total Fees	$0.00
Estimated Strategy Capacity	$230000000.00
Lowest Capacity Asset	SPY R735QTJ8XC9X

```
#region imports
from AlgorithmImports import *
#endregion
class WolverineBrokerageExampleAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2021, 1, 1)
        self.SetCash(100000)

        self.SetBrokerageModel(BrokerageName.Wolverine, AccountType.Margin)
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	1
Average Win	0%
Average Loss	0%
Compounding Annual Return	1.533%
Drawdown	11.200%
Expectancy	0
Net Profit	2.869%
Sharpe Ratio	0.199
Probabilistic Sharpe Ratio	9.459%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0.004
Beta	0.389
Annual Standard Deviation	0.063
Annual Variance	0.004
Information Ratio	-0.311
Tracking Error	0.099
Treynor Ratio	0.032
Total Fees	$0.00
Estimated Strategy Capacity	$230000000.00
Lowest Capacity Asset	SPY R735QTJ8XC9X

```
#region imports
    using QuantConnect.Brokerages;
    using QuantConnect.Data;
    using QuantConnect.Orders;
#endregion
namespace QuantConnect.Algorithm.CSharp
{
    public class WolverineBrokerageExampleAlgorithm : QCAlgorithm
    {
        private Symbol _symbol;
```

Deploy Live Algorithms

You must have an available live trading node for each live trading algorithm you deploy.

Follow these steps to deploy a live algorithm:

1. Open the project that you want to deploy.
2. Click the Deploy Live icon.
3. On the Deploy Live page, click the Brokerage field and then click Wolverine Execution Services from the drop-down menu.
4. Enter your Wolverine Execution Services credentials.

   Your account details are not saved on QuantConnect.

5. Click the Node field and then click the live trading node that you want to use from the drop-down menu.
6. If your brokerage account has existing cash holdings, follow these steps (see video):
   1. In the Algorithm Cash State section, click Show .
   2. Click Add Currency .
   3. Enter the currency ticker (for example, USD or CAD) and a quantity.
7. If your brokerage account has existing position holdings, follow these steps (see video):
   1. In the Algorithm Holdings State section, click Show .
   2. Click Add Holding .
   3. Enter the symbol ID, symbol, quantity, and average price.
8. (Optional) Set up notifications .
9. Configure the Automatically restart algorithm setting.

By enabling automatic restarts, the algorithm will use best efforts to restart the algorithm if it fails due to a runtime error. This can help improve the algorithm's resilience to temporary outages such as a brokerage API disconnection.

10. Click Deploy .

The deployment process can take up to 5 minutes. When the algorithm deploys, the live results page displays. If you know your brokerage positions before you deployed, you can verify they have been loaded properly by checking your equity value in the runtime statistics, your cashbook holdings, and your position holdings.

10.2.14 Zerodha

Introduction

QuantConnect enables you to run your algorithms in live mode with real-time market data. We have successfully hosted more than 200,000 live algorithms and have had more than $22B in volume traded on our servers since 2015. Brokerages supply a connection to the exchanges so that you can automate orders using LEAN. You can use multiple data feeds in live trading algorithms.

Zerodha was founded by Nithin Kamath in 2010 with the goal to break all barriers that traders and investors face in India in terms of cost, support, and technology. Zerodha provides access to India Equities for clients in India with no minimum balance required. Zerodha also provides a mutual fund investment platform and an interactive portfolio dashboard.

To view the implementation of the Zerodha brokerage integration, see the Lean.Brokerages.Zerodha repository .

Account Types

Zerodha supports cash and margin accounts. To set the account type in an algorithm, see the Zerodha brokerage model documentation .

Zerodha only supports trading Indian Rupees, so set the account currency of your algorithm to INR.

Create an Account

To create a Zerodha account, follow the account creation wizard on the Zerodha website.

You will need API credentials to deploy live algorithms with your brokerage account. After you open your Zerodha account, follow these steps to get your API credentials:

1. Create a Kite Connect developer account .
2. On the My apps page, click Create new app .
3. On the Create a new app page, fill in the form.

   For the Redirect URL field, if you don't have a redirect URL, use https://zerodha.com.

4. Click Create .
5. Copy and save your API key and API secret.
6. In a terminal, run the following Python script:

```
from kiteconnect import KiteConnect
import webbrowser

api_key = input ("Enter your API key: ")
kite = KiteConnect(api_key=api_key)
â€‹webbrowser.open(kite.login_url())
request_token = input ("Enter your request_token: ")
api_secret = input ("Enter your API secret: ")
data = kite.generate_session(request_token, api_secret=api_secret)
kite.set_access_token(data["access_token"])
print(f"Your access_token is {data['access_token']}")
```

   Input the data that the script requests. When the script opens your redirect URL page, log in and then copy the request token that's in the URL parameters.

   When the script prints your access token. Copy and save it somewhere safe.

Paper Trading

Zerodha doesn't support paper trading.

Asset Classes

Our Zerodha integration supports trading Indian Equities .

Data Feeds

The Zerodha data feed provides India Equities data during live trading.

Orders

We model the Zerodha API by supporting several order types, order properties, and order updates. When you deploy live algorithms, you can place manual orders through the IDE.

Order Types

The following table describes the available order types for each asset class that our Zerodha integration supports:

Order Type	India Equity
MarketOrder	✓
LimitOrder	✓
StopMarketOrder	✓
StopLimitOrder	✓

Order Properties

We model custom order properties from the Zerodha API. The following table describes the members of the `IndiaOrderProperties` object that you can set to customize order execution:

Property	Description
`Exchange`	Select the exchange for sending the order to. The following instructions are supported: • `NSE` • `BSE`
`ProductType`	A `ProductType` instruction to apply to the order. The `IndiaProductType` enumeration has the following members:
`TimeInForce`	A TimeInForce instruction to apply to the order. The following instructions are supported: • `Day` • `GoodTilCanceled` • `GoodTilDate`

Updates

We model the Samco Zerodha by supporting order updates .

Handling Splits

If you're using raw data normalization and you have active orders with a limit, stop, or trigger price in the market for a US Equity when a stock split occurs, the following properties of your orders automatically adjust to reflect the stock split:

- Quantity
- Limit price
- Stop price

- Trigger price

Fees

To view the Zerodha trading fees, see the Charges page on the Zerodha website. To view how we model their fees, see Fees .

Margin

We model buying power and margin calls to ensure your algorithm stays within the margin requirements. The amount of margin available depends on the Equity and product type. To check the amount of margin available for each asset, see the Margin Calculator on the Zerodha website.

Slippage

Orders through Zerodha do not experience slippage in backtests. In live trading, your orders may experience slippage.

To view how we model Zerodha slippage, see Slippage .

Fills

We fill market orders immediately and completely in backtests. In live trading, if the quantity of your market orders exceeds the quantity available at the top of the order book, your orders are filled according to what is available in the order book.

To view how we model Zerodha order fills, see Fills .

Settlements

If you trade with a margin account, trades settle immediately

To view how we model settlement for Zerodha trades, see Settlement .

Security and Stability

When you deploy live algorithms with Zerodha, we don't save your brokerage account credentials.

Deposits and Withdraws

You can deposit and withdraw cash from your brokerage account while you run an algorithm that's connected to the account. We sync the algorithm's cash holdings with the cash holdings in your brokerage account every day at 7:45 AM Eastern Time (ET).

Demo Algorithm

The following algorithm demonstrates the functionality of the Zerodha brokerage:

- Charts
- Statistics
- Code
  - main.py
  - research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0%
Average Loss	0%
Compounding Annual Return	-0.007%
Drawdown	0.000%
Expectancy	0
Net Profit	-0.008%
Sharpe Ratio	-0.561
Probabilistic Sharpe Ratio	2.648%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0
Beta	0
Annual Standard Deviation	0
Annual Variance	0
Information Ratio	-1.492
Tracking Error	0.109
Treynor Ratio	-1.093
Total Fees	$0.02
Estimated Strategy Capacity	$160000000000.00
Lowest Capacity Asset	YESBANK TA8Y7QR36M31

```
class ZerodhaBrokerageExampleAlgorithm(QCAlgorithm):

    def Initialize(self):
        self.SetStartDate(2021, 1, 1)
        self.SetAccountCurrency("INR")
        self.SetCash(100000)

        self.SetBrokerageModel(BrokerageName.Zerodha, AccountType.Margin)

        self.symbol = self.AddEquity("YESBANK", Resolution.Minute, Market.India).Symbol
```

- Charts
- Statistics
- Code
  - Main.cs
  - Research.ipynb

Clone Algorithm

QUANTCONNECT

Overall Statistics	
Total Trades	2
Average Win	0%
Average Loss	0%
Compounding Annual Return	-0.007%
Drawdown	0.000%
Expectancy	0
Net Profit	-0.008%
Sharpe Ratio	-0.561
Probabilistic Sharpe Ratio	2.648%
Loss Rate	0%
Win Rate	0%
Profit-Loss Ratio	0
Alpha	-0
Beta	0
Annual Standard Deviation	0
Annual Variance	0
Information Ratio	-1.492
Tracking Error	0.109
Treynor Ratio	-1.093
Total Fees	$0.02
Estimated Strategy Capacity	$160000000000.00
Lowest Capacity Asset	YESBANK TA8Y7QR36M31

```
using QuantConnect.Orders;

namespace QuantConnect.Algorithm.CSharp
{
    public class ZerodhaBrokerageExampleAlgorithm : QCAlgorithm
    {
        private Symbol _symbol;
        private bool _ordered = false;

        public override void Initialize()
```

Deploy Live Algorithms

You must have an available live trading node for each live trading algorithm you deploy.

Follow these steps to deploy a live trading algorithm:

1. Open the project that you want to deploy.
2. Click the Deploy Live icon.
3. On the Deploy Live page, click the Brokerage field and then click Zerodha from the drop-down menu.
4. Enter your Kite Connect access token and key.

   To get your access token and key, see Account Types . Your account details aren't saved on QuantConnect.

5. Click the Product Type field and then click one of the following options from the drop-down menu:

Product Type	Description
MIS	Intraday products
CNC	Delivery products
NRML	Carry forward products

6. Click the Trading Segment field and then click one of the following options from the drop-down menu:

Trading Segment	Description
EQUITY	For trading Equities on the National Stock Exchange of India (NSE) or the Bombay Stock Exchange (BSE)
COMMODITY	For trading commodities on the Multi Commodity Exchange of India (MCX)

7. Click the History Subscription field and then click Yes or No from the drop-down menu.

   Use this field to declare whether you have a history API subscription on your Kite Connect account.

8. Click the Node field and then click the live trading node that you want to use from the drop-down menu.
9. (Optional) Set up notifications .
10. Configure the Automatically restart algorithm setting.

    By enabling automatic restarts, the algorithm will use best efforts to restart the algorithm if it fails due to a runtime error. This can help improve the algorithm's resilience to temporary outages such as a brokerage API disconnection.

11. Click Deploy .

The deployment process can take up to 5 minutes. When the algorithm deploys, the live results page displays.

10.2.15 FIX Connections

Introduction

The Financial Information eXchange (FIX) is the standard electronic communications protocol for front-office messaging. The FIX community includes about 300 firms, including major investment banks.

Supported Connections

The following FIX connections are available on QuantConnect:

Name	Integration Implementation	Model Implementation
Raiffeisen Bank International	Lean.Brokerages.RaiffeisenBankInternational	RBIBrokerageModel.cs

10.2.16 Unsupported Brokerages

Introduction

New brokerages can be added if the brokerage has an API that is popular, stable, and officially supported by the brokerage. To add a new brokerage to the platform, contact us .

10.3 Data Feeds

Data feeds are a stream of asset prices and quotes delivered to your trading algorithm during live execution. You need live data feeds to inject data into your algorithm so that you can make real-time trading decisions and so that the values of the securities in your portfolio are updated. You can source data from QuantConnect or your brokerage.

US Equities

From all US Equity markets

Crypto

From several exchanges

Crypto Futures

From Binance

CFD

From Oanda

Forex

From Oanda

Futures

From several Futures markets

Future Options

From several FOP markets

Alternative Data

Supplied by data vendors

Brokerage Data Feeds

Sourced directly from brokerages

See Also

Brokerage Data Feeds
Datasets
Brokerages

10.3.1 US Equities

Introduction

The US Equities data feed is a stream of security trades and quotes delivered to your trading algorithm during live execution. Live data feeds enable you to make real-time trades and update the value of the securities in your portfolio.

The US Equity Security Master data feed provides a live stream of corporate actions. The US Fundamentals data feed provides daily updates on company fundamentals. The US Equities Short Availability data feed provides the number of shares that are available for short sellers to borrow.

Sourcing

The US Equities data feed consolidates market data across all of the exchanges. Over-the-Counter (OTC) trades are excluded. The data feed is powered by the Securities Information Processor (SIP), so it has 100% market coverage. In contrast, free platforms that display data feeds like the Better Alternative Trading System (BATS) only have about 6-7% market coverage .

We provide live splits, dividends, and corporate actions for US companies. We deliver them to your algorithm before the trading day starts.

Universe Selection

The US Equities data feed enables you to create a dynamic universe of securities.

Coarse-Fine Universe

The live data for coarse and fine universe selection arrives at 7 AM Eastern Time (ET), so coarse and fine universe selection runs for live algorithms between 7 and 8 AM ET. This timing allows you to place trades before the market opens. Don't schedule anything for midnight because the universe selection data isn't ready yet.

```
AddUniverse(SelectCoarse, SelectFine);
```

```
self.AddUniverse(self.SelectCoarse, self.SelectFine)
```

ETF Constituent Universe

The US Equities data feed enables you to create a universe of securities to match the constituents of an ETF. For more information about ETF universes, see ETF Constituents Selection .

```
var spy = AddEquity("SPY").Symbol;
AddUniverse(Universe.ETF(spy, UniverseSettings, ETFConstituentsFilter));
```

```
spy = self.AddEquity("SPY").Symbol
self.AddUniverse(self.Universe.ETF(spy, self.UniverseSettings, self.ETFConstituentsFilter))
```

Bar Building

We aggregate ticks to build bars.

Discrepancies

In live trading, bars are built using the exchange timestamps with microsecond accuracy. This microsecond-by-microsecond processing of the ticks can mean that the individual bars between live trading and backtesting can have slightly different ticks. As a result, it's possible for a tick to be counted in different bars between backtesting and live trading, which can lead to bars having slightly different open, high, low, close, and volume values.

Opening and Closing Auctions

The opening and closing price of the day is set by very specific opening and closing auction ticks. When a stock like Apple is listed, itâ€™s listed on Nasdaq. The open auction tick on Nasdaq is the price thatâ€™s used as the official open of the day. NYSE, BATS, and other exchanges also have opening auctions, but the only official opening price for Apple is the opening auction on the exchange where it was listed.

We set the opening and closing prices of the first and last bars of the day to the official auction prices. This process is used for second, minute, hour, and daily bars for the 9:30 AM and 4:30 PM Eastern Time (ET) prices. In contrast, other platforms might not be using the correct opening and closing prices.

The official auction prices are usually emitted 2-30 seconds after the market open and close. We do our best to use the official opening and closing prices in the bars we build, but the delay can be so large that there isn't enough time to update the opening and closing price of the bar before it's injected into your algorithms. For example, if you subscribe to second resolution data, we wait until the end of the second for the opening price but most second resolution data wonâ€™t get the official opening price. If you subscribe to minute resolution data, we wait until the end of the minute for the opening auction price. Most of the time, youâ€™ll get the actual opening auction price with minute resolution data, but there are always exceptions. Nasdaq and NYSE can have delays in publishing the opening auction price, but we donâ€™t have control over those issues and we have to emit the data on time so that you get the bar you are expecting.

Excluded Ticks

The bar-building process can exclude ticks. If a tick is excluded, its volume is aggregated in the bar but its price is not aggregated in the bar. Ticks are excluded if any of the following statements are true:

- The tick is suspicious.
- The tick is from the FINRA exchange and meets our price and volume thresholds.
- The trade has none of the following included `TradeConditionFlags` and at least one of the following excluded `TradeConditionFlags` :

TradeConditionFlags	Status	Description
Regular	Included	A trade made without stated conditions is deemed the regular way for settlement on the third business day following the transaction date.
FormT	Included	Trading in extended hours enables investors to react quickly to events that typically occur outside regular market hours, such as earnings reports. However, liquidity may be constrained during such Form T trading, resulting in wide bid-ask spreads.
Cash	Included	A transaction that requires delivery of securities and payment on the same day the trade takes place.
ExtendedHours	Included	Identifies a trade that was executed outside of regular primary market hours and is reported as an extended hours trade.
NextDay	Included	A transaction that requires the delivery of securities on the first business day following the trade date.
OfficialClose	Included	Indicates the "official" closing value determined by a Market Center. This transaction report will contain the market center generated closing price.
OfficialOpen	Included	Indicates the 'Official' open value as determined by a Market Center. This transaction report will contain the market center generated opening price.
ClosingPrints	Included	The transaction that constituted the trade-through was a single priced closing transaction by the Market Center.
OpeningPrints	Included	The trade that constituted the trade-through was a single priced opening transaction by the Market Center.
IntermarketSweep	Excluded	The transaction that constituted the trade-through was the execution of an order identified as an Intermarket Sweep Order.
TradeThroughExempt	Excluded	Denotes whether or not a trade is exempt (Rule 611).
OddLot	Excluded	Denotes the trade is an odd lot less than a 100 shares.

- The quote has a size of less than 100 shares.
- The quote has one of the following `QuoteConditionFlags` :

QuoteConditionFlags	Description
Closing	Indicates that this quote was the last quote for a security for that Participant.
NewsDissemination	Denotes a regulatory trading halt when relevant news influencing the security is being disseminated. Trading is suspended until the primary market determines that an adequate publication or disclosure of information has occurred.
NewsPending	Denotes a regulatory Trading Halt due to an expected news announcement, which may influence the security. An Opening Delay or Trading Halt may be continued once the news has been disseminated.
TradingRangeIndication	Denotes the probable trading range (Bid and Offer prices, no sizes) of a security that is not Opening Delayed or Trading Halted. The Trading Range Indication is used prior to or after the opening of a security.
OrderImbalance	Denotes a non-regulatory halt condition where there is a significant imbalance of buy or sell orders.
Resume	Indicates that trading for a Participant is no longer suspended in a security that had been Opening Delayed or Trading Halted.

- The quote has none of the following `QuoteConditionFlags` :

QuoteConditionFlags	Description
Regular	This condition is used for the majority of quotes to indicate a normal trading environment.
Slow	This condition is used to indicate that the quote is a Slow Quote on both the bid and offer sides due to a Set Slow List that includes high price securities.
Gap	While in this mode, auto-execution is not eligible, the quote is then considered manual and non-firm in the bid and offer, and either or both sides can be traded through as per Regulation NMS.
OpeningQuote	This condition can be disseminated to indicate that this quote was the opening quote for a security for that Participant.

QuoteConditionFlags	Description
Resume	For extremely active periods of short duration. While in this mode, the UTP Participant will enter quotations on a best Indicate that trading for a Participant is no longer suspended in a security which had been Opening Delayed or Trading Halted.

In the preceding tables, Participant refers to the entities on page 19 of the Consolidated Tape System Multicast Output Binary Specification .

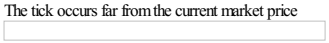
Suspicious Ticks

Tick price data is raw and unfiltered, so it can contain a lot of noise. If a tick is not tradable, we flag it as suspicious. This process makes the bars a more realistic representation of what you could execute in live trading. If you use tick data, avoid using suspicious ticks in your algorithms as informative data points. We recommend only using tick data if you understand the risks and are able to perform your own tick filtering. Ticks are flagged as suspicious in the following situations:

- The tick occurs below the best bid or above the best ask

  This image shows a tick that occurred above the best ask price of a security. The green line represents the best ask of the security, the blue line represents the best bid of the security, and the red dots represent trade ticks. The ticks between the best bid and ask occur from filling hidden orders. The tick that occurred above the best ask price is flagged as suspicious.

- The tick occurs far from the current market price

  This image shows a tick that occurred far from the price of the security. The red dots represent trade ticks. The tick that occurred far from the market price is flagged as suspicious.

- The tick occurs on a dark pool
- The tick is rolled back
- The tick is reported late

Delivery

Most live trading algorithms run on co-located servers racked in Equinix. Co-location reduces several factors that can interfere with your algorithm, including downtime from internet outages, equipment repairs, and natural disasters.

Live data takes time to travel from the source to your algorithm. The US Equities data feed has a latency of 20-50 milliseconds. QuantConnect is not designed for high-frequency trading.

Pricing

The US Equities data feed is free.

10.3.2 Crypto

Introduction

The Crypto data feed is a stream of security trades and quotes delivered to your trading algorithm during live execution. Live data feeds enable you to make real-time trades and update the value of the securities in your portfolio.

Sourcing

The Crypto data feed uses WebSockets to gather market data from the following sources:

- Binance & Binance US
- Bitfinex
- Coinbase
- Kraken

Universe Selection

The Crypto data feeds enable you to create a dynamic universe of securities. The live data for Crypto universe selection arrives at 4 PM Coordinated Universal Time (UTC), so universe selection runs for live algorithms between 4 PM and 4:30 PM. Don't schedule anything for midnight because the universe selection data isn't ready yet.

```
AddUniverse(CryptoCoarseFundamentalUniverse(Market.Bitfinex, UniverseSettings, UniverseSelectionFilter));
```

```
self.AddUniverse(CryptoCoarseFundamentalUniverse(Market.Bitfinex, self.UniverseSettings, self.UniverseSelectionFilter))
```

To view an example for each Crypto market, see the Universe Selection section of the Crypto market in the Dataset Market .

Bar Building

We aggregate ticks to build bars.

In live trading, bars are built using the exchange timestamps with microsecond accuracy. This microsecond-by-microsecond processing of the ticks can mean that the individual bars between live trading and backtesting can have slightly different ticks. As a result, it's possible for a tick to be counted in different bars between backtesting and live trading, which can lead to bars having slightly different open, high, low, close, and volume values.

Delivery

Most live trading algorithms run on co-located servers racked in Equinix. Co-location reduces several factors that can interfere with your algorithm, including downtime from internet outages, equipment repairs, and natural disasters.

Live data takes time to travel from the source to your algorithm. The Crypto data feed has a latency of 20-50 milliseconds. QuantConnect is not designed for high-frequency trading.

Pricing

The Crypto data feed is free.

10.3.3 Crypto Futures

Introduction

The Crypto Futures data feed is a stream of security trades and quotes delivered to your trading algorithm during live execution. Live data feeds enable you to make real-time trades and update the value of the securities in your portfolio.

Sourcing

The Crypto Futures data feed uses WebSockets to gather market data from Binance.

Bar Building

We aggregate ticks to build bars.

In live trading, bars are built using the exchange timestamps with microsecond accuracy. This microsecond-by-microsecond processing of the ticks can mean that the individual bars between live trading and backtesting can have slightly different ticks. As a result, it's possible for a tick to be counted in different bars between backtesting and live trading, which can lead to bars having slightly different open, high, low, close, and volume values.

Delivery

Most live trading algorithms run on co-located servers racked in Equinix. Co-location reduces several factors that can interfere with your algorithm, including downtime from internet outages, equipment repairs, and natural disasters.

Live data takes time to travel from the source to your algorithm. The Crypto Futures data feed has a latency of 20-50 milliseconds. QuantConnect is not designed for high-frequency trading.

Pricing

The Crypto Futures data feed is free.

10.3.4 CFD

Introduction

The CFD data feed is a stream of contract trades and quotes delivered to your trading algorithm during live execution. Live data feeds enable you to make real-time trades and update the value of the securities in your portfolio.

Sourcing

The CFD data feed consolidates market data from OANDA.

Bar Building

We aggregate ticks to build bars.

In live trading, bars are built using the exchange timestamps with microsecond accuracy. This microsecond-by-microsecond processing of the ticks can mean that the individual bars between live trading and backtesting can have slightly different ticks. As a result, it's possible for a tick to be counted in different bars between backtesting and live trading, which can lead to bars having slightly different open, high, low, close, and volume values.

Delivery

Most live trading algorithms run on co-located servers racked in Equinix. Co-location reduces several factors that can interfere with your algorithm, including downtime from internet outages, equipment repairs, and natural disasters.

Live data takes time to travel from the source to your algorithm. The CFD data feed has a latency of 20-50 milliseconds. QuantConnect is not designed for high-frequency trading.

Pricing

The CFD data feed is free.

10.3.5 Forex

Introduction

The Forex data feed is a stream of forex pairs trades and quotes delivered to your trading algorithm during live execution. Live data feeds enable you to make real-time trades and update the value of the securities in your portfolio.

Sourcing

The Forex data feed consolidates market data from OANDA.

Bar Building

We aggregate ticks to build bars.

In live trading, bars are built using the exchange timestamps with microsecond accuracy. This microsecond-by-microsecond processing of the ticks can mean that the individual bars between live trading and backtesting can have slightly different ticks. As a result, it's possible for a tick to be counted in different bars between backtesting and live trading, which can lead to bars having slightly different open, high, low, close, and volume values.

Delivery

Most live trading algorithms run on co-located servers racked in Equinix. Co-location reduces several factors that can interfere with your algorithm, including downtime from internet outages, equipment repairs, and natural disasters.

Live data takes time to travel from the source to your algorithm. The Forex data feed has a latency of 20-50 milliseconds. QuantConnect is not designed for high-frequency trading.

Pricing

The Forex data feed is free.

10.3.6 Futures

Introduction

The Futures data feed is a stream of contracts trades, quotes and open interest delivered to your trading algorithm during live execution. Live data feeds enable you to make real-time trades and update the value of the securities in your portfolio.

Sourcing

The Futures data feed consolidates market data across the following markets:

- CBOT
- CME
- COMEX
- ICE
- NYMEX

The data feed is powered by the Chicago Mercantile Exchange (CME).

The data feed doesn't include the CFE market. For `Futures.Indices.VIX` , use a combination of the QuantConnect and IB data feeds. For more details about this option, see Hybrid QuantConnect Data Feed .

Bar Building

We aggregate ticks to build bars.

In live trading, bars are built using the exchange timestamps with microsecond accuracy. This microsecond-by-microsecond processing of the ticks can mean that the individual bars between live trading and backtesting can have slightly different ticks. As a result, it's possible for a tick to be counted in different bars between backtesting and live trading, which can lead to bars having slightly different open, high, low, close, and volume values.

Delivery

Most live trading algorithms run on co-located servers racked in Equinix. Co-location reduces several factors that can interfere with your algorithm, including downtime from internet outages, equipment repairs, and natural disasters.

Live data takes time to travel from the source to your algorithm. The Futures data feed has a latency of 20-50 milliseconds. QuantConnect is not designed for high-frequency trading.

Pricing

The Futures data feed is free.

10.3.7 Future Options

Introduction

The Futures Options data feed is a stream of contracts trades, quotes and open interest delivered to your trading algorithm during live execution. Live data feeds enable you to make real-time trades and update the value of the securities in your portfolio.

Sourcing

The Future Options data feed consolidates market data across the following markets:

- CBOT
- CME
- COMEX
- NYMEX

Bar Building

We aggregate ticks to build bars.

In live trading, bars are built using the exchange timestamps with microsecond accuracy. This microsecond-by-microsecond processing of the ticks can mean that the individual bars between live trading and backtesting can have slightly different ticks. As a result, it's possible for a tick to be counted in different bars between backtesting and live trading, which can lead to bars having slightly different open, high, low, close, and volume values.

Delivery

Most live trading algorithms run on co-located servers racked in Equinix. Co-location reduces several factors that can interfere with your algorithm, including downtime from internet outages, equipment repairs, and natural disasters.

Live data takes time to travel from the source to your algorithm. The Future Options data feed has a latency of 20-50 milliseconds. QuantConnect is not designed for high-frequency trading.

Pricing

The Future Options data feed is free.

10.3.8 Alternative Data

Introduction

Alternative data feeds stream live alternative data into your algorithms. We add a live trading data feed for each of the alternative datasets we integrate.

Sourcing

We source alternative data feeds directly from data vendors . To view all of the integrated data vendors, see the Dataset Market .

Delivery

The delivery schedule of alternative data feeds depends on the specific data feed you're using. We inject the data into your algorithms when the vendor provides the data. For most alternative data feeds, the data is updated on a daily or hourly basis. Some data feeds, like the Tiingo News Feed or Benzinga News Feed , include a live stream. In these cases, we deliver the data as a live stream to your algorithm.

Most live trading algorithms run on co-located servers racked in Equinix. Co-location reduces several factors that can interfere with your algorithm, including downtime from internet outages, equipment repairs, and natural disasters.

Live data takes time to travel from the source to your algorithm. The latency of the alternative data feeds depends on the specific data feed you're using.

Pricing

Refer to the Dataset Market listings.

10.3.9 Brokerage Data Feeds

Brokerage data feeds are streams of live security prices that come directly from the respective brokerage. If you use a brokerage data feed and request historical data, the historical data comes from the brokerage. LEAN accepts all resolutions of data from brokerage data feeds.

Interactive Brokers

US Equities, Equity Options, Future Options, Index, and Index Options

Samco

Indian Equities

Tradier

US Equities and Equity Options

Zerodha

Indian Equities

See Also

Data Feeds
Datasets
Brokerages

10.3.9.1 Interactive Brokers

Introduction

Interactive Brokers (IB) was founded by Thomas Peterffy in 1993 with the goal to "create technology to provide liquidity on better terms. Compete on price, speed, size, diversity of global products and advanced trading tools". IB provides access to trading Equities, ETFs, Options, Futures, Future Options, Forex, Gold, Warrants, Bonds, and Mutual Funds for clients in over 200 countries and territories with no minimum deposit. IB also provides paper trading, a trading platform, and educational services.

The Interactive Brokers (IB) data feed streams live asset prices from IB. If you use this data feed and request historical data, the historical data comes from IB.

Sourcing

The IB data feed comes directly from IB. For more information about the data source, see the Trader Workstation API documentation.

If you use the IB data feed, IB only provides the security price data. We provide the following auxiliary datasets from the Dataset Market:

- US Equity Security Master
- US Futures Security Master
- Universe selection datasets
- Non-streaming alternative datasets

Universe Selection

Universe selection is available with the IB data feed.

```
AddUniverse(CoarseUniverseSelection, FineUniverseSelection);
```

```
self.AddUniverse(self.CoarseUniverseSelection, self.FineUniverseSelection)
```

The universe selection data comes from our Dataset Market, not the TWS market scanners. Universe selection with the IB data feed occurs around 6-7 AM Eastern Time (ET) on Tuesday to Friday and at 2 AM ET on Sunday. Universe selection data isn't available when the IB servers are closed. To check the IB server status, see the Current System Status page on the IB website.

The IB data feed can stream data for up to 100 assets by default, but IB may let you stream more than 100 assets based on your commissions and equity value. For more information about data feed limits from IB, see the Market Data Pricing Overview page on the IB website. If IB lets you stream more than 100 assets, set the `DataSubscriptionLimit` of your algorithm to the new limit from IB.

```
Settings.DataSubscriptionLimit = 150;
```

```
self.Settings.DataSubscriptionLimit = 150
```

Bar Building

The data feed is a summarized snapshot of the trades and quotes at roughly 300 milliseconds per snapshot.

Alternative Data

Brokerage data feeds support most alternative data feeds, except feeds that stream real-time intraday data. Streaming data feeds, like the Tiingo News Feed and Benzinga News Feed, require the QuantConnect data feed. The hybrid QuantConnect-IB data feed supports streaming data feeds.

Hybrid QuantConnect Data Feed

When you deploy a live algorithm with the IB brokerage, you can use the QC data feed, the IB data feed, or both. If you use both data feeds, Lean gives priority to the QuantConnect data feed. If our data feed doesn't have a stream for the securities you request, Lean uses the IB data feed. This process makes it possible to use our data feed for Equity universe selection and then place Options trades on the securities in the universe. If you use the QC data feed, the assets that you subscribe to don't contribute to the IB data feed limit.

Historical Data

If you get historical data from IB through a history request or a warm-up period, the historical data has the following characteristics:

- Second resolution data is limited to six months of history.
- The historical data excludes delisted Equities and expired Options.
- The historical data excludes expired Futures after two years.

The following quotas are in place for tick and second resolution historical data:

- You can have up to 50 simultaneous requests.
- You can make up to 60 requests within any 10-minute period.

In the preceding quotas, `TradeBar` and `QuoteBar` data count as separate requests. For example, if you request `TradeBar` and `QuoteBar` data for SPY, it counts as two requests.

For more information about historical data from IB, see Historical Data Limitations in the IB documentation.

Pricing

To use IB data feeds in your algorithms, subscribe to IB market data. We support all of the IB data subscriptions that are related to the securities and markets we support. Members usually subscribe to the following IB market data:

- US Securities Snapshot and Futures Value Bundle
- US Equity and Options Add-On Streaming Bundle
- CFE Enhanced Top of Book (L1 for VIX Futures)

To see the latest prices, check the Market Data Pricing Overview page on the IB website. IB can take up to 24 hours to process subscription requests. So after you subscribe to data, you need to wait 24 hours before you can use it in your algorithms. When you subscribe to data, IB only assigns your data subscription to one of your accounts. If you want to assign the subscription to a different account, for example, a paper trading account instead of a live trading account, then contact IB.

10.3.9.2 Samco

Introduction

Samco was founded by Jimeet Modi in2015 with a mission of providing retail investors access to sophisticated financial technology that can assist retail investors in creating wealth at a low cost. Samco provides access to India Equities for clients in India with no minimum balance. Samco also provides stock ratings, mutual funds, and a mini-portfolio investment platform.

The Samco data feed streams live asset prices from Samco. If you use this data feed and request historical data, the historical data comes from Samco.

Sourcing

The Samco data feed comes directly from Samco. For more information about the data source, see the StockNote API documentation .

If you use the Samco data feed, Samco only provides the security price data. We provide the following auxiliary datasets from the Dataset Market:

- India Equity Security Master
- Non-streaming alternative datasets

Universe Selection

Universe selection isn't available with the Samco data feed.

Bar Building

The data feed consolidates prices and quotes across all of the Indian exchanges. For a complete list of exchange and securities, see the ScripMaster file from the StockNote API documentation.

Alternative Data

Brokerage data feeds support most alternative data feeds, except feeds that stream real-time intraday data. Streaming data feeds, like the Tiingo News Feed and Benzinga News Feed , require the QuantConnect data feed.

Pricing

The Samco data feed is free. To access the data feed, you just need an active Samco account.

10.3.9.3 Tradier

Introduction

Tradier was founded by Dan Raju, Peter Laptewicz, Jason Barry, Jeyashree Chidambaram, and Steve Agalloco in 2012 with the goal to "deliver a choice of low-cost, high-value brokerage services to traders". Tradier provides access to trading Equities and Options for clients in over 250 countries and territories with no minimum deposit for cash accounts . Tradier also delivers custody, clearing, execution, and billing on behalf of registered advisors.

The Tradier data feeds are streams of Equity and Option prices directly from Tradier. If you use this data feed and request historical data, the historical data comes from Tradier. If you deploy to the demo environment, Tradier doesn't offer streaming market data due to exchange restrictions related to delayed data, so you must use our data feed.

Sourcing

The Tradier data feed comes directly from Tradier. For more information about the data source, see the Tradier API documentation .

If you use the Tradier data feed, Tradier only provides the security price data. We provide the following auxiliary datasets from the Dataset Market:

- US Equity Security Master
- Universe selection datasets
- Non-streaming alternative datasets

Universe Selection

Universe selection is available with the Tradier data feed.

```
AddUniverse(CoarseUniverseSelection, FineUniverseSelection);
```

```
self.AddUniverse(self.CoarseUniverseSelection, self.FineUniverseSelection)
```

Bar Building

The data feed is a stream of asset prices collected by WebSockets and distributed to algorithms on the platform.

Alternative Data

Brokerage data feeds support most alternative data feeds, except feeds that stream real-time intraday data. Streaming data feeds, like the Tiingo News Feed and Benzinga News Feed , require the QuantConnect data feed.

Pricing

The Tradier data feed is free for Tradier subscription accounts. If you have a free Tradier account, you may have to pay inactivity and maintenance fees. If you have less than $2,000 in total account value and less than 2 executed trades in 1 year, the inactivity fee is $50. If you have less than 2 executed trades per month, the international account monthly maintenance fee is $20. To view the latest prices, see the Pricing page on the Tradier website.

10.3.9.4 Zerodha

Introduction

Zerodha was founded by Nithin Kamath in 2010 with the goal to break all barriers that traders and investors face in India in terms of cost, support, and technology. Zerodha provides access to India Equities for clients in India with no minimum balance required. Zerodha also provides a mutual fund investment platform and an interactive portfolio dashboard.

The Zerodha data feed streams live asset prices from Zerodha. If you use this data feed and request historical data, the historical data comes from Zerodha.

Sourcing

The Zerodha data feed comes directly from Zerodha. For more information about the data source, see the Kite Connect API documentation .

If you use the Zerodha data feed, Zerodha only provides the security price data. We provide the following auxiliary datasets from the Dataset Market:

- India Equity Security Master
- Non-streaming alternative datasets

Universe Selection

Universe selection isn't available with the Zerodha data feed.

Bar Building

The data feed consolidates prices and quotes across all of the Indian exchanges.

Alternative Data

Brokerage data feeds support most alternative data feeds, except feeds that stream real-time intraday data. Streaming data feeds, like the Tiingo News Feed and Benzinga News Feed , require the QuantConnect data feed.

Pricing

The Zerodha data feed costs â‚¹2000/month for retail users. To view the latest prices, see the What are the charges for KITE APIs? page on the Zerodha website.

10.4 Deployment

Introduction

Deploy your trading algorithms live to receive real-time market data and submit orders on our co-located servers. As your algorithms run, you can view their performance in the Algorithm Lab. Since the algorithms run in QuantConnect Cloud, you can close the IDE without interrupting the execution of your algorithms. Deploying your algorithms to live trading through QuantConnect is cheaper than purchasing server space, setting up data feeds, and maintaining the software on your own. To deploy your algorithms on QuantConnect, you just need to follow the Deploy Live Algorithms section in the guide of your brokerage .

Resources

You need a live trading node for each algorithm that you deploy to our co-located servers. Several models of live trading nodes are available. More powerful live trading nodes allow you to run algorithms with larger universes and give you more time for machine learning training . The following table shows the specifications of the live trading node models:

Name	Number of Cores	Processing Speed (GHz)	RAM (GB)	GPU
L-MICRO	1	2.6	0.5	0
L1-1	1	2.6	1	0
L1-2	1	2.6	2	0
L2-4	2	2.6	4	0

Refer to the Pricing page to see the price of each live trading node model.

Node Quotas

You need a live trading node for each simultaneous algorithm that you deploy. We do not support sub algorithms or sharing a server with multiple algorithms. The tier of your organization determines the number of live trading nodes the organization can have. The following number of live trading nodes are available for each tier:

Tier	Node Quota
Free	0
Quant Researcher	2
Team	10
Trading Firm	Unlimited
Institution	Unlimited

To deploy multiple algorithms using a single brokerage, create sub-accounts in your brokerage account so that each algorithm has its own set of brokerage connection credentials.

Ram Allocations

Members often use 8-32GB of RAM in backtesting and are concerned that their algorithms will not work in live trading since live trading nodes have 512MB to 4GB of RAM. Backtesting nodes have more RAM because data is injected into your algorithm roughly 100,000x faster during backtests than live trading. You use more RAM in backtesting because many data objects are cached to achieve such fast speed. In live trading, 512MB to 4GB of RAM is sufficient for almost all use cases.

Wizard

Use the deployment wizard in the Algorithm Lab to deploy your algorithms to live trading . The deployment wizard lets you select a brokerage, enter your brokerage credentials, select a data feed, select a live trading node, set up notifications, and configure automatic algorithm restarts.

Most of the brokerages automatically load your cash holdings, position holdings, and submitted orders so that you can view your portfolio state on the live results page . For brokerages that don't automatically load your holdings, you can enter your cash and position holdings in the deployment wizard.

Unsupported Assets

If you have unsupported assets in your brokerage account when you deploy, Lean can't calculate the portfolio value correctly, so margin calculations are wrong. To avoid issues, if your account has unsupported assets, Lean automatically exits on deployment. For a list of supported assets, see the asset class dataset listing .

Automatic Restarts

Automatic restarts use best efforts to restart your algorithm if it fails due to a runtime error or an API disconnection. Automatic restarts reduce the risk of your algorithm missing a trade during periods of downtime. If you enable automatic restarts when you deploy your algorithm and your algorithm fails, your algorithm will try five times to restart. After five unsuccessful restarts, your algorithm won't attempt to restart again.

Security

Your code is stored in a database, isolated from the internet. When the code leaves the database, it is compiled and obfuscated before being deployed to the cloud. If the cloud servers were compromised, this process makes it difficult to read your strategy.

As we've seen over recent years, there can never be any guarantee of security with online websites. However, we deploy all modern and common security procedures. We deploy nightly software updates to keep the server up to date with the latest security patches. We also use SSH key login to avoid reliance on passwords. Internally, we use processes to ensure only a handful of people have access to the database and we always restrict logins to never use root credentials.

See our Security and IP documentation for more information.

10.5 Notifications

Introduction

Set up some live trading notifications so that you are notified of market events and your algorithm's performance. We support email, SMS, webhooks, and Telegram notifications. If you set up notifications in the deployment wizard, we will notify you when your algorithm places orders or emits insights. To be notified at other moments in your algorithm, create notifications in your code files with the `NotificationManager` . Lean ignores notifications during backtests. To view the number of notification you can send for free, see the Live Trading Notification Quotas .

Email

Email notifications can include up to 10KB of text content in the message body. These notifications can be slow since they go through your email provider. If you don't receive an email notification that you're expecting, check your junk folders.

Follow these steps to set up email notifications in the deployment wizard:

1. On the Deploy Live page, enable at least one of the notification types.

   The following table shows the supported notification types:

Notification Type	Description
Order Events	Notifications for when the algorithm receives `OrderEvent` objects
Insights	Notifications for when the algorithm emits `Insight` objects

2. Click Email .
3. Enter an email address.
4. Enter a subject.
5. Click Add .

   To add more email notifications, click Add Notification and then continue from step 2.

SMS

SMS notifications are the only type of notification that you don't need an internet connection to receive.

Follow these steps to set up SMS notifications in the deployment wizard:

1. On the Deploy Live page, enable at least one of the notification types.

   The following table shows the supported notification types:

Notification Type	Description
Order Events	Notifications for when the algorithm receives `OrderEvent` objects
Insights	Notifications for when the algorithm emits `Insight` objects

2. Click SMS .
3. Enter a phone number.
4. Click Add .

   To add more SMS notifications, click Add Notification and then continue from step 2.

Webhooks

Webhook notifications are an HTTP-POST request to a URL you provide. The request is sent with a timeout of 300s. You can process these notifications on your web server however you want. For instance, you can inject the content of the notifications into your server's database or use it to create other notifications on your own server.

Follow these steps to set up webhook notifications in the deployment wizard:

1. On the Deploy Live page, enable at least one of the notification types.

   The following table shows the supported notification types:

Notification Type	Description
Order Events	Notifications for when the algorithm receives `OrderEvent` objects
Insights	Notifications for when the algorithm emits `Insight` objects

2. Click Webhook .
3. Enter a URL.
4. If you want to add header information, click Add Header and then enter a key and value.

   Repeat this step to add multiple header keys and values.

5. Click Add .

   To add more webhook notifications, click Add Notification and then continue from step 2.

Telegram

Telegram notifications are automated messages to a Telegram group.

Follow these steps to set up Telegram notifications in the deployment wizard:

1. On the Deploy Live page, enable at least one of the notification types.

   The following table shows the supported notification types:

Notification Type	Description
Order Events	Notifications for when the algorithm receives `OrderEvent` objects
Insights	Notifications for when the algorithm emits `Insight` objects

2. Create a new Telegram group.
3. Add a bot to your Telegram group.

   To create a bot, chat with @BotFather and follow its instructions. If you want to use our bot, the username is @quantconnect_notifications_bot.

4. On the live deployment wizard, click Telegram .
5. Enter your user Id or group Id.

   Your group Id is in the URL when you open your group chat in the Telegram web interface. For example, the group Id of web.telegram.org/z/#-503016366 is -503016366.

6. If you are not using our notification bot, enter the token of your bot.
7. Click Add .

   To add more Telegram notifications, click Add Notification and then continue from step 2.

Terms of Use

The notification system can't be used for data distribution.

10.6 Results

Introduction

The live results page shows your algorithm's live trading performance. Review the results page to see how your algorithm has been performing and to investigate ways to improve it.

View Live Results

The live results page automatically displays when you deploy a live algorithm . The page presents the algorithm's equity curve, holdings, trades, logs, server statistics, and much more information.

The content in the live results page updates as your algorithm executes. You can close or refresh the window without interrupting the algorithm because the live trading node processes on our servers. If you close the page, you can view all of your live projects to open the page again.

Runtime Statistics

The banner at the top of the live results page displays the performance statistics of your algorithm.

The following table describes the default runtime statistics:

Statistic	Description
Equity	The total portfolio value if all of the holdings were sold at current market rates.
Fees	The total quantity of fees paid for all the transactions.
Holdings	The absolute sum of the items in the portfolio.
Net Profit	The dollar-value return across the entire trading period.
PSR	The probability that the estimated Sharpe ratio of an algorithm is greater than a benchmark (1).
Return	The rate of return across the entire trading period.
Unrealized	The amount of profit a portfolio would capture if it liquidated all open positions and paid the fees for transacting and crossing the spread.
Volume	The total value of assets traded for all of an algorithm's transactions.

To add a custom runtime statistic, call the `SetRuntimeStatistic` method with a `name` and `value` . The `value` argument can be a `string` or a number.

```
SetRuntimeStatistic(name, value);
```

```
self.SetRuntimeStatistic(name, value)
```

If you stop and redeploy a live algorithm, the runtime statistics are reset.

Built-in Charts

The live results page displays the equity curve of your algorithm so that you can analyze its performance in real-time.

The following table describes the series in the Strategy Equity chart:

Series	Description
Equity	The live equity curve of your algorithm.
Out of Sample Backtest	The backtest equity curve of your algorithm during the live trading period.
Meta	Points in time when you deployed your algorithm, stopped your algorithm, and when your algorithm encountered a runtime error.

Custom Charts

The results page shows the custom charts that you create.

Supported Chart Types

We support the following types of charts:

If you use `SeriesType.Candle` and plot enough values, the plot displays candlesticks. However, the `Plot` method only accepts one numerical value per time step, so you can't plot candles that represent the open, high, low, and close values of each bar in your algorithm. The charting software automatically groups the data points you provide to create the candlesticks, so you can't control the period of time that each candlestick represents.

To create other types of charts, save the plot data in the ObjectStore and then load it into the Research Environment. In the Research Environment, you can create other types of charts with third-party charting packages .

Supported Markers

When you create scatter plots, you can set a marker symbol. We support the following marker symbols:

Chart Quotas

Custom charts are limited to 4,000 data points. Intensive charting requires hundreds of megabytes of data, which is too much to stream online or display in a web browser. If you exceed the quota, the Cloud Terminal displays the following message:

Exceeded maximum points per chart, data skipped

You can create up to 10 custom chart series per algorithm. If you exceed the quota, your algorithm stops executing and the Cloud Terminal displays the following message:

Exceeded maximum series count: Each backtest can have up to 10 series in total.

Demonstration

For more information about creating custom charts, see Charting .

Adjust Charts

You can manipulate the charts displayed on the live results page.

Toggle Charts

To display and hide a chart on the live results page, in the Select Chart section, click the name of a chart.

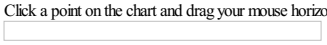
Toggle Chart Series

To display and hide a series on a chart on the live results page, click the name of a series at the top of a chart.

Adjust the Display Period

To zoom in and out of a time series chart on the live results page, perform either of the following actions:

- Click the 1m , 3m , 1y , or All period in the top-right corner of the chart.
- Click a point on the chart and drag your mouse horizontally to highlight a specific period of time in the chart.



If you adjust the zoom on a chart, it affects all of the charts.

After you zoom in on a chart, slide the horizontal bar at the bottom of the chart to adjust the time frame that displays.

Resize Charts

To resize a chart on the live results page, hover over the bottom-right corner of the chart. When the resize cursor appears, hold the left mouse button and then drag to the desired size.

Move Charts

To move a chart on the live results page, click, hold, and drag the chart title.

Refresh Charts

Refreshing the charts on the live results page resets the zoom level on all the charts. If you refresh the charts while your algorithm is executing, only the data that was seen by the Lean engine after you refreshed the charts is displayed. To refresh the charts, in the Select Chart section, click the reset icon.

Holdings

The Holdings tab on the live results page displays your positions and cash.

The following table describes the properties that display for each of your positions:

Property	Description
Symbol	The ticker of the security.
Average Price	The average price that you paid for the position.
Quantity	The size of your position.
Market Value	The value of your position if sold with market orders.
Unrealized	The unrealized profit of your position, including fees and spread costs.

The values in the positions section update as new data points are injected into your algorithm. The cash section displays the quantity of each currency in your algorithm's `CashBook` . View the Holdings tab to see your holdings, add security subscriptions , and place manual orders . To view all of your current holdings and active data subscriptions, enable the Show All Portfolio check box.

Orders

The live results page displays the orders of your algorithm and you can download them to your local machine.

View in the GUI

To see the orders that your algorithm created, open the live results page and then click the Orders tab. If there are more than 10 orders, use the pagination tools at the bottom of the Orders Summary table to see all of the orders. Click on an individual order in the Orders Summary table to reveal additional information regarding the following:

- Submissions
- Fills
- Partial fills
- Updates
- Cancellations
- Option contract exercises and expiration

The timestamps in the Order Summary table are based in Eastern Time (ET).

Download CSV

To download the orders in CSV format, open the live results page, click the Orders tab, and then click Download Orders . The content of the CSV file is the content displayed in the Orders Summary table when the table rows are

collapsed.

Insights

The live results page displays the insights of your algorithm and you can download them to your local machine.

View in the GUI

To see the insights your algorithm has emitted, open the live result page and then click the Insights tab. If there are more than 10 insights, use the pagination tools at the bottom of the Insights Summary table to see all of the insights. The timestamps in the Insights Summary table are based in Eastern Time (ET).

Download JSON

To download the insights in JSON format, open the live result page, click the Insights tab, and then click Download Insights . The timestamps in the CSV file are based in Coordinated Universal Time (UTC).

Logs

The Logs tab on the live results page displays all of the logging statements and status messages your algorithm creates. Their timestamps in the log file are in Coordinated Universal Time (UTC). The status messages include all of the points in time when your algorithm deployed, encountered an error, sent an order, or quit executing. It's good practice to add logs in live algorithms because then you can see what is happening while it executes. If you stop and redeploy your algorithm, the logs are retained. You can view the log file on the live results page or download them to your local machine.

View in the GUI

To see the log file your algorithm has created, open the live results page and then click the Logs tab.

Download Log File

To download the log file, open the live result page, click the Logs tab, and then click Download Logs .

Project Files

The live results page displays the project files used to deploy the algorithm. To view the files, click the Code tab. By default, the main.py or Main.cs file displays. To view other files in the project, click the file name and then select a different file from the drop-down menu.

View All Live Projects

The Your Strategies section of the Strategy Explorer page displays the status of all the live algorithms in your organizations. To view the page, log in to the Algorithm Lab and then, in the left navigation bar, click Strategy Explorer .

Errors

If your live algorithm throws a runtime error, it stops executing and we send you an email. If you enabled automatic restarts when you deployed your algorithm, your algorithm will try five times to restart.

10.7 Algorithm Control

Introduction

The algorithm control features on the live results page let you adjust your algorithm while it is executing live so that you can perform actions that are not written in the project files. The control features let you intervene in the execution of your algorithm and make adjustments. For instance, you can create security subscriptions, place trades, stop the algorithm, and update the algorithm.

Add Security Subscriptions

The live results page enables you to manually create security subscriptions for your algorithm instead of calling the `Add securityType` methods in your code files. If you add security subscriptions to your algorithm, you can place manual trades through the IDE without having to edit and redeploy the algorithm. Follow these steps to add security subscriptions:

1. Open your algorithm's live results page .
2. In the Holdings tab, click Add Security .
3. Enter the symbol, security type, resolution, leverage, and market of the security you want to add.
4. If you want the data for the security to be filled-forward, check the Fill Forward check box.
5. If you want to subscribe to extended market hours for the security, check the Extended Market Hours check box.
6. Click Add Security .

You can't manually remove security subscriptions from the IDE.

Place Manual Trades

The live results page lets you manually place orders instead of calling the automated methods in your project files. You can use any order type that is supported by the brokerage that you used when deploying the algorithm. To view the supported order types of your brokerage, see the Orders section of your brokerage model . Some example situations where it may be helpful to place manual orders instead of stopping and redeploying the algorithm include the following:

- Your brokerage account had holdings in it before you deployed your algorithm
- Your algorithm had bugs in it that caused it to purchase the wrong security
- You want to add a hedge to your portfolio without adjusting the algorithm code
- You want to rebalance your portfolio before the rebalance date

Note that it's not currently possible to cancel manual orders.

Follow these steps to place manual orders:

1. Open your algorithm's live results page .
2. In the Holdings tab, if the security you want to trade isn't listed, click Show All Portfolio .
3. If the security you want to trade still isn't listed, subscribe to the security .
4. Click the security you want to trade.
5. Click Create Order or Liquidate .
6. If you clicked Create Order , enter an order quantity.
7. Click the Type field and then click an order type from the drop-down menu.
8. Click Submit Order .

Liquidate Positions

The live results page has a Liquidate button that acts as a "kill switch" to sell all of your portfolio holdings. If your algorithm has a bug in it that caused it to purchase a lot of securities that you didn't want, this button let's you easily liquidate your portfolio instead of placing many manual trades. When you click the Liquidate button, if the market is open for an asset you hold, the algorithm liquidates it with market orders. If the market is not open, the algorithm places market on open orders. After the algorithm submits the liquidation orders, it stops executing.

Follow these steps to liquidate your positions:

1. Open your algorithm's live results page .
2. Click Liquidate .
3. Click Liquidate again.

Stop the Algorithm

The live trading results page has a Stop button to immediately stop your algorithm from executing. When you stop a live algorithm, your portfolio holdings are retained. Stop your algorithm if you want to perform any of the following actions:

- Update your project's code files
- Upgrade the live trading node
- Update the settings you entered into the deployment wizard
- Place manual orders through your brokerage account instead of the web IDE

Furthermore, if you receive new securities in your portfolio because of a reverse merger, you also need to stop and redeploy the algorithm.

LEAN actively terminates live algorithms when it detects interference outside of the algorithm's control to avoid conflicting race conditions between the owner of the account and the algorithm, so avoid manipulating your brokerage account and placing manual orders on your brokerage account while your algorithm is running. If you need to adjust your brokerage account holdings, stop the algorithm, manually place your trades, and then redeploy the algorithm.

Follow these steps to stop your algorithm:

1. Open your algorithm's live results page .
2. Click Stop .
3. Click Stop again.

Update the Algorithm

If you need to adjust your algorithm's project files or parameter values , stop your algorithm, make your changes, and then redeploy your algorithm. You can't adjust your algorithm's code or parameter values while your algorithm executes. When you stop and redeploy a live algorithm, your project's live equity curve is retained between the deployments. To erase the equity curve history, clone the project and then redeploy the cloned version of the project.

To update parameters in live mode, add a Schedule Event that downloads a remote file and uses its contents to update the parameter values.

```
private Dictionary _parameters = new();
public override void Initialize()
{
    if (LiveMode)
    {
        Schedule.On(
            DateRules.EveryDay(),
            TimeRules.Every(TimeSpan.FromMinutes(1)),
            ()=>
            {
                var content = Download(urlToRemoteFile);
                // Convert content to _parameters
            });
    }
}
```

```
def Initialize(self):
    self.parameters = { }
    if self.LiveMode:
        def download_parameters():
            content = self.Download(url_to_remote_file)
            # Convert content to self.parameters

        self.Schedule.On(self.DateRules.EveryDay(), self.TimeRules.Every(timedelta(minutes=1)), download_parameters)
```

10.8 Reconciliation

Introduction

Algorithms usually perform differently between backtesting and live trading over the same time period. Backtests are simulations where we model reality as close as possible, but the modeling isn't always perfect. To measure the performance differences, we run an out-of-sample (OSS) backtest in parallel to all of your live trading deployments. The live results page displays the live equity curve and the OOS backtest equity curve of your algorithms.

If your algorithm is perfectly reconciled, it has an exact overlap between its live and OOS backtest equity curves. Deviations mean that the performance of your algorithm has differed between the two execution modes. Several factors can contribute to the deviations.

Differences From Data

The data that your algorithm uses can cause differences between backtesting and live trading performance.

Look-Ahead Bias

The Time Frontier minimizes the risk of look-ahead bias in backtests, but it does not completely eliminate the risk of look-ahead bias. For instance, if you use a custom dataset that contains look-ahead bias, your algorithm's live and backtest equity curves may deviate. To avoid look-ahead bias with custom datasets, set a `Period` on your custom data points so that your algorithm receives the data points after the `Time + Period` .

Discrete Time Steps

In backtests, we inject data into your algorithm at predictable times, according to the data resolution. In live trading, we inject data into your algorithm when new data is available. Therefore, if your algorithm has a condition with a specific time (i.e. time is 9:30:15), the condition may work in backtests but it will always fail in live trading since live data has microsecond precision. To avoid issues, either use a time range in your condition (i.e. 9:30:10 < time < 9:30:20), use a rounded time, or use a Scheduled Event.

Custom Data Emission Times

Custom data is often timestamped to midnight, but the data point may not be available in reality until several days after that point. If your custom dataset is prone to this delay, your backtest may not fetch the same data at the same time or frequency that your live trading algorithm receives the data, leading to deviations between backtesting and live trading. To avoid issues, ensure the timestamps of your custom dataset are the times when the data points would be available in reality.

In backtesting, LEAN and custom data are perfectly synchonized. In live trading, daily and hourly data from a custom data source are not because of the frequency that LEAN checks the data source depends on the `resolution` argument. The following table shows the polling frequency of each resolution:

Resolution	Update Frequency
Daily	Every 30 minutes
Hour	Every 30 minutes
Minute	Every minute
Second	Every second
Tick	Constantly checks for new data

Split Adjustment of Indicators

Backtests use adjusted price data by default. Therefore, if you don't change the data normalization mode , the indicators in your backtests are updated with adjusted price data. In contrast, if a split or dividend occurs in live trading, your indicators will temporarily contain price data from before the corporate event and price data from after the corporate event. If this occurs, your indicators will produce different signals in your backtests compared to your live trading deployment. To avoid issues, reset and warm up your indicators when your algorithm receives a corporate event.

Tick Slice Sizes

In backtesting, we collect ticks into slices that span 1 millisecond before injecting them into your algorithm. In live trading, we collect ticks into slices that span up to 70 milliseconds before injecting them into your algorithm. This difference in slice sizes can cause deviations between your algorithm's live and OOS backtest equity curves. To avoid issues, ensure your strategy logic is compatible with both slice sizes.

Differences From Modeling

The modeling that your algorithm uses can cause differences between backtesting and live trading performance.

Reality Modeling Error

We provide brokerage models to model fees, slippage, and order fills in backtests. However, these model predictions may not always match the fees that your live algorithm incurs, leading to deviations between backtesting and live trading. You can adjust the reality models that your algorithm uses to more accurately reflect the specific assets that you're trading. For more information about reality models, see Reality Modeling .

Market Impact

We don't currently model market impact. So, if you are trading large orders, your fill prices can be better during backtesting than live trading, causing deviations between backtesting and live trading. To avoid issues, implement a custom fill model in your backtests that incorporates market impact.

Fills

In backtests, orders fill immediately. In live trading, they are sent to your brokerage and take about half a second to execute. If you fill an order in a backtest with stale data, deviations between backtesting and live trading can occur because the order is filled at a price that is likely different from the real market price. Stale order fills commonly occur in backtests when you create a Scheduled Event with an incompatible data resolution. For instance, if you subscribe to hourly data, place a Scheduled Event for 11:15 AM, and fill an order during the Scheduled Event, the order will fill at a stale price because the data between 11:00 AM and 11:15 AM is missing. To avoid stale fills, only place orders when your algorithm receives price data.

In live trading, your brokerage provides the fill price of your orders. Since the backtesting brokerage models do not know the price at which live orders are filled, the fill price of backtest orders is based on the best price available in the current backtesting data. Similarly, limit orders can fill at different prices between backtesting and live trading. In backtesting, limit orders fill as soon as the limit price is hit. In live trading, your brokerage may fill the same limit order at a different price or fail to fill the order, depending on the position of your order in their order book.

Borrowing Costs

We do not currently simulate the cost of borrowing shorts in backtests. Therefore, if your algorithm takes short positions, deviations can occur between backtesting and live trading. We are working on adding the functionality to model borrowing fees. Subscribe to GitHub Issue #4563 to track the feature progress.

Differences From Brokerage

The brokerage that your algorithm uses can cause differences between backtesting and live trading performance.

Portfolio Allocations on Small Accounts

If you trade a small portfolio, it's difficult to achieve accurate portfolio allocations because shares are usually sold in whole numbers. For instance, you likely can't allocate exactly 10% of your portfolio to a security. You can use fractional shares to achieve accurate portfolio allocations, but not all brokerages support fractional shares. To get the closest results when backtesting and live trading over the same period, ensure both algorithms have the same starting cash balance.

Different Backtest Parameters

If you don't start your backtest and live deployment on the same date with the same holdings, deviations can occur between backtesting and live trading. To avoid issues, ensure your backtest parameters are the same as your live deployment.

Non-deterministic State From Algorithm Restarts

If you stop and redeploy your live trading algorithm, it needs to restart in a stateful way or else deviations can occur between backtesting and live trading. To avoid issues, redeploy your algorithm in a stateful way using the `SetWarmUp` and `History` methods. Furthermore, use the `ObjectStore` to save state information between your live trading deployments.

Existing Portfolio Securities

If you deploy your algorithm to live trading with a brokerage account that has existing holdings, your live trading equity curve reflects your existing positions, but the backtesting curve won't. Therefore, if you have existing positions in your brokerage account when you deploy your algorithm to live trading, deviations will occur between backtesting and live trading. To avoid issues, deploy your algorithm to live trading using a separate brokerage account or subaccount that does not have existing positions.

Brokerage Limitations

We provide brokerage models that support specific order types and model your buying power. In backtesting, we simulate your ordersÂ with the brokerage model you select. In live trading, we send your orders to your brokerage for execution. If the brokerage model that you use in backtesting is not the same brokerage that you use in live trading, deviations may occur between backtesting and live trading. The deviations can occur if your live brokerage doesn't support the order types that you use or if the backtesting brokerage models your buying power with a different methodology than the real brokerage. To avoid brokerage model issues, set the brokerage model in your backtest to the same brokerage that you use in live trading.

10.9 Risks

Introduction

There are risks associated with deploying your algorithms to live trading. Strategy, portfolio, market, counterparty, operational, and error risks can cause you to lose capital. Some of these risks can be out of your control, but there are ways that you can mitigate them.

Strategy

Strategy risk is the risk that results from designing a strategy based on a statistical model. If you ignore the underlying assumptions of the statistical model, you are exposed to strategy risk. Even if you test that the model assumptions are held, if the market environment changes, the new environment may violate the underlying assumptions of the model after you have deployed it to live trading. Additionally, your strategy development process may be prone to overfitting, survivorship bias, or look-ahead bias , which increases your exposure to strategy risk. To address strategy risk, use rolling parameters when training your statistical models and perform the required statistical tests before training such models.

Portfolio

Portfolio risk is the risk associated with your portfolio as a whole. For instance, you're exposed to portfolio risk if you allocate too much of your portfolio to a particular factor, the capacity of your trading strategies reduces, or the correlation of the strategies in your portfolio increases. To address portfolio risk, diversify your portfolio among multiple factors, monitor the rolling capacity of your trading strategies, and frequently check the correlation of your trading strategies.

Market

Market risk, also known as systematic risk, is the risk that the value of your portfolio will decrease due to the value of the entire market decreasing. Market risk is caused by changes in interest rates, changes in currency exchange rates, geopolitical events, natural disasters, wars, terrorist attacks, and economic recessions. Additionally, central bank announcements and changes to monetary policy can increase overall market volatility and market risk. To address market risk, you can increase diversification, reduce your portfolio beta , hedge your positions with put Options, or hedge against volatility with volatility index securities.

Counterparty

Counterparty risk is the risk that a counterparty with which you engage won't pay an obligation that they have made with you. Most commonly, counterparty risk is associated with the risk that your brokerage goes out of business without returning the trading capital that you have in your brokerage account. Brokerages can go bankrupt just like any other business. To address counterparty risk, diversify your portfolio across multiple brokers that have a strong reputation. If you allocate your capital across multiple brokers and one of them goes out of business, you won't lose all of your trading capital.

Operational

Operational risks are the risks within your fund that relate to business operations, such as business risks, regulatory risks, trading infrastructure risks, and the risks of employees committing fraud or quitting. Operational risks are a result of the nature of a trading business, having employees, and regulatory changes. To address operational risks, stay up to date on potential regulatory changes, only hire employees that have signed contracts that protect your firm, use open-source trading infrastructure that's maintained by experts (Lean), and use co-located servers so that you don't need to tend to hardware failures and internet outages.

Error

Error risk is the risk associated with errors occurring in your strategy logic or trading infrastructure. Error risks occur because bugs naturally arise in trading algorithms and the underlying engine that the algorithms use to execute. The Lean trading engine has been under constant development for over 10 years, but there are always more improvements that can be implemented. To address error risk, backtest your trading algorithm before deploying it live to test if it has coding errors, stay up to date on the Lean GitHub Issues , and have close access to your email at all times. If your trading algorithm fails, we notify you through email. You can also enable automatic restarts when you deploy algorithms.

11 Optimization

Parameter optimization is the process of finding the optimal algorithm parameters to maximize or minimize an objective function. For instance, you can optimize your indicator parameters to maximize the Sharpe ratio that your algorithm achieves over a backtest. Optimization can help you adjust your strategy to achieve better backtesting performance, but be wary of overfitting. If you select parameter values that model the past too closely, your algorithm may not be robust enough to perform well using out-of-sample data.

Getting Started

Learn the basics

Parameters

Variables being optimized

Objectives

Target metric of performance

Strategies

How parameters are adjusted

Deployment

Run optimization jobs

Results

Your performance dashboard

See Also

Running Optimizations
Reviewing Results

11.1 Getting Started

Introduction

Parameter optimization is the process of finding the optimal algorithm parameters to maximize or minimize an objective function. For instance, you can optimize your indicator parameters to maximize the Sharpe ratio that your algorithm achieves over a backtest. Optimization can help you adjust your strategy to achieve better backtesting performance, but be wary of overfitting. If you select parameter values that model the past too closely, your algorithm may not be robust enough to perform well using out-of-sample data.

Launch Optimization Jobs

The following video demonstrates how to launch an optimization job:

You need the following to optimize parameters:

- At least one algorithm parameter in your project.
- The `GetParameter` method or `Parameter` attribute in your project.
- A successful backtest of the project.
- QuantConnect Credit (QCC) in your organization.

Follow these steps to optimize parameters:

1. Open the project that contains the parameters you want to optimize.
2. In the top-right corner of the IDE, click the Optimize icon.
3. On the Optimization page, in the Parameter & Constraints section, enter the name of the parameter to optimize.

   The parameter name must match a parameter name in the Project panel.

4. Enter the minimum and maximum parameter values.
5. Click the gear icon next to the parameter and then enter a step size.
6. If you want to add a second parameter to optimize, click Add Parameter .

   You can optimize a maximum of two parameters. To optimize more parameters, run local optimizations .

7. If you want to add optimization constraints, follow these steps:
   1. Click Add Constraint .
   2. Click the target field and then select a target from the drop-down menu.
   3. Click the operation field and then an operation from the drop-down menu.
   4. Enter a constraint value.
8. In the Estimated Number and Cost of Backtests , click an optimization node .
9. Select a maximum number of nodes to use.
10. In the Estimated Number and Cost of Backtests , click the Choose Optimization Strategy field and then select a strategy from the drop-down menu.
11. Click the Select Target field and then select a target from the drop-down menu.

    The target (also known as objective) is the performance metric the optimizer uses to compare the backtest performance of different parameter values.

12. Click Maximize or Minimize to maximize or minimize the optimization target, respectively.
13. Click Launch Optimization .

    The optimization results page displays. As the optimization job runs, you can close or refresh the window without interrupting the job because the nodes are processing on our servers.

    To abort a running optimization job, in the Status panel, click Abort and then click Yes .

View Individual Backtest Results

The optimization results page displays a Backtests table that includes all of the backtests that ran during the optimization job. The table lists the parameter values of the backtests in the optimization job and their resulting values for the objectives.

Open the Backtest Results Page

To open the backtest result page of one of the backtests in the optimization job, click a backtest in the table.

Download the Table

To download the table, right-click one of the rows, and then click Export > CSV Export .

Filter the Table

Follow these steps to apply filters to the Backtests table:

1. On the right edge of the Backtests table, click Filters .
2. Click the name of the column to which you want the filter to be applied.
3. If the column you selected is numerical, click the operation field and then select one of the operations from the drop-down menu.
4. Fill the fields below the operation you selected.

Toggle Table Columns

Follow these steps to hide and show columns in the Backtests table:

1. On the right edge of the Backtests table, click Columns .
2. Select the columns you want to include in the Backtests table and deselect the columns you want to exclude.

Sort the Table Columns

In the Backtests table, click one of the column names to sort the table by that column.

View All Optimizations

Follow these steps to view all of the optimization results of a project:

1. Open the project that contains the optimization results you want to view.
2. At the top of the IDE, click the Results icon.

   A table containing all of the backtest and optimization results for the project is displayed. If there is a play icon to the left of the name, it's a backtest result . If there is a fast-forward icon next to the name, it's an optimization result .

3. (Optional) In the top-right corner, select the Show field and then select one of the options from the drop-down menu to filter the table by backtest or optimization results.
4. (Optional) In the bottom-right corner, click the Hide Error check box to remove backtest and optimization results from the table that had a runtime error.
5. (Optional) Use the pagination tools at the bottom to change the page.
6. (Optional) Click a column name to sort the table by that column.
7. Click a row in the table to open the results page of that backtest or optimization.

Rename Optimizations

We give an arbitrary name (for example, "Smooth Apricot Chicken") to your optimization result files, but you can follow these steps to rename them:

1. Hover over the optimization you want to rename and then click the pencil icon that appears.
2. Enter the new name and then press Enter .

Delete Optimizations

Hover over the optimization you want to delete and then click the trash can icon that appears to delete the optimization result.

11.2 Parameters

Introduction

Parameters are project variables that your algorithm uses to define the value of internal variables like indicator arguments or the length of lookback windows.

Parameters are stored outside of your algorithm code, but we inject the values of the parameters into your algorithm when you launch an optimization job . The optimizer adjusts the value of your project parameters across a range and step size that you define to minimize or maximize an objective function. To optimize some parameters, add some parameters to your project and add the `GetParameter` method to your code files.

Set Parameters

Algorithm parameters are hard-coded values for variables in your project that are set outside of the code files. Add parameters to your projects to remove hard-coded values from your code files and to perform parameter optimizations. You can add parameters, set default parameter values, and remove parameters from your projects.

Add Parameters

Follow these steps to add an algorithm parameter to a project:

1. Open the project .
2. In the Project panel, click Add New Parameter .
3. Enter the parameter name.

   The parameter name must be unique in the project.

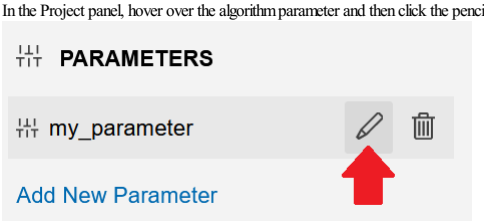
4. Enter the default value.
5. Click Create Parameter .

To get the parameter values into your algorithm, see Get Parameters .

Set Default Parameter Values

Follow these steps to set the default value of an algorithm parameter in a project:

1. Open the project .
2. In the Project panel, hover over the algorithm parameter and then click the pencil icon that appears.



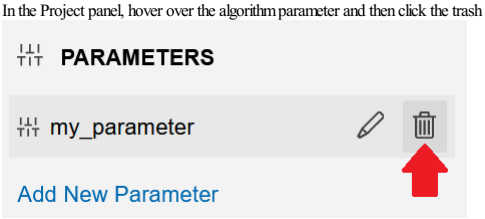
3. Enter a default value for the parameter and then click Save .

   The Project panel displays the default parameter value next to the parameter name.

Delete Parameters

Follow these steps to delete an algorithm parameter in a project:

1. Open the project .
2. In the Project panel, hover over the algorithm parameter and then click the trash can icon that appears.



3. Remove the `GetParameter` calls that were associated with the parameter from your code files.

Get Parameters

To get the parameter values from the Project panel into your algorithm, see Get Parameters .

Number of Parameters

The cloud optimizer can optimize up to 2 parameters. There are several reasons for this quota. First, the optimizer only supports the grid search strategy , which is very inefficient. This strategy tests every permutation of parameter values, so the number of backtests that the optimization job must run explodes as you add more parameters. Second, the parameter charts that display the optimization results are limited to two dimensions. Third, if you optimize with many variables, it increases the likelihood of overfitting to historical data.

We plan to upgrade the parameter charts on the optimization results page to support 3D surface charts. When we upgrade the visualization technology and add more efficient optimization strategies, we will increase the number of parameters you can optimize in the cloud. To optimize more parameters now, run local optimizations with the CLI .

11.3 Objectives

Introduction

An optimization objective is the performance metric that's used to compare the backtest performance of different parameter values. The optimizer currently supports the compound annual growth rate (CAGR), drawdown, Sharpe ratio, and Probabilistic Sharpe ratio (PSR) as optimization objectives. When the optimization job finishes, the results page displays the value of the objective with respect to the parameter values.

CAGR

CAGR is the yearly return that would be required to generate the return over the backtest period. CAGR is calculated as

$$ (\frac{e}{s})^{\frac{1}{y}}-1 $$

where s is starting equity, e is ending equity, and y is the number of years in the backtest period. The benefit of using CAGR as the objective is that it maximizes the return of your algorithm over the entire backtest. The drawback of using CAGR is that it may cause your algorithm to have more volatile returns, which increases the difficulty of keeping your algorithm deployed in live mode.

Drawdown

Drawdown is the largest peak to trough decline in your algorithm's equity curve. Drawdown is calculated as

$$ 1- \frac{v^{t\ge s}_{min}}{v^s_{max}} $$

where v^s_{max} is the maximum equity value up to time s and $v^{t\ge s}_{min}$ is the minimum equity value at time t where $t\ge s$. The following image illustrates how the max drawdown is calculated:

During the first highlighted period in the preceding image, the equity curve dropped from 106,027 to 93,949 (11.4%). During the second highlighted period, the equity curve dropped from 112,848 to 99,576 (11.8%). Since 11.8% > 11.4%, the max drawdown of the equity curve is 11.8%.

The benefit of using drawdown as the objective is that it's psychologically easier to keep an algorithm deployed in live mode if the algorithm doesn't experience large drawdowns. The drawback of using drawdown is that it may limit the potential returns of your algorithm.

Sharpe

The Sharpe ratio measures the excess returns relative to a benchmark, divided by the standard deviation of those returns. The Sharpe ratio is calculated as

$$ \frac{E[R_p - R_b]}{\sigma_p} $$

where R_p is the returns of your portfolio, R_b is the returns of the benchmark, and σ_p is the standard deviation of your algorithm's excess returns. By default, Lean uses a 0% risk-free rate, so $R_b=0$. The benefit of using the Sharpe ratio as the objective is that it maximizes returns while minimizing the return volatility. It's usually psychologically easier to keep a live algorithm deployed if it has minimal swings in equity than if it has large swings in equity. The drawback of using the Sharpe ratio is that it may limit your potential returns in favor of a less volatile equity curve.

PSR

The PSR is the probability that the estimated Sharpe ratio of your algorithm is greater than the Sharpe ratio of the benchmark. PSR is calculated as

\[P\left(\hat{SR} > SR^{\ast}\right) = CDF\left(\frac{(\hat{SR} - SR^{\ast})\sqrt{n-1}}{\sqrt{1 - \hat{\gamma}_{3}\hat{SR} + \frac{\hat{\gamma}_{4}-1}{4}\hat{SR}^{2}}}\right) \]

where $SR^{\ast}$ is the Sharpe ratio of the benchmark, $\hat{SR}$ is the Sharpe ratio of your algorithm, n is the number of trading days, $\hat{\gamma}_{3}$ is the skewness of your algorithm's returns, $\hat{\gamma}_{4}$ is the kurtosis of your algorithm's returns, and CDF is the normal cumulative distribution function. The benefit of using the PSR as the objective is that it maximizes the probability of your algorithm's Sharpe ratio outperforming the benchmark Sharpe ratio. The drawback of using the PSR is that, like the Sharpe ratio objective, optimizing the PSR may limit your potential returns in favor of a less volatile equity curve.

Constraints

Constraints filter out backtests from your optimization results that do not conform to your desired range of statistical results. Constraints consist of a target, operator, and a numerical value. For instance, you can add a constraint that the optimization backtests must have a Sharpe ratio >= 1 to be included in the optimization results. Constraints are optional, but you can use them to incorporate multiple objective functions into a single optimization job.

11.4 Strategies

Introduction

Optimization strategies control how the optimizer adjusts parameters for each new backtest that's run in the optimization job. Grid search is the only strategy currently available, but you can contribute new optimization strategies.

Grid Search

Grid search is the most complete but the most expensive strategy because it takes a brute force approach and tests all the combinations of parameter values. If you are optimizing one parameter, the grid search strategy selects the values of the parameters based on the starting value, ending value, and step size that you provide. If you optimize two parameters, the grid search strategy searches the Cartesian product of possible values for each parameter. The following animation shows the process of using grid search to optimize two parameters:

In the preceding animation, grid search tests all of the parameter combinations. The axes represent the possible values of each parameter. Gray squares represent backtests in the optimization queue, orange squares represent successful backtests, and black squares represent failed backtests. In this example, several squares are colored at the same time because the optimization job is using multiple optimization nodes .

Why Backtests Fail

When backtests fail during optimization, it is usually because the selected parameter values cause a divide-by-zero error or an index out-of-range exception.

Strategy Benefit and Drawback

The benefit of the grid search strategy is that it is the most comprehensive optimization strategy. The drawback of the strategy is it can be an expensive option because of the curse of dimensionality.

Contribute Strategies

You can contribute any optimization strategy that is popular in the literature and is not already implemented. To view the optimization strategies that are already implemented, see our GitHub repository . If you contribute a strategy, you'll receive some QuantConnect Credit , you'll be shown as a contributor to Lean on your GitHub profile, and your work will be used in the Algorithm Lab by our community of over 235,000 quants.

To contribute optimization strategies, submit a pull request to the Lean GitHub repository . In your pull request, provide an explanation of the strategy and some relevant resources so that we can add the strategy to our documentation. For an example implementation, see the GridSearchOptimizationStrategy .

11.5 Deployment

Introduction

Deploy optimization jobs for your trading algorithms to optimize your algorithm parameters for the objective that you specify. The optimizer runs concurrent backtests to optimize your algorithm parameter using up to 24 nodes. As the optimization runs, the results are displayed and updated in real-time.

Resources

The optimization nodes that backtest your algorithm are not the backtesting nodes in your organization. The optimization nodes are a cluster of nodes that exclusively run optimization jobs. The optimization can concurrently run multiple backtests if you use multiple nodes, but the maximum number of nodes you can use depends on the node type. The following table describes the node types:

Type	Description	Number of Cores	RAM (GB)	Max Cluster Size
O2-8	Relatively simple strategies with less than 100 assets	2	8	10
O4-12	Strategies with less than 500 assets and simple universe selections	4	12	6
O8-16	Complex strategies and machine learning	8	16	4

The following table shows the training quotas of the optimization node types:

Type	Capacity (min)	Refill Rate (min/day)
O2-8	30	5
O4-12	60	10
O8-16	90	15

Cost

You can rent optimization nodes on a time basis. The deployment wizard estimates the total cost of your optimization job based on the results of the last successful backtest of your algorithm, the number of parameters , and the optimization strategy . Therefore, you must run a backtest of your algorithm before the deployment wizard can estimate the cost of the optimization job. The final cost that you pay can vary from the estimate. For instance, if your backtest used parameters that were favorable for speedy execution, the estimate can be lower than the final cost.

You can use multiple nodes to speed up the optimization job without the job costing more because you use each node for a shorter period of time. However, there is a spin-up time of roughly 15 seconds on each backtest, so it can sometimes cost more to use many nodes when you factor in the spin-up time. You pay for optimizations with your organization's QuantConnect Credit balance. If you have your own hardware, you can run local optimizations with your own data and hardware.

Launch Optimization Jobs

You need the following to optimize parameters:

- At least one algorithm parameter in your project.
- The `GetParameter` method or `Parameter` attribute in your project.
- A successful backtest of the project.
- QuantConnect Credit (QCC) in your organization.

Follow these steps to optimize parameters:

1. Open the project that contains the parameters you want to optimize.
2. In the top-right corner of the IDE, click the Optimize icon.
3. On the Optimization page, in the Parameter & Constraints section, enter the name of the parameter to optimize.

   The parameter name must match a parameter name in the Project panel.

4. Enter the minimum and maximum parameter values.
5. Click the gear icon next to the parameter and then enter a step size.
6. If you want to add a second parameter to optimize, click Add Parameter .

   You can optimize a maximum of two parameters. To optimize more parameters, run local optimizations .

7. If you want to add optimization constraints, follow these steps:
   1. Click Add Constraint .
   2. Click the target field and then select a target from the drop-down menu.
   3. Click the operation field and then an operation from the drop-down menu.
   4. Enter a constraint value.
8. In the Estimated Number and Cost of Backtests , click an optimization node .
9. Select a maximum number of nodes to use.
10. In the Estimated Number and Cost of Backtests , click the Choose Optimization Strategy field and then select a strategy from the drop-down menu.
11. Click the Select Target field and then select a target from the drop-down menu.

    The target (also known as objective) is the performance metric the optimizer uses to compare the backtest performance of different parameter values.

12. Click Maximize or Minimize to maximize or minimize the optimization target, respectively.
13. Click Launch Optimization .

    The optimization results page displays. As the optimization job runs, you can close or refresh the window without interrupting the job because the nodes are processing on our servers.

    To abort a running optimization job, in the Status panel, click Abort and then click Yes .

11.6 Results

Introduction

The optimization results page shows your algorithm's performance with the various parameter values. Review the results page to see how your algorithm has performed during the backtests and to investigate how you might improve your algorithm before live trading.

View Optimization Results

The optimization results page automatically displays when you launch an optimization job . The page presents the algorithm's equity curves, parameters, target values, server statistics, and much more information.

The content in the optimization results page updates as your optimization job executes. You can close or refresh the window without interrupting the job because the optimization nodes process on our servers. If you close the page, you can view all of the project's optimizations to open the page again.

Runtime Statistics

The banner at the top of the optimization results page displays the performance statistics of the optimization job.

The banner updates in real-time as the optimization job progresses on our servers. The following table describes the runtime statistics:

Statistic	Description
Completed	The number of backtests that have successfully completed
Failed	The number of backtests that have failed during execution
Running	The number of backtests that are currently running
In Queue	The number of backtests that are waiting to start
Average Length	The average amount of time to complete one of the backtests
Total Runtime	The total runtime of the optimization job
Total	The total number of backtests run in the optimization job
Consumed	The amount of QuantConnect Credit that was used to perform the optimization

Equity Curves

The optimization results page displays a Strategy Equities chart so that you can analyze the equity curves of the individual backtests in the optimization job.

The equity curves of the backtests update in real-time as the optimization job runs. View the Strategy Equities chart to see how the parameter values affect the equity of your algorithm, to see how sensitive the returns are to the range of parameters selected by the optimizer, and to take a closer look at specific times in the backtest history.

Parameter Charts

The optimization results page displays parameter charts to show the relationship between the parameter value(s) selected by the optimizer and the value of several objectives. If your optimization job has one parameter, the result page displays a scatter plot for each objective. If your optimization job has two parameters, the result page displays a heat map for each objective.

Parameter Stability

Zones in the heatmap where the color of adjacent cells are relatively consistent represent areas where the objectives are stable. In these areas, the value of the objectives is not significantly influenced by the parameter values. The following image shows the parameter chart of an optimization job. The highlighted area identifies combinations of parameter values that stabilize the objective function.

Supported Objectives

You can add parameter charts for the following objectives:

- Alpha
- Annual Standard Deviation
- Annual Variance
- Average Loss
- Average Win
- Beta
- Compounding Annual Return
- Drawdown
- Estimated Strategy Capacity
- Expectancy
- Information Ratio
- Loss Rate
- Net Profit
- Probabilistic Sharpe Ratio (PSR)
- Profit-Loss Ratio
- Sharpe Ratio
- Total Fees
- Total Trades
- Tracking Error
- Treynor Ratio
- Win Rate

Add Parameter Charts

Follow these steps to add a parameter chart to the optimization results page:

1. In the Parameter Chart panel, click the plus icon.
2. Click the Objective field and then select an objective from the drop-down menu.
3. Click the Parameter 1 field and then select a parameter from the drop-down menu.
4. If there are two parameters in the optimization, click the Parameter 2 field and then select a parameter from the drop-down menu.
5. Click Create Chart .

   The optimization results page displays the new chart.

Individual Backtest Results

The optimization results page displays a Backtests table that includes all of the backtests that ran during the optimization job. The table lists the parameter values of the backtests in the optimization job and their resulting values for the objectives.

Open the Backtest Results Page

To open the backtest result page of one of the backtests in the optimization job, click a backtest in the table.

Download the Table

To download the table, right-click one of the rows, and then click Export > CSV Export .

Filter the Table

Follow these steps to apply filters to the Backtests table:

1. On the right edge of the Backtests table, click Filters .
2. Click the name of the column to which you want the filter to be applied.
3. If the column you selected is numerical, click the operation field and then select one of the operations from the drop-down menu.
4. Fill the fields below the operation you selected.

Toggle Table Columns

Follow these steps to hide and show columns in the Backtests table:

1. On the right edge of the Backtests table, click Columns .
2. Select the columns you want to include in the Backtests table and deselect the columns you want to exclude.

Sort the Table Columns

In the Backtests table, click one of the column names to sort the table by that column.

Server Stats

The optimization results page displays a Server Statistics section to show the status of the nodes running the optimization job.

The following image shows an example of the Server Statistics section:

The following table describes the information that the Server Statistics section displays:

Property	Description
CPU	The total CPU usage and the CPU usage of each node
RAM	The total RAM usage of the RAM usage of each node
HOST	The node model and the number of nodes used to run the optimization
Uptime	The length of time that the optimization job has ran

View the Server Statistics section to see the amount of CPU power and RAM the optimization job demands. If your algorithm is demanding a lot of resources, use more powerful nodes on the next optimization job or improve the efficiency of your algorithm.

Errors

The following table describes common optimization errors:

Error	Description
Runtime Errors	If a backtest in your optimization job throws a runtime error, the backtest will not complete but you will still be charged.
Data Overload	If a backtest in your optimization job produces more than 700MB of data, then Lean can't upload the results and the optimization job appears to never be complete.

View All Optimizations

Follow these steps to view all of the optimization results of a project:

1. Open the project that contains the optimization results you want to view.
2. At the top of the IDE, click the Results icon.

   A table containing all of the backtest and optimization results for the project is displayed. If there is a play icon to the left of the name, it's a backtest result . If there is a fast-forward icon next to the name, it's an optimization result .

3. (Optional) In the top-right corner, select the Show field and then select one of the options from the drop-down menu to filter the table by backtest or optimization results.
4. (Optional) In the bottom-right corner, click the Hide Error check box to remove backtest and optimization results from the table that had a runtime error.
5. (Optional) Use the pagination tools at the bottom to change the page.
6. (Optional) Click a column name to sort the table by that column.
7. Click a row in the table to open the results page of that backtest or optimization.

Rename Optimizations

We give an arbitrary name (for example, "Smooth Apricot Chicken") to your optimization result files, but you can follow these steps to rename them:

1. Hover over the optimization you want to rename and then click the pencil icon that appears.
2. Enter the new name and then press Enter .

Delete Optimizations

Hover over the optimization you want to delete and then click the trash can icon that appears to delete the optimization result.

12 Data Storage

Introduction

The ObjectStore is an organization-specific key-value storage location to save and retrieve data in QuantConnect's cache. Similar to a dictionary or hash table, a key-value store is a storage system that saves and retrieves objects by using keys. A key is a unique string that is associated with a single record in the key-value store and a value is an object being stored. Some common use cases of the ObjectStore include the following:

- Transporting data between the backtesting environment and the research environment.
- Training machine learning models in the research environment before deploying them to live trading.

The ObjectStore is shared across the entire organization. Using the same key, you can access data across all projects in an organization.

Supported Types

The ObjectStore has helper methods to store strings, JSON objects, XML objects, and bytes.

The ObjectStore has helper methods to store strings and bytes.

```
ObjectStore.Save(stringKey, stringValue);
ObjectStore.SaveJson<T>(jsonKey, jsonValue);
ObjectStore.SaveXml<T>(xmlKey, xmlValue);
ObjectStore.SaveBytes(bytesKey, bytesValue);
```

```
self.ObjectStore.Save(string_key, string_value)
self.ObjectStore.SaveBytes(bytes_key, bytes_value)
```

To store an object that is in a different format, you need to encode it to one of the supported data types. For instance, if you train a machine learning model and it is in binary format, encode it into base 64 before saving it.

The ObjectStore also has helper methods to retrieve the stored objects.

```
var stringValue = ObjectStore.Read(stringKey);
var jsonValue = ObjectStore.SaveJson<T>(jsonKey);
var xmlValue = ObjectStore.SaveXml<T>(xmlKey);
var bytesValue = ObjectStore.SaveBytes(bytesKey);
```

```
string_value = self.ObjectStore.Read(string_key)
bytes_value = self.ObjectStore.ReadBytes(bytes_key)
```

For complete examples of using the ObjectStore, see Object Store .

Storage Sizes

All organizations get 50 MB of free storage in the ObjectStore. Paid organizations can subscribe to more storage space. The following table shows the cost of the supported storage sizes:

Storage Size (GB)	Monthly Cost ($)
0.05	0
2	10
5	20
10	50
50	100

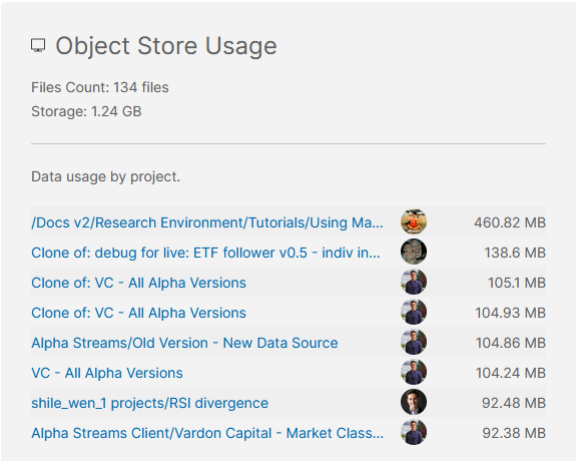
Research to Live Considerations

When you deploy a live algorithm, you can access the data within minutes of modifying the ObjectStore. Ensure your algorithm is able to handle a changing dataset.

The live environment's access to the ObjectStore is much slower than in research and backtesting. Limit the individual objects to less than 50 MB to prevent live trading access issues.

Monitor Usage

The Resources page shows the total storage used in your organization and the storage used by individual projects so that you can easily manage your storage space. To view the page, log in to the Algorithm Lab and then, in the left navigation bar, click Organization > Resources.



Edit Storage Plan

You need storage billing permissions and a paid organization to edit the size of the organization's ObjectStore.

Follow these steps to edit the amount of storage available in your organization's ObjectStore:

1. Log in to the Algorithm Lab.
2. In the left navigation bar, click Organization > Resources .
3. On the Resources page, scroll down to the Storage Resources and then click Add Object Store Capacity .
4. On the Pricing page, select a storage plan.
5. Click Proceed to Checkout .

Delete Storage

To free up storage space, delete the key-value pairs in the ObjectStore by calling the `Delete` method with a key.

```
ObjectStore.Delete(key);
```

```
self.ObjectStore.Delete(key)
```

13 Community

The QuantConnect community consists of over 235,000 quants and investors with diverse backgrounds. Our platform supports several channels of communication so that our members can discuss with the core team, other community members, and third-party contractors. Our community members are a great source for assistance when creating trading algorithms, but we recommend all users complete our base training material before requesting assistance from other members.

Code of Conduct

Community policies and expectations

Forum

Discuss with other members

Discord

Outside of our forum

Profile

Your public profile page

Integration Partners

Quant professionals for hire

Affiliates

Monetize your platform

See Also

Using the Forum
Managing Your Profile

13.1 Code of Conduct

Introduction

The QuantConnect community consists of 235,000 quants and investors with diverse backgrounds. Our platform supports several channels of communication so that our members can discuss with the core team, other community members, and third-party contractors. Our community members are a great source for assistance when creating trading algorithms, but we recommend all users complete our base training material before requesting assistance from other members.

Our community forum was created to be a hub for sharing quality quantitative science insights, discussions on quantitative philosophies, and solving problems. Our goal is to embrace new ways of thinking while keeping a friendly, welcoming environment where people feel comfortable amongst their peers. Our community's exceptional and diverse range of opinions and experiences make us a unique platform. Coming together with mutual respect and courtesy can unite us in growing together as quants.

We ask that our users adhere to the community code of conduct to ensure QuantConnect remains a safe, healthy environment for high-quality quantitative trading discussions.

Expectations

If you're here to get help, make it as easy as possible for others to help you. Follow our guidelines and remember that our community is made possible by volunteers.

Be clear and constructive when giving feedback, and be open when receiving it. Edits, comments, and suggestions are healthy parts of our community.

If you're here to help others, be patient and welcoming. Learning how to participate in our community can be hard. Offer support if you see someone struggling or otherwise in need of help.

Be inclusive and respectful. Avoid sarcasm and be careful with jokes â€” tone is hard to decipher online. Prefer gender-neutral language when uncertain. If a situation makes it hard to be friendly, stop participating and move on.

The following table shows examples of friendly and unfriendly content:

Unfriendly	Friendly
"Google is free!"	"I think googling this might provide you with more helpful information."
"Obviously that's wrong because..."	"I think I can help you with this! Try this..."
"Can you speak English?"	"I think you're trying to say ____. Is that correct?"
"Your strategy will never work because ___."	"Here are some suggestions for your strategy..."

Policies

Please follow our community policies.

Respect

We want our forum to be a place of general respect for one another. Keep interactions constructive, but friendly and lighthearted. Remember â€” we're all here to help one another and keep our community strong.

Due Diligence

We have hundreds of quants posting their questions. Your question is important, but make sure to check a few places before posting. We've worked hard on providing comprehensive documentation and bootcamp tutorials â€” make sure to check there first. Next, try Googling the concept to see if you can get another perspective before posting. Furthermore, our Debugger is a great tool to identify bugs in the code logic.

Relevancy

Keep posts related to algorithmic trading or quantitative finance.

Patience

Be patient with the community responses to your questions. Keep in mind the community are volunteers contributing to your quantitative growth voluntarily. When possible, answer your own questions to leave a path for future readers. Avoid "bumps", "+1", "Any Update?", double posting, thread hijacking , or necro-bumping discussions. Contributions are often rewarded with QuantConnect Credit .

Bigotry

QuantConnect is firmly rooted in our policy against bigotry in our community forum. We are vehemently against racist, sexist, xenophobic, homophobic, or otherwise discriminatory behavior in our community. Any language that may offend anyone based on race, sexual orientation, gender, religion, will not be tolerated.

Harassment

Bullying is not a part of our core culture at QuantConnect. We do not tolerate bullying, sexual harassment, profanity, threats of violence or otherwise, or any sexual harassment in our community forum.

Bug Reports / Data Issues

We request bug reports be sent to the Support Team . The forums are not an effective bug tracking tool and often the reported issue is simply confusion on how the platform operates. For data issues please report the specific dates, times, contracts, and type of issue to the Data Explorer Issues List . This is a system we've designed to track, fix and notify users when issues are fixed.

Promotional Activity

Spam and other forms of promotional activity isn't permitted in the community forum. Posts that deliver immediate value to the readers are permitted such as sharing a well-performing algorithm with an attribution to the author's company in the code comments.

Can Someone Make My Algorithm?

We understand people are at different stages of their quant-growth, but if you are soliciting assistance you should have completed Boot Camp , and attach a backtest or code snippet with your best attempt at building your algorithm. This shows respect for the reader's time. You will need to know how to code to use QuantConnect.

Reporting Violations

We monitor our forum diligently, but if you see something unsavory, please message us to report the activity.

About This Code of Conduct

We aspire to have a welcoming community filled with high-quality discussions about quantitative and algorithmic trading. Since our founding in November 2012, we ran an informal code of conduct and evolved those principles to support the community. Starting January 1st, 2021 as the scale of community content surpassed our ability to review each post, we have sought to write down these guidelines to provide transparency and framework for productive discussions. We welcome your feedback and expect this code of conduct to evolve over time.

13.2 Forum

Introduction

The QuantConnect forum is a place to discuss with other community members, the core QuantConnect team, and our Integration Partners . Use the forum to spark interesting discussions, ask for assistance from other members, and voice your opinion on our products. You must complete 30% of the Bootcamp lessons in the Learning Center to unlock the ability to post to the forum.

Discussions

Discussions are a set of forum posts and comments about a targeted subject. We occasionally post to forum discussions to announce new features, tutorials, and examples. If you have completed 30% of the Bootcamp lessons, you can contribute to discussions. Create discussions to connect with other members or to ask for guidance on a specific problem that you are facing, but always perform your own research and tests before asking other members for assistance.

The forum supports the following functionality:

- Attaching backtest results and research notebooks
- Bookmarking discussions
- Generating links to comments to share
- Publishing text content that includes bold, italics, lists, links, and emojis
- Rewarding discussion content with QuantConnect Credit and upvotes
- Sharing code snippets, photos, videos, and files
- Tagging other members

Sharing Backtests

You can attach project files and backtest results to your forum comments. Sharing your project files is helpful in the following situations:

- Contributing to a discussion about trading strategies and implementations
- Highlighting an issue with data or Lean
- Requesting assistance with an algorithm or research notebook
- Sharing your interesting research and backtests with the community

Sharing backtests is helpful in these situations because it enables other members to reproduce your results. However, since Lean is under constant development, old backtests that are attached to forum discussions can sometimes produce different results than they did in the past.

You can only attach backtests to the forum if they don't throw errors. If you want to share a backtest that throws errors, call the `Quit` method before the error occurs.

Credit

To show your appreciation for contributions in the forum, give some QuantConnect Credit (QCC) rewards . The following table shows the available QCC rewards:

Award	Description	Cost (QCC)
Silver Award	A simple token of recognition from one quant to another. Keep up the great work.	80
Gold Award	This is some great work! Gold star!	600
Platinum Award	Highly resistant to oxidation, this award is for those contributions which will stand the test of time. Strong, classic, and useful for most high technology products.	1,200
Medal for Excellence	The QuantConnect Medal for Excellence is awarded by a member of the QuantConnect staff for exceptional contributions to the QuantConnect community.	3,000
Plutonium Award	Nuclear hot! This post is incredible and deserves recognition as such. Show the author your appreciation for their work.	2,000
Docs Shakespeare	You've left a mark by a contribution to the documentation for the community. Your edits and examples will be followed for generations to come.	500
Nobel Laureate	Bestowed in recognition of quantitative advances.	80
Spaghetti Code Award	Following the intricate flows of code noodle, this code compiles and runs.	300
Jedi Quant	Quant promise you show. Your code channel the force into. Hmmmmmm.	200
Live Trader	That looks like a wild ride.	50
Today I Learned	Thank you for upgrading my brain.	50
Machine Unlearning	I plan on letting the computers do all the work for me.	150
Totally Overfit	I see parameters everywhere.	80
Mind Blown Award	Something incredibly amazing, mind-boggling, and you're shocked senseless.	80
Research Rembrandt	A Jupyter notebook work of art, pulling together all the right hues and plots to be a true masterpiece.	100
	Let's get a boat and start a fund together. Did I mention the boat?	800

Award	Description	Cost (QCC)
Cayman Island Award		
Printing Money Award	Awarded to profitable algorithms.	100
Stronger Together Award	We're stronger together. Let's make this happen!	120
Appreciate the Support Award	Quant trading is a hard road, we could all use a hand.	250
Master Craftsman	You're a master craftsman, taking raw materials and molding them into works of art for the good of the world.	600

Notifications

Follow discussions to receive email notifications when members post new content. You can follow all forum discussions or individual discussions.

All Discussions

Follow these steps to subscribe or unsubscribe from all forum discussions:

1. Log in to your account.
2. Open the forum homepage .
3. Click Email Notifications .

Individual Discussions

Follow these steps to toggle your subscription to individual discussions:

1. Log in to your account.
2. Open the forum homepage .
3. Open the discussion for which you want to toggle your subscription.
4. On the discussion page, click Follow Discussion .

Search Discussions

You can search the forum with keywords or tags.

Search With Keywords

Open the forum homepage , click the magnifying glass icon, and then enter some keywords to search the forum.

Filter Discussions by Tags

Open the forum homepage and then, in the Filter Discussion by Tags section, click some tags to filter the discussions by those tags.

Create Discussions

Follow these steps to create a new forum discussion:

1. Log in to your account.
2. Open the forum homepage .
3. Click Start New Discussion .
4. Enter the discussion title.
5. Enter the discussion content.
6. (Optional) Follow these steps to attach a backtest:
    1. Click Attach Backtest .
    2. Click the Project field and then select the project that contains the backtest that you want to attach from the drop-down menu.

        You can only attach projects that you own.

    3. Click the Backtest field and then select a backtest from the drop-down menu.

    When you attach a backtest, you're sharing all of the project files that were used to generate the backtest results.
7. (Optional) Under the Discussion Tags section, click Show All and then click all of the tags that are relevant to the discussion.
8. Click Publish .

Post Comments

You can only post comments on open forum discussions. To open a closed discussion, contact us .

Follow these steps to post a comment on a forum discussion:

1. Log in to your account.
2. Open the forum homepage .
3. Click the discussion on which you want to comment.
4. At the bottom of the discussion page, enter your comment.
5. (Optional) Follow these steps to attach a backtest:
    1. Click Attach Backtest .
    2. Click the Project field and then select the project that contains the backtest you want to attach from the drop-down menu.

        You can only attach projects that you own.

    3. Click the Backtest field and then select a backtest from the drop-down menu.

    When you attach a backtest, you're sharing all of the project files that were used to generate the backtest results.
6. Click Comment .

    Your comment is appended to the discussion.

Accept Answers

You can only accept answers on discussions you create.

Follow these steps to accept an answer on your discussion:

1. Log in to your account.
2. Open the forum homepage .
3. Open the discussion that contains the answer that you want to accept.

4. On the comment that you want to accept, click Accept Answer .

   The discussion closes and a copy of the accepted answer is added as the first comment in the discussion.

Give Rewards

You need sufficient QuantConnect Credit (QCC) in your organization to give QCC rewards.

Follow these steps to reward a discussion or comment:

1. Log in to your account.
2. Open the forum homepage .
3. Open the discussion to which you want to give your reward.
4. If you want to give a QCC reward, follow these steps:
   1. On the discussion page, if you want to reward the discussion, click Reward Discussion . If you want to reward a comment in the discussion, on the comment that you want to reward, click Reward Answer .
   2. Click a QCC reward.
   3. Click Send Reward .

      The reward displays and the QCC value of the reward is deducted from your organization's balance.
5. If you want to give an upvote, click the thumbs-up icon on the comment that you want to reward.

Share Comments

Follow these steps to share a link to a comment in the forum:

1. Open the forum homepage .
2. Click the discussion that contains the comment that you want to share.
3. On the discussion page, in the bottom-right corner of the comment that you want to share, click the share icon and then click one of the share options.

Edit Comments

You can edit comments only within five minutes after publishing the comments.

Follow these steps to edit a comment:

1. Log in to your account.
2. Open the forum homepage .
3. Click the discussion that contains the comment that you want to edit.
4. On the discussion page, in the top-right corner of the comment that you want to edit, click the three dots icon and then click Edit .
5. (Optional) Update the comment text.
6. (Optional) Follow these steps to update the attached backtest:
   1. Click Update Backtest .
   2. Click the Project field and then select the project that contains the backtest that you want to attach from the drop-down menu.

      You can only attach projects that you own.
   3. Click the Backtest field and then select a backtest from the drop-down menu.
7. Click Update .

   The updated comment displays.

Delete Comments

Email us the comment URL and a reason for deleting the comment. We will consider deleting the comment.

Mark Discussions Read

You can mark individual discussions or all discussions as read.

Individual Discussions

Log in to your account, open the forum homepage , and then open a discussion to mark the discussion as read.

All Discussions

Follow these steps to mark all discussions as read:

1. Log in to your account.
2. Open the forum homepage .
3. At the top of the forum homepage, click the three dots icon and then click Mark all as read

   The notification bubbles at the top of the page disappear.

Access Member Profiles

Open the forum homepage , click a discussion, and then click a member's profile image to view the member's profile page.

Manage Bookmarks

Follow these steps to bookmark a discussion:

1. Log in to your account.
2. Open the forum homepage .
3. Click on the discussion that you want to bookmark.
4. On the discussion page, in the bottom-right corner of the first comment, click the bookmark icon.

   Click the bookmark icon again to remove the discussion from your bookmarks.

On the forum homepage, click Bookmarked to view all of the discussions that you have bookmarked.

13.3 Discord

Introduction

Join the QuantConnect Discord server to chat with community members and the core QuantConnect team in real-time.

Channels

The following table shows the channels in our Discord server:

Category	Channel
QuantConnect News	welcome
QuantConnect News	rules
QuantConnect News	announcements
QuantConnect News	github-activity
Support Zone	live-trading-support
Support Zone	general-community-support
Support Zone	realtime-support
Open-Source	lean
Open-Source	lean-pr-review
Open-Source	lean-cli
Open-Source	docs-feedback
Open-Source	internationalization
Styles and Strategies	trading-discussion
Styles and Strategies	portfolio-strategies
Styles and Strategies	spx-weeklies
Other Discussions	local-platform
Other Discussions	off-topic

Code of Conduct

The Discord server is governed by the normal rules in the Code of Conduct .

13.4 Profile

Introduction

Your QuantConnect profile page is your place to customize your community appearance. Your profile page is accessible to everyone from the forum. The page displays your personal information, your preferred programming language, your latest forum activity, and the badges that you have earned.

Personal Information

Your profile page displays the following personal information about you:

- Username
- Title
- Picture
- Biography
- Social media accounts
- Website
- The date that you joined QuantConnect

You can edit your profile image, username, title, biography, social media links, and email.

Profile Image

Follow these steps to update your profile image:

1. Log in to your account.
2. In the top navigation, click yourUsername > Change Profile Image .
3. Click Browse .
4. Select a file from your local machine and then click Open .

   Your profile image must be in gif , jpg , or png format and less than 1MB in size.

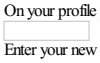
5. Click Save .

   "Profile image changed successfully" displays.

Username

Follow these steps to update your username:

1. Log in to your account.
2. In the top navigation, click yourUsername > My Profile .
3. On your profile page, hover over your username and then click the Edit box that appears.



4. Enter your new username and then click OK .

   Your new username displays.

Title

Follow these steps to update your title:

1. Log in to your account.
2. In the top navigation, click yourUsername > My Profile .
3. On your profile page, hover over your title and then click the Edit box that appears.

4. Enter your new title and then click OK .

   Your new title displays.

Biography

Follow these steps to update your biography:

1. Log in to your account.
2. In the top navigation, click yourUsername > My Profile .
3. On your profile page, hover over your biography and then click the Edit box that appears.

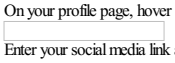
4. Enter your biography and then click OK .

   Your new biography displays.

Social Media Links

Follow these steps to update your social media links:

1. Log in to your account.
2. In the top navigation, click yourUsername > My Profile .
3. On your profile page, hover over the social media icon that you want to update and then click the Edit box that appears.



4. Enter your social media link and then click OK .

Email

Email us to change the email associated with your QuantConnect account.

Organizations

Your profile page displays all of the organizations of which you are a member and the date that you joined each organization.

Activity

Your profile page displays your last five forum posts and your community engagement statistics. The forum post display lets other members read your recent posts and lets you find the comments that you recently posted. The page displays your community engagement statistics like the number of experience points that you've earned, the number of comments that you've posted, the number of discussions that you've created, and the number of algorithms that you've shared in the forum. Your community engagement statistics are displayed so that other members can see how active you are in the forum.

In addition to your forum activity, your profile page also platform engagement statistics like the number of backtests that you've run and the proportion of the Bootcamp lessons that you've completed. Your platform engagement statistics are displayed so that other members can see how experienced you are with QuantConnect and Lean.

Badges

Badges are fun to collect and show your experience with QuantConnect. Your profile page displays the badges you've earned. The following table describes the badges you can earn:

Badge	Description
Algorithm Sharing	Share an algorithm with the community.

Badge	Description
Chatter Box	Post 100+ comments in the forum.
Backtester	Run 100+ backtests.
Live Trading	Deploy a live trading algorithm.
Parallel Live Trading	Run multiple live trading algorithms simultaneously.
Connected With Discord	Connect your QuantConnect account with our Discord server .
C# Programming	Set C# as your preferred programming language.
Python Programming	Set Python as your preferred programming language.
Popular	Receive 10+ upvotes from different members.
Boot Camp Private	Finish the easy tutorial.
Boot Camp Corporal	Finish the intermediate tutorial.
Boot Camp Major	Finish the advanced tutorial.
Notebook Sharing	Attach a research notebook to a post in the community forum.
Broadcaster	Attach a live running algorithm to a forum discussion.
Cloner	Clone 100+ algorithms from the community forum.
Bug Hunter	Report 10+ bugs.
Open Source Contributor	Make a pull request to one of our GitHub repositories .
Alpha Streams Developer	Add 1 Alpha to the Alpha Streams Market.
Alpha Streams Artisan	Add 3 Alphas to the Alpha Streams Market.
Alpha Streams Master	Add 10 Alphas to the Alpha Streams Market.
Alpha Winner #1	Place first in the Alpha competition.
Alpha Winner #2	Place second in the Alpha competition.
Alpha Winner #3	Place third in the Alpha competition.
Community Editor	Provide great service to community members in the forum or the Discord server.
Outstanding Community Service	Provide excellent service to community members in the forum or the Discord server.

Log In

Follow these steps to log in to your account:

1. In the top navigation bar, click Sign In .

   The Sign In to QuantConnect page displays.

2. If you signed up with your Google or Facebook account, click Sign In with Google or Sign In with Facebook and then follow the prompts.
3. If you signed up with your email address, enter your email address, enter your password, and then click Sign In .
4. If you have 2FA enabled on your account, in the Please Enter 2FA Code field, enter the authentication code from the Google Authenticator app on your mobile device and then click Sign In .

Toggle 2FA

Follow these steps to activate or deactivate two-factor authentication (2FA):

1. Log in to your account.
2. In the top navigation bar, click yourUsername > My Account .
3. If you want to activate 2FA, follow these steps:
   1. On your Account page, in the Security section, click Activate .
   2. Enter your email address and then click Continue .
   3. Follow the steps in the pop-up window and then click Activate Two Factor .

      "Updated successfully" displays.

4. If you want to deactivate 2FA, follow these steps:
   1. On your Account page, in the Security section, click Deactivate .
   2. Enter your email address and then click Continue .

      "Success" displays.

Request API Token

Follow these steps to request an API token:

1. Log in to your account.
2. In the top navigation bar, click yourUsername > My Account .
3. On your Account page, in the Security section, click Request Email With Token and Your User-Id for API Requests .
4. Click OK .

We email you your user Id and API token.

Close Active Sessions

Follow these steps to close active sessions:

1. Log in to your account.
2. In the top navigation bar, click yourUsername > My Account .

3. On your Account page, in the Security section, click Sign Out next to the session that you want to close.

   "Session terminated successfully" displays.

Manage Email Subscriptions

Follow these steps to manage your email subscriptions:

1. Log in to your account.
2. In the top navigation bar, click yourUsername > My Account .
3. On your Account page, in the Account Settings section, click Manage Email Subscriptions .
4. Select the subscriptions you want and deselect the subscriptions you don't want.

   "Preferences updated successfully" displays.

Change Password

Follow these steps to change your password:

1. Log in to your account.
2. In the top navigation bar, click yourUsername > Change Password .
3. Enter your current password.
4. Enter your new password.
5. Enter your new password again to confirm.
6. Press Reset Password .
7. Enter your email address and then click Continue .

Sign Out

In the top navigation bar, click yourUsername > Sign Out to sign out.

Deactivate Account

Follow these steps to deactivate your account:

1. Log in to your account.
2. In the top navigation bar, click yourUsername > My Account .
3. At the bottom of your Account page, click Deactivate Account .
4. Click Deactivate Account .
5. Enter your email address and then click Continue .

   The QuantConnect homepage displays. Log in to reactivate your account.

Remove Account

Contact us to remove your QuantConnect account from our servers.

13.5 Integration Partners

Introduction

Our Integration Partners are hand-picked, independent consultants and companies with a solid track record of operational excellence with QuantConnect. They offer guidance, consultation, teaching, coding development services. Let them help take your idea to a fully implemented algorithmic trading strategy.

Hire Integration Partners

The Integration Partners are thoroughly vetted and must pass a test to provide their services to the community. They decide their pricing and the services that they provide. Their services range from beginner to advanced dives into the algorithm framework, strategy and project consultation, developing simple to complex algorithms, portfolio optimization, and more. Hire them on the Integration Partners page to connect with a QuantConnect expert, to progress your development skills, or to outsource your algorithm development.

Join the Integration Partner Team

To list your services on the Integration Partners page, contact us and pass our test. If you are accepted and community members hire you, you receive 100% of the revenue. As an Integration Partner, you set your own pricing, define your own services, pick your own hours, and get paid to work on your quant trading specialty. Our Integration Partner program is a great addition to resumes.

13.6 Affiliates

Introduction

Our Affiliate Program gives influential quants, financial strategists, and traders an opportunity to use their platforms to share QuantConnect and subsequently share in profits from referrals.

Earning Potential

On a monthly basis, you'll receive 10% of sales that originate for your referral. You'll receive this for up to 12 months after the individual you refer subscribes to our services. For example, if you refer a customer who signs up for our $280/month plan and the custom stays with us for 12+ months, you'll receive 12 payments of $28 for a total of $336 over the course of the first year. If the customer leaves after five months, you'll receive compensation for those five months.

YouTube Case Study

Chris is a YouTube host with an audience of more than 20,000 subscribers where he posts trading and finance-related content in an easy-to-digest format. He is often appreciated for his ability to simply present complex concepts. Chris originally organically posted a review of various algorithmic trading platforms that caught our eye, so we invited him to join the QuantConnect Affiliate Program.

Over a period of two months, Chris and the QuantConnect team collaborated to design a series of video topics that would be interesting for the broader community. Chris always had complete content ownership and artistic control over his content. He often sought a â€œcontent reviewâ€ from the QuantConnect team in a private Slack channel to help maintain the technical accuracy and quality of the videos. We were mindful to update him about new features that might impact or improve his videos. Each time Chris published a video, we would re-share it on all our social media channels to increase his video reach. We were always motivated to get high-quality content to the community so they can learn more effectively.

Through his channel and the partnership with QuantConnect, Chris was able to create a long-term passive income from his content. Each month, he drives approximately 500 users to QuantConnect and 10-15 subscribe to become long-term clients. Within six months, he has built a passive recurring revenue stream of $250/month.

Become a Partner

To apply for the QuantConnect Affiliate Program, fill in the application form.

14 API Reference

The QuantConnect REST API lets you communicate with our cloud servers through URL endpoints.

Authentication

Login to use the API

Project Management

Create, read, update, and delete projects

File Management

Create, read, update, and delete files from a project

Compiling Code

Compile codes for backtest and live trading

Backtest Management

Create, read, update, and delete backtests from a project

Live Management

Create, read, and update live algorithms

Downloading Data

Download data for backtest usage

Reports

Generate reports for backtests to evaluate performance

14.1 Authentication

Introduction

Make authenticated REST requests to the QuantConnect API with your User-id and API-Token. Use a simple API endpoint to verify the authentication is working correctly.

The base URL of QuantConnect API is `https://www.quantconnect.com/api/v2` .

Authenticating Requests

Requests to QuantConnect API v2 require a hashed combination of time, and the API token. The unixtime stamp combination serves as a nonce token as each request is sent with a different signature but never requires sending the API token itself.

Hashing

Follow the below example to create a hashed token for authentication.

```
#r "nuget:RestSharp"
using System;
using System.Security.Cryptography;
using RestSharp;

// Get timestamp
var stamp = ((DateTimeOffset)DateTime.UtcNow).ToUnixTimeSeconds();
var timeStampedToken = $"{<yourApiToken>}:{stamp}";

// Get hashed API token
var crypt = new SHA256Managed();
var hashToken = crypt.ComputeHash(Encoding.UTF8.GetBytes(timeStampedToken), 0, Encoding.UTF8.GetByteCount(timeStampedToken));
var hash = new StringBuilder();
foreach (var theByte in hashToken)
{
    hash.Append(theByte.ToString("x2"));
}
var apiToken = hash.ToString();
```

```
import base64
import hashlib
import time

# Get timestamp
timestamp = str(int(time.time()))
time_stamped_token = "<your_api_token>" + ':' + timestamp

# Get hased API token
hashed_token = hashlib.sha256(time_stamped_token.encode('utf-8')).hexdigest()
authentication = "{}:{}".format(<your_user_id>, hashed_token)
api_token = base64.b64encode(authentication.encode('utf-8')).decode('ascii')
```

Make API Request

Follow the below example to install the hashing into the authenticator and make an API request.

Follow the below example to install the hashing into the headings and make an API request.

```
// Create REST client and install authenticator.
var client = new RestClient("<requestUrl>");
client.Authenticator = new HttpBasicAuthenticator(
    "<yourUserId>",
    hash.ToString()
);

// Create Request and add timestamp header (optional: Json Content).
var request = new RestRequest();
request.AddHeader("Timestamp", stamp.ToString());

// Make POST request.
var response = await client.PostAsync(request);
var content = response.Content
```

```
# Create headers dictionary.
headers = {
    'Authorization': 'Basic %s' % api_token,
    'Timestamp': timestamp
}

# Create POST Request with headers (optional: Json Content as data argument).
response = requests.post("<request_url>",
                         data = {},
                         headers = headers)
content = response.text
```

Authenticated State Request

Authentication status check endpoint to verify the hashing function is working successfully. The `/authenticate` API does not require any information, but just an authenticated hash in the header.

Authenticated State Responses

The `/authenticate` API provides a response in the following format:

200 Success

`RestResponse` Model - Base API response class for the QuantConnect API.

success	`boolean`	Indicate if the API request was successful.
Example		`{ "success": true }`

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string`	Header

14.2 Project Management

The QuantConnect REST API lets you manage your projects on our cloud servers through URL endpoints.

Create Project

Read Project

Update Project

Delete Project

Read Project Nodes

Update Project Nodes

14.2.1 Create Project

Introduction

Create a new project in your default organization.

Description

Create a project with the specified name and programming language. If the project-name already exists, API call returns success:false with exception details in the errors array.

Request

Name and language of the project to create. The `/projects/create` API accepts requests in the following format:

`CreateProjectRequest` Model - Request to create a project.

name	`string` Project name.
language	`string Enum` Programming langage to use. Options : ['C#', 'Py']

Example	{
	"name": "string",
	"language": "C#"
	}

Responses

The `/projects/create` API provides a response in the following format:

200 Success

`ProjectListResponse` Model - Project list response.

projects	`Project Array` List of projects for the authenticated user.
success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	{
	"projects": [
	{
	"projectId": 0,
	"name": "string",
	"created": "2021-11-26T15:18:27.693Z",
	"modified": "2021-11-26T15:18:27.693Z",
	"language": "C#"
	}
	],
	"success": true,
	"errors": [
	"string"
	]
	}

`Project` Model - Response from reading a project by id.

projectId	`integer` Project id.
name	`string` Name of the project.
created	`string($date-time)` Date the project was created.
modified	`string($date-time)` Modified date for the project.
language	`string Enum` Programming language of the project. Options : ['C#', 'Py']

Example	{
	"projectId": 0,
	"name": "string",
	"created": "2021-11-26T15:18:27.693Z",
	"modified": "2021-11-26T15:18:27.693Z",
	"language": "C#"
	}

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header

14.2.2 Read Project

Introduction

If a ReadProjectRequest is passed, get details about that single project. If no request body is passed, list details of all projects.

Request

The projectId for the project to read, or nothing to get a list of all projects. The `/projects/read` API accepts requests in the following format:

`ReadProjectRequest` Model - Request to get details about a specific project.

projectId	`integer` Id of the project.

Example	{
	"projectId": 0
	}

Responses

The `/projects/read` API provides a response in the following format:

200 Success

`ProjectListResponse` Model - Project list response.

projects	`Project Array` List of projects for the authenticated user.
success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	{
	"projects": [
	{
	"projectId": 0,
	"name": "string",
	"created": "2021-11-26T15:18:27.693Z",
	"modified": "2021-11-26T15:18:27.693Z",
	"language": "C#"
	}
	],
	"success": true,
	"errors": [
	"string"
	]
	}

`Project` Model - Response from reading a project by id.

projectId	`integer` Project id.
name	`string` Name of the project.
created	`string($date-time)` Date the project was created.
modified	`string($date-time)` Modified date for the project.
language	`string Enum` Programming language of the project. Options : ['C#', 'Py']

Example	{
	"projectId": 0,
	"name": "string",
	"created": "2021-11-26T15:18:27.693Z",
	"modified": "2021-11-26T15:18:27.693Z",
	"language": "C#"
	}

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header

14.2.3 Update Project

Introduction

Update a project name, description or parameters.

Request

The `/projects/update` API accepts requests in the following format:

`UpdateProjectRequest` Model - Update a project name, description or parameters.

projectId	`integer` Project Id to which the file belongs.
name	`string` The new name for the project.
description	`object` The new description for the project.

Example	<pre>{
	"projectId": 0,
	"name": "string",
	"description":
	}</pre>

Responses

The `/projects/update` API provides a response in the following format:

200 Success

`RestResponse` Model - Base API response class for the QuantConnect API.

success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	<pre>{
	"success": true,
	"errors": [
	"string"
	]
	}</pre>

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header	

14.2.4 Delete Project

Introduction

Delete a project.

Request

The `/projects/delete` API accepts requests in the following format:

`DeleteProjectRequest` Model - Request to delete a project.

projectId	`integer` Project Id to which the file belongs.

Example	<pre>{
	"projectId": 0
	}</pre>

Responses

The `/projects/delete` API provides a response in the following format:

200 Success

`RestResponse` Model - Base API response class for the QuantConnect API.

success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	<pre>{
	"success": true,
	"errors": [
	"string"
	]
	}</pre>

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header	

14.2.5 Read Project Nodes

Introduction

Read all nodes in a project.

Request

The `/projects/nodes/read` API accepts requests in the following format:

`ReadProjectNodesRequest` Model - Request to get details about nodes of a specific organization.

projectId	`string` Project Id to which the nodes refer.

Example	<pre>{
	"projectId": "string"
	}</pre>

Responses

The `/projects/nodes/read` API provides a response in the following format:

200 Success

`ProjectNodesResponse` Model - Response received when reading all nodes of a project.

nodes	`#/components/schemas/ProjectNodes` List of project nodes..
autoSelectNode	`boolean` Indicate if a node is automatically selected.
success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	<pre>{
	"nodes": ,
	"autoSelectNode": true,
	"success": true,
	"errors": [
	"string"
	]
	}</pre>

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header

14.2.6 Update Project Nodes

Introduction

Update the active state of some nodes to true. If you don't provide any nodes, all the nodes become inactive and AutoSelectNode is true.

Request

The `/projects/nodes/update` API accepts requests in the following format:

`UpdateProjectNodesRequest` Model - Request to update the nodes of a project.

projectId	`integer` Project Id to which the nodes refer.
nodes	`string Array` List of node Id to update.

Example	`{`
	`"projectId": 0,`
	`"nodes": [`
	`"string"`
	`]`
	`}`

Responses

The `/projects/nodes/update` API provides a response in the following format:

200 Success

`ProjectNodesResponse` Model - Response received when reading all nodes of a project.

nodes	`#/components/schemas/ProjectNodes` List of project nodes..
autoSelectNode	`boolean` Indicate if a node is automatically selected.
success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	`{`
	`"nodes": ,`
	`"autoSelectNode": true,`
	`"success": true,`
	`"errors": [`
	`"string"`
	`]`
	`}`

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header

14.3 File Management

The QuantConnect REST API lets you manage your files on our cloud servers through URL endpoints.

Create File

Read File

Update File

Delete File

14.3.1 Create File

Introduction

Add a file to given project.

Request

Project, file name and file content to create. The `/files/create` API accepts requests in the following format:

`CreateProjectFileRequest` Model - Request to add a file to a project.

projectId	`integer` Project Id to which the file belongs.
name	`string` *example: main.py* The name of the new file.
content	`string` The content of the new file.

Example	<pre>{
	"projectId": 0,
	"name": "main.py",
	"content": "string"
	}</pre>

Responses

The `/files/create` API provides a response in the following format:

200 Success

`ProjectFilesResponse` Model - Response received when reading files from a project.

files	`ProjectFile Array` List of project file information.
success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	<pre>{
	"files": [
	{
	"name": "string",
	"content": "string",
	"modified": "2021-11-26T15:18:27.693Z"
	}
	],
	"success": true,
	"errors": [
	"string"
	]
	}</pre>

`ProjectFile` Model - File for a project.

name	`string` Name of a project file.
content	`string` Contents of the project file.
modified	`string($date-time)` DateTime project file was modified.

Example	<pre>{
	"name": "string",
	"content": "string",
	"modified": "2021-11-26T15:18:27.693Z"
	}</pre>

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header

14.3.2 Read File

Introduction

If a ReadSingleFileRequest is passed, reads that file from the project. If a ReadAllFilesRequest is passed, reads all files in the project.

Request

An array list of files from the project requested. The `/files/read` API accepts requests in the following format:

`ReadFilesRequest` Model - Request to read all files from a project.

projectId	`integer` Project Id to which the file belongs.
fileName	`string` Optional. The name of the file that should be updated.

Example	{
	"projectId": 0,
	"fileName": "string"
	}

Responses

The `/files/read` API provides a response in the following format:

200 Success

`ProjectFilesResponse` Model - Response received when reading files from a project.

files	`ProjectFile Array` List of project file information.
success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	{
	"files": [
	{
	"name": "string",
	"content": "string",
	"modified": "2021-11-26T15:18:27.693Z"
	}
	],
	"success": true,
	"errors": [
	"string"
	]
	}

`ProjectFile` Model - File for a project.

name	`string` Name of a project file.
content	`string` Contents of the project file.
modified	`string($date-time)` DateTime project file was modified.

Example	{
	"name": "string",
	"content": "string",
	"modified": "2021-11-26T15:18:27.693Z"
	}

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header

14.3.3 Update File

Introduction

If an UpdateProjectFileNameRequest is passed, update the name of a file. If a UpdateProjectFileContentsRequest is passed, update the contents of a file.

Request

Information about the file to update along with the new properties to set. The `/files/update` API accepts requests in the following format:

`UpdateFileNameRequest` Model - Request to update the name of a file.

projectId	`integer` Project Id to which the file belongs.
oldFileName	`string` The current name of the new file.
newFileName	`string` The new name for the file.

Example	`{`
	`"projectId": 0,`
	`"oldFileName": "string",`
	`"newFileName": "string"`
	`}`

`UpdateFileContentsRequest` Model - Request to update the contents of a file.

projectId	`integer` Project Id to which the file belongs.
fileName	`string` The name of the file that should be updated.
newFileContents	`string` The new contents of the file.

Example	`{`
	`"projectId": 0,`
	`"fileName": "string",`
	`"newFileContents": "string"`
	`}`

Responses

The `/files/update` API provides a response in the following format:

200 Success

`RestResponse` Model - Base API response class for the QuantConnect API.

success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	`{`
	`"success": true,`
	`"errors": [`
	`"string"`
	`]`
	`}`

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header

14.3.4 Delete File

Introduction

Delete a file in a project

Request

Project Id and filename to specify the file for deletion. The `/files/delete` API accepts requests in the following format:

`DeleteFileRequest` Model - Request to delete a file in a project.

projectId	`integer` Project Id to which the file belongs.
name	`string` The name of the file that should be deleted.

Example	<pre>{
	"projectId": 0,
	"name": "string"
	}</pre>

Responses

The `/files/delete` API provides a response in the following format:

200 Success

`RestResponse` Model - Base API response class for the QuantConnect API.

success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	<pre>{
	"success": true,
	"errors": [
	"string"
	]
	}</pre>

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string`	Header

14.4 Compiling Code

The QuantConnect REST API lets you compile your projects on our cloud servers through URL endpoints.

Create Compilation Job

Read Compilation Result

14.4.1 Create Compilation Job

Introduction

Asynchronously create a compile job request for a project

Request

Project Id specifying project to build. The `/compile/create` API accepts requests in the following format:

`CreateCompileRequest` Model - Request to compile a project.

projectId	integer Project Id we wish to compile.	

Example	<pre>{
	"projectId": 0
	}</pre>	

Responses

The `/compile/create` API provides a response in the following format:

200 Success

`CompileResponse` Model - Response from the compiler on a build event.

compileId	string Compile Id for a sucessful build.	
state	string Enum True on successful compile. Options : ['InQueue', 'BuildSuccess', 'BuildError']	
logs	string Array Logs of the compilation request.	
success	boolean Indicate if the API request was successful.	
errors	string Array List of errors with the API call.	

Example	<pre>{
	"compileId": "string",
	"state": "InQueue",
	"logs": [
	"string"
	],
	"success": true,
	"errors": [
	"string"
	]
	}</pre>	

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	string Header	

14.4.2 Read Compilation Result

Introduction

Read a compile packet job result.

Request

Read a compile result for a specific Project Id and Compile Id. The `/compile/read` API accepts requests in the following format:

`ReadCompileRequest` Model - Request to read a compile packet job.

projectId	`integer` Project Id we sent for compile.
compileId	`string` Compile Id returned during the creation request.

Example	<pre>{
	"projectId": 0,
	"compileId": "string"
	}</pre>

Responses

The `/compile/read` API provides a response in the following format:

200 Success

`CompileResponse` Model - Response from the compiler on a build event.

compileId	`string` Compile Id for a sucessful build.
state	`string Enum` True on successful compile. Options : ['InQueue', 'BuildSuccess', 'BuildError']
logs	`string Array` Logs of the compilation request.
success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	<pre>{
	"compileId": "string",
	"state": "InQueue",
	"logs": [
	"string"
	],
	"success": true,
	"errors": [
	"string"
	]
	}</pre>

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header

14.5 Backtest Management

The QuantConnect REST API lets you manage backtests on our cloud servers through URL endpoints.

Create Backtest

Read Backtest

Update Backtest

Delete Backtest

14.5.1 Create Backtest

Introduction

Create a new backtest request and get the backtest Id.

Request

Create a new backtest given a project Id and compile Id. The `/backtests/create` API accepts requests in the following format:

`CreateBacktestRequest` Model - Request to create a new backtest.

projectId	`integer` Project Id we sent for compile.
compileId	`string` Compile Id for the project to backtest.
backtestName	`string` Name for the new backtest.

Example	<pre>{
	"projectId": 0,
	"compileId": "string",
	"backtestName": "string"
	}</pre>

Responses

The `/backtests/create` API provides a response in the following format:

200 Success

`BacktestResponse` Model - Packet container for carrying Backtest results.

name	`string` Name of the backtest.
note	`string` Note on the backtest attached by the user.
backtestId	`string` Assigned backtest Id.
completed	`boolean` Boolean true when the backtest is completed.
progress	`number` Progress of the backtest in percent 0-1.
result	`BacktestResult object` Results object class. Results are exhaust from backtest or live algorithms running in LEAN.
error	`string` Backtest error message.
stacktrace	`string` Backtest error stacktrace.
created	`string($date-time)` Backtest creation date and time.
success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.


```
{
"name": "string",
"note": "string",
"backtestId": "string",
"completed": true,
"progress": 0,
"result": {
"RollingWindow": {
"TradeStatistics": {
"StartDateTime": "2021-11-26T15:18:27.693Z",
"EndDateTime": "2021-11-26T15:18:27.693Z",
"TotalNumberOfTrades": 0,
"NumberOfWinningTrades": 0,
"NumberOfLosingTrades": 0,
"TotalProfitLoss": 0,
"TotalProfit": 0,
"TotalLoss": 0,
"LargestProfit": 0,
"LargestLoss": 0,
"AverageProfitLoss": 0,
"AverageProfit": 0,
"AverageLoss": 0,
"AverageTradeDuration": "string",
"AverageWinningTradeDuration": "string",
"AverageLosingTradeDuration": "string",
"MedianTradeDuration": "string",
"MedianWinningTradeDuration": "string",
"MedianLosingTradeDuration": "string",
"MaxConsecutiveWinningTrades": 0,
"MaxConsecutiveLosingTrades": 0,
"ProfitLossRatio": 0,
"WinLossRatio": 0,
"WinRate": 0,
"LossRate": 0,
"AverageMAE": 0,
"AverageMFE": 0,
"LargestMAE": 0,
"LargestMFE": 0,
"MaximumClosedTradeDrawdown": 0,
"MaximumIntraTradeDrawdown": 0,
"ProfitLossStandardDeviation": 0,
"ProfitLossDownsideDeviation": 0,
"ProfitFactor": 0,
"SharpeRatio": 0,
"SortinoRatio": 0,
"ProfitToMaxDrawdownRatio": 0,
"MaximumEndTradeDrawdown": 0,
"AverageEndTradeDrawdown": 0,
"MaximumDrawdownDuration": "string",
"TotalFees": 0
},
"PortfolioStatistics": {
"RiskFreeRate": 0,
"AverageWinRate": 0,
"AverageLossRate": 0,
"ProfitLossRatio": 0,
"WinRate": 0,
"LossRate": 0,
"Expectancy": 0,
"CompoundingAnnualReturn": 0,
"Drawdown": 0,
"TotalNetProfit": 0,
"SharpeRatio": 0,
"ProbabilisticSharpeRatio": 0,
"Alpha": 0,
```

`BacktestResponse` **Model - Packet container for carrying Backtest results.**

Example


```
"Beta": 0,
"AnnualStandardDeviation": 0,
"AnnualVariance": 0,
"InformationRatio": 0,
"TrackingError": 0,
"TreynorRatio": 0
},
"ClosedTrades": [
{
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"EntryTime": "2021-11-26T15:18:27.693Z",
"EntryPrice": 0,
"Direction": "Long",
"Quantity": 0,
"ExitTime": "2021-11-26T15:18:27.693Z",
"ExitPrice": 0,
"ProfitLoss": 0,
"TotalFees": 0,
"MAE": 0,
"MFE": 0,
"Duration": "string",
"EndTradeDrawdown": 0
}
]
},
"TotalPerformance": {
"TradeStatistics": {
"StartDateTime": "2021-11-26T15:18:27.693Z",
"EndDateTime": "2021-11-26T15:18:27.693Z",
"TotalNumberOfTrades": 0,
"NumberOfWinningTrades": 0,
"NumberOfLosingTrades": 0,
"TotalProfitLoss": 0,
"TotalProfit": 0,
"TotalLoss": 0,
"LargestProfit": 0,
"LargestLoss": 0,
"AverageProfitLoss": 0,
"AverageProfit": 0,
"AverageLoss": 0,
"AverageTradeDuration": "string",
"AverageWinningTradeDuration": "string",
"AverageLosingTradeDuration": "string",
"MedianTradeDuration": "string",
"MedianWinningTradeDuration": "string",
"MedianLosingTradeDuration": "string",
"MaxConsecutiveWinningTrades": 0,
"MaxConsecutiveLosingTrades": 0,
"ProfitLossRatio": 0,
"WinLossRatio": 0,
"WinRate": 0,
"LossRate": 0,
"AverageMAE": 0,
"AverageMFE": 0,
"LargestMAE": 0,
"LargestMFE": 0,
"MaximumClosedTradeDrawdown": 0,
"MaximumIntraTradeDrawdown": 0,
"ProfitLossStandardDeviation": 0,
"ProfitLossDownsideDeviation": 0,
"ProfitFactor": 0,
"SharpeRatio": 0,
"SortinoRatio": 0,
"ProfitToMaxDrawdownRatio": 0,
"MaximumEndTradeDrawdown": 0,
"AverageEndTradeDrawdown": 0,
"MaximumDrawdownDuration": "string",
"TotalFees": 0
},
"PortfolioStatistics": {
"RiskFreeRate": 0,
"AverageWinRate": 0,
"AverageLossRate": 0,
"ProfitLossRatio": 0,
"WinRate": 0,
"LossRate": 0,
"Expectancy": 0,
"CompoundingAnnualReturn": 0,
"Drawdown": 0,
"TotalNetProfit": 0,
"SharpeRatio": 0,
"ProbabilisticSharpeRatio": 0,
"Alpha": 0,
"Beta": 0,
"AnnualStandardDeviation": 0,
"AnnualVariance": 0,
"InformationRatio": 0,
"TrackingError": 0,
"TreynorRatio": 0
},
"ClosedTrades": [
{
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"EntryTime": "2021-11-26T15:18:27.693Z",
"EntryPrice": 0,
"Direction": "Long",
"Quantity": 0,
"ExitTime": "2021-11-26T15:18:27.693Z",
"ExitPrice": 0,
"ProfitLoss": 0,
"TotalFees": 0,
"MAE": 0,
"MFE": 0,
"Duration": "string",
"EndTradeDrawdown": 0
}
]
},
"AlphaRuntimeStatistics": {
"MeanPopulationScore": {
"UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
"Direction": 0,
"Magnitude": 0,
"IsFinalScore": true
},
"RollingAveragedPopulationScore": {
"UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
"Direction": 0,
"Magnitude": 0,
"IsFinalScore": true
},
"LongCount": "string",
```

BacktestResponse **Model - Packet container for carrying Backtest results.**


```
                                        "ShortCount": "string",
                                        "LongShortRatio": 0,
                    "TotalAccumulatedEstimatedAlphaValue": 0,
                                "KellyCriterionEstimate": 0,
                        "KellyCriterionProbabilityValue": 0,
                                        "FitnessScore": 0,
                                    "PortfolioTurnover": 0,
                                "ReturnOverMaxDrawdown": 0,
                                        "SortinoRatio": 0,
                            "EstimatedMonthlyAlphaValue": 0,
                        "TotalInsightsGenerated": "string",
                            "TotalInsightsClosed": "string",
                "TotalInsightsAnalysisCompleted": "string",
                    "MeanPopulationEstimatedInsightValue": 0
                                                        },
                                                "Charts": {
                                            "Name": "string",
                                    "ChartType": "Overlay",
                                                "Series": {
                                            "Name": "string",
                                            "Unit": "string",
                                                "Index": 0,
                                                "Values": [
                                                          {
                                                "x": "string",
                                                      "y": 0
                                                          }
                                                        ],
                                        "SeriesType": "Line",
                                            "Color": "string",
                                "ScatterMarkerSymbol": "none"
                                                          }
                                                        },
                                                "Orders": {
                                                    "Id": 0,
                                        "ContingentId": 0,
                                            "BrokerId": [
                                                "string"
                                                        ],
                                                "Symbol": {
                                        "Value": "string",
                                            "ID": "string",
                                        "Permtick": "string"
                                                        },
                                                "Price": 0,
                                "PriceCurrency": "string",
                        "Time": "2021-11-26T15:18:27.693Z",
                "CreatedTime": "2021-11-26T15:18:27.693Z",
                "LastFillTime": "2021-11-26T15:18:27.693Z",
            "LastUpdateTime": "2021-11-26T15:18:27.693Z",
                "CanceledTime": "2021-11-26T15:18:27.693Z",
                                            "Quantity": 0,
                                        "Type": "Market",
                                         "Status": "New",
                                         "Tag": "string",
                                    "SecurityType": "Base",
                                        "Direction": "Buy",
                                                "Value": 0,
                                    "OrderSubmissionData": {
                                            "BidPrice": 0,
                                            "AskPrice": 0,
                                            "LastPrice": 0
                                                        },
                                        "IsMarketable": true
                                                        },
                                            "OrderEvents": [
                                                          {
                                                "OrderId": 0,
                                                    "Id": 0,
                                                "Symbol": {
                                        "Value": "string",
                                            "ID": "string",
                                        "Permtick": "string"
                                                        },
                    "UtcTime": "2021-11-26T15:18:27.693Z",
                                                "Status": {
                                            "OrderId": 0,
                                                    "Id": 0,
                                                "Symbol": {
                                        "Value": "string",
                                            "ID": "string",
                                        "Permtick": "string"
                                                        },
                    "UtcTime": "2021-11-26T15:18:27.693Z",
                                            "Status": "New",
                                             "FillPrice": 0,
                            "FillPriceCurrency": "string",
                                        "FillQuantity": 0,
                                        "Direction": "Buy",
                                        "Message": "string",
                                    "IsAssignment": true,
                                            "StopPrice": 0,
                                            "LimitPrice": 0,
                                               "Quantity": 0
                                                        },
                                               "OrderFee": {
                                                  "Value": {
                                                "Amount": 0,
                                        "Currency": "string"
                                                          }
                                                        },
                                             "FillPrice": 0,
                            "FillPriceCurrency": "string",
                                        "FillQuantity": 0,
                                        "Direction": "Buy",
                                        "Message": "string",
                                    "IsAssignment": true,
                                            "StopPrice": 0,
                                            "LimitPrice": 0,
                                               "Quantity": 0
                                                          }
                                                        ],
                                    "ProfitLoss": "number",
                                    "Statistics": "string",
                            "RuntimeStatistics": "string",
                              "ServerStatistics": "string"
                                                        },
                                        "error": "string",
                                    "stacktrace": "string",
                    "created": "2021-11-26T15:18:27.693Z",
                                            "success": true,
                                                "errors": [
                                                   "string"
                                                          ]
                                                          }
```


BacktestResult **Model - Results object class. Results are exhaust from backtest or live algorithms running in LEAN.**

RollingWindow

AlgorithmPerformance object

Rolling window detailed statistics.

BacktestResult - Model - Results object class. Results are exhaust from backtest or live algorithms running in LEAN.

TotalPerformance	AlgorithmPerformance object The AlgorithmPerformance class is a wrapper for TradeStatistics and PortfolioStatistics.
AlphaRuntimeStatistics	AlphaRuntimeStatistics object Contains insight population run time statistics.
Charts	Chart object Charts updates for the live algorithm since the last result packet.
Orders	Order object Order updates since the last result packet.
OrderEvents	OrderEvent Array OrderEvent updates since the last result packet.
ProfitLoss	number object Trade profit and loss information since the last algorithm result packet.
Statistics	string object Statistics information sent during the algorithm operations.
RuntimeStatistics	string object Runtime banner/updating statistics in the title banner of the live algorithm GUI.
ServerStatistics	string object Server status information, including CPU and RAM usage.


```
{
  "RollingWindow": {
    "TradeStatistics": {
      "StartDateTime": "2021-11-26T15:18:27.693Z",
      "EndDateTime": "2021-11-26T15:18:27.693Z",
      "TotalNumberOfTrades": 0,
      "NumberOfWinningTrades": 0,
      "NumberOfLosingTrades": 0,
      "TotalProfitLoss": 0,
      "TotalProfit": 0,
      "TotalLoss": 0,
      "LargestProfit": 0,
      "LargestLoss": 0,
      "AverageProfitLoss": 0,
      "AverageProfit": 0,
      "AverageLoss": 0,
      "AverageTradeDuration": "string",
      "AverageWinningTradeDuration": "string",
      "AverageLosingTradeDuration": "string",
      "MedianTradeDuration": "string",
      "MedianWinningTradeDuration": "string",
      "MedianLosingTradeDuration": "string",
      "MaxConsecutiveWinningTrades": 0,
      "MaxConsecutiveLosingTrades": 0,
      "ProfitLossRatio": 0,
      "WinLossRatio": 0,
      "WinRate": 0,
      "LossRate": 0,
      "AverageMAE": 0,
      "AverageMFE": 0,
      "LargestMAE": 0,
      "LargestMFE": 0,
      "MaximumClosedTradeDrawdown": 0,
      "MaximumIntraTradeDrawdown": 0,
      "ProfitLossStandardDeviation": 0,
      "ProfitLossDownsideDeviation": 0,
      "ProfitFactor": 0,
      "SharpeRatio": 0,
      "SortinoRatio": 0,
      "ProfitToMaxDrawdownRatio": 0,
      "MaximumEndTradeDrawdown": 0,
      "AverageEndTradeDrawdown": 0,
      "MaximumDrawdownDuration": "string",
      "TotalFees": 0
    },
    "PortfolioStatistics": {
      "RiskFreeRate": 0,
      "AverageWinRate": 0,
      "AverageLossRate": 0,
      "ProfitLossRatio": 0,
      "WinRate": 0,
      "LossRate": 0,
      "Expectancy": 0,
      "CompoundingAnnualReturn": 0,
      "Drawdown": 0,
      "TotalNetProfit": 0,
      "SharpeRatio": 0,
      "ProbabilisticSharpeRatio": 0,
      "Alpha": 0,
      "Beta": 0,
      "AnnualStandardDeviation": 0,
      "AnnualVariance": 0,
      "InformationRatio": 0,
      "TrackingError": 0,
      "TreynorRatio": 0
    },
    "ClosedTrades": [
      {
        "Symbol": {
          "Value": "string",
          "ID": "string",
          "Permtick": "string"
        },
        "EntryTime": "2021-11-26T15:18:27.693Z",
        "EntryPrice": 0,
        "Direction": "Long",
        "Quantity": 0,
        "ExitTime": "2021-11-26T15:18:27.693Z",
        "ExitPrice": 0,
        "ProfitLoss": 0,
        "TotalFees": 0,
        "MAE": 0,
        "MFE": 0,
        "Duration": "string",
        "EndTradeDrawdown": 0
      }
    ]
  },
  "TotalPerformance": {
    "TradeStatistics": {
      "StartDateTime": "2021-11-26T15:18:27.693Z",
      "EndDateTime": "2021-11-26T15:18:27.693Z",
      "TotalNumberOfTrades": 0,
      "NumberOfWinningTrades": 0,
      "NumberOfLosingTrades": 0,
      "TotalProfitLoss": 0,
      "TotalProfit": 0,
      "TotalLoss": 0,
      "LargestProfit": 0,
      "LargestLoss": 0,
      "AverageProfitLoss": 0,
      "AverageProfit": 0,
      "AverageLoss": 0,
      "AverageTradeDuration": "string",
      "AverageWinningTradeDuration": "string",
      "AverageLosingTradeDuration": "string",
      "MedianTradeDuration": "string",
      "MedianWinningTradeDuration": "string",
```

`BacktestResult` Model - Results object class. Results are exhaust from backtest or live algorithms running in LEAN.

Example


```
"MedianLosingTradeDuration": "string",
"MaxConsecutiveWinningTrades": 0,
"MaxConsecutiveLosingTrades": 0,
"ProfitLossRatio": 0,
"WinLossRatio": 0,
"WinRate": 0,
"LossRate": 0,
"AverageMAE": 0,
"AverageMFE": 0,
"LargestMAE": 0,
"LargestMFE": 0,
"MaximumClosedTradeDrawdown": 0,
"MaximumIntraTradeDrawdown": 0,
"ProfitLossStandardDeviation": 0,
"ProfitLossDownsideDeviation": 0,
"ProfitFactor": 0,
"SharpeRatio": 0,
"SortinoRatio": 0,
"ProfitToMaxDrawdownRatio": 0,
"MaximumEndTradeDrawdown": 0,
"AverageEndTradeDrawdown": 0,
"MaximumDrawdownDuration": "string",
"TotalFees": 0
},
"PortfolioStatistics": {
"RiskFreeRate": 0,
"AverageWinRate": 0,
"AverageLossRate": 0,
"ProfitLossRatio": 0,
"WinRate": 0,
"LossRate": 0,
"Expectancy": 0,
"CompoundingAnnualReturn": 0,
"Drawdown": 0,
"TotalNetProfit": 0,
"SharpeRatio": 0,
"ProbabilisticSharpeRatio": 0,
"Alpha": 0,
"Beta": 0,
"AnnualStandardDeviation": 0,
"AnnualVariance": 0,
"InformationRatio": 0,
"TrackingError": 0,
"TreynorRatio": 0
},
"ClosedTrades": [
{
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"EntryTime": "2021-11-26T15:18:27.693Z",
"EntryPrice": 0,
"Direction": "Long",
"Quantity": 0,
"ExitTime": "2021-11-26T15:18:27.693Z",
"ExitPrice": 0,
"ProfitLoss": 0,
"TotalFees": 0,
"MAE": 0,
"MFE": 0,
"Duration": "string",
"EndTradeDrawdown": 0
}
]
},
"AlphaRuntimeStatistics": {
"MeanPopulationScore": {
"UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
"Direction": 0,
"Magnitude": 0,
"IsFinalScore": true
},
"RollingAveragedPopulationScore": {
"UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
"Direction": 0,
"Magnitude": 0,
"IsFinalScore": true
},
"LongCount": "string",
"ShortCount": "string",
"LongShortRatio": 0,
"TotalAccumulatedEstimatedAlphaValue": 0,
"KellyCriterionEstimate": 0,
"KellyCriterionProbabilityValue": 0,
"FitnessScore": 0,
"PortfolioTurnover": 0,
"ReturnOverMaxDrawdown": 0,
"SortinoRatio": 0,
"EstimatedMonthlyAlphaValue": 0,
"TotalInsightsGenerated": "string",
"TotalInsightsClosed": "string",
"TotalInsightsAnalysisCompleted": "string",
"MeanPopulationEstimatedInsightValue": 0
},
"Charts": {
"Name": "string",
"ChartType": "Overlay",
"Series": {
"Name": "string",
"Unit": "string",
"Index": 0,
"Values": [
{
"x": "string",
"y": 0
}
],
"SeriesType": "Line",
"Color": "string",
"ScatterMarkerSymbol": "none"
}
},
"Orders": {
"Id": 0,
"ContingentId": 0,
"BrokerId": [
"string"
],
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"Price": 0,
"PriceCurrency": "string",
"Time": "2021-11-26T15:18:27.693Z",
"CreatedTime": "2021-11-26T15:18:27.693Z",
"LastFillTime": "2021-11-26T15:18:27.693Z",
```

`BacktestResult` Model - Results object class. Results are exhaust from backtest or live algorithms running in LEAN.


```
"LastUpdateTime": "2021-11-26T15:18:27.693Z",
"CanceledTime": "2021-11-26T15:18:27.693Z",
"Quantity": 0,
"Type": "Market",
"Status": "New",
"Tag": "string",
"SecurityType": "Base",
"Direction": "Buy",
"Value": 0,
"OrderSubmissionData": {
"BidPrice": 0,
"AskPrice": 0,
"LastPrice": 0
},
"IsMarketable": true
},
"OrderEvents": [
{
"OrderId": 0,
"Id": 0,
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"UtcTime": "2021-11-26T15:18:27.693Z",
"Status": {
"OrderId": 0,
"Id": 0,
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"UtcTime": "2021-11-26T15:18:27.693Z",
"Status": "New",
"FillPrice": 0,
"FillPriceCurrency": "string",
"FillQuantity": 0,
"Direction": "Buy",
"Message": "string",
"IsAssignment": true,
"StopPrice": 0,
"LimitPrice": 0,
"Quantity": 0
},
"OrderFee": {
"Value": {
"Amount": 0,
"Currency": "string"
}
},
"FillPrice": 0,
"FillPriceCurrency": "string",
"FillQuantity": 0,
"Direction": "Buy",
"Message": "string",
"IsAssignment": true,
"StopPrice": 0,
"LimitPrice": 0,
"Quantity": 0
}
],
"ProfitLoss": "number",
"Statistics": "string",
"RuntimeStatistics": "string",
"ServerStatistics": "string"
}
```


`AlgorithmPerformance` Model - The AlgorithmPerformance class is a wrapper for TradeStatistics and PortfolioStatistics.

TradeStatistics	`TradeStatistics object` A set of statistics calculated from a list of closed trades.
PortfolioStatistics	`PortfolioStatistics object` Represents a set of statistics calculated from equity and benchmark samples.
ClosedTrades	`Trade Array` The algorithm statistics on portfolio.

Example


```
{
"TradeStatistics": {
"StartDateTime": "2021-11-26T15:18:27.693Z",
"EndDateTime": "2021-11-26T15:18:27.693Z",
"TotalNumberOfTrades": 0,
"NumberOfWinningTrades": 0,
"NumberOfLosingTrades": 0,
"TotalProfitLoss": 0,
"TotalProfit": 0,
"TotalLoss": 0,
"LargestProfit": 0,
"LargestLoss": 0,
"AverageProfitLoss": 0,
"AverageProfit": 0,
"AverageLoss": 0,
"AverageTradeDuration": "string",
"AverageWinningTradeDuration": "string",
"AverageLosingTradeDuration": "string",
"MedianTradeDuration": "string",
"MedianWinningTradeDuration": "string",
"MedianLosingTradeDuration": "string",
"MaxConsecutiveWinningTrades": 0,
"MaxConsecutiveLosingTrades": 0,
"ProfitLossRatio": 0,
"WinLossRatio": 0,
"WinRate": 0,
"LossRate": 0,
"AverageMAE": 0,
"AverageMFE": 0,
"LargestMAE": 0,
"LargestMFE": 0,
"MaximumClosedTradeDrawdown": 0,
"MaximumIntraTradeDrawdown": 0,
"ProfitLossStandardDeviation": 0,
"ProfitLossDownsideDeviation": 0,
"ProfitFactor": 0,
"SharpeRatio": 0,
"SortinoRatio": 0,
"ProfitToMaxDrawdownRatio": 0,
"MaximumEndTradeDrawdown": 0,
"AverageEndTradeDrawdown": 0,
"MaximumDrawdownDuration": "string",
"TotalFees": 0
},
"PortfolioStatistics": {
"RiskFreeRate": 0,
"AverageWinRate": 0,
"AverageLossRate": 0,
"ProfitLossRatio": 0,
"WinRate": 0,
"LossRate": 0,
"Expectancy": 0,
```

`AlgorithmPerformance` Model - The AlgorithmPerformance class is a wrapper for TradeStatistics and PortfolioStatistics.


```
"CompoundingAnnualReturn": 0,
"Drawdown": 0,
"TotalNetProfit": 0,
"SharpeRatio": 0,
"ProbabilisticSharpeRatio": 0,
"Alpha": 0,
"Beta": 0,
"AnnualStandardDeviation": 0,
"AnnualVariance": 0,
"InformationRatio": 0,
"TrackingError": 0,
"TreynorRatio": 0
},
"ClosedTrades": [
{
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"EntryTime": "2021-11-26T15:18:27.693Z",
"EntryPrice": 0,
"Direction": "Long",
"Quantity": 0,
"ExitTime": "2021-11-26T15:18:27.693Z",
"ExitPrice": 0,
"ProfitLoss": 0,
"TotalFees": 0,
"MAE": 0,
"MFE": 0,
"Duration": "string",
"EndTradeDrawdown": 0
}
]
}
```


`TradeStatistics` Model - A set of statistics calculated from a list of closed trades.

StartDateTime	`string($date-time)` The entry date/time of the first trade.
EndDateTime	`string($date-time)` The exit date/time of the first trade.
TotalNumberOfTrades	`integer` The total number of trades.
NumberOfWinningTrades	`integer` The total number of winning trades.
NumberOfLosingTrades	`integer` The total number of losing trades.
TotalProfitLoss	`number` The total profit/loss for all trades (as symbol currency).
TotalProfit	`number` The total profit for all winning trades (as symbol currency).
TotalLoss	`number` The total loss for all losing trades (as symbol currency).
LargestProfit	`number` The largest profit in a single trade (as symbol currency).
LargestLoss	`number` The largest loss in a single trade (as symbol currency).
AverageProfitLoss	`number` The average profit/loss (a.k.a. Expectancy or Average Trade) for all trades (as symbol currency).
AverageProfit	`number` The average profit for all winning trades (as symbol currency).
AverageLoss	`number` The average loss for all winning trades (as symbol currency).
AverageTradeDuration	`string` The average duration for all trades.
AverageWinningTradeDuration	`string` The average duration for all winning trades.
AverageLosingTradeDuration	`string` The average duration for all losing trades.
MedianTradeDuration	`string` The median duration for all trades.
MedianWinningTradeDuration	`string` The median duration for all winning trades.
MedianLosingTradeDuration	`string` The median duration for all losing trades.
MaxConsecutiveWinningTrades	`integer` The maximum number of consecutive winning trades.
MaxConsecutiveLosingTrades	`integer` The maximum number of consecutive losing trades.
ProfitLossRatio	`number` The ratio of the average profit per trade to the average loss per trade.
WinLossRatio	`number` The ratio of the number of winning trades to the number of losing trades.
WinRate	`number` The ratio of the number of winning trades to the total number of trades.
LossRate	`number` The ratio of the number of losing trades to the total number of trades.
AverageMAE	`number` The average Maximum Adverse Excursion for all trades.
AverageMFE	`number` The average Maximum Adverse Excursion for all trades.
LargestMAE	`number` The average Maximum Favorable Excursion for all trades.
LargestMFE	`number` The largest Maximum Adverse Excursion in a single trade (as symbol currency).
MaximumClosedTradeDrawdown	`number` The maximum closed-trade drawdown for all trades (as symbol currency).
MaximumIntraTradeDrawdown	`number` The maximum intra-trade drawdown for all trades (as symbol currency).
ProfitLossStandardDeviation	`number` The standard deviation of the profits/losses for all trades (as symbol currency).
ProfitLossDownsideDeviation	`number` The downside deviation of the profits/losses for all trades (as symbol currency).
ProfitFactor	`number` The ratio of the total profit to the total loss.
SharpeRatio	`number` The ratio of the average profit/loss to the standard deviation.
SortinoRatio	`number` The ratio of the average profit/loss to the downside deviation.
ProfitToMaxDrawdownRatio	`number` The ratio of the total profit/loss to the maximum closed trade drawdown.
	`number`

MaximumEndTradeDrawdown	The maximum amount of profit given back by a single trade before exit (as symbol currency).
AverageEndTradeDrawdown	number The average amount of profit given back by all trades before exit (as symbol currency).
MaximumDrawdownDuration	string The maximum amount of time to recover from a drawdown (longest time between new equity highs or peaks).
TotalFees	number The sum of fees for all trades.

`TradeStatistics` Model - A set of statistics calculated from a list of closed trades.

Example


```
{
  "StartDateTime": "2021-11-26T15:18:27.693Z",
  "EndDateTime": "2021-11-26T15:18:27.693Z",
  "TotalNumberOfTrades": 0,
  "NumberOfWinningTrades": 0,
  "NumberOfLosingTrades": 0,
  "TotalProfitLoss": 0,
  "TotalProfit": 0,
  "TotalLoss": 0,
  "LargestProfit": 0,
  "LargestLoss": 0,
  "AverageProfitLoss": 0,
  "AverageProfit": 0,
  "AverageLoss": 0,
  "AverageTradeDuration": "string",
  "AverageWinningTradeDuration": "string",
  "AverageLosingTradeDuration": "string",
  "MedianTradeDuration": "string",
  "MedianWinningTradeDuration": "string",
  "MedianLosingTradeDuration": "string",
  "MaxConsecutiveWinningTrades": 0,
  "MaxConsecutiveLosingTrades": 0,
  "ProfitLossRatio": 0,
  "WinLossRatio": 0,
  "WinRate": 0,
  "LossRate": 0,
  "AverageMAE": 0,
  "AverageMFE": 0,
  "LargestMAE": 0,
  "LargestMFE": 0,
  "MaximumClosedTradeDrawdown": 0,
  "MaximumIntraTradeDrawdown": 0,
  "ProfitLossStandardDeviation": 0,
  "ProfitLossDownsideDeviation": 0,
  "ProfitFactor": 0,
  "SharpeRatio": 0,
  "SortinoRatio": 0,
  "ProfitToMaxDrawdownRatio": 0,
  "MaximumEndTradeDrawdown": 0,
  "AverageEndTradeDrawdown": 0,
  "MaximumDrawdownDuration": "string",
  "TotalFees": 0
}
```


`PortfolioStatistics` Model - Represents a set of statistics calculated from equity and benchmark samples.

RiskFreeRate	number The current defined risk free annual return rate.
AverageWinRate	number The average rate of return for winning trades.
AverageLossRate	number The average rate of return for losing trades.
ProfitLossRatio	number The ratio of the average win rate to the average loss rate.
WinRate	number The ratio of the number of winning trades to the total number of trades.
LossRate	number The ratio of the number of losing trades to the total number of trades.
Expectancy	number The expected value of the rate of return.
CompoundingAnnualReturn	number Annual compounded returns statistic based on the final-starting capital and years.
Drawdown	number Drawdown maximum percentage.
TotalNetProfit	number The total net profit percentage.
SharpeRatio	number Sharpe ratio with respect to risk free rate: measures excess of return per unit of risk.
ProbabilisticSharpeRatio	number Probabilistic Sharpe Ratio is a probability measure associated with the Sharpe ratio. It informs us of the probability that the estimated Sharpe ratio is greater than a chosen benchmark.
Alpha	number Algorithm "Alpha" statistic - abnormal returns over the risk free rate and the relationshio (beta) with the benchmark returns.
Beta	number Algorithm beta statistic - the covariance between the algorithm and benchmark performance, divided by benchmark variance.
AnnualStandardDeviation	number Annualized standard deviation.
AnnualVariance	number Annualized variance statistic calculation using the daily performance variance and trading days per year.
InformationRatio	number Information ratio - risk adjusted return.
TrackingError	number Tracking error volatility (TEV) statistic - a measure of how closely a portfolio follows the index to which it is benchmarked.
TreynorRatio	number Treynor ratio statistic is a measurement of the returns earned in excess of that which could have been earned on an investment that has no diversifiable risk.

Example


```
{
  "RiskFreeRate": 0,
  "AverageWinRate": 0,
  "AverageLossRate": 0,
  "ProfitLossRatio": 0,
  "WinRate": 0,
  "LossRate": 0,
  "Expectancy": 0,
  "CompoundingAnnualReturn": 0,
  "Drawdown": 0,
  "TotalNetProfit": 0,
  "SharpeRatio": 0,
  "ProbabilisticSharpeRatio": 0,
  "Alpha": 0,
  "Beta": 0,
  "AnnualStandardDeviation": 0,
  "AnnualVariance": 0,
  "InformationRatio": 0,
  "TrackingError": 0,
  "TreynorRatio": 0
}
```


`Trade` Model - Represents a closed trade.

Symbol	Symbol object Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
EntryTime	string($date-time) The date and time the trade was opened.

`Trade` Model - Represents a closed trade.

EntryPrice	number The price at which the trade was opened (or the average price if multiple entries).
Direction	string Enum Direction of a trade. Options : ['Long', 'Short']
Quantity	number The total unsigned quantity of the trade.
ExitTime	string($date-time) The date and time the trade was closed.
ExitPrice	number The price at which the trade was closed (or the average price if multiple exits).
ProfitLoss	number The gross profit/loss of the trade (as account currency).
TotalFees	number The total fees associated with the trade (always positive value) (as account currency).
MAE	number The Maximum Adverse Excursion (as account currency).
MFE	number The Maximum Favorable Excursion (as account currency).
Duration	string The duration of the trade.
EndTradeDrawdown	number The amount of profit given back before the trade was closed.

Example	{
	"Symbol": {
	"Value": "string",
	"ID": "string",
	"Permtick": "string"
	},
	"EntryTime": "2021-11-26T15:18:27.693Z",
	"EntryPrice": 0,
	"Direction": "Long",
	"Quantity": 0,
	"ExitTime": "2021-11-26T15:18:27.693Z",
	"ExitPrice": 0,
	"ProfitLoss": 0,
	"TotalFees": 0,
	"MAE": 0,
	"MFE": 0,
	"Duration": "string",
	"EndTradeDrawdown": 0
	}

`Symbol` Model - Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.

Value	string The current symbol for this ticker.
ID	string The security identifier for this symbol.
Permtick	string The current symbol for this ticker.

Example	{
	"Value": "string",
	"ID": "string",
	"Permtick": "string"
	}

`TradeDirection` Model - Direction of a trade.

TradeDirection	string Enum Direction of a trade. Options : ['Long', 'Short']

Example	{
	"TradeDirection": "Long"
	}

`AlphaRuntimeStatistics` Model - Contains insight population run time statistics.

MeanPopulationScore	InsightScore object Defines the scores given to a particular insight.
RollingAveragedPopulationScore	InsightScore object Defines the scores given to a particular insight.
LongCount	string Gets the total number of insights with an up direction.
ShortCount	string Gets the total number of insights with a down direction.
LongShortRatio	number The ratio of InsightDirection.Up over InsightDirection.Down.
TotalAccumulatedEstimatedAlphaValue	number The total accumulated estimated value of trading all insights.
KellyCriterionEstimate	number Score of the strategy's insights predictive power.
KellyCriterionProbabilityValue	number The p-value or probability value of the KellyCriterionEstimate.
FitnessScore	number Score of the strategy's performance, and suitability for the Alpha Stream Market.
PortfolioTurnover	number Measurement of the strategies trading activity with respect to the portfolio value. Calculated as the sales volume with respect to the average total portfolio value.
ReturnOverMaxDrawdown	number Provides a risk adjusted way to factor in the returns and drawdown of the strategy. It is calculated by dividing the Portfolio Annualized Return by the Maximum Drawdown seen during the backtest.
SortinoRatio	number Gives a relative picture of the strategy volatility. It is calculated by taking a portfolio's annualized rate of return and subtracting the risk free rate of return.
EstimatedMonthlyAlphaValue	number Suggested Value of the Alpha On A Monthly Basis For Licensing.
TotalInsightsGenerated	string The total number of insight signals generated by the algorithm.
TotalInsightsClosed	string The total number of insight signals generated by the algorithm.
TotalInsightsAnalysisCompleted	string The total number of insight signals generated by the algorithm.
MeanPopulationEstimatedInsightValue	number Gets the mean estimated insight value.

Example	{
	"MeanPopulationScore": {
	"UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
	"Direction": 0,
	"Magnitude": 0,
	"IsFinalScore": true
	},
	"RollingAveragedPopulationScore": {
	"UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
	"Direction": 0,
	"Magnitude": 0,
	"IsFinalScore": true
	},
	"LongCount": "string",
	"ShortCount": "string",
	"LongShortRatio": 0,

`AlphaRuntimeStatistics` Model - Contains insight population run time statistics.

```
"TotalAccumulatedEstimatedAlphaValue": 0,
"KellyCriterionEstimate": 0,
"KellyCriterionProbabilityValue": 0,
"FitnessScore": 0,
"PortfolioTurnover": 0,
"ReturnOverMaxDrawdown": 0,
"SortinoRatio": 0,
"EstimatedMonthlyAlphaValue": 0,
"TotalInsightsGenerated": "string",
"TotalInsightsClosed": "string",
"TotalInsightsAnalysisCompleted": "string",
"MeanPopulationEstimatedInsightValue": 0
}
```

`InsightScore` Model - Defines the scores given to a particular insight

UpdatedTimeUtc	`string($date-time)`	The time these scores were last updated.
Direction	`number`	The direction score.
Magnitude	`number`	The magnitude score.
IsFinalScore	`boolean`	Is the insight past its expiry time and score can be finalized.

Example

```
{
"UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
"Direction": 0,
"Magnitude": 0,
"IsFinalScore": true
}
```

`Chart` Model - Single Parent Chart Object for Custom Charting.

Name	`string`	Name of the Chart.
ChartType	`string Enum`	Type of the Chart, Overlayed or Stacked. Options : ['Overlay', 'Stacked']
Series	`Series object`	List of Series Objects for this Chart.

Example

```
{
"Name": "string",
"ChartType": "Overlay",
"Series": {
"Name": "string",
"Unit": "string",
"Index": 0,
"Values": [
{
"x": "string",
"y": 0
}
],
"SeriesType": "Line",
"Color": "string",
"ScatterMarkerSymbol": "none"
}
}
```

`Series` Model - Chart Series Object - Series data and properties for a chart.

Name	`string`	Name of the series.
Unit	`string`	Axis for the chart series.
Index	`integer`	Index/position of the series on the chart.
Values	`ChartPoint Array`	Values for the series plot. These values are assumed to be in ascending time order (first points earliest, last points latest).
SeriesType	`string Enum`	Chart type for the series. Options : ['Line', 'Scatter', 'Candle', 'Bar', 'Flag', 'StackedArea', 'Pie', 'Treemap']
Color	`string`	Color the series.
ScatterMarkerSymbol	`string Enum`	Shape or symbol for the marker in a scatter plot. Options : ['none', 'circle', 'square', 'diamond', 'triangle', 'triangle-down']

Example

```
{
"Name": "string",
"Unit": "string",
"Index": 0,
"Values": [
{
"x": "string",
"y": 0
}
],
"SeriesType": "Line",
"Color": "string",
"ScatterMarkerSymbol": "none"
}
```

`ChartPoint` Model - Location on a chart containing the X-Y location

x	`string`	Time of this chart point: lower case for javascript encoding simplicty.
y	`number`	Value of this chart point: lower case for javascript encoding simplicty.

Example

```
{
"x": "string",
"y": 0
}
```

`Order` Model - Order struct for placing new trade.

Id	`integer`	Order ID.
ContingentId	`integer`	Order Id to process before processing this order.
BrokerId	`string Array`	Brokerage Id for this order for when the brokerage splits orders into multiple pieces.
Symbol	`Symbol object`	Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
Price	`number`	Price of the Order.
PriceCurrency	`string`	Currency for the order price.
Time	`string($date-time)`	Gets the utc time the order was created.
CreatedTime	`string($date-time)`	Gets the utc time this order was created. Alias for Time.
LastFillTime	`string($date-time)`	Gets the utc time the last fill was received, or null if no fills have been received.
LastUpdateTime	`string($date-time)`	Gets the utc time this order was last updated, or null if the order has not been updated.

`Order` Model - Order struct for placing new trade.

CanceledTime	`string($date-time)`	Gets the utc time this order was canceled, or null if the order was not canceled.
Quantity	`number`	Number of shares to execute.
Type	`string Enum`	Order type. Options : ['Market', 'Limit', 'StopMarket', 'StopLimit', 'MarketOnOpen', 'MarketOnClose', 'OptionExercise']
Status	`string Enum`	Status of the Order. Options : ['New', 'Submitted', 'PartiallyFilled', 'Filled', 'Canceled', 'None', 'Invalid', 'CancelPending', 'UpdateSubmitted']
Tag	`string`	Tag the order with some custom data.
SecurityType	`string Enum`	Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']
Direction	`string Enum`	Direction of the order. Options : ['Buy', 'Sell', 'Hold']
Value	`number`	Gets the executed value of this order. If the order has not yet filled, then this will return zero.
OrderSubmissionData	`OrderSubmissionData object`	Stores time and price information available at the time an order was submitted.
IsMarketable	`boolean`	Returns true if the order is a marketable order.

Example


```
{
  "Id": 0,
  "ContingentId": 0,
  "BrokerId": [
    "string"
  ],
  "Symbol": {
    "Value": "string",
    "ID": "string",
    "Permtick": "string"
  },
  "Price": 0,
  "PriceCurrency": "string",
  "Time": "2021-11-26T15:18:27.693Z",
  "CreatedTime": "2021-11-26T15:18:27.693Z",
  "LastFillTime": "2021-11-26T15:18:27.693Z",
  "LastUpdateTime": "2021-11-26T15:18:27.693Z",
  "CanceledTime": "2021-11-26T15:18:27.693Z",
  "Quantity": 0,
  "Type": "Market",
  "Status": "New",
  "Tag": "string",
  "SecurityType": "Base",
  "Direction": "Buy",
  "Value": 0,
  "OrderSubmissionData": {
    "BidPrice": 0,
    "AskPrice": 0,
    "LastPrice": 0
  },
  "IsMarketable": true
}
```


`SecurityType` Model - Type of tradable security / underlying asset.

SecurityType	`string Enum`	Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']

Example


```
{
  "SecurityType": "Base"
}
```


`OrderDirection` Model - Direction of the order.

OrderDirection	`string Enum`	Direction of the order. Options : ['Buy', 'Sell', 'Hold']

Example


```
{
  "OrderDirection": "Buy"
}
```


`OrderSubmissionData` Model - Stores time and price information available at the time an order was submitted.

BidPrice	`number`	The bid price at an order submission time.
AskPrice	`number`	The ask price at an order submission time.
LastPrice	`number`	The current price at an order submission time.

Example


```
{
  "BidPrice": 0,
  "AskPrice": 0,
  "LastPrice": 0
}
```


`OrderEvent` Model - Change in an order state applied to user algorithm portfolio

OrderId	`integer`	Id of the order this event comes from.
Id	`integer`	The unique order event Id for each order.
Symbol	`Symbol object`	Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
UtcTime	`string($date-time)`	The date and time of this event (UTC).
Status	`OrderStatus object`	Messaging class signifying a change in an order state and record the change in the users algorithm portfolio.
OrderFee	`OrderFee object`	The order fee associated with the specified order.
FillPrice	`number`	Fill price information about the order.
FillPriceCurrency	`string`	Currency for the fill price.
FillQuantity	`number`	Number of shares of the order that was filled in this event.
Direction	`string Enum`	Direction of the order. Options : ['Buy', 'Sell', 'Hold']
Message	`string`	Any message from the exchange.
IsAssignment	`boolean`	True if the order event is an assignment.
StopPrice	`number`	The current stop price.
LimitPrice	`number`	The current limit price.
Quantity	`number`	The current order quantity.


```
{
  "OrderId": 0,
  "Id": 0,
```

`OrderEvent` Model - Change in an order state applied to user algorithm portfolio

Example


```
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"UtcTime": "2021-11-26T15:18:27.693Z",
"Status": {
"OrderId": 0,
"Id": 0,
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"UtcTime": "2021-11-26T15:18:27.693Z",
"Status": "New",
"FillPrice": 0,
"FillPriceCurrency": "string",
"FillQuantity": 0,
"Direction": "Buy",
"Message": "string",
"IsAssignment": true,
"StopPrice": 0,
"LimitPrice": 0,
"Quantity": 0
},
"OrderFee": {
"Value": {
"Amount": 0,
"Currency": "string"
}
},
"FillPrice": 0,
"FillPriceCurrency": "string",
"FillQuantity": 0,
"Direction": "Buy",
"Message": "string",
"IsAssignment": true,
"StopPrice": 0,
"LimitPrice": 0,
"Quantity": 0
}
```


`OrderStatus` Model - Messaging class signifying a change in an order state and record the change in the users algorithm portfolio.

OrderId	integer	Id of the order this event comes from.
Id	integer	The unique order event Id for this order.
Symbol	Symbol object	Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
UtcTime	string($date-time)	The date and time of this event.
Status	string Enum	Status of the Order. Options : ['New', 'Submitted', 'PartiallyFilled', 'Filled', 'Canceled', 'None', 'Invalid', 'CancelPending', 'UpdateSubmitted']
FillPrice	number	Fill price information about the order.
FillPriceCurrency	string	Currency for the fill price.
FillQuantity	number	Number of shares of the order that was filled in this event.
Direction	string Enum	Direction of the order. Options : ['Buy', 'Sell', 'Hold']
Message	string	Any message from the exchange.
IsAssignment	boolean	Order event is an allocation of trades from ITM option assignment.
StopPrice	number	The current stop price.
LimitPrice	number	The current limit price.
Quantity	number	The current order quantity.

Example


```
{
"OrderId": 0,
"Id": 0,
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"UtcTime": "2021-11-26T15:18:27.693Z",
"Status": "New",
"FillPrice": 0,
"FillPriceCurrency": "string",
"FillQuantity": 0,
"Direction": "Buy",
"Message": "string",
"IsAssignment": true,
"StopPrice": 0,
"LimitPrice": 0,
"Quantity": 0
}
```


`Status` Model - Status of the Order.

Status	string Enum	Status of the Order. Options : ['New', 'Submitted', 'PartiallyFilled', 'Filled', 'Canceled', 'None', 'Invalid', 'CancelPending', 'UpdateSubmitted']

Example


```
{
"Status": "New"
}
```


`OrderFee` Model - The order fee associated with the specified order.

Value	CashAmount object	Represents a cash amount which can be converted to account currency using a currency converter.

Example


```
{
"Value": {
"Amount": 0,
"Currency": "string"
}
}
```


`CashAmount` Model - Represents a cash amount which can be converted to account currency using a currency converter.

Amount	number	The amount of cash.
Currency	string	The currency in which the cash amount is denominated.

Example


```
{
"Amount": 0,
"Currency": "string"
}
```

401 Authentication Error

`UnauthorizedError` **Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.**

www_authenticate	`string`	Header

14.5.2 Read Backtest

The QuantConnect REST API lets you read backtest results from our cloud servers through URL endpoints.

Backtest Statistics

Portfolio

Orders

14.5.2.1 Backtest Statistics

Introduction

If a backtest Id is provided, read out that backtest from the project, otherwise list all the backtests for the project.

Request

Fetch the results for the project Id and backtest Id provided. The `/backtests/read` API accepts requests in the following format:

`ReadBacktestRequest` Model - Request to read a single backtest from a project.

projectId	`integer` Id of the project from which to read one or multiple backtests.
backtestId	`string` When provided, specific backtest Id to read.

Example	<pre>{
	"projectId": 0,
	"backtestId": "string"
	}</pre>

Responses

The `/backtests/read` API provides a response in the following format:

200 Success

`BacktestResponse` Model - Packet container for carrying Backtest results.

name	`string` Name of the backtest.
note	`string` Note on the backtest attached by the user.
backtestId	`string` Assigned backtest Id.
completed	`boolean` Boolean true when the backtest is completed.
progress	`number` Progress of the backtest in percent 0-1.
result	`BacktestResult object` Results object class. Results are exhaust from backtest or live algorithms running in LEAN.
error	`string` Backtest error message.
stacktrace	`string` Backtest error stacktrace.
created	`string($date-time)` Backtest creation date and time.
success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.


```
{
"name": "string",
"note": "string",
"backtestId": "string",
"completed": true,
"progress": 0,
"result": {
"RollingWindow": {
"TradeStatistics": {
"StartDateTime": "2021-11-26T15:18:27.693Z",
"EndDateTime": "2021-11-26T15:18:27.693Z",
"TotalNumberOfTrades": 0,
"NumberOfWinningTrades": 0,
"NumberOfLosingTrades": 0,
"TotalProfitLoss": 0,
"TotalProfit": 0,
"TotalLoss": 0,
"LargestProfit": 0,
"LargestLoss": 0,
"AverageProfitLoss": 0,
"AverageProfit": 0,
"AverageLoss": 0,
"AverageTradeDuration": "string",
"AverageWinningTradeDuration": "string",
"AverageLosingTradeDuration": "string",
"MedianTradeDuration": "string",
"MedianWinningTradeDuration": "string",
"MedianLosingTradeDuration": "string",
"MaxConsecutiveWinningTrades": 0,
"MaxConsecutiveLosingTrades": 0,
"ProfitLossRatio": 0,
"WinLossRatio": 0,
"WinRate": 0,
"LossRate": 0,
"AverageMAE": 0,
"AverageMFE": 0,
"LargestMAE": 0,
"LargestMFE": 0,
"MaximumClosedTradeDrawdown": 0,
"MaximumIntraTradeDrawdown": 0,
"ProfitLossStandardDeviation": 0,
"ProfitLossDownsideDeviation": 0,
"ProfitFactor": 0,
"SharpeRatio": 0,
"SortinoRatio": 0,
"ProfitToMaxDrawdownRatio": 0,
"MaximumEndTradeDrawdown": 0,
"AverageEndTradeDrawdown": 0,
"MaximumDrawdownDuration": "string",
"TotalFees": 0
},
"PortfolioStatistics": {
"RiskFreeRate": 0,
"AverageWinRate": 0,
"AverageLossRate": 0,
"ProfitLossRatio": 0,
"WinRate": 0,
"LossRate": 0,
"Expectancy": 0,
"CompoundingAnnualReturn": 0,
"Drawdown": 0,
"TotalNetProfit": 0,
"SharpeRatio": 0,
"ProbabilisticSharpeRatio": 0,
"Alpha": 0,
"Beta": 0,
"AnnualStandardDeviation": 0,
"AnnualVariance": 0,
"InformationRatio": 0,
```

`BacktestResponse` Model - Packet container for carrying Backtest results.

Example


```
                                                "TrackingError": 0,
                                                "TreynorRatio": 0
                                            },
                                        "ClosedTrades": [
                                            {
                                                "Symbol": {
                                            "Value": "string",
                                                "ID": "string",
                                            "Permtick": "string"
                                            },
                                    "EntryTime": "2021-11-26T15:18:27.693Z",
                                            "EntryPrice": 0,
                                        "Direction": "Long",
                                            "Quantity": 0,
                                     "ExitTime": "2021-11-26T15:18:27.693Z",
                                            "ExitPrice": 0,
                                        "ProfitLoss": 0,
                                            "TotalFees": 0,
                                                "MAE": 0,
                                                "MFE": 0,
                                        "Duration": "string",
                                        "EndTradeDrawdown": 0
                                            }
                                            ]
                                        },
                                    "TotalPerformance": {
                                     "TradeStatistics": {
                              "StartDateTime": "2021-11-26T15:18:27.693Z",
                                "EndDateTime": "2021-11-26T15:18:27.693Z",
                                   "TotalNumberOfTrades": 0,
                                "NumberOfWinningTrades": 0,
                                 "NumberOfLosingTrades": 0,
                                      "TotalProfitLoss": 0,
                                         "TotalProfit": 0,
                                           "TotalLoss": 0,
                                       "LargestProfit": 0,
                                        "LargestLoss": 0,
                                   "AverageProfitLoss": 0,
                                       "AverageProfit": 0,
                                        "AverageLoss": 0,
                             "AverageTradeDuration": "string",
                           "AverageWinningTradeDuration": "string",
                            "AverageLosingTradeDuration": "string",
                                 "MedianTradeDuration": "string",
                           "MedianWinningTradeDuration": "string",
                            "MedianLosingTradeDuration": "string",
                                 "MaxConsecutiveWinningTrades": 0,
                                  "MaxConsecutiveLosingTrades": 0,
                                      "ProfitLossRatio": 0,
                                        "WinLossRatio": 0,
                                             "WinRate": 0,
                                            "LossRate": 0,
                                         "AverageMAE": 0,
                                         "AverageMFE": 0,
                                         "LargestMAE": 0,
                                         "LargestMFE": 0,
                                "MaximumClosedTradeDrawdown": 0,
                                 "MaximumIntraTradeDrawdown": 0,
                               "ProfitLossStandardDeviation": 0,
                               "ProfitLossDownsideDeviation": 0,
                                          "ProfitFactor": 0,
                                           "SharpeRatio": 0,
                                          "SortinoRatio": 0,
                                "ProfitToMaxDrawdownRatio": 0,
                                 "MaximumEndTradeDrawdown": 0,
                                 "AverageEndTradeDrawdown": 0,
                            "MaximumDrawdownDuration": "string",
                                              "TotalFees": 0
                                            },
                                  "PortfolioStatistics": {
                                       "RiskFreeRate": 0,
                                     "AverageWinRate": 0,
                                    "AverageLossRate": 0,
                                    "ProfitLossRatio": 0,
                                             "WinRate": 0,
                                            "LossRate": 0,
                                          "Expectancy": 0,
                                "CompoundingAnnualReturn": 0,
                                           "Drawdown": 0,
                                      "TotalNetProfit": 0,
                                         "SharpeRatio": 0,
                                "ProbabilisticSharpeRatio": 0,
                                               "Alpha": 0,
                                                "Beta": 0,
                                "AnnualStandardDeviation": 0,
                                       "AnnualVariance": 0,
                                     "InformationRatio": 0,
                                        "TrackingError": 0,
                                         "TreynorRatio": 0
                                            },
                                        "ClosedTrades": [
                                            {
                                                "Symbol": {
                                            "Value": "string",
                                                "ID": "string",
                                            "Permtick": "string"
                                            },
                                    "EntryTime": "2021-11-26T15:18:27.693Z",
                                            "EntryPrice": 0,
                                        "Direction": "Long",
                                            "Quantity": 0,
                                     "ExitTime": "2021-11-26T15:18:27.693Z",
                                            "ExitPrice": 0,
                                        "ProfitLoss": 0,
                                            "TotalFees": 0,
                                                "MAE": 0,
                                                "MFE": 0,
                                        "Duration": "string",
                                        "EndTradeDrawdown": 0
                                            }
                                            ]
                                        },
                                "AlphaRuntimeStatistics": {
                                   "MeanPopulationScore": {
                              "UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
                                           "Direction": 0,
                                           "Magnitude": 0,
                                      "IsFinalScore": true
                                    },
                            "RollingAveragedPopulationScore": {
                              "UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
                                           "Direction": 0,
                                           "Magnitude": 0,
                                      "IsFinalScore": true
                                    },
                                 "LongCount": "string",
                                "ShortCount": "string",
                                   "LongShortRatio": 0,
                          "TotalAccumulatedEstimatedAlphaValue": 0,
                                "KellyCriterionEstimate": 0,
```

`BacktestResponse` Model - Packet container for carrying Backtest results.


```
                "KellyCriterionProbabilityValue": 0,
                                  "FitnessScore": 0,
                              "PortfolioTurnover": 0,
                            "ReturnOverMaxDrawdown": 0,
                                  "SortinoRatio": 0,
                          "EstimatedMonthlyAlphaValue": 0,
                       "TotalInsightsGenerated": "string",
                         "TotalInsightsClosed": "string",
                "TotalInsightsAnalysisCompleted": "string",
                   "MeanPopulationEstimatedInsightValue": 0
                                                        },
                                               "Charts": {
                                            "Name": "string",
                                         "ChartType": "Overlay",
                                               "Series": {
                                            "Name": "string",
                                            "Unit": "string",
                                                  "Index": 0,
                                                  "Values": [
                                                            {
                                               "x": "string",
                                                      "y": 0
                                                            }
                                                          ],
                                         "SeriesType": "Line",
                                            "Color": "string",
                                  "ScatterMarkerSymbol": "none"
                                                        }
                                                      },
                                             "Orders": {
                                                "Id": 0,
                                          "ContingentId": 0,
                                               "BrokerId": [
                                                    "string"
                                                          ],
                                             "Symbol": {
                                          "Value": "string",
                                             "ID": "string",
                                        "Permtick": "string"
                                                        },
                                             "Price": 0,
                                    "PriceCurrency": "string",
                            "Time": "2021-11-26T15:18:27.693Z",
                       "CreatedTime": "2021-11-26T15:18:27.693Z",
                     "LastFillTime": "2021-11-26T15:18:27.693Z",
                   "LastUpdateTime": "2021-11-26T15:18:27.693Z",
                     "CanceledTime": "2021-11-26T15:18:27.693Z",
                                          "Quantity": 0,
                                        "Type": "Market",
                                         "Status": "New",
                                         "Tag": "string",
                                    "SecurityType": "Base",
                                        "Direction": "Buy",
                                             "Value": 0,
                                  "OrderSubmissionData": {
                                             "BidPrice": 0,
                                             "AskPrice": 0,
                                             "LastPrice": 0
                                                        },
                                     "IsMarketable": true
                                                      },
                                         "OrderEvents": [
                                                        {
                                           "OrderId": 0,
                                               "Id": 0,
                                            "Symbol": {
                                          "Value": "string",
                                             "ID": "string",
                                        "Permtick": "string"
                                                        },
                                "UtcTime": "2021-11-26T15:18:27.693Z",
                                             "Status": {
                                           "OrderId": 0,
                                               "Id": 0,
                                            "Symbol": {
                                          "Value": "string",
                                             "ID": "string",
                                        "Permtick": "string"
                                                        },
                                "UtcTime": "2021-11-26T15:18:27.693Z",
                                          "Status": "New",
                                           "FillPrice": 0,
                                 "FillPriceCurrency": "string",
                                           "FillQuantity": 0,
                                         "Direction": "Buy",
                                        "Message": "string",
                                      "IsAssignment": true,
                                           "StopPrice": 0,
                                        "LimitPrice": 0,
                                           "Quantity": 0
                                                        },
                                           "OrderFee": {
                                              "Value": {
                                            "Amount": 0,
                                         "Currency": "string"
                                                        }
                                                      },
                                           "FillPrice": 0,
                                 "FillPriceCurrency": "string",
                                           "FillQuantity": 0,
                                         "Direction": "Buy",
                                        "Message": "string",
                                      "IsAssignment": true,
                                           "StopPrice": 0,
                                        "LimitPrice": 0,
                                           "Quantity": 0
                                                        }
                                                      ],
                                   "ProfitLoss": "number",
                                   "Statistics": "string",
                                "RuntimeStatistics": "string",
                                  "ServerStatistics": "string"
                                                      },
                                           "error": "string",
                                      "stacktrace": "string",
                            "created": "2021-11-26T15:18:27.693Z",
                                          "success": true,
                                              "errors": [
                                                 "string"
                                                          ]
                                                        }
```


`BacktestList` Model - Collection container for a list of Backtest objects for a project.

backtests	`BacktestResponse Array`	Array of BacktestResponse objects.
success	`boolean`	Indicate if the API request was successful.
	`string Array`	

errors — List of errors with that API call

`BacktestList` Model - Collection container for a list of Backtest objects for a project.


```
{
  "backtests": [
    {
      "name": "string",
      "note": "string",
      "backtestId": "string",
      "completed": true,
      "progress": 0,
      "result": {
        "RollingWindow": {
          "TradeStatistics": {
            "StartDateTime": "2021-11-26T15:18:27.693Z",
            "EndDateTime": "2021-11-26T15:18:27.693Z",
            "TotalNumberOfTrades": 0,
            "NumberOfWinningTrades": 0,
            "NumberOfLosingTrades": 0,
            "TotalProfitLoss": 0,
            "TotalProfit": 0,
            "TotalLoss": 0,
            "LargestProfit": 0,
            "LargestLoss": 0,
            "AverageProfitLoss": 0,
            "AverageProfit": 0,
            "AverageLoss": 0,
            "AverageTradeDuration": "string",
            "AverageWinningTradeDuration": "string",
            "AverageLosingTradeDuration": "string",
            "MedianTradeDuration": "string",
            "MedianWinningTradeDuration": "string",
            "MedianLosingTradeDuration": "string",
            "MaxConsecutiveWinningTrades": 0,
            "MaxConsecutiveLosingTrades": 0,
            "ProfitLossRatio": 0,
            "WinLossRatio": 0,
            "WinRate": 0,
            "LossRate": 0,
            "AverageMAE": 0,
            "AverageMFE": 0,
            "LargestMAE": 0,
            "LargestMFE": 0,
            "MaximumClosedTradeDrawdown": 0,
            "MaximumIntraTradeDrawdown": 0,
            "ProfitLossStandardDeviation": 0,
            "ProfitLossDownsideDeviation": 0,
            "ProfitFactor": 0,
            "SharpeRatio": 0,
            "SortinoRatio": 0,
            "ProfitToMaxDrawdownRatio": 0,
            "MaximumEndTradeDrawdown": 0,
            "AverageEndTradeDrawdown": 0,
            "MaximumDrawdownDuration": "string",
            "TotalFees": 0
          },
          "PortfolioStatistics": {
            "RiskFreeRate": 0,
            "AverageWinRate": 0,
            "AverageLossRate": 0,
            "ProfitLossRatio": 0,
            "WinRate": 0,
            "LossRate": 0,
            "Expectancy": 0,
            "CompoundingAnnualReturn": 0,
            "Drawdown": 0,
            "TotalNetProfit": 0,
            "SharpeRatio": 0,
            "ProbabilisticSharpeRatio": 0,
            "Alpha": 0,
            "Beta": 0,
            "AnnualStandardDeviation": 0,
            "AnnualVariance": 0,
            "InformationRatio": 0,
            "TrackingError": 0,
            "TreynorRatio": 0
          },
          "ClosedTrades": [
            {
              "Symbol": {
                "Value": "string",
                "ID": "string",
                "Permtick": "string"
              },
              "EntryTime": "2021-11-26T15:18:27.693Z",
              "EntryPrice": 0,
              "Direction": "Long",
              "Quantity": 0,
              "ExitTime": "2021-11-26T15:18:27.693Z",
              "ExitPrice": 0,
              "ProfitLoss": 0,
              "TotalFees": 0,
              "MAE": 0,
              "MFE": 0,
              "Duration": "string",
              "EndTradeDrawdown": 0
            }
          ]
        },
        "TotalPerformance": {
          "TradeStatistics": {
            "StartDateTime": "2021-11-26T15:18:27.693Z",
            "EndDateTime": "2021-11-26T15:18:27.693Z",
            "TotalNumberOfTrades": 0,
            "NumberOfWinningTrades": 0,
            "NumberOfLosingTrades": 0,
            "TotalProfitLoss": 0,
            "TotalProfit": 0,
            "TotalLoss": 0,
            "LargestProfit": 0,
            "LargestLoss": 0,
            "AverageProfitLoss": 0,
            "AverageProfit": 0,
            "AverageLoss": 0,
            "AverageTradeDuration": "string",
            "AverageWinningTradeDuration": "string",
            "AverageLosingTradeDuration": "string",
            "MedianTradeDuration": "string",
            "MedianWinningTradeDuration": "string",
            "MedianLosingTradeDuration": "string",
            "MaxConsecutiveWinningTrades": 0,
            "MaxConsecutiveLosingTrades": 0,
            "ProfitLossRatio": 0,
            "WinLossRatio": 0,
            "WinRate": 0,
            "LossRate": 0,
            "AverageMAE": 0,
            "AverageMFE": 0,
            "LargestMAE": 0,
            "LargestMFE": 0,
            "MaximumClosedTradeDrawdown": 0,
            "MaximumIntraTradeDrawdown": 0,
            "ProfitLossStandardDeviation": 0,
```

BacktestList Model - Collection container for a list of Backtest objects for a project.

Example


```
                "ProfitLossDownsideDeviation": 0,
                                "ProfitFactor": 0,
                                "SharpeRatio": 0,
                                "SortinoRatio": 0,
                        "ProfitToMaxDrawdownRatio": 0,
                         "MaximumEndTradeDrawdown": 0,
                         "AverageEndTradeDrawdown": 0,
                  "MaximumDrawdownDuration": "string",
                                    "TotalFees": 0
                                                },
                            "PortfolioStatistics": {
                                  "RiskFreeRate": 0,
                                "AverageWinRate": 0,
                               "AverageLossRate": 0,
                               "ProfitLossRatio": 0,
                                        "WinRate": 0,
                                       "LossRate": 0,
                                     "Expectancy": 0,
                         "CompoundingAnnualReturn": 0,
                                      "Drawdown": 0,
                                "TotalNetProfit": 0,
                                   "SharpeRatio": 0,
                        "ProbabilisticSharpeRatio": 0,
                                         "Alpha": 0,
                                          "Beta": 0,
                         "AnnualStandardDeviation": 0,
                                "AnnualVariance": 0,
                              "InformationRatio": 0,
                                 "TrackingError": 0,
                                   "TreynorRatio": 0
                                                },
                                   "ClosedTrades": [
                                                 {
                                         "Symbol": {
                                  "Value": "string",
                                     "ID": "string",
                                "Permtick": "string"
                                                },
            "EntryTime": "2021-11-26T15:18:27.693Z",
                                    "EntryPrice": 0,
                                "Direction": "Long",
                                      "Quantity": 0,
             "ExitTime": "2021-11-26T15:18:27.693Z",
                                     "ExitPrice": 0,
                                    "ProfitLoss": 0,
                                     "TotalFees": 0,
                                           "MAE": 0,
                                           "MFE": 0,
                               "Duration": "string",
                               "EndTradeDrawdown": 0
                                                 }
                                                 ]
                                                },
                         "AlphaRuntimeStatistics": {
                            "MeanPopulationScore": {
        "UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
                                     "Direction": 0,
                                     "Magnitude": 0,
                                "IsFinalScore": true
                                                },
                  "RollingAveragedPopulationScore": {
        "UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
                                     "Direction": 0,
                                     "Magnitude": 0,
                                "IsFinalScore": true
                                                },
                             "LongCount": "string",
                            "ShortCount": "string",
                               "LongShortRatio": 0,
           "TotalAccumulatedEstimatedAlphaValue": 0,
                        "KellyCriterionEstimate": 0,
                "KellyCriterionProbabilityValue": 0,
                                  "FitnessScore": 0,
                            "PortfolioTurnover": 0,
                         "ReturnOverMaxDrawdown": 0,
                                  "SortinoRatio": 0,
                     "EstimatedMonthlyAlphaValue": 0,
                  "TotalInsightsGenerated": "string",
                     "TotalInsightsClosed": "string",
         "TotalInsightsAnalysisCompleted": "string",
            "MeanPopulationEstimatedInsightValue": 0
                                                },
                                         "Charts": {
                                   "Name": "string",
                             "ChartType": "Overlay",
                                         "Series": {
                                   "Name": "string",
                                   "Unit": "string",
                                          "Index": 0,
                                          "Values": [
                                                 {
                                        "x": "string",
                                                "y": 0
                                                 }
                                                ],
                               "SeriesType": "Line",
                                  "Color": "string",
                        "ScatterMarkerSymbol": "none"
                                                 }
                                                },
                                         "Orders": {
                                            "Id": 0,
                                  "ContingentId": 0,
                                       "BrokerId": [
                                            "string"
                                                ],
                                         "Symbol": {
                                  "Value": "string",
                                     "ID": "string",
                                "Permtick": "string"
                                                },
                                          "Price": 0,
                          "PriceCurrency": "string",
                  "Time": "2021-11-26T15:18:27.693Z",
          "CreatedTime": "2021-11-26T15:18:27.693Z",
         "LastFillTime": "2021-11-26T15:18:27.693Z",
       "LastUpdateTime": "2021-11-26T15:18:27.693Z",
         "CanceledTime": "2021-11-26T15:18:27.693Z",
                                      "Quantity": 0,
                                   "Type": "Market",
                                    "Status": "New",
                                     "Tag": "string",
                             "SecurityType": "Base",
                                 "Direction": "Buy",
                                          "Value": 0,
                            "OrderSubmissionData": {
                                      "BidPrice": 0,
                                      "AskPrice": 0,
                                      "LastPrice": 0
                                                },
```

`BacktestList` Model - Collection container for a list of Backtest objects for a project.


```
                                                    "IsMarketable": true
                                                                    },
                                                        "OrderEvents": [
                                                                      {
                                                           "OrderId": 0,
                                                               "Id": 0,
                                                            "Symbol": {
                                                       "Value": "string",
                                                          "ID": "string",
                                                    "Permtick": "string"
                                                                    },
                                        "UtcTime": "2021-11-26T15:18:27.693Z",
                                                            "Status": {
                                                           "OrderId": 0,
                                                               "Id": 0,
                                                            "Symbol": {
                                                       "Value": "string",
                                                          "ID": "string",
                                                    "Permtick": "string"
                                                                    },
                                        "UtcTime": "2021-11-26T15:18:27.693Z",
                                                          "Status": "New",
                                                           "FillPrice": 0,
                                              "FillPriceCurrency": "string",
                                                        "FillQuantity": 0,
                                                       "Direction": "Buy",
                                                      "Message": "string",
                                                     "IsAssignment": true,
                                                           "StopPrice": 0,
                                                          "LimitPrice": 0,
                                                             "Quantity": 0
                                                                    },
                                                           "OrderFee": {
                                                              "Value": {
                                                             "Amount": 0,
                                                      "Currency": "string"
                                                                     }
                                                                    },
                                                           "FillPrice": 0,
                                              "FillPriceCurrency": "string",
                                                        "FillQuantity": 0,
                                                       "Direction": "Buy",
                                                      "Message": "string",
                                                     "IsAssignment": true,
                                                           "StopPrice": 0,
                                                          "LimitPrice": 0,
                                                             "Quantity": 0
                                                                     }
                                                                    ],
                                                   "ProfitLoss": "number",
                                                   "Statistics": "string",
                                              "RuntimeStatistics": "string",
                                                "ServerStatistics": "string"
                                                                    },
                                                        "error": "string",
                                                   "stacktrace": "string",
                                        "created": "2021-11-26T15:18:27.693Z",
                                                          "success": true,
                                                               "errors": [
                                                                  "string"
                                                                     ]
                                                                     }
                                                                    ],
                                                          "success": true,
                                                               "errors": [
                                                                  "string"
                                                                     ]
                                                                     }
```


`BacktestResult` Model - Results object class. Results are exhaust from backtest or live algorithms running in LEAN.

RollingWindow	`AlgorithmPerformance object` Rolling window detailed statistics.
TotalPerformance	`AlgorithmPerformance object` The AlgorithmPerformance class is a wrapper for TradeStatistics and PortfolioStatistics.
AlphaRuntimeStatistics	`AlphaRuntimeStatistics object` Contains insight population run time statistics.
Charts	`Chart object` Charts updates for the live algorithm since the last result packet.
Orders	`Order object` Order updates since the last result packet.
OrderEvents	`OrderEvent Array` OrderEvent updates since the last result packet.
ProfitLoss	`number object` Trade profit and loss information since the last algorithm result packet.
Statistics	`string object` Statistics information sent during the algorithm operations.
RuntimeStatistics	`string object` Runtime banner/updating statistics in the title banner of the live algorithm GUI.
ServerStatistics	`string object` Server status information, including CPU and RAM usage.


```
                                                                    {
                                                         "RollingWindow": {
                                                       "TradeStatistics": {
                                        "StartDateTime": "2021-11-26T15:18:27.693Z",
                                          "EndDateTime": "2021-11-26T15:18:27.693Z",
                                                   "TotalNumberOfTrades": 0,
                                                 "NumberOfWinningTrades": 0,
                                                  "NumberOfLosingTrades": 0,
                                                       "TotalProfitLoss": 0,
                                                           "TotalProfit": 0,
                                                             "TotalLoss": 0,
                                                         "LargestProfit": 0,
                                                           "LargestLoss": 0,
                                                     "AverageProfitLoss": 0,
                                                         "AverageProfit": 0,
                                                           "AverageLoss": 0,
                                                 "AverageTradeDuration": "string",
                                           "AverageWinningTradeDuration": "string",
                                            "AverageLosingTradeDuration": "string",
                                                   "MedianTradeDuration": "string",
                                            "MedianWinningTradeDuration": "string",
                                             "MedianLosingTradeDuration": "string",
                                                 "MaxConsecutiveWinningTrades": 0,
                                                  "MaxConsecutiveLosingTrades": 0,
                                                         "ProfitLossRatio": 0,
                                                            "WinLossRatio": 0,
                                                                 "WinRate": 0,
                                                                "LossRate": 0,
                                                              "AverageMAE": 0,
                                                              "AverageMFE": 0,
                                                              "LargestMAE": 0,
                                                              "LargestMFE": 0,
                                                  "MaximumClosedTradeDrawdown": 0,
                                                   "MaximumIntraTradeDrawdown": 0,
```

`BacktestResult` Model - Results object class. Results are exhaust from backtest or live algorithms running in LEAN.


```
                                                "ProfitLossStandardDeviation": 0,
                                                "ProfitLossDownsideDeviation": 0,
                                                        "ProfitFactor": 0,
                                                        "SharpeRatio": 0,
                                                        "SortinoRatio": 0,
                                                "ProfitToMaxDrawdownRatio": 0,
                                                "MaximumEndTradeDrawdown": 0,
                                                "AverageEndTradeDrawdown": 0,
                                            "MaximumDrawdownDuration": "string",
                                                        "TotalFees": 0
                                                    },
                                                "PortfolioStatistics": {
                                                        "RiskFreeRate": 0,
                                                    "AverageWinRate": 0,
                                                    "AverageLossRate": 0,
                                                    "ProfitLossRatio": 0,
                                                            "WinRate": 0,
                                                            "LossRate": 0,
                                                        "Expectancy": 0,
                                                "CompoundingAnnualReturn": 0,
                                                            "Drawdown": 0,
                                                    "TotalNetProfit": 0,
                                                        "SharpeRatio": 0,
                                                "ProbabilisticSharpeRatio": 0,
                                                            "Alpha": 0,
                                                            "Beta": 0,
                                                "AnnualStandardDeviation": 0,
                                                        "AnnualVariance": 0,
                                                    "InformationRatio": 0,
                                                        "TrackingError": 0,
                                                        "TreynorRatio": 0
                                                    },
                                                    "ClosedTrades": [
                                                        {
                                                            "Symbol": {
                                                        "Value": "string",
                                                            "ID": "string",
                                                    "Permtick": "string"
                                                    },
                                        "EntryTime": "2021-11-26T15:18:27.693Z",
                                                        "EntryPrice": 0,
                                                    "Direction": "Long",
                                                            "Quantity": 0,
                                        "ExitTime": "2021-11-26T15:18:27.693Z",
                                                        "ExitPrice": 0,
                                                    "ProfitLoss": 0,
                                                        "TotalFees": 0,
                                                            "MAE": 0,
                                                            "MFE": 0,
                                                "Duration": "string",
                                                "EndTradeDrawdown": 0
                                                        }
                                                    ]
                                                },
                                                "TotalPerformance": {
                                                    "TradeStatistics": {
                                    "StartDateTime": "2021-11-26T15:18:27.693Z",
                                      "EndDateTime": "2021-11-26T15:18:27.693Z",
                                                    "TotalNumberOfTrades": 0,
                                                "NumberOfWinningTrades": 0,
                                                "NumberOfLosingTrades": 0,
                                                    "TotalProfitLoss": 0,
                                                        "TotalProfit": 0,
                                                        "TotalLoss": 0,
                                                    "LargestProfit": 0,
                                                    "LargestLoss": 0,
                                                "AverageProfitLoss": 0,
                                                    "AverageProfit": 0,
                                                    "AverageLoss": 0,
                                            "AverageTradeDuration": "string",
                                        "AverageWinningTradeDuration": "string",
                                         "AverageLosingTradeDuration": "string",
                                                "MedianTradeDuration": "string",
                                        "MedianWinningTradeDuration": "string",
                                         "MedianLosingTradeDuration": "string",
                                            "MaxConsecutiveWinningTrades": 0,
                                             "MaxConsecutiveLosingTrades": 0,
                                                    "ProfitLossRatio": 0,
                                                    "WinLossRatio": 0,
                                                        "WinRate": 0,
                                                        "LossRate": 0,
                                                    "AverageMAE": 0,
                                                    "AverageMFE": 0,
                                                    "LargestMAE": 0,
                                                    "LargestMFE": 0,
                                            "MaximumClosedTradeDrawdown": 0,
                                             "MaximumIntraTradeDrawdown": 0,
                                            "ProfitLossStandardDeviation": 0,
                                            "ProfitLossDownsideDeviation": 0,
                                                    "ProfitFactor": 0,
                                                    "SharpeRatio": 0,
                                                    "SortinoRatio": 0,
                                                "ProfitToMaxDrawdownRatio": 0,
                                                "MaximumEndTradeDrawdown": 0,
                                                "AverageEndTradeDrawdown": 0,
                                            "MaximumDrawdownDuration": "string",
                                                        "TotalFees": 0
                                                    },
                                                "PortfolioStatistics": {
                                                        "RiskFreeRate": 0,
                                                    "AverageWinRate": 0,
                                                    "AverageLossRate": 0,
                                                    "ProfitLossRatio": 0,
                                                            "WinRate": 0,
                                                            "LossRate": 0,
                                                        "Expectancy": 0,
                                                "CompoundingAnnualReturn": 0,
                                                            "Drawdown": 0,
                                                    "TotalNetProfit": 0,
                                                        "SharpeRatio": 0,
                                                "ProbabilisticSharpeRatio": 0,
                                                            "Alpha": 0,
                                                            "Beta": 0,
                                                "AnnualStandardDeviation": 0,
                                                        "AnnualVariance": 0,
                                                    "InformationRatio": 0,
                                                        "TrackingError": 0,
                                                        "TreynorRatio": 0
                                                    },
                                                    "ClosedTrades": [
                                                        {
                                                            "Symbol": {
                                                        "Value": "string",
                                                            "ID": "string",
                                                    "Permtick": "string"
                                                    },
                                        "EntryTime": "2021-11-26T15:18:27.693Z",
                                                        "EntryPrice": 0,
                                                    "Direction": "Long",
                                                            "Quantity": 0,
```


Example

BacktestResult **Model - Results object class. Results are exhaust from backtest or live algorithms running in LEAN.**


```
                                                    "ExitTime": "2021-11-26T15:18:27.693Z",
                                                                                "ExitPrice": 0,
                                                                              "ProfitLoss": 0,
                                                                               "TotalFees": 0,
                                                                                    "MAE": 0,
                                                                                    "MFE": 0,
                                                                          "Duration": "string",
                                                                          "EndTradeDrawdown": 0
                                                                                            }
                                                                                            ]
                                                                                           },
                                                                   "AlphaRuntimeStatistics": {
                                                                      "MeanPopulationScore": {
                                                  "UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
                                                                                "Direction": 0,
                                                                                "Magnitude": 0,
                                                                          "IsFinalScore": true
                                                                                           },
                                                            "RollingAveragedPopulationScore": {
                                                  "UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
                                                                                "Direction": 0,
                                                                                "Magnitude": 0,
                                                                          "IsFinalScore": true
                                                                                           },
                                                                        "LongCount": "string",
                                                                       "ShortCount": "string",
                                                                          "LongShortRatio": 0,
                                                     "TotalAccumulatedEstimatedAlphaValue": 0,
                                                                  "KellyCriterionEstimate": 0,
                                                          "KellyCriterionProbabilityValue": 0,
                                                                            "FitnessScore": 0,
                                                                       "PortfolioTurnover": 0,
                                                                   "ReturnOverMaxDrawdown": 0,
                                                                            "SortinoRatio": 0,
                                                               "EstimatedMonthlyAlphaValue": 0,
                                                            "TotalInsightsGenerated": "string",
                                                               "TotalInsightsClosed": "string",
                                                   "TotalInsightsAnalysisCompleted": "string",
                                                      "MeanPopulationEstimatedInsightValue": 0
                                                                                           },
                                                                                  "Charts": {
                                                                                    "Name": "string",
                                                                        "ChartType": "Overlay",
                                                                                  "Series": {
                                                                                    "Name": "string",
                                                                                    "Unit": "string",
                                                                                  "Index": 0,
                                                                                  "Values": [
                                                                                            {
                                                                                     "x": "string",
                                                                                          "y": 0
                                                                                            }
                                                                                           ],
                                                                          "SeriesType": "Line",
                                                                             "Color": "string",
                                                                  "ScatterMarkerSymbol": "none"
                                                                                            }
                                                                                           },
                                                                                  "Orders": {
                                                                                     "Id": 0,
                                                                             "ContingentId": 0,
                                                                                 "BrokerId": [
                                                                                     "string"
                                                                                           ],
                                                                                  "Symbol": {
                                                                             "Value": "string",
                                                                                "ID": "string",
                                                                          "Permtick": "string"
                                                                                           },
                                                                                  "Price": 0,
                                                                     "PriceCurrency": "string",
                                                            "Time": "2021-11-26T15:18:27.693Z",
                                                     "CreatedTime": "2021-11-26T15:18:27.693Z",
                                                    "LastFillTime": "2021-11-26T15:18:27.693Z",
                                                  "LastUpdateTime": "2021-11-26T15:18:27.693Z",
                                                    "CanceledTime": "2021-11-26T15:18:27.693Z",
                                                                                "Quantity": 0,
                                                                              "Type": "Market",
                                                                               "Status": "New",
                                                                               "Tag": "string",
                                                                        "SecurityType": "Base",
                                                                            "Direction": "Buy",
                                                                                  "Value": 0,
                                                                       "OrderSubmissionData": {
                                                                                 "BidPrice": 0,
                                                                                 "AskPrice": 0,
                                                                                 "LastPrice": 0
                                                                                           },
                                                                          "IsMarketable": true
                                                                                           },
                                                                              "OrderEvents": [
                                                                                            {
                                                                                 "OrderId": 0,
                                                                                     "Id": 0,
                                                                                  "Symbol": {
                                                                             "Value": "string",
                                                                                "ID": "string",
                                                                          "Permtick": "string"
                                                                                           },
                                                            "UtcTime": "2021-11-26T15:18:27.693Z",
                                                                                  "Status": {
                                                                                 "OrderId": 0,
                                                                                     "Id": 0,
                                                                                  "Symbol": {
                                                                             "Value": "string",
                                                                                "ID": "string",
                                                                          "Permtick": "string"
                                                                                           },
                                                            "UtcTime": "2021-11-26T15:18:27.693Z",
                                                                                "Status": "New",
                                                                                 "FillPrice": 0,
                                                                  "FillPriceCurrency": "string",
                                                                             "FillQuantity": 0,
                                                                            "Direction": "Buy",
                                                                           "Message": "string",
                                                                          "IsAssignment": true,
                                                                                "StopPrice": 0,
                                                                               "LimitPrice": 0,
                                                                                  "Quantity": 0
                                                                                           },
                                                                                 "OrderFee": {
                                                                                    "Value": {
                                                                                   "Amount": 0,
                                                                          "Currency": "string"
                                                                                            }
                                                                                           },
                                                                                 "FillPrice": 0,
                                                                  "FillPriceCurrency": "string",
                                                                             "FillQuantity": 0,
                                                                            "Direction": "Buy",
```

```
                "Message": "string",
                "IsAssignment": true,
                "StopPrice": 0,
                "LimitPrice": 0,
                "Quantity": 0
                }
            ],
        "ProfitLoss": "number",
        "Statistics": "string",
    "RuntimeStatistics": "string",
      "ServerStatistics": "string"
    }
```


`BacktestResult` **Model - Results object class. Results are exhaust from backtest or live algorithms running in LEAN.**

`AlgorithmPerformance` **Model - The AlgorithmPerformance class is a wrapper for TradeStatistics and PortfolioStatistics.**

	`TradeStatistics object`
TradeStatistics	A set of statistics calculated from a list of closed trades.
	`PortfolioStatistics object`
PortfolioStatistics	Represents a set of statistics calculated from equity and benchmark samples.
	`Trade Array`
ClosedTrades	The algorithm statistics on portfolio.

Example


```
{
    "TradeStatistics": {
  "StartDateTime": "2021-11-26T15:18:27.693Z",
    "EndDateTime": "2021-11-26T15:18:27.693Z",
      "TotalNumberOfTrades": 0,
    "NumberOfWinningTrades": 0,
     "NumberOfLosingTrades": 0,
          "TotalProfitLoss": 0,
              "TotalProfit": 0,
                "TotalLoss": 0,
            "LargestProfit": 0,
                "LargestLoss": 0,
        "AverageProfitLoss": 0,
            "AverageProfit": 0,
              "AverageLoss": 0,
    "AverageTradeDuration": "string",
  "AverageWinningTradeDuration": "string",
   "AverageLosingTradeDuration": "string",
        "MedianTradeDuration": "string",
  "MedianWinningTradeDuration": "string",
   "MedianLosingTradeDuration": "string",
        "MaxConsecutiveWinningTrades": 0,
     "MaxConsecutiveLosingTrades": 0,
          "ProfitLossRatio": 0,
            "WinLossRatio": 0,
                "WinRate": 0,
              "LossRate": 0,
            "AverageMAE": 0,
            "AverageMFE": 0,
            "LargestMAE": 0,
            "LargestMFE": 0,
    "MaximumClosedTradeDrawdown": 0,
     "MaximumIntraTradeDrawdown": 0,
   "ProfitLossStandardDeviation": 0,
   "ProfitLossDownsideDeviation": 0,
          "ProfitFactor": 0,
            "SharpeRatio": 0,
            "SortinoRatio": 0,
      "ProfitToMaxDrawdownRatio": 0,
       "MaximumEndTradeDrawdown": 0,
     "AverageEndTradeDrawdown": 0,
  "MaximumDrawdownDuration": "string",
              "TotalFees": 0
    },
  "PortfolioStatistics": {
          "RiskFreeRate": 0,
        "AverageWinRate": 0,
      "AverageLossRate": 0,
      "ProfitLossRatio": 0,
              "WinRate": 0,
            "LossRate": 0,
          "Expectancy": 0,
  "CompoundingAnnualReturn": 0,
              "Drawdown": 0,
     "TotalNetProfit": 0,
            "SharpeRatio": 0,
   "ProbabilisticSharpeRatio": 0,
                "Alpha": 0,
                 "Beta": 0,
  "AnnualStandardDeviation": 0,
        "AnnualVariance": 0,
      "InformationRatio": 0,
         "TrackingError": 0,
           "TreynorRatio": 0
    },
    "ClosedTrades": [
      {
          "Symbol": {
        "Value": "string",
          "ID": "string",
       "Permtick": "string"
      },
  "EntryTime": "2021-11-26T15:18:27.693Z",
        "EntryPrice": 0,
      "Direction": "Long",
          "Quantity": 0,
   "ExitTime": "2021-11-26T15:18:27.693Z",
        "ExitPrice": 0,
      "ProfitLoss": 0,
       "TotalFees": 0,
            "MAE": 0,
            "MFE": 0,
    "Duration": "string",
    "EndTradeDrawdown": 0
      }
    ]
}
```


`TradeStatistics` **Model - A set of statistics calculated from a list of closed trades.**

StartDateTime	`string($date-time)`	The entry date/time of the first trade.
EndDateTime	`string($date-time)`	The exit date/time of the first trade.
TotalNumberOfTrades	`integer`	The total number of trades.
NumberOfWinningTrades	`integer`	The total number of winning trades.
NumberOfLosingTrades	`integer`	The total number of losing trades.
TotalProfitLoss	`number`	The total profit/loss for all trades (as symbol currency).
TotalProfit	`number`	The total profit for all winning trades (as symbol currency).

TotalLoss	The total loss for all losing trades (as symbol currency). number
LargestProfit	The largest profit in a single trade (as symbol currency). number
LargestLoss	The largest loss in a single trade (as symbol currency). number
AverageProfitLoss	The average profit/loss (a.k.a. Expectancy or Average Trade) for all trades (as symbol currency). number
AverageProfit	The average profit for all winning trades (as symbol currency). number
AverageLoss	The average loss for all winning trades (as symbol currency). string
AverageTradeDuration	The average duration for all trades. string
AverageWinningTradeDuration	The average duration for all winning trades. string
AverageLosingTradeDuration	The average duration for all losing trades. string
MedianTradeDuration	The median duration for all trades. string
MedianWinningTradeDuration	The median duration for all winning trades. string
MedianLosingTradeDuration	The median duration for all losing trades. integer
MaxConsecutiveWinningTrades	The maximum number of consecutive winning trades. integer
MaxConsecutiveLosingTrades	The maximum number of consecutive losing trades. number
ProfitLossRatio	The ratio of the average profit per trade to the average loss per trade. number
WinLossRatio	The ratio of the number of winning trades to the number of losing trades. number
WinRate	The ratio of the number of winning trades to the total number of trades. number
LossRate	The ratio of the number of losing trades to the total number of trades. number
AverageMAE	The average Maximum Adverse Excursion for all trades. number
AverageMFE	The average Maximum Adverse Excursion for all trades. number
LargestMAE	The average Maximum Favorable Excursion for all trades. number
LargestMFE	The largest Maximum Adverse Excursion in a single trade (as symbol currency). number
MaximumClosedTradeDrawdown	The maximum closed-trade drawdown for all trades (as symbol currency). number
MaximumIntraTradeDrawdown	The maximum intra-trade drawdown for all trades (as symbol currency). number
ProfitLossStandardDeviation	The standard deviation of the profits/losses for all trades (as symbol currency). number
ProfitLossDownsideDeviation	The downside deviation of the profits/losses for all trades (as symbol currency). number
ProfitFactor	The ratio of the total profit to the total loss. number
SharpeRatio	The ratio of the average profit/loss to the standard deviation. number
SortinoRatio	The ratio of the average profit/loss to the downside deviation. number
ProfitToMaxDrawdownRatio	The ratio of the total profit/loss to the maximum closed trade drawdown. number
MaximumEndTradeDrawdown	The maximum amount of profit given back by a single trade before exit (as symbol currency). number
AverageEndTradeDrawdown	The average amount of profit given back by all trades before exit (as symbol currency). string
MaximumDrawdownDuration	The maximum amount of time to recover from a drawdown (longest time between new equity highs or peaks). number
TotalFees	The sum of fees for all trades.

`TradeStatistics` **Model - A set of statistics calculated from a list of closed trades.**

Example


```
{
  "StartDateTime": "2021-11-26T15:18:27.693Z",
  "EndDateTime": "2021-11-26T15:18:27.693Z",
  "TotalNumberOfTrades": 0,
  "NumberOfWinningTrades": 0,
  "NumberOfLosingTrades": 0,
  "TotalProfitLoss": 0,
  "TotalProfit": 0,
  "TotalLoss": 0,
  "LargestProfit": 0,
  "LargestLoss": 0,
  "AverageProfitLoss": 0,
  "AverageProfit": 0,
  "AverageLoss": 0,
  "AverageTradeDuration": "string",
  "AverageWinningTradeDuration": "string",
  "AverageLosingTradeDuration": "string",
  "MedianTradeDuration": "string",
  "MedianWinningTradeDuration": "string",
  "MedianLosingTradeDuration": "string",
  "MaxConsecutiveWinningTrades": 0,
  "MaxConsecutiveLosingTrades": 0,
  "ProfitLossRatio": 0,
  "WinLossRatio": 0,
  "WinRate": 0,
  "LossRate": 0,
  "AverageMAE": 0,
  "AverageMFE": 0,
  "LargestMAE": 0,
  "LargestMFE": 0,
  "MaximumClosedTradeDrawdown": 0,
  "MaximumIntraTradeDrawdown": 0,
  "ProfitLossStandardDeviation": 0,
  "ProfitLossDownsideDeviation": 0,
  "ProfitFactor": 0,
  "SharpeRatio": 0,
  "SortinoRatio": 0,
  "ProfitToMaxDrawdownRatio": 0,
  "MaximumEndTradeDrawdown": 0,
  "AverageEndTradeDrawdown": 0,
  "MaximumDrawdownDuration": "string",
  "TotalFees": 0
}
```


`PortfolioStatistics` **Model - Represents a set of statistics calculated from equity and benchmark samples.**

RiskFreeRate	number

`PortfolioStatistics` Model - Represents a set of statistics calculated from equity and benchmark samples.

	The current defined risk free annual return rate.
AverageWinRate	`number` The average rate of return for winning trades.
AverageLossRate	`number` The average rate of return for losing trades.
ProfitLossRatio	`number` The ratio of the average win rate to the average loss rate.
WinRate	`number` The ratio of the number of winning trades to the total number of trades.
LossRate	`number` The ratio of the number of losing trades to the total number of trades.
Expectancy	`number` The expected value of the rate of return.
CompoundingAnnualReturn	`number` Annual compounded returns statistic based on the final-starting capital and years.
Drawdown	`number` Drawdown maximum percentage.
TotalNetProfit	`number` The total net profit percentage.
SharpeRatio	`number` Sharpe ratio with respect to risk free rate: measures excess of return per unit of risk.
ProbabilisticSharpeRatio	`number` Probabilistic Sharpe Ratio is a probability measure associated with the Sharpe ratio. It informs us of the probability that the estimated Sharpe ratio is greater than a chosen benchmark.
Alpha	`number` Algorithm "Alpha" statistic - abnormal returns over the risk free rate and the relationshio (beta) with the benchmark returns.
Beta	`number` Algorithm beta statistic - the covariance between the algorithm and benchmark performance, divided by benchmark variance.
AnnualStandardDeviation	`number` Annualized standard deviation.
AnnualVariance	`number` Annualized variance statistic calculation using the daily performance variance and trading days per year.
InformationRatio	`number` Information ratio - risk adjusted return.
TrackingError	`number` Tracking error volatility (TEV) statistic - a measure of how closely a portfolio follows the index to which it is benchmarked.
TreynorRatio	`number` Treynor ratio statistic is a measurement of the returns earned in excess of that which could have been earned on an investment that has no diversifiable risk.

Example	{
	"RiskFreeRate": 0,
	"AverageWinRate": 0,
	"AverageLossRate": 0,
	"ProfitLossRatio": 0,
	"WinRate": 0,
	"LossRate": 0,
	"Expectancy": 0,
	"CompoundingAnnualReturn": 0,
	"Drawdown": 0,
	"TotalNetProfit": 0,
	"SharpeRatio": 0,
	"ProbabilisticSharpeRatio": 0,
	"Alpha": 0,
	"Beta": 0,
	"AnnualStandardDeviation": 0,
	"AnnualVariance": 0,
	"InformationRatio": 0,
	"TrackingError": 0,
	"TreynorRatio": 0
	}

`Trade` Model - Represents a closed trade.

Symbol	`Symbol object` Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
EntryTime	`string($date-time)` The date and time the trade was opened.
EntryPrice	`number` The price at which the trade was opened (or the average price if multiple entries).
Direction	`string Enum` Direction of a trade. Options : ['Long', 'Short']
Quantity	`number` The total unsigned quantity of the trade.
ExitTime	`string($date-time)` The date and time the trade was closed.
ExitPrice	`number` The price at which the trade was closed (or the average price if multiple exits).
ProfitLoss	`number` The gross profit/loss of the trade (as account currency).
TotalFees	`number` The total fees associated with the trade (always positive value) (as account currency).
MAE	`number` The Maximum Adverse Excursion (as account currency).
MFE	`number` The Maximum Favorable Excursion (as account currency).
Duration	`string` The duration of the trade.
EndTradeDrawdown	`number` The amount of profit given back before the trade was closed.

Example	{
	"Symbol": {
	"Value": "string",
	"ID": "string",
	"Permtick": "string"
	},
	"EntryTime": "2021-11-26T15:18:27.693Z",
	"EntryPrice": 0,
	"Direction": "Long",
	"Quantity": 0,
	"ExitTime": "2021-11-26T15:18:27.693Z",
	"ExitPrice": 0,
	"ProfitLoss": 0,
	"TotalFees": 0,
	"MAE": 0,
	"MFE": 0,
	"Duration": "string",
	"EndTradeDrawdown": 0
	}

`Symbol` Model - Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.

Value	`string` The current symbol for this ticker.
ID	`string` The security identifier for this symbol.
	`string`

`Symbol` Model - Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.

Permtick	The current symbol for this ticker.
Example	{ "Value": "string", "ID": "string", "Permtick": "string" }

`TradeDirection` Model - Direction of a trade.

TradeDirection	string Enum Direction of a trade. Options : ['Long', 'Short']
Example	{ "TradeDirection": "Long" }

`AlphaRuntimeStatistics` Model - Contains insight population run time statistics.

MeanPopulationScore	InsightScore object Defines the scores given to a particular insight.
RollingAveragedPopulationScore	InsightScore object Defines the scores given to a particular insight.
LongCount	string Gets the total number of insights with an up direction.
ShortCount	string Gets the total number of insights with a down direction.
LongShortRatio	number The ratio of InsightDirection.Up over InsightDirection.Down.
TotalAccumulatedEstimatedAlphaValue	number The total accumulated estimated value of trading all insights.
KellyCriterionEstimate	number Score of the strategy's insights predictive power.
KellyCriterionProbabilityValue	number The p-value or probability value of the KellyCriterionEstimate.
FitnessScore	number Score of the strategy's performance, and suitability for the Alpha Stream Market.
PortfolioTurnover	number Measurement of the strategies trading activity with respect to the portfolio value. Calculated as the sales volume with respect to the average total portfolio value.
ReturnOverMaxDrawdown	number Provides a risk adjusted way to factor in the returns and drawdown of the strategy. It is calculated by dividing the Portfolio Annualized Return by the Maximum Drawdown seen during the backtest.
SortinoRatio	number Gives a relative picture of the strategy volatility. It is calculated by taking a portfolio's annualized rate of return and subtracting the risk free rate of return.
EstimatedMonthlyAlphaValue	number Suggested Value of the Alpha On A Monthly Basis For Licensing.
TotalInsightsGenerated	string The total number of insight signals generated by the algorithm.
TotalInsightsClosed	string The total number of insight signals generated by the algorithm.
TotalInsightsAnalysisCompleted	string The total number of insight signals generated by the algorithm.
MeanPopulationEstimatedInsightValue	number Gets the mean estimated insight value.
Example	{ "MeanPopulationScore": { "UpdatedTimeUtc": "2021-11-26T15:18:27.693Z", "Direction": 0, "Magnitude": 0, "IsFinalScore": true }, "RollingAveragedPopulationScore": { "UpdatedTimeUtc": "2021-11-26T15:18:27.693Z", "Direction": 0, "Magnitude": 0, "IsFinalScore": true }, "LongCount": "string", "ShortCount": "string", "LongShortRatio": 0, "TotalAccumulatedEstimatedAlphaValue": 0, "KellyCriterionEstimate": 0, "KellyCriterionProbabilityValue": 0, "FitnessScore": 0, "PortfolioTurnover": 0, "ReturnOverMaxDrawdown": 0, "SortinoRatio": 0, "EstimatedMonthlyAlphaValue": 0, "TotalInsightsGenerated": "string", "TotalInsightsClosed": "string", "TotalInsightsAnalysisCompleted": "string", "MeanPopulationEstimatedInsightValue": 0 }

`InsightScore` Model - Defines the scores given to a particular insight

UpdatedTimeUtc	string($date-time) The time these scores were last updated.
Direction	number The direction score.
Magnitude	number The magnitude score.
IsFinalScore	boolean Is the insight past its expiry time and score can be finalized.
Example	{ "UpdatedTimeUtc": "2021-11-26T15:18:27.693Z", "Direction": 0, "Magnitude": 0, "IsFinalScore": true }

`Chart` Model - Single Parent Chart Object for Custom Charting.

Name	string Name of the Chart.
ChartType	string Enum Type of the Chart, Overlayed or Stacked. Options : ['Overlay', 'Stacked']
Series	Series object List of Series Objects for this Chart.
Example	{ "Name": "string", "ChartType": "Overlay", "Series": { "Name": "string", "Unit": "string", "Index": 0, "Values": [{ "x": "string", "y": 0 }], "SeriesType": "Line",

```
    "Color": "string",
    "ScatterMarkerSymbol": "none"
  }
}
```


`Series` Model - Chart Series Object - Series data and properties for a chart.

Name	`string` Name of the series.
Unit	`string` Axis for the chart series.
Index	`integer` Index/position of the series on the chart.
Values	`ChartPoint Array` Values for the series plot. These values are assumed to be in ascending time order (first points earliest, last points latest).
SeriesType	`string Enum` Chart type for the series. Options : ['Line', 'Scatter', 'Candle', 'Bar', 'Flag', 'StackedArea', 'Pie', 'Treemap']
Color	`string` Color the series.
ScatterMarkerSymbol	`string Enum` Shape or symbol for the marker in a scatter plot. Options : ['none', 'circle', 'square', 'diamond', 'triangle', 'triangle-down']
Example	`{ "Name": "string", "Unit": "string", "Index": 0, "Values": [ { "x": "string", "y": 0 } ], "SeriesType": "Line", "Color": "string", "ScatterMarkerSymbol": "none" }`

`ChartPoint` Model - Location on a chart containing the X-Y location

x	`string` Time of this chart point: lower case for javascript encoding simplicty.
y	`number` Value of this chart point: lower case for javascript encoding simplicty.
Example	`{ "x": "string", "y": 0 }`

`Order` Model - Order struct for placing new trade.

Id	`integer` Order ID.
ContingentId	`integer` Order Id to process before processing this order.
BrokerId	`string Array` Brokerage Id for this order for when the brokerage splits orders into multiple pieces.
Symbol	`Symbol object` Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
Price	`number` Price of the Order.
PriceCurrency	`string` Currency for the order price.
Time	`string($date-time)` Gets the utc time the order was created.
CreatedTime	`string($date-time)` Gets the utc time this order was created. Alias for Time.
LastFillTime	`string($date-time)` Gets the utc time the last fill was received, or null if no fills have been received.
LastUpdateTime	`string($date-time)` Gets the utc time this order was last updated, or null if the order has not been updated.
CanceledTime	`string($date-time)` Gets the utc time this order was canceled, or null if the order was not canceled.
Quantity	`number` Number of shares to execute.
Type	`string Enum` Order type. Options : ['Market', 'Limit', 'StopMarket', 'StopLimit', 'MarketOnOpen', 'MarketOnClose', 'OptionExercise']
Status	`string Enum` Status of the Order. Options : ['New', 'Submitted', 'PartiallyFilled', 'Filled', 'Canceled', 'None', 'Invalid', 'CancelPending', 'UpdateSubmitted']
Tag	`string` Tag the order with some custom data.
SecurityType	`string Enum` Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']
Direction	`string Enum` Direction of the order. Options : ['Buy', 'Sell', 'Hold']
Value	`number` Gets the executed value of this order. If the order has not yet filled, then this will return zero.
OrderSubmissionData	`OrderSubmissionData object` Stores time and price information available at the time an order was submitted.
IsMarketable	`boolean` Returns true if the order is a marketable order.
Example	`{ "Id": 0, "ContingentId": 0, "BrokerId": ["string"], "Symbol": { "Value": "string", "ID": "string", "Permtick": "string" }, "Price": 0, "PriceCurrency": "string", "Time": "2021-11-26T15:18:27.693Z", "CreatedTime": "2021-11-26T15:18:27.693Z", "LastFillTime": "2021-11-26T15:18:27.693Z", "LastUpdateTime": "2021-11-26T15:18:27.693Z", "CanceledTime": "2021-11-26T15:18:27.693Z", "Quantity": 0, "Type": "Market", "Status": "New", "Tag": "string", "SecurityType": "Base", "Direction": "Buy", "Value": 0, "OrderSubmissionData": { "BidPrice": 0, "AskPrice": 0, "LastPrice": 0`

`Order` Model - Order struct for placing new trade.


```
},
"IsMarketable": true
}
```


`SecurityType` Model - Type of tradable security / underlying asset.

SecurityType — string Enum
Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']

Example


```
{
"SecurityType": "Base"
}
```


`OrderDirection` Model - Direction of the order.

OrderDirection — string Enum
Direction of the order. Options : ['Buy', 'Sell', 'Hold']

Example


```
{
"OrderDirection": "Buy"
}
```


`OrderSubmissionData` Model - Stores time and price information available at the time an order was submitted.

BidPrice — number
The bid price at an order submission time.

AskPrice — number
The ask price at an order submission time.

LastPrice — number
The current price at an order submission time.

Example


```
{
"BidPrice": 0,
"AskPrice": 0,
"LastPrice": 0
}
```


`OrderEvent` Model - Change in an order state applied to user algorithm portfolio

OrderId — integer
Id of the order this event comes from.

Id — integer
The unique order event Id for each order.

Symbol — Symbol object
Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.

UtcTime — string($date-time)
The date and time of this event (UTC).

Status — OrderStatus object
Messaging class signifying a change in an order state and record the change in the users algorithm portfolio.

OrderFee — OrderFee object
The order fee associated with the specified order.

FillPrice — number
Fill price information about the order.

FillPriceCurrency — string
Currency for the fill price.

FillQuantity — number
Number of shares of the order that was filled in this event.

Direction — string Enum
Direction of the order. Options : ['Buy', 'Sell', 'Hold']

Message — string
Any message from the exchange.

IsAssignment — boolean
True if the order event is an assignment.

StopPrice — number
The current stop price.

LimitPrice — number
The current limit price.

Quantity — number
The current order quantity.

Example


```
{
"OrderId": 0,
"Id": 0,
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"UtcTime": "2021-11-26T15:18:27.693Z",
"Status": {
"OrderId": 0,
"Id": 0,
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"UtcTime": "2021-11-26T15:18:27.693Z",
"Status": "New",
"FillPrice": 0,
"FillPriceCurrency": "string",
"FillQuantity": 0,
"Direction": "Buy",
"Message": "string",
"IsAssignment": true,
"StopPrice": 0,
"LimitPrice": 0,
"Quantity": 0
},
"OrderFee": {
"Value": {
"Amount": 0,
"Currency": "string"
}
},
"FillPrice": 0,
"FillPriceCurrency": "string",
"FillQuantity": 0,
"Direction": "Buy",
"Message": "string",
"IsAssignment": true,
"StopPrice": 0,
"LimitPrice": 0,
"Quantity": 0
}
```


`OrderStatus` Model - Messaging class signifying a change in an order state and record the change in the users algorithm portfolio.

OrderId — integer
Id of the order this event comes from.

Id — integer
The unique order event Id for this order.

Symbol — Symbol object
Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.

UtcTime — string($date-time)
The date and time of this event.

`OrderStatus` Model - Messaging class signifying a change in an order state and record the change in the users algorithm portfolio.

Status	`string Enum` Status of the Order. Options : ['New', 'Submitted', 'PartiallyFilled', 'Filled', 'Canceled', 'None', 'Invalid', 'CancelPending', 'UpdateSubmitted']
FillPrice	`number` Fill price information about the order.
FillPriceCurrency	`string` Currency for the fill price.
FillQuantity	`number` Number of shares of the order that was filled in this event.
Direction	`string Enum` Direction of the order. Options : ['Buy', 'Sell', 'Hold']
Message	`string` Any message from the exchange.
IsAssignment	`boolean` Order event is an allocation of trades from ITM option assignment.
StopPrice	`number` The current stop price.
LimitPrice	`number` The current limit price.
Quantity	`number` The current order quantity.

Example	{
	"OrderId": 0,
	"Id": 0,
	"Symbol": {
	"Value": "string",
	"ID": "string",
	"Permtick": "string"
	},
	"UtcTime": "2021-11-26T15:18:27.693Z",
	"Status": "New",
	"FillPrice": 0,
	"FillPriceCurrency": "string",
	"FillQuantity": 0,
	"Direction": "Buy",
	"Message": "string",
	"IsAssignment": true,
	"StopPrice": 0,
	"LimitPrice": 0,
	"Quantity": 0
	}

`Status` Model - Status of the Order.

Status	`string Enum` Status of the Order. Options : ['New', 'Submitted', 'PartiallyFilled', 'Filled', 'Canceled', 'None', 'Invalid', 'CancelPending', 'UpdateSubmitted']

Example	{
	"Status": "New"
	}

`OrderFee` Model - The order fee associated with the specified order.

Value	`CashAmount object` Represents a cash amount which can be converted to account currency using a currency converter.

Example	{
	"Value": {
	"Amount": 0,
	"Currency": "string"
	}
	}

`CashAmount` Model - Represents a cash amount which can be converted to account currency using a currency converter.

Amount	`number` The amount of cash.
Currency	`string` The currency in which the cash amount is denominated.

Example	{
	"Amount": 0,
	"Currency": "string"
	}

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header

14.5.2.2 Portfolio

Introduction

Read out the portfolio state of a backtest.

Request

Fetch the portfolio state of a backtest for the project Id and backtest Id. The `/backtests/read/portfolio` API accepts requests in the following format:

`ReadBacktestPortfolioRequest` Model - Request to read the portfolio state from a backtest in a project.

projectId	`integer` Id of the project from which to read the backtest.
backtestId	`string` Id of the backtest from which to read the portfolio state.

Example	{
	"projectId": 0,
	"backtestId": "string"
	}

Responses

The `/backtests/read/portfolio` API provides a response in the following format:

200 Success

`BacktestPortfolioResponse` Model - Contains holdings and cash of the backtest instance.

portfolio	`Portfolio object` /.

Example	{
	"portfolio": {
	"Holdings": {
	"Symbol": {
	"Value": "string",
	"ID": "string",
	"Permtick": "string"
	},
	"Type": "Base",
	"CurrencySymbol": "$",
	"AveragePrice": 0,
	"Quantity": 0,
	"MarketPrice": 0,
	"ConversionRate": 0,
	"MarketValue": 0,
	"UnrealizedPnl": 0
	},
	"Cash": {
	"Symbol": "string",
	"Amount": 0,
	"ConversionRate": 0,
	"CurrencySymbol": ,
	"ValueInAccountCurrency": 0
	}
	}
	}

`Portfolio` Model

Holdings	`Holding object` Dictionary of algorithm holdings information.
Cash	`Cash object` Represents a holding of a currency in cash.

Example	{
	"Holdings": {
	"Symbol": {
	"Value": "string",
	"ID": "string",
	"Permtick": "string"
	},
	"Type": "Base",
	"CurrencySymbol": "$",
	"AveragePrice": 0,
	"Quantity": 0,
	"MarketPrice": 0,
	"ConversionRate": 0,
	"MarketValue": 0,
	"UnrealizedPnl": 0
	},
	"Cash": {
	"Symbol": "string",
	"Amount": 0,
	"ConversionRate": 0,
	"CurrencySymbol": ,
	"ValueInAccountCurrency": 0
	}
	}

`Holding` Model - Live results object class for packaging live result data.

Symbol	`Symbol object` Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
Type	`string Enum` Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']
CurrencySymbol	`string` *example: $* The currency symbol of the holding.
AveragePrice	`number` Average Price of our Holding in the currency the symbol is traded in.
Quantity	`number` Quantity of the Symbol we hold.
MarketPrice	`number` Current Market Price of the Asset in the currency the symbol is traded in.
ConversionRate	`number` Current market conversion rate into the account currency.
MarketValue	`number` Current market value of the holding.
UnrealizedPnl	`number` Current unrealized P/L of the holding.

Example	{
	"Symbol": {
	"Value": "string",
	"ID": "string",
	"Permtick": "string"
	},
	"Type": "Base",
	"CurrencySymbol": "$",
	"AveragePrice": 0,
	"Quantity": 0,

`Holding` Model - Live results object class for packaging live result data.

```
"MarketPrice": 0,
"ConversionRate": 0,
"MarketValue": 0,
"UnrealizedPnl": 0
}
```

`Symbol` Model - Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.

Value	string	The current symbol for this ticker.
ID	string	The security identifier for this symbol.
Permtick	string	The current symbol for this ticker.

Example

```
{
"Value": "string",
"ID": "string",
"Permtick": "string"
}
```

`SecurityType` Model - Type of tradable security / underlying asset.

SecurityType	string Enum	Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']

Example

```
{
"SecurityType": "Base"
}
```

`Cash` Model - Represents a holding of a currency in cash.

Symbol	string	Gets the symbol used to represent this cash.
Amount	number	Gets or sets the amount of cash held.
ConversionRate	number	The currency conversion rate to the account base currency.
CurrencySymbol	object	The symbol of the currency, such as $.
ValueInAccountCurrency	number	The value of the currency cash in the account base currency.

Example

```
{
"Symbol": "string",
"Amount": 0,
"ConversionRate": 0,
"CurrencySymbol": ,
"ValueInAccountCurrency": 0
}
```

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	string	Header

14.5.2.3 Orders

Introduction

Read out the orders of a backtest.

Request

Fetch the orders of a backtest for the project Id, backtest Id and steps provided. The `/backtests/read/orders` API accepts requests in the following format:

`ReadBacktestOrdersRequest` Model - Request to read orders from a backtest.

start	`integer` Starting index of the orders to be fetched. Required if end > 100.
end	`integer` Last index of the orders to be fetched. Note that end - start must be less than 100.
projectId	`integer` Id of the project from which to read the backtest.
backtestId	`string` Id of the backtest from which to read the orders.
Example	`{ "start": 0, "end": 0, "projectId": 0, "backtestId": "string" }`

Responses

The `/backtests/read/orders` API provides a response in the following format:

200 Success

`BacktestOrdersResponse` Model - Contains orders and the number of orders of the backtest in the request criteria.

Orders	`Order object` /.
Length	`integer` Total number of returned orders.
Example	`{ "Orders": { "Id": 0, "ContingentId": 0, "BrokerId": ["string"], "Symbol": { "Value": "string", "ID": "string", "Permtick": "string" }, "Price": 0, "PriceCurrency": "string", "Time": "2021-11-26T15:18:27.693Z", "CreatedTime": "2021-11-26T15:18:27.693Z", "LastFillTime": "2021-11-26T15:18:27.693Z", "LastUpdateTime": "2021-11-26T15:18:27.693Z", "CanceledTime": "2021-11-26T15:18:27.693Z", "Quantity": 0, "Type": "Market", "Status": "New", "Tag": "string", "SecurityType": "Base", "Direction": "Buy", "Value": 0, "OrderSubmissionData": { "BidPrice": 0, "AskPrice": 0, "LastPrice": 0 }, "IsMarketable": true }, "Length": 0 }`

`Order` Model - Order struct for placing new trade.

Id	`integer` Order ID.
ContingentId	`integer` Order Id to process before processing this order.
BrokerId	`string Array` Brokerage Id for this order for when the brokerage splits orders into multiple pieces.
Symbol	`Symbol object` Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
Price	`number` Price of the Order.
PriceCurrency	`string` Currency for the order price.
Time	`string($date-time)` Gets the utc time the order was created.
CreatedTime	`string($date-time)` Gets the utc time this order was created. Alias for Time.
LastFillTime	`string($date-time)` Gets the utc time the last fill was received, or null if no fills have been received.
LastUpdateTime	`string($date-time)` Gets the utc time this order was last updated, or null if the order has not been updated.
CanceledTime	`string($date-time)` Gets the utc time this order was canceled, or null if the order was not canceled.
Quantity	`number` Number of shares to execute.
Type	`string Enum` Order type. Options : ['Market', 'Limit', 'StopMarket', 'StopLimit', 'MarketOnOpen', 'MarketOnClose', 'OptionExercise']
Status	`string Enum` Status of the Order. Options : ['New', 'Submitted', 'PartiallyFilled', 'Filled', 'Canceled', 'None', 'Invalid', 'CancelPending', 'UpdateSubmitted']
Tag	`string` Tag the order with some custom data.
SecurityType	`string Enum` Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']
Direction	`string Enum` Direction of the order. Options : ['Buy', 'Sell', 'Hold']
Value	`number` Gets the executed value of this order. If the order has not yet filled, then this will return zero.
	`OrderSubmissionData object`

OrderSubmissionData

Stores time and price information available at the time an order was submitted.

IsMarketable

boolean

Returns true if the order is a marketable order.

Example


```
{
  "Id": 0,
  "ContingentId": 0,
  "BrokerId": [
    "string"
  ],
  "Symbol": {
    "Value": "string",
    "ID": "string",
    "Permtick": "string"
  },
  "Price": 0,
  "PriceCurrency": "string",
  "Time": "2021-11-26T15:18:27.693Z",
  "CreatedTime": "2021-11-26T15:18:27.693Z",
  "LastFillTime": "2021-11-26T15:18:27.693Z",
  "LastUpdateTime": "2021-11-26T15:18:27.693Z",
  "CanceledTime": "2021-11-26T15:18:27.693Z",
  "Quantity": 0,
  "Type": "Market",
  "Status": "New",
  "Tag": "string",
  "SecurityType": "Base",
  "Direction": "Buy",
  "Value": 0,
  "OrderSubmissionData": {
    "BidPrice": 0,
    "AskPrice": 0,
    "LastPrice": 0
  },
  "IsMarketable": true
}
```


`Order` Model - Order struct for placing new trade.

`Symbol` Model - Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.

Value

string

The current symbol for this ticker.

ID

string

The security identifier for this symbol.

Permtick

string

The current symbol for this ticker.

Example


```
{
  "Value": "string",
  "ID": "string",
  "Permtick": "string"
}
```


`SecurityType` Model - Type of tradable security / underlying asset.

SecurityType

string Enum

Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']

Example


```
{
  "SecurityType": "Base"
}
```


`OrderDirection` Model - Direction of the order.

OrderDirection

string Enum

Direction of the order. Options : ['Buy', 'Sell', 'Hold']

Example


```
{
  "OrderDirection": "Buy"
}
```


`OrderSubmissionData` Model - Stores time and price information available at the time an order was submitted.

BidPrice

number

The bid price at an order submission time.

AskPrice

number

The ask price at an order submission time.

LastPrice

number

The current price at an order submission time.

Example


```
{
  "BidPrice": 0,
  "AskPrice": 0,
  "LastPrice": 0
}
```


401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate

string

Header

14.5.3 Update Backtest

Introduction

Update a backtest name or note

Request

A JSON object containing info about the backtest and new name. The `/backtests/update` API accepts requests in the following format:

`UpdateBacktestRequest` Model - Request to update a backtest's name.

projectId	`integer` Project Id for the backtest we want to update.
backtestId	`string` Backtest Id we want to update.
name	`string` Name we'd like to assign to the backtest.
note	`string` Note attached to the backtest.

Example	`{`
	`"projectId": 0,`
	`"backtestId": "string",`
	`"name": "string",`
	`"note": "string"`
	`}`

Responses

The `/backtests/update` API provides a response in the following format:

200 Success

`RestResponse` Model - Base API response class for the QuantConnect API.

success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	`{`
	`"success": true,`
	`"errors": [`
	`"string"`
	`]`
	`}`

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header

14.5.4 Delete Backtest

Introduction

Delete a backtest from the specified project and backtestId

Request

Information required to delete the backtest. The `/backtests/delete` API accepts requests in the following format:

`DeleteBacktestRequest` Model - Request to delete a backtest.

projectId	`integer` Project Id for the backtest we want to delete.
backtestId	`string` Backtest Id we want to delete.

Example	{
	"projectId": 0,
	"backtestId": "string"
	}

Responses

The `/backtests/delete` API provides a response in the following format:

200 Success

`RestResponse` Model - Base API response class for the QuantConnect API.

success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	{
	"success": true,
	"errors": [
	"string"
	]
	}

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header

14.6 Live Management

The QuantConnect REST API lets you manage your live algorithms on our cloud servers through URL endpoints.

Create Live Algorithm

Read Live Algorithm

Update Live Algorithm

14.6.1 Create Live Algorithm

Introduction

Create a live algorithm

Request

Project, compile and brokerage login information for deploying a live algorithm. The `/live/create` API accepts requests in the following format:

`CreateLiveAlgorithmRequest` Model - Request to create a live algorithm.

projectId	`integer` Project Id.
compileId	`string` Compile Id.
serverType	`string` Type of server instance that will run the algorithm.
baseLiveAlgorithmSettings	`BaseLiveAlgorithmSettings object` Base class for settings that must be configured per Brokerage to create new algorithms via the API.
versionId	`string` *example: -1* The version of the Lean used to run the algorithm. -1 is master, however, sometimes this can create problems with live deployments. If you experience problems using, try specifying the version of Lean you would like to use.

Example	{
	"projectId": 0,
	"compileId": "string",
	"serverType": "string",
	"baseLiveAlgorithmSettings": {
	"id": "string",
	"user": "string",
	"password": "string",
	"environment": "live",
	"account": "string"
	},
	"versionId": "-1"
	}

`BaseLiveAlgorithmSettings` Model - Base class for settings that must be configured per Brokerage to create new algorithms via the API.

id	`string` 'Interactive' / 'FXCM' / 'Oanda' / 'Tradier' /'PaperTrading'.
user	`string` Username associated with brokerage.
password	`string` Password associated with brokerage.
environment	`string Enum` Represents the types of environments supported by brokerages for trading. Options : ['live', 'paper']
account	`string` Account of the associated brokerage.

Example	{
	"id": "string",
	"user": "string",
	"password": "string",
	"environment": "live",
	"account": "string"
	}

`BrokerageEnvironment` Model - Represents the types of environments supported by brokerages for trading.

BrokerageEnvironment	`string Enum` Represents the types of environments supported by brokerages for trading. Options : ['live', 'paper']

Example	{
	"BrokerageEnvironment": "live"
	}

Responses

The `/live/create` API provides a response in the following format:

200 Success

`LiveAlgorithm` Model - Live algorithm instance result from the QuantConnect Rest API.

projectId	`integer` Project Id for the live instance.
deployId	`string` Unique live algorithm deployment identifier (similar to a backtest id).
status	`string Enum` States of a live deployment. Options : ['DeployError', 'InQueue', 'Running', 'Stopped', 'Liquidated', 'Deleted', 'Completed', 'RuntimeError', 'Invalid', 'LoggingIn', 'Initializing', 'History']
launched	`string($date-time)` Datetime the algorithm was launched in UTC.
stopped	`string($date-time)` Datetime the algorithm was stopped in UTC, null if its still running.
brokerage	`string Enum` Brokerage. Options : ['Interactive', 'FXCM', 'Oanda', 'Tradier', 'PaperTrading', 'Alpaca', 'Bitfinex', 'Binance', 'GDAX']
subscription	`string` Chart we're subscribed to.
error	`string` Live algorithm error message from a crash or algorithm runtime error.
success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	{
	"projectId": 0,
	"deployId": "string",
	"status": "DeployError",
	"launched": "2021-11-26T15:18:27.693Z",
	"stopped": "2021-11-26T15:18:27.693Z",
	"brokerage": "Interactive",
	"subscription": "string",
	"error": "string",
	"success": true,
	"errors": [
	"string"
	]
	}

`AlgorithmStatus` Model - States of a live deployment.

AlgorithmStatus	`string Enum` States of a live deployment. Options : ['DeployError', 'InQueue', 'Running', 'Stopped', 'Liquidated', 'Deleted', 'Completed', 'RuntimeError', 'Invalid', 'LoggingIn', 'Initializing', 'History']

Example	{
	"AlgorithmStatus": "DeployError"
	}

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string`	Header

14.6.2 Read Live Algorithm

The QuantConnect REST API lets you read your live algorithm results from our cloud servers through URL endpoints.

Live Algorithm Statistics

Logs

Portfolio State

Orders

14.6.2.1 Live Algorithm Statistics

Introduction

If a ReadLiveAlgorithmRequest is provided details on a live algorithm are returned. If a ListLiveAlgorithmsRequest is passed get a list of live running algorithms.

Request

Dynamic arguement to specify whether seeking single project or list response. The `/live/read` API accepts requests in the following format:

`ReadLiveAlgorithmRequest` Model - Request to read out a single algorithm.

projectId	`integer` Id of the project to read.
deployId	`string` Specific instance Id to read.

Example	<pre>{
	"projectId": 0,
	"deployId": "string"
	}</pre>

`ListLiveAlgorithmsRequest` Model - Request for a list of live running algorithms.

status	`string Enum` States of a live deployment. Options : ['DeployError', 'InQueue', 'Running', 'Stopped', 'Liquidated', 'Deleted', 'Completed', 'RuntimeError', 'Invalid', 'LoggingIn', 'Initializing', 'History']
start	`string($date-time)` Earliest launched time of the algorithms.
end	`string($date-time)` Latest launched time of the algorithms.

Example	<pre>{
	"status": "DeployError",
	"start": "2021-11-26T15:18:27.693Z",
	"end": "2021-11-26T15:18:27.693Z"
	}</pre>

`AlgorithmStatus` Model - States of a live deployment.

AlgorithmStatus	`string Enum` States of a live deployment. Options : ['DeployError', 'InQueue', 'Running', 'Stopped', 'Liquidated', 'Deleted', 'Completed', 'RuntimeError', 'Invalid', 'LoggingIn', 'Initializing', 'History']

Example	<pre>{
	"AlgorithmStatus": "DeployError"
	}</pre>

Responses

The `/live/read` API provides a response in the following format:

200 Success

`LiveAlgorithmResults` Model - Details a live algorithm from the live/read API endpoint.

LiveResults	`LiveResultsData object` Holds information about the state and operation of the live running algorithm.
success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

<pre>{
"LiveResults": {
"version": 0,
"resolution": "10minute",
"results": {
"Holdings": {
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"Type": "Base",
"CurrencySymbol": "$",
"AveragePrice": 0,
"Quantity": 0,
"MarketPrice": 0,
"ConversionRate": 0,
"MarketValue": 0,
"UnrealizedPnl": 0
},
"Cash": {
"Symbol": "string",
"Amount": 0,
"ConversionRate": 0,
"CurrencySymbol": ,
"ValueInAccountCurrency": 0
},
"AlphaRuntimeStatistics": {
"MeanPopulationScore": {
"UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
"Direction": 0,
"Magnitude": 0,
"IsFinalScore": true
},
"RollingAveragedPopulationScore": {
"UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
"Direction": 0,
"Magnitude": 0,
"IsFinalScore": true
},
"LongCount": "string",
"ShortCount": "string",
"LongShortRatio": 0,
"TotalAccumulatedEstimatedAlphaValue": 0,
"KellyCriterionEstimate": 0,
"KellyCriterionProbabilityValue": 0,
"FitnessScore": 0,
"PortfolioTurnover": 0,
"ReturnOverMaxDrawdown": 0,
"SortinoRatio": 0,
"EstimatedMonthlyAlphaValue": 0,
"TotalInsightsGenerated": "string",
"TotalInsightsClosed": "string",
"TotalInsightsAnalysisCompleted": "string",
"MeanPopulationEstimatedInsightValue": 0
},
"Charts": {
"Name": "string",
"ChartType": "Overlay",
"Series": {
"Name": "string",
"Unit": "string",
"Index": 0,
"Values": [
{
"x": "string",
"y": 0</pre>

`LiveAlgorithmResults` Model - Details a live algorithm from the live/read API endpoint.

Example


```
}
],
"SeriesType": "Line",
"Color": "string",
"ScatterMarkerSymbol": "none"
}
},
"Orders": {
"Id": 0,
"ContingentId": 0,
"BrokerId": [
"string"
],
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"Price": 0,
"PriceCurrency": "string",
"Time": "2021-11-26T15:18:27.693Z",
"CreatedTime": "2021-11-26T15:18:27.693Z",
"LastFillTime": "2021-11-26T15:18:27.693Z",
"LastUpdateTime": "2021-11-26T15:18:27.693Z",
"CanceledTime": "2021-11-26T15:18:27.693Z",
"Quantity": 0,
"Type": "Market",
"Status": "New",
"Tag": "string",
"SecurityType": "Base",
"Direction": "Buy",
"Value": 0,
"OrderSubmissionData": {
"BidPrice": 0,
"AskPrice": 0,
"LastPrice": 0
},
"IsMarketable": true
},
"OrderEvents": [
{
"OrderId": 0,
"Id": 0,
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"UtcTime": "2021-11-26T15:18:27.693Z",
"Status": {
"OrderId": 0,
"Id": 0,
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"UtcTime": "2021-11-26T15:18:27.693Z",
"Status": "New",
"FillPrice": 0,
"FillPriceCurrency": "string",
"FillQuantity": 0,
"Direction": "Buy",
"Message": "string",
"IsAssignment": true,
"StopPrice": 0,
"LimitPrice": 0,
"Quantity": 0
},
"OrderFee": {
"Value": {
"Amount": 0,
"Currency": "string"
}
},
"FillPrice": 0,
"FillPriceCurrency": "string",
"FillQuantity": 0,
"Direction": "Buy",
"Message": "string",
"IsAssignment": true,
"StopPrice": 0,
"LimitPrice": 0,
"Quantity": 0
}
],
"ProfitLoss": "number",
"Statistics": "string",
"RuntimeStatistics": "string",
"ServerStatistics": "string"
}
},
"success": true,
"errors": [
"string"
]
}
```


`LiveList` Model - List of the live algorithms running which match the requested status.

Algorithms	`LiveAlgorithm Array`	Algorithm list matching the requested status.
success	`boolean`	Indicate if the API request was successful.
errors	`string Array`	List of errors with the API call.

Example


```
{
"Algorithms": [
{
"projectId": 0,
"deployId": "string",
"status": "DeployError",
"launched": "2021-11-26T15:18:27.693Z",
"stopped": "2021-11-26T15:18:27.693Z",
"brokerage": "Interactive",
"subscription": "string",
"error": "string",
"success": true,
"errors": [
"string"
]
}
],
"success": true,
"errors": [
"string"
]
}
```

`LiveResultsData` Model - Holds information about the state and operation of the live running algorithm.

version
integer
Results version.

resolution
string Enum
Storage format of the charting data. Options : ['10minute', 'minute', 'second']

results
LiveResult object
Live results object class for packaging live result data.

Example


```
{
  "version": 0,
  "resolution": "10minute",
  "results": {
    "Holdings": {
      "Symbol": {
        "Value": "string",
        "ID": "string",
        "Permtick": "string"
      },
      "Type": "Base",
      "CurrencySymbol": "$",
      "AveragePrice": 0,
      "Quantity": 0,
      "MarketPrice": 0,
      "ConversionRate": 0,
      "MarketValue": 0,
      "UnrealizedPnl": 0
    },
    "Cash": {
      "Symbol": "string",
      "Amount": 0,
      "ConversionRate": 0,
      "CurrencySymbol": ,
      "ValueInAccountCurrency": 0
    },
    "AlphaRuntimeStatistics": {
      "MeanPopulationScore": {
        "UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
        "Direction": 0,
        "Magnitude": 0,
        "IsFinalScore": true
      },
      "RollingAveragedPopulationScore": {
        "UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
        "Direction": 0,
        "Magnitude": 0,
        "IsFinalScore": true
      },
      "LongCount": "string",
      "ShortCount": "string",
      "LongShortRatio": 0,
      "TotalAccumulatedEstimatedAlphaValue": 0,
      "KellyCriterionEstimate": 0,
      "KellyCriterionProbabilityValue": 0,
      "FitnessScore": 0,
      "PortfolioTurnover": 0,
      "ReturnOverMaxDrawdown": 0,
      "SortinoRatio": 0,
      "EstimatedMonthlyAlphaValue": 0,
      "TotalInsightsGenerated": "string",
      "TotalInsightsClosed": "string",
      "TotalInsightsAnalysisCompleted": "string",
      "MeanPopulationEstimatedInsightValue": 0
    },
    "Charts": {
      "Name": "string",
      "ChartType": "Overlay",
      "Series": {
        "Name": "string",
        "Unit": "string",
        "Index": 0,
        "Values": [
          {
            "x": "string",
            "y": 0
          }
        ],
        "SeriesType": "Line",
        "Color": "string",
        "ScatterMarkerSymbol": "none"
      }
    },
    "Orders": {
      "Id": 0,
      "ContingentId": 0,
      "BrokerId": [
        "string"
      ],
      "Symbol": {
        "Value": "string",
        "ID": "string",
        "Permtick": "string"
      },
      "Price": 0,
      "PriceCurrency": "string",
      "Time": "2021-11-26T15:18:27.693Z",
      "CreatedTime": "2021-11-26T15:18:27.693Z",
      "LastFillTime": "2021-11-26T15:18:27.693Z",
      "LastUpdateTime": "2021-11-26T15:18:27.693Z",
      "CanceledTime": "2021-11-26T15:18:27.693Z",
      "Quantity": 0,
      "Type": "Market",
      "Status": "New",
      "Tag": "string",
      "SecurityType": "Base",
      "Direction": "Buy",
      "Value": 0,
      "OrderSubmissionData": {
        "BidPrice": 0,
        "AskPrice": 0,
        "LastPrice": 0
      },
      "IsMarketable": true
    },
    "OrderEvents": [
      {
        "OrderId": 0,
        "Id": 0,
        "Symbol": {
          "Value": "string",
          "ID": "string",
          "Permtick": "string"
        },
        "UtcTime": "2021-11-26T15:18:27.693Z",
        "Status": {
          "OrderId": 0,
          "Id": 0,
          "Symbol": {
            "Value": "string",
            "ID": "string",
```

`LiveResultsData` Model - Holds information about the state and operation of the live running algorithm.


```
"Permtick": "string"
},
"UtcTime": "2021-11-26T15:18:27.693Z",
"Status": "New",
"FillPrice": 0,
"FillPriceCurrency": "string",
"FillQuantity": 0,
"Direction": "Buy",
"Message": "string",
"IsAssignment": true,
"StopPrice": 0,
"LimitPrice": 0,
"Quantity": 0
},
"OrderFee": {
"Value": {
"Amount": 0,
"Currency": "string"
}
},
"FillPrice": 0,
"FillPriceCurrency": "string",
"FillQuantity": 0,
"Direction": "Buy",
"Message": "string",
"IsAssignment": true,
"StopPrice": 0,
"LimitPrice": 0,
"Quantity": 0
}
],
"ProfitLoss": "number",
"Statistics": "string",
"RuntimeStatistics": "string",
"ServerStatistics": "string"
}
}
```


`ChartResolution` Model - Storage format of the charting data

ChartResolution	`string Enum` Storage format of the charting data. Options : ['10minute', 'minute', 'second']

Example	`{`
	`"ChartResolution": "10minute"`
	`}`

`LiveResult` Model - Live results object class for packaging live result data.

Holdings	`Holding object` Dictionary of algorithm holdings information.
Cash	`Cash object` Represents a holding of a currency in cash.
AlphaRuntimeStatistics	`AlphaRuntimeStatistics object` Contains insight population run time statistics.
Charts	`Chart object` Charts updates for the live algorithm since the last result packet.
Orders	`Order object` Order updates since the last result packet.
OrderEvents	`OrderEvent Array` OrderEvent updates since the last result packet.
ProfitLoss	`number object` Trade profit and loss information since the last algorithm result packet.
Statistics	`string object` Statistics information sent during the algorithm operations.
RuntimeStatistics	`string object` Runtime banner/updating statistics in the title banner of the live algorithm GUI.
ServerStatistics	`string object` Server status information, including CPU and RAM usage.


```
{
"Holdings": {
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"Type": "Base",
"CurrencySymbol": "$",
"AveragePrice": 0,
"Quantity": 0,
"MarketPrice": 0,
"ConversionRate": 0,
"MarketValue": 0,
"UnrealizedPnl": 0
},
"Cash": {
"Symbol": "string",
"Amount": 0,
"ConversionRate": 0,
"CurrencySymbol": ,
"ValueInAccountCurrency": 0
},
"AlphaRuntimeStatistics": {
"MeanPopulationScore": {
"UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
"Direction": 0,
"Magnitude": 0,
"IsFinalScore": true
},
"RollingAveragedPopulationScore": {
"UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
"Direction": 0,
"Magnitude": 0,
"IsFinalScore": true
},
"LongCount": "string",
"ShortCount": "string",
"LongShortRatio": 0,
"TotalAccumulatedEstimatedAlphaValue": 0,
"KellyCriterionEstimate": 0,
"KellyCriterionProbabilityValue": 0,
"FitnessScore": 0,
"PortfolioTurnover": 0,
"ReturnOverMaxDrawdown": 0,
"SortinoRatio": 0,
"EstimatedMonthlyAlphaValue": 0,
"TotalInsightsGenerated": "string",
"TotalInsightsClosed": "string",
"TotalInsightsAnalysisCompleted": "string",
"MeanPopulationEstimatedInsightValue": 0
},
"Charts": {
"Name": "string",
"ChartType": "Overlay",
"Series": {
"Name": "string",
"Unit": "string",
```

`LiveResult` Model - Live results object class for packaging live result data.

Example


```
"Index": 0,
"Values": [
{
"x": "string",
"y": 0
}
],
"SeriesType": "Line",
"Color": "string",
"ScatterMarkerSymbol": "none"
}
},
"Orders": {
"Id": 0,
"ContingentId": 0,
"BrokerId": [
"string"
],
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"Price": 0,
"PriceCurrency": "string",
"Time": "2021-11-26T15:18:27.693Z",
"CreatedTime": "2021-11-26T15:18:27.693Z",
"LastFillTime": "2021-11-26T15:18:27.693Z",
"LastUpdateTime": "2021-11-26T15:18:27.693Z",
"CanceledTime": "2021-11-26T15:18:27.693Z",
"Quantity": 0,
"Type": "Market",
"Status": "New",
"Tag": "string",
"SecurityType": "Base",
"Direction": "Buy",
"Value": 0,
"OrderSubmissionData": {
"BidPrice": 0,
"AskPrice": 0,
"LastPrice": 0
},
"IsMarketable": true
},
"OrderEvents": [
{
"OrderId": 0,
"Id": 0,
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"UtcTime": "2021-11-26T15:18:27.693Z",
"Status": {
"OrderId": 0,
"Id": 0,
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"UtcTime": "2021-11-26T15:18:27.693Z",
"Status": "New",
"FillPrice": 0,
"FillPriceCurrency": "string",
"FillQuantity": 0,
"Direction": "Buy",
"Message": "string",
"IsAssignment": true,
"StopPrice": 0,
"LimitPrice": 0,
"Quantity": 0
},
"OrderFee": {
"Value": {
"Amount": 0,
"Currency": "string"
}
},
"FillPrice": 0,
"FillPriceCurrency": "string",
"FillQuantity": 0,
"Direction": "Buy",
"Message": "string",
"IsAssignment": true,
"StopPrice": 0,
"LimitPrice": 0,
"Quantity": 0
}
],
"ProfitLoss": "number",
"Statistics": "string",
"RuntimeStatistics": "string",
"ServerStatistics": "string"
}
```


`Holding` Model - Live results object class for packaging live result data.

Symbol	`Symbol object`	Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
Type	`string Enum`	Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']
CurrencySymbol	`string` *example: $*	The currency symbol of the holding.
AveragePrice	`number`	Average Price of our Holding in the currency the symbol is traded in.
Quantity	`number`	Quantity of the Symbol we hold.
MarketPrice	`number`	Current Market Price of the Asset in the currency the symbol is traded in.
ConversionRate	`number`	Current market conversion rate into the account currency.
MarketValue	`number`	Current market value of the holding.
UnrealizedPnl	`number`	Current unrealized P/L of the holding.


```
{
"Symbol": {
"Value": "string",
"ID": "string",
"Permtick": "string"
},
"Type": "Base",
"CurrencySymbol": "$",
```

Example

`Holding` Model - Live results object class for packaging live result data.


```
"AveragePrice": 0,
"Quantity": 0,
"MarketPrice": 0,
"ConversionRate": 0,
"MarketValue": 0,
"UnrealizedPnl": 0
}
```


`Symbol` Model - Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.

Value	`string`	The current symbol for this ticker.
ID	`string`	The security identifier for this symbol.
Permtick	`string`	The current symbol for this ticker.

Example


```
{
"Value": "string",
"ID": "string",
"Permtick": "string"
}
```


`SecurityType` Model - Type of tradable security / underlying asset.

SecurityType	`string Enum`	Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']

Example


```
{
"SecurityType": "Base"
}
```


`Cash` Model - Represents a holding of a currency in cash.

Symbol	`string`	Gets the symbol used to represent this cash.
Amount	`number`	Gets or sets the amount of cash held.
ConversionRate	`number`	The currency conversion rate to the account base currency.
CurrencySymbol	`object`	The symbol of the currency, such as $.
ValueInAccountCurrency	`number`	The value of the currency cash in the account base currency.

Example


```
{
"Symbol": "string",
"Amount": 0,
"ConversionRate": 0,
"CurrencySymbol": ,
"ValueInAccountCurrency": 0
}
```


`AlphaRuntimeStatistics` Model - Contains insight population run time statistics.

MeanPopulationScore	`InsightScore object`	Defines the scores given to a particular insight.
RollingAveragedPopulationScore	`InsightScore object`	Defines the scores given to a particular insight.
LongCount	`string`	Gets the total number of insights with an up direction.
ShortCount	`string`	Gets the total number of insights with a down direction.
LongShortRatio	`number`	The ratio of InsightDirection.Up over InsightDirection.Down.
TotalAccumulatedEstimatedAlphaValue	`number`	The total accumulated estimated value of trading all insights.
KellyCriterionEstimate	`number`	Score of the strategy's insights predictive power.
KellyCriterionProbabilityValue	`number`	The p-value or probability value of the KellyCriterionEstimate.
FitnessScore	`number`	Score of the strategy's performance, and suitability for the Alpha Stream Market.
PortfolioTurnover	`number`	Measurement of the strategies trading activity with respect to the portfolio value. Calculated as the sales volume with respect to the average total portfolio value.
ReturnOverMaxDrawdown	`number`	Provides a risk adjusted way to factor in the returns and drawdown of the strategy. It is calculated by dividing the Portfolio Annualized Return by the Maximum Drawdown seen during the backtest.
SortinoRatio	`number`	Gives a relative picture of the strategy volatility. It is calculated by taking a portfolio's annualized rate of return and subtracting the risk free rate of return.
EstimatedMonthlyAlphaValue	`number`	Suggested Value of the Alpha On A Monthly Basis For Licensing.
TotalInsightsGenerated	`string`	The total number of insight signals generated by the algorithm.
TotalInsightsClosed	`string`	The total number of insight signals generated by the algorithm.
TotalInsightsAnalysisCompleted	`string`	The total number of insight signals generated by the algorithm.
MeanPopulationEstimatedInsightValue	`number`	Gets the mean estimated insight value.

Example


```
{
"MeanPopulationScore": {
"UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
"Direction": 0,
"Magnitude": 0,
"IsFinalScore": true
},
"RollingAveragedPopulationScore": {
"UpdatedTimeUtc": "2021-11-26T15:18:27.693Z",
"Direction": 0,
"Magnitude": 0,
"IsFinalScore": true
},
"LongCount": "string",
"ShortCount": "string",
"LongShortRatio": 0,
"TotalAccumulatedEstimatedAlphaValue": 0,
"KellyCriterionEstimate": 0,
"KellyCriterionProbabilityValue": 0,
"FitnessScore": 0,
"PortfolioTurnover": 0,
"ReturnOverMaxDrawdown": 0,
"SortinoRatio": 0,
"EstimatedMonthlyAlphaValue": 0,
"TotalInsightsGenerated": "string",
"TotalInsightsClosed": "string",
"TotalInsightsAnalysisCompleted": "string",
"MeanPopulationEstimatedInsightValue": 0
}
```


`InsightScore` Model - Defines the scores given to a particular insight

UpdatedTimeUtc	`string($date-time)`	The time these scores were last updated.

`InsightScore` Model - Defines the scores given to a particular insight

Direction	number	The direction score.
Magnitude	number	The magnitude score.
IsFinalScore	boolean	Is the insight past its expiry time and score can be finalized.
Example		{ "UpdatedTimeUtc": "2021-11-26T15:18:27.693Z", "Direction": 0, "Magnitude": 0, "IsFinalScore": true }

`Chart` Model - Single Parent Chart Object for Custom Charting.

Name	string	Name of the Chart.
ChartType	string Enum	Type of the Chart, Overlayed or Stacked. Options : ['Overlay', 'Stacked']
Series	Series object	List of Series Objects for this Chart.
Example		{ "Name": "string", "ChartType": "Overlay", "Series": { "Name": "string", "Unit": "string", "Index": 0, "Values": [{ "x": "string", "y": 0 }], "SeriesType": "Line", "Color": "string", "ScatterMarkerSymbol": "none" } }

`Series` Model - Chart Series Object - Series data and properties for a chart.

Name	string	Name of the series.
Unit	string	Axis for the chart series.
Index	integer	Index/position of the series on the chart.
Values	ChartPoint Array	Values for the series plot. These values are assumed to be in ascending time order (first points earliest, last points latest).
SeriesType	string Enum	Chart type for the series. Options : ['Line', 'Scatter', 'Candle', 'Bar', 'Flag', 'StackedArea', 'Pie', 'Treemap']
Color	string	Color the series.
ScatterMarkerSymbol	string Enum	Shape or symbol for the marker in a scatter plot. Options : ['none', 'circle', 'square', 'diamond', 'triangle', 'triangle-down']
Example		{ "Name": "string", "Unit": "string", "Index": 0, "Values": [{ "x": "string", "y": 0 }], "SeriesType": "Line", "Color": "string", "ScatterMarkerSymbol": "none" }

`ChartPoint` Model - Location on a chart containing the X-Y location

x	string	Time of this chart point: lower case for javascript encoding simplicty.
y	number	Value of this chart point: lower case for javascript encoding simplicty.
Example		{ "x": "string", "y": 0 }

`Order` Model - Order struct for placing new trade.

Id	integer	Order ID.
ContingentId	integer	Order Id to process before processing this order.
BrokerId	string Array	Brokerage Id for this order for when the brokerage splits orders into multiple pieces.
Symbol	Symbol object	Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
Price	number	Price of the Order.
PriceCurrency	string	Currency for the order price.
Time	string($date-time)	Gets the utc time the order was created.
CreatedTime	string($date-time)	Gets the utc time this order was created. Alias for Time.
LastFillTime	string($date-time)	Gets the utc time the last fill was received, or null if no fills have been received.
LastUpdateTime	string($date-time)	Gets the utc time this order was last updated, or null if the order has not been updated.
CanceledTime	string($date-time)	Gets the utc time this order was canceled, or null if the order was not canceled.
Quantity	number	Number of shares to execute.
Type	string Enum	Order type. Options : ['Market', 'Limit', 'StopMarket', 'StopLimit', 'MarketOnOpen', 'MarketOnClose', 'OptionExercise']
Status	string Enum	Status of the Order. Options : ['New', 'Submitted', 'PartiallyFilled', 'Filled', 'Canceled', 'None', 'Invalid', 'CancelPending', 'UpdateSubmitted']
Tag	string	Tag the order with some custom data.
SecurityType	string Enum	Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']
Direction	string Enum	

`Order` Model - Order struct for placing new trade.

Value	number Gets the executed value of this order. If the order has not yet filled, then this will return zero.
OrderSubmissionData	OrderSubmissionData object Stores time and price information available at the time an order was submitted.
IsMarketable	boolean Returns true if the order is a marketable order.

Example


```
{
  "Id": 0,
  "ContingentId": 0,
  "BrokerId": [
    "string"
  ],
  "Symbol": {
    "Value": "string",
    "ID": "string",
    "Permtick": "string"
  },
  "Price": 0,
  "PriceCurrency": "string",
  "Time": "2021-11-26T15:18:27.693Z",
  "CreatedTime": "2021-11-26T15:18:27.693Z",
  "LastFillTime": "2021-11-26T15:18:27.693Z",
  "LastUpdateTime": "2021-11-26T15:18:27.693Z",
  "CanceledTime": "2021-11-26T15:18:27.693Z",
  "Quantity": 0,
  "Type": "Market",
  "Status": "New",
  "Tag": "string",
  "SecurityType": "Base",
  "Direction": "Buy",
  "Value": 0,
  "OrderSubmissionData": {
    "BidPrice": 0,
    "AskPrice": 0,
    "LastPrice": 0
  },
  "IsMarketable": true
}
```


`OrderDirection` Model - Direction of the order.

OrderDirection	string Enum Direction of the order. Options : ['Buy', 'Sell', 'Hold']

Example


```
{
  "OrderDirection": "Buy"
}
```


`OrderSubmissionData` Model - Stores time and price information available at the time an order was submitted.

BidPrice	number The bid price at an order submission time.
AskPrice	number The ask price at an order submission time.
LastPrice	number The current price at an order submission time.

Example


```
{
  "BidPrice": 0,
  "AskPrice": 0,
  "LastPrice": 0
}
```


`OrderEvent` Model - Change in an order state applied to user algorithm portfolio

OrderId	integer Id of the order this event comes from.
Id	integer The unique order event Id for each order.
Symbol	Symbol object Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
UtcTime	string($date-time) The date and time of this event (UTC).
Status	OrderStatus object Messaging class signifying a change in an order state and record the change in the users algorithm portfolio.
OrderFee	OrderFee object The order fee associated with the specified order.
FillPrice	number Fill price information about the order.
FillPriceCurrency	string Currency for the fill price.
FillQuantity	number Number of shares of the order that was filled in this event.
Direction	string Enum Direction of the order. Options : ['Buy', 'Sell', 'Hold']
Message	string Any message from the exchange.
IsAssignment	boolean True if the order event is an assignment.
StopPrice	number The current stop price.
LimitPrice	number The current limit price.
Quantity	number The current order quantity.

Example


```
{
  "OrderId": 0,
  "Id": 0,
  "Symbol": {
    "Value": "string",
    "ID": "string",
    "Permtick": "string"
  },
  "UtcTime": "2021-11-26T15:18:27.693Z",
  "Status": {
    "OrderId": 0,
    "Id": 0,
    "Symbol": {
      "Value": "string",
      "ID": "string",
      "Permtick": "string"
    },
    "UtcTime": "2021-11-26T15:18:27.693Z",
    "Status": "New",
    "FillPrice": 0,
    "FillPriceCurrency": "string",
    "FillQuantity": 0,
    "Direction": "Buy",
    "Message": "string",
    "IsAssignment": true,
    "StopPrice": 0,
    "LimitPrice": 0,
    "Quantity": 0
```

`OrderEvent` Model - Change in an order state applied to user algorithm portfolio

OrderId	integer Id of the order this event comes from.
Id	integer The unique order event Id for this order.
Symbol	Symbol object Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
UtcTime	string($date-time) The date and time of this event.
Status	string Enum Status of the Order. Options : ['New', 'Submitted', 'PartiallyFilled', 'Filled', 'Canceled', 'None', 'Invalid', 'CancelPending', 'UpdateSubmitted']
FillPrice	number Fill price information about the order.
FillPriceCurrency	string Currency for the fill price.
FillQuantity	number Number of shares of the order that was filled in this event.
Direction	string Enum Direction of the order. Options : ['Buy', 'Sell', 'Hold']
Message	string Any message from the exchange.
IsAssignment	boolean Order event is an allocation of trades from ITM option assignment.
StopPrice	number The current stop price.
LimitPrice	number The current limit price.
Quantity	number The current order quantity.

Example	{
	"OrderId": 0,
	"Id": 0,
	"Symbol": {
	"Value": "string",
	"ID": "string",
	"Permtick": "string"
	},
	"UtcTime": "2021-11-26T15:18:27.693Z",
	"Status": "New",
	"FillPrice": 0,
	"FillPriceCurrency": "string",
	"FillQuantity": 0,
	"Direction": "Buy",
	"Message": "string",
	"IsAssignment": true,
	"StopPrice": 0,
	"LimitPrice": 0,
	"Quantity": 0
	}

`OrderStatus` Model - Messaging class signifying a change in an order state and record the change in the users algorithm portfolio.

OrderId	integer Id of the order this event comes from.
Id	integer The unique order event Id for this order.
Symbol	Symbol object Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
UtcTime	string($date-time) The date and time of this event.
Status	string Enum Status of the Order. Options : ['New', 'Submitted', 'PartiallyFilled', 'Filled', 'Canceled', 'None', 'Invalid', 'CancelPending', 'UpdateSubmitted']
FillPrice	number Fill price information about the order.
FillPriceCurrency	string Currency for the fill price.
FillQuantity	number Number of shares of the order that was filled in this event.
Direction	string Enum Direction of the order. Options : ['Buy', 'Sell', 'Hold']
Message	string Any message from the exchange.
IsAssignment	boolean Order event is an allocation of trades from ITM option assignment.
StopPrice	number The current stop price.
LimitPrice	number The current limit price.
Quantity	number The current order quantity.

Example	{
	"OrderId": 0,
	"Id": 0,
	"Symbol": {
	"Value": "string",
	"ID": "string",
	"Permtick": "string"
	},
	"UtcTime": "2021-11-26T15:18:27.693Z",
	"Status": "New",
	"FillPrice": 0,
	"FillPriceCurrency": "string",
	"FillQuantity": 0,
	"Direction": "Buy",
	"Message": "string",
	"IsAssignment": true,
	"StopPrice": 0,
	"LimitPrice": 0,
	"Quantity": 0
	}

`Status` Model - Status of the Order.

Status	string Enum Status of the Order. Options : ['New', 'Submitted', 'PartiallyFilled', 'Filled', 'Canceled', 'None', 'Invalid', 'CancelPending', 'UpdateSubmitted']

Example	{
	"Status": "New"
	}

`OrderFee` Model - The order fee associated with the specified order.

Value	CashAmount object Represents a cash amount which can be converted to account currency using a currency converter.

Example	{
	"Value": {
	"Amount": 0,
	"Currency": "string"
	}
	}

`CashAmount` Model - Represents a cash amount which can be converted to account currency using a currency converter.

Amount	number The amount of cash.
Currency	string The currency in which the cash amount is denominated.

Example	{
	"Amount": 0,
	"Currency": "string"
	}

`LiveAlgorithm` Model - Live algorithm instance result from the QuantConnect Rest API.

projectId	integer Project Id for the live instance.
deployId	string Unique live algorithm deployment identifier (similar to a backtest id).
status	string Enum States of a live deployment. Options : ['DeployError', 'InQueue', 'Running', 'Stopped', 'Liquidated', 'Deleted', 'Completed', 'RuntimeError', 'Invalid', 'LoggingIn', 'Initializing', 'History']
launched	string($date-time) Datetime the algorithm was launched in UTC.
stopped	string($date-time) Datetime the algorithm was stopped in UTC, null if its still running.
brokerage	string Enum Brokerage. Options : ['Interactive', 'FXCM', 'Oanda', 'Tradier', 'PaperTrading', 'Alpaca', 'Bitfinex', 'Binance', 'GDAX']
subscription	string Chart we're subscribed to.
error	string Live algorithm error message from a crash or algorithm runtime error.
success	boolean Indicate if the API request was successful.
	string Array

errors

`LiveAlgorithm` Model - Live algorithm instance result from the QuantConnect Rest API.

List of errors with the API call.

Example


```
{
  "projectId": 0,
  "deployId": "string",
  "status": "DeployError",
  "launched": "2021-11-26T15:18:27.693Z",
  "stopped": "2021-11-26T15:18:27.693Z",
  "brokerage": "Interactive",
  "subscription": "string",
  "error": "string",
  "success": true,
  "errors": [
    "string"
  ]
}
```


`AlgorithmStatus` Model - States of a live deployment.

AlgorithmStatus

`string Enum`

States of a live deployment. Options : ['DeployError', 'InQueue', 'Running', 'Stopped', 'Liquidated', 'Deleted', 'Completed', 'RuntimeError', 'Invalid', 'LoggingIn', 'Initializing', 'History']

Example


```
{
  "AlgorithmStatus": "DeployError"
}
```


401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate

`string`
Header

14.6.2.2 Logs

Introduction

Get the logs of a specific live algorithm.

Request

Information about the algorithm to read live logs from. The `/live/read/log` API accepts requests in the following format:

`ReadLiveLogsRequest` Model - Request to read the logs of a specific algorithm.

format	`object` *example: json* Format of the log results.
projectId	`integer` Project Id of the live running algorithm.
algorithmId	`string` Deploy Id (Algorithm Id) of the live running algorithm.
start	`integer` No logs will be returned before this unixtime.
end	`integer` No logs will be returned after this unixtime.

Example	`{`
	`"format": "json",`
	`"projectId": 0,`
	`"algorithmId": "string",`
	`"start": 0,`
	`"end": 0`
	`}`

Responses

The `/live/read/log` API provides a response in the following format:

200 Success

`ReadLiveLogsResponse` Model - Logs from a live algorithm.

LiveLogs	`string Array` List of logs from the live algorithm.
success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	`{`
	`"LiveLogs": [`
	`"string"`
	`],`
	`"success": true,`
	`"errors": [`
	`"string"`
	`]`
	`}`

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header

14.6.2.3 Portfolio State

Introduction

Read out the portfolio state of a live algorithm.

Request

Fetch the live portfolio state for the project Id provided. The `/live/read/portfolio` API accepts requests in the following format:

`ReadLivePortfolioRequest` Model - Request to read the portfolio state from a live algorithm.

projectId	`integer` Id of the project from which to read the live algorithm.

Example	{
	"projectId": 0
	}

Responses

The `/live/read/portfolio` API provides a response in the following format:

200 Success

`LivePortfolioResponse` Model - Contains holdings and cash of the live algorithm in the request criteria.

portfolio	`Portfolio object` /.

Example	{
	"portfolio": {
	"Holdings": {
	"Symbol": {
	"Value": "string",
	"ID": "string",
	"Permtick": "string"
	},
	"Type": "Base",
	"CurrencySymbol": "$",
	"AveragePrice": 0,
	"Quantity": 0,
	"MarketPrice": 0,
	"ConversionRate": 0,
	"MarketValue": 0,
	"UnrealizedPnl": 0
	},
	"Cash": {
	"Symbol": "string",
	"Amount": 0,
	"ConversionRate": 0,
	"CurrencySymbol": ,
	"ValueInAccountCurrency": 0
	}
	}
	}

`Portfolio` Model

Holdings	`Holding object` Dictionary of algorithm holdings information.
Cash	`Cash object` Represents a holding of a currency in cash.

Example	{
	"Holdings": {
	"Symbol": {
	"Value": "string",
	"ID": "string",
	"Permtick": "string"
	},
	"Type": "Base",
	"CurrencySymbol": "$",
	"AveragePrice": 0,
	"Quantity": 0,
	"MarketPrice": 0,
	"ConversionRate": 0,
	"MarketValue": 0,
	"UnrealizedPnl": 0
	},
	"Cash": {
	"Symbol": "string",
	"Amount": 0,
	"ConversionRate": 0,
	"CurrencySymbol": ,
	"ValueInAccountCurrency": 0
	}
	}

`Holding` Model - Live results object class for packaging live result data.

Symbol	`Symbol object` Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
Type	`string Enum` Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']
CurrencySymbol	`string` *example: $* The currency symbol of the holding.
AveragePrice	`number` Average Price of our Holding in the currency the symbol is traded in.
Quantity	`number` Quantity of the Symbol we hold.
MarketPrice	`number` Current Market Price of the Asset in the currency the symbol is traded in.
ConversionRate	`number` Current market conversion rate into the account currency.
MarketValue	`number` Current market value of the holding.
UnrealizedPnl	`number` Current unrealized P/L of the holding.

Example	{
	"Symbol": {
	"Value": "string",
	"ID": "string",
	"Permtick": "string"
	},
	"Type": "Base",
	"CurrencySymbol": "$",
	"AveragePrice": 0,
	"Quantity": 0,
	"MarketPrice": 0,
	"ConversionRate": 0,
	"MarketValue": 0,

`Holding` Model - Live results object class for packaging live result data.

```
"UnrealizedPnl": 0
}
```

`Symbol` Model - Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.

Value	string	The current symbol for this ticker.
ID	string	The security identifier for this symbol.
Permtick	string	The current symbol for this ticker.

Example

```
{
  "Value": "string",
  "ID": "string",
  "Permtick": "string"
}
```

`SecurityType` Model - Type of tradable security / underlying asset.

SecurityType	string Enum	Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']

Example

```
{
  "SecurityType": "Base"
}
```

`Cash` Model - Represents a holding of a currency in cash.

Symbol	string	Gets the symbol used to represent this cash.
Amount	number	Gets or sets the amount of cash held.
ConversionRate	number	The currency conversion rate to the account base currency.
CurrencySymbol	object	The symbol of the currency, such as $.
ValueInAccountCurrency	number	The value of the currency cash in the account base currency.

Example

```
{
  "Symbol": "string",
  "Amount": 0,
  "ConversionRate": 0,
  "CurrencySymbol": ,
  "ValueInAccountCurrency": 0
}
```

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	string	Header

14.6.2.4 Orders

Introduction

Read out the orders of a live algorithm.

Request

Fetch the orders of a live algorithm for the project Id and steps provided. The `/live/read/orders` API accepts requests in the following format:

`ReadLiveOrdersRequest` Model - Request to read orders from a live algorithm.

start	integer	Starting index of the orders to be fetched. Required if end > 100.
end	integer	Last index of the orders to be fetched. Note that end - start must be less than 100.
projectId	integer	Id of the project from which to read the live algorithm.

Example

```
{
  "start": 0,
  "end": 0,
  "projectId": 0
}
```

Responses

The `/live/read/orders` API provides a response in the following format:

200 Success

`LiveOrdersResponse` Model - Contains orders and the number of orders of the live algorithm in the request criteria.

orders	Order object	/.
Length	integer	Total number of returned orders.
latestOrderTimestamp	integer	Timestamp of the latest order event.

Example

```
{
  "orders": {
    "Id": 0,
    "ContingentId": 0,
    "BrokerId": [
      "string"
    ],
    "Symbol": {
      "Value": "string",
      "ID": "string",
      "Permtick": "string"
    },
    "Price": 0,
    "PriceCurrency": "string",
    "Time": "2021-11-26T15:18:27.693Z",
    "CreatedTime": "2021-11-26T15:18:27.693Z",
    "LastFillTime": "2021-11-26T15:18:27.693Z",
    "LastUpdateTime": "2021-11-26T15:18:27.693Z",
    "CanceledTime": "2021-11-26T15:18:27.693Z",
    "Quantity": 0,
    "Type": "Market",
    "Status": "New",
    "Tag": "string",
    "SecurityType": "Base",
    "Direction": "Buy",
    "Value": 0,
    "OrderSubmissionData": {
      "BidPrice": 0,
      "AskPrice": 0,
      "LastPrice": 0
    },
    "IsMarketable": true
  },
  "Length": 0,
  "latestOrderTimestamp": 0
}
```

`Order` Model - Order struct for placing new trade.

Id	integer	Order ID.
ContingentId	integer	Order Id to process before processing this order.
BrokerId	string Array	Brokerage Id for this order for when the brokerage splits orders into multiple pieces.
Symbol	Symbol object	Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.
Price	number	Price of the Order.
PriceCurrency	string	Currency for the order price.
Time	string($date-time)	Gets the utc time the order was created.
CreatedTime	string($date-time)	Gets the utc time this order was created. Alias for Time.
LastFillTime	string($date-time)	Gets the utc time the last fill was received, or null if no fills have been received.
LastUpdateTime	string($date-time)	Gets the utc time this order was last updated, or null if the order has not been updated.
CanceledTime	string($date-time)	Gets the utc time this order was canceled, or null if the order was not canceled.
Quantity	number	Number of shares to execute.
Type	string Enum	Order type. Options : ['Market', 'Limit', 'StopMarket', 'StopLimit', 'MarketOnOpen', 'MarketOnClose', 'OptionExercise']
Status	string Enum	Status of the Order. Options : ['New', 'Submitted', 'PartiallyFilled', 'Filled', 'Canceled', 'None', 'Invalid', 'CancelPending', 'UpdateSubmitted']
Tag	string	Tag the order with some custom data.
SecurityType	string Enum	Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']
Direction	string Enum	Direction of the order. Options : ['Buy', 'Sell', 'Hold']
Value	number	Gets the executed value of this order. If the order has not yet filled, then this will return zero.
	OrderSubmissionData object	

OrderSubmissionData

Stores time and price information available at the time an order was submitted.

IsMarketable

boolean

Returns true if the order is a marketable order.

Example


```
{
  "Id": 0,
  "ContingentId": 0,
  "BrokerId": [
    "string"
  ],
  "Symbol": {
    "Value": "string",
    "ID": "string",
    "Permtick": "string"
  },
  "Price": 0,
  "PriceCurrency": "string",
  "Time": "2021-11-26T15:18:27.693Z",
  "CreatedTime": "2021-11-26T15:18:27.693Z",
  "LastFillTime": "2021-11-26T15:18:27.693Z",
  "LastUpdateTime": "2021-11-26T15:18:27.693Z",
  "CanceledTime": "2021-11-26T15:18:27.693Z",
  "Quantity": 0,
  "Type": "Market",
  "Status": "New",
  "Tag": "string",
  "SecurityType": "Base",
  "Direction": "Buy",
  "Value": 0,
  "OrderSubmissionData": {
    "BidPrice": 0,
    "AskPrice": 0,
    "LastPrice": 0
  },
  "IsMarketable": true
}
```


`Symbol` Model - Represents a unique security identifier. This is made of two components, the unique SID and the Value. The value is the current ticker symbol while the SID is constant over the life of a security.

Value

string

The current symbol for this ticker.

ID

string

The security identifier for this symbol.

Permtick

string

The current symbol for this ticker.

Example


```
{
  "Value": "string",
  "ID": "string",
  "Permtick": "string"
}
```


`SecurityType` Model - Type of tradable security / underlying asset.

SecurityType

string Enum

Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']

Example


```
{
  "SecurityType": "Base"
}
```


`OrderDirection` Model - Direction of the order.

OrderDirection

string Enum

Direction of the order. Options : ['Buy', 'Sell', 'Hold']

Example


```
{
  "OrderDirection": "Buy"
}
```


`OrderSubmissionData` Model - Stores time and price information available at the time an order was submitted.

BidPrice

number

The bid price at an order submission time.

AskPrice

number

The ask price at an order submission time.

LastPrice

number

The current price at an order submission time.

Example


```
{
  "BidPrice": 0,
  "AskPrice": 0,
  "LastPrice": 0
}
```


401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate

string

Header

14.6.3 Update Live Algorithm

The QuantConnect REST API lets you update your live algorithms on our cloud servers through URL endpoints.

Liquidate Live Portfolio

Stop Live Algorithm

14.6.3.1 Liquidate Live Portfolio

Introduction

Liquidate a live algorithm from the specified project and deployId.

Request

Information about the live algorithm to liquidate. The `/live/update/liquidate` API accepts requests in the following format:

`LiquidateLiveAlgorithmRequest` Model - Request to liquidate a live algorithm.

projectId	`integer` Project Id for the live instance we want to liquidate.

Example	`{`
	`"projectId": 0`
	`}`

Responses

The `/live/update/liquidate` API provides a response in the following format:

200 Success

`RestResponse` Model - Base API response class for the QuantConnect API.

success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	`{`
	`"success": true,`
	`"errors": [`
	`"string"`
	`]`
	`}`

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header

14.6.3.2 Stop Live Algorithm

Introduction

Stop a live algorithm from the specified project and deployId.

Request

Information about the project to delete. The `/live/update/stop` API accepts requests in the following format:

`StopLiveAlgorithmRequest` **Model - Request to stop a live algorithm.**

projectId	`integer` Project Id for the live instance we want to stop.

Example	<pre>{
	"projectId": 0
	}</pre>

Responses

The `/live/update/stop` API provides a response in the following format:

200 Success

`RestResponse` **Model - Base API response class for the QuantConnect API.**

success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	<pre>{
	"success": true,
	"errors": [
	"string"
	]
	}</pre>

401 Authentication Error

`UnauthorizedError` **Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.**

www_authenticate	`string` Header	

14.7 Downloading Data

The QuantConnect REST API lets you generate links to download data from our cloud servers through URL endpoints. To use the CLI to download data, see Downloading Data .

Read Downloaded Data

14.7.1 Read Downloaded Data

Introduction

Get the link to the downloadable data.

Request

The `/data/read` API accepts requests in the following format:

`DataDownloadRequest` Model - Request for a link to downloadable data.

format	object *example: link*	Format for returning data, link or download.
ticker	string	/.
type	string Enum	Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']
resolution	string Enum	Resolution of data requested. Options : ['Tick', 'Second', 'Minute', 'Hour', 'Daily']
market	string Enum	Fungible market of the underlying security. Options : ['usa', 'oanda', 'fxcm', 'dukascopy', 'bitfinex', 'cmeglobex', 'nymex', 'cbot', 'ice', 'cboe', 'nse', 'comex', 'cme', 'sgx', 'hkfe', 'gdax', 'kraken', 'bitstamp', 'okcoin', 'bithumb', 'binance', 'poloniex', 'coinone', 'hitbtc', 'bittrex']
date	string *example: yyyyMMdd*	Date of the data requested yyyyMMdd.

Example		{
		"format": "link",
		"ticker": "string",
		"type": "Base",
		"resolution": "Tick",
		"market": "usa",
		"date": "yyyyMMdd"
		}

`SecurityType` Model - Type of tradable security / underlying asset.

SecurityType	string Enum	Type of tradable security / underlying asset. Options : ['Base', 'Equity', 'Option', 'Commodity', 'Forex', 'Future', 'Cfd', 'Crypto']

Example		{
		"SecurityType": "Base"
		}

`Resolution` Model - Resolution of data requested.

Resolution	string Enum	Resolution of data requested. Options : ['Tick', 'Second', 'Minute', 'Hour', 'Daily']

Example		{
		"Resolution": "Tick"
		}

`Market` Model - Fungible market of the underlying security.

Market	string Enum	Fungible market of the underlying security. Options : ['usa', 'oanda', 'fxcm', 'dukascopy', 'bitfinex', 'cmeglobex', 'nymex', 'cbot', 'ice', 'cboe', 'nse', 'comex', 'cme', 'sgx', 'hkfe', 'gdax', 'kraken', 'bitstamp', 'okcoin', 'bithumb', 'binance', 'poloniex', 'coinone', 'hitbtc', 'bittrex']

Example		{
		"Market": "usa"
		}

Responses

The `/data/read` API provides a response in the following format:

200 Success

`ReadDataLinkResponse` Model - Response from reading purchased data.

link	string	Link to the data.
success	boolean	Indicate if the API request was successful.
errors	string Array	List of errors with the API call.

Example		{
		"link": "string",
		"success": true,
		"errors": [
		"string"
		]
		}

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	string	Header

14.8 Reports

The QuantConnect REST API lets you access your backtest reports from our cloud servers through URL endpoints.

Backtest Report

14.8.1 Backtest Report

Introduction

Read out the report of a backtest in the project Id specified

Request

A JSON object containing info about the project to delete. The `/backtests/read/report` API accepts requests in the following format:

`BacktestReportRequest` Model - Request to read out the report of a backtest.

projectId	`integer` Id of the project to read.
backtestId	`string` Specific backtest Id to read.

Example	<pre>{
	"projectId": 0,
	"backtestId": "string"
	}</pre>

Responses

The `/backtests/read/report` API provides a response in the following format:

200 Success

`BacktestReport` Model - Backtest Report Response wrapper.

report	`string` HTML data of the report with embedded base64 images.
success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	<pre>{
	"report": "string",
	"success": true,
	"errors": [
	"string"
	]
	}</pre>

`RequestFailedError` Model - The API method call could not be completed as requested.

success	`boolean` Indicate if the API request was successful.
errors	`string Array` List of errors with the API call.

Example	<pre>{
	"success": true,
	"errors": [
	"string"
	]
	}</pre>

401 Authentication Error

`UnauthorizedError` Model - Unauthorized response from the API. Key is missing, invalid, or timestamp is too old for hash.

www_authenticate	`string` Header