Building a Trading System

In the initial chapters of this book, we learned how to create a trading strategy by analyzing historical data. In this chapter, we are going to study how to convert data analysis into realtime software that will connect to a real exchange to actually apply the theory that you've
previously learned.

We will describe the functional components supporting the trading strategy based on the algorithm created in the previous chapters. We will be using Python to build a small trading system. We will use the algorithms to build a trading system capable of trading.

This chapter will cover the following topics:

- Understanding the trading system
- Building a trading system in Python
- Designing a limit order book

Understanding the trading system

A trading system will help you to automate your trading strategy. When you choose to build this kind of software, you need to take the following into consideration:

- **Asset class**: When you code, knowing which asset class will be used in your trading system will modify the data structure of this software. Each asset class is idiosyncratic and has its own set of features. US stocks are mainly traded on two exchanges (NY Stock Exchange and NASDAQ). There are about 6,000 companies (symbols) listed on these two exchanges. Unlike equities, **Foreign Exchange** (**FX**) has six major currency pairs, six minor currency pairs, and six more exotic currency pairs. We can add more currency pairs, but there will not be more than 100 currency pairs. However, there will be hundreds of market players (banks, brokers).
- **Trading strategy type (high frequency, long-term position)**: Depending upon the type of strategies, the design of the software architecture will be impacted. High-frequency trading strategies require sending orders very rapidly. A regular trading system for US equities will decide to send an order within microseconds. A system trading on the Chicago Mercantile Exchange (CME) could work within nanoseconds. Based on this observation, the technology will be critical in the choice of designing the software. If we just refer to the programming language, Python is not adapted to speed and we will preferably choose C++ or Java. If we want to take a long-term position such as many days, the speed allowing a trader to get a liquidity faster than others will not be important. A programming language such as Python will be fast enough to reach this goal.
- **The number of users (the number of trading strategies)**: When the number of traders increases, the number of trading strategies increases. This means that the number of orders is higher. Before sending an order to an exchange, we need to check the validity of the orders we are about to send: checking whether the overall position for a given instrument has not been reached. In trading world, we have more and more regulations moderating trading strategies. To follow that our trading strategy respect the regulation, we will test the compliance of the orders that we want to send. All these checks will add some calculation time. If we have too many orders, we will need to have all these verification done sequentially for one given instrument. If the software is not fast enough, it will slow down the orders to go out. So having more users will require a faster trading system.

These parameters modify the conception of the trading system you are going to build. It is essential to have a clear description of the requirements when you build a trading system.

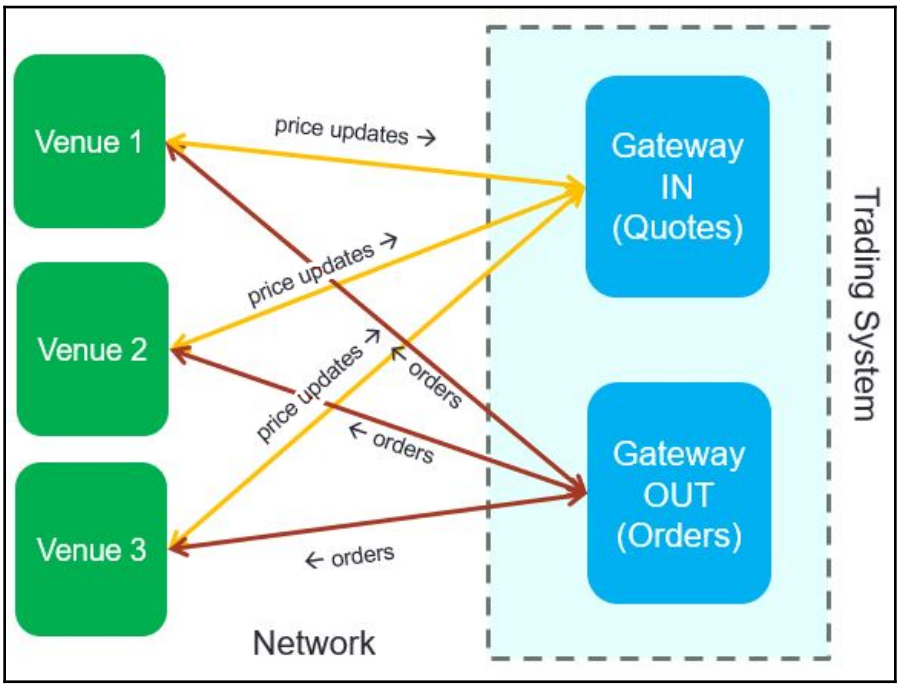
Because the goal of a trading system is to support your trading ideas. The trading system will collect the information that your trading strategy needs and be in charge of sending orders and receiving responses from the market regarding this order. The main functionalities will be to collect the data (most of the time this will be price updates). If the trading strategy needs to get some quantitative data involving earnings, fed announcements (more generally news), these news will also trigger orders. When the trading strategy decide the direction of the position. the trading system will send orders accordingly . The trading system will also decide which specific exchange will be the best to get the order filled for the requested price and for the requested volume.

Gateways

A trading system collects price updates and sends orders on your behalf. In order to get to that, you need to code all the steps that you would do if you were trading without any trading system. If you would like to make money by buying low and selling high, you will need to choose the product you will use to trade. Once you select this product, you want to receive the order from the other traders. The other traders will provide you their intention (their orders) to trade a financial asset by indicating the side, the price and the quantity. As soon as you receive enough orders for the product that you want to trade, you can choose the trader you are going to make a deal with. You will make your decision based on the price of this asset. If you want to resell this asset later one, it will be important to buy it for a low price. When you agree with a price, you will indicate the other trader that you will want to buy for the advertised price. When the deal is done, you now own this product. You will proceed the same way when you want to sell it at a higher price. We formalize this way of trading using functional units:

- **Data handling**: Collecting price updates coming from the venues you will choose to trade with (exchanges, ECNs, dark pools). This component (called a **gateway** in the following diagram) is one of the most critical of the trading system. The task of this component is to get the book for a given instrument from an exchange to the trading system. This component will be linked to the network and it will get connected to exchanges receiving and sending streams to communicate with it.

The following diagram represents the location of the gateways in the trading system. They are the input and the output of the trading system:



The diagram shows the following:

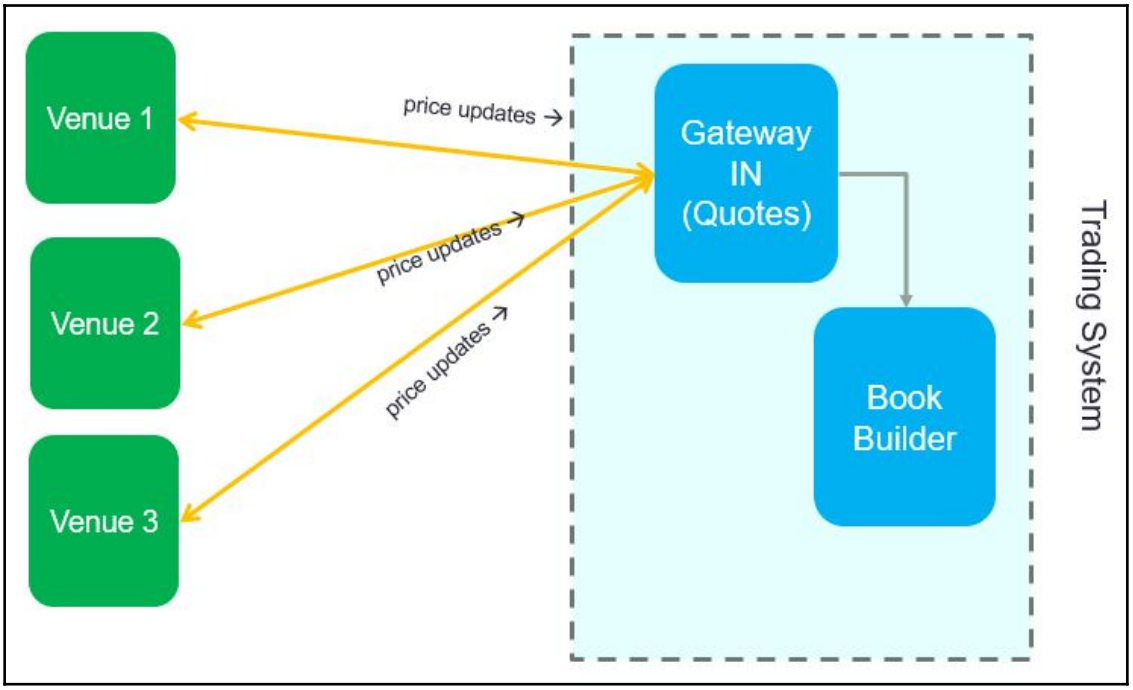
- The venues represent traders, exchanges, ecns, and dark pools.
- The getaways and venues can be linked by different ways (they are represented using arrows).
- We can use a wire, wireless network, internet, microwave, or fibers. All these different network media have different characteristics in terms of speed, data loss, and bandwidth.
- We can observe the arrows are bidirectional for the price updates and the orders. There is a protocol to ask for price updates.
- The gateway will initiate a network connection with the venue, authenticate itself, and subscribe to a given instrument to start receiving price updates (we will explain this part in more detail later).

- The gateway taking care of orders also receives and send messages. When an order is created, it is sent through the network to the venue.
- If the venue receives this order, an acknowledgment of this order will be sent. When this order has met a matching order, a trade will be sent to the trading system.

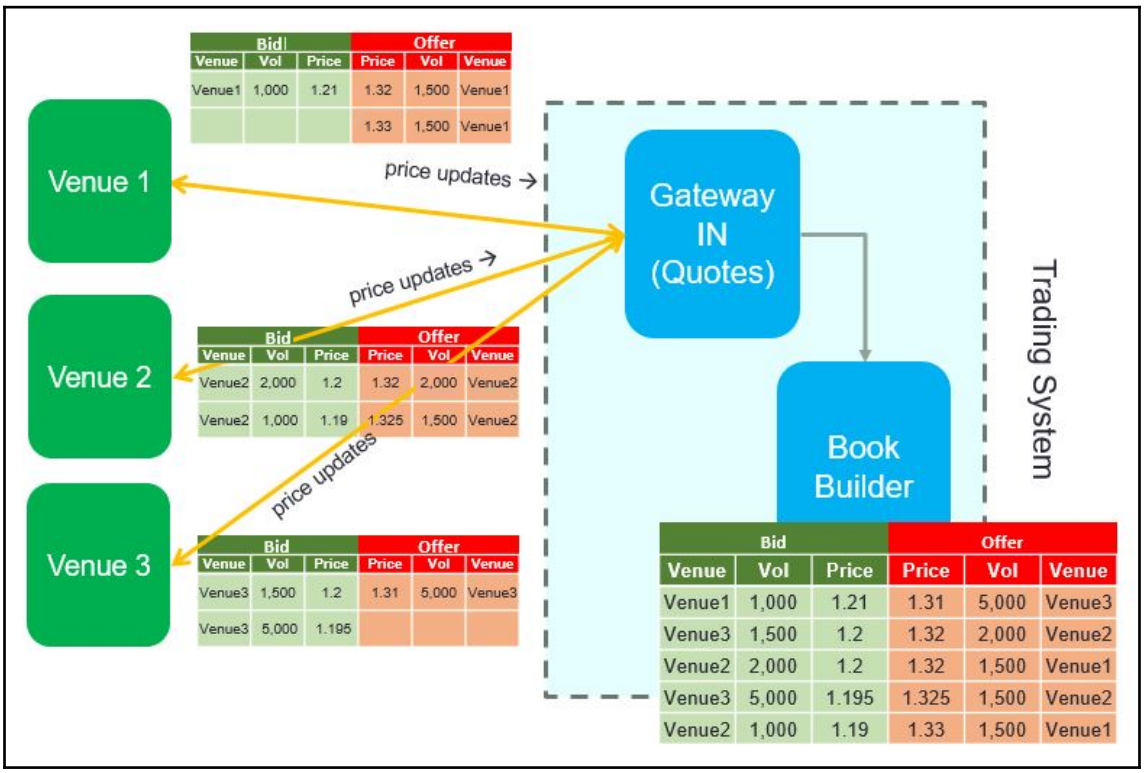
Order book management

The main task of data handling is to replicate the limit order book from the venues into your trading system. To combine all the different books you receive, the **book builder** will be in charge of gathering the prices and sorting them for your strategies.

In the following diagram, the price updates are converted by the gateway then transferred to the book builder. The book builder will use the books received by the gateways from the venues and it will gather and sort all the price updates:



In the following diagram, we use an example of an **order book** for a given financial product. Since we have three venues, we observe three different books:



Bid			Offer		
Venue	Vol	Price	Price	Vol	Venue
Venue1	1,000	1.21	1.32	1,500	Venue1
			1.33	1,500	Venue1

Bid			Offer		
Venue	Vol	Price	Price	Vol	Venue
Venue2	2,000	1.2	1.32	2,000	Venue2
Venue2	1,000	1.19	1.325	1,500	Venue2

Bid			Offer		
Venue	Vol	Price	Price	Vol	Venue
Venue3	1,500	1.2	1.31	5,000	Venue3
Venue3	5,000	1.195			

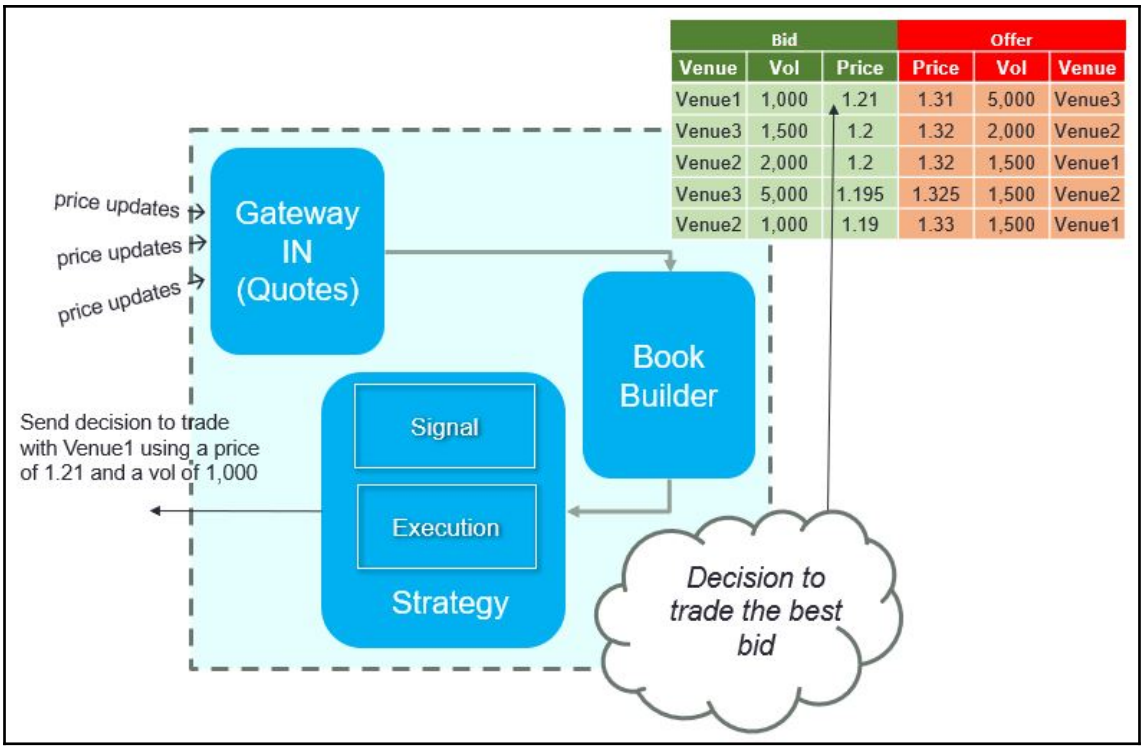
Bid			Offer		
Venue	Vol	Price	Price	Vol	Venue
Venue1	1,000	1.21	1.31	5,000	Venue3
Venue3	1,500	1.2	1.32	2,000	Venue2
Venue2	2,000	1.2	1.32	1,500	Venue1
Venue3	5,000	1.195	1.325	1,500	Venue2
Venue2	1,000	1.19	1.33	1,500	Venue1

The diagram shows the following:

- In these books, you can see for each row there is an order.
- For instance, in the bid list of Venue 1, there is a trader willing to buy 1,000 shares for $1.21. On the other side is the list of people willing to sell.
- You can expect the offer (or ask) price to always be higher than the bid price.
- Indeed, if you could buy for a smaller amount than you could sell, it would be too easy to make money.
- The task of the book builder is to get the three books from the three venues collected by the gateways. The book builder regroups the three books and sort the orders.

Strategy

The trading strategy is the brain of the system. This is where your algorithm representing your trading idea will be implemented. Let's have a look at the diagram:



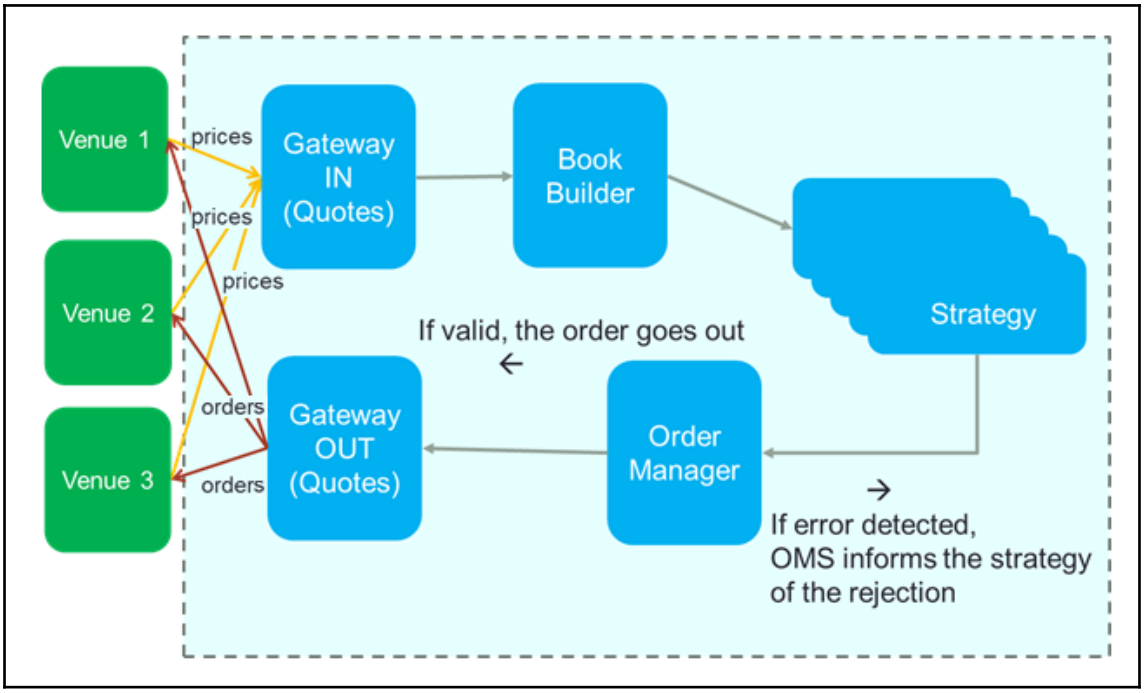
The diagram shows the following:

- The trading strategy is divided into two main components: **signal** and **execution**. In this book, the numerous strategies we saw in the first part can be called signal.
- The signals represent the indication of getting a long or a short position. For instance, in the dual moving average crossover momentum strategy, when the two average lines were crossing, a signal to go long or go short was generated.
- The signal component of this strategy only focuses on generating signals. However, having the intention (a signal) does not guarantee you to get the liquidity you are interested in. For instance, in high-frequency trading, it is highly likely your orders will be rejected because of the speed of your trading system.

- The execution part of the strategy will take care of handling response from the market. This part decides what to do for any responses from the market. For instance, what should happen when the order is rejected? You should continue trying to get an equivalent liquidity, another price. That's an important part you will need to focus how to implement.

Order management system

The **order management system** (**OMS)** is the component that collects the orders sent from the strategies. The OMS keeps track of the order life cycle (creation, execution, amendment, cancelation, and rejection). Trading strategy orders are gathered in the OMS. The OMS may reject orders if an order is malformed or not valid (too large a quantity, wrong direction, erroneous prices, excessive outstanding position, or order type not handled by the exchange). When an error is detected in the OMS, the order does not go out from the trading system. The rejection happens earlier. Consequently, the trading strategy can respond faster than if the order was rejected by the exchange. Let's have a look at the following diagram, which illustrates these features of the OMS:



Critical components

Gateways, a book builder, strategies, and an OMS are the critical components of any trading system. They gather the essential functions you need to start trading. We measure the performance of a trading system in terms of speed by adding the processing time of all the critical components. We start a timer when a price update gets into the entrance of the trading system and we stop the timer when the order triggered by this price update goes out from the system. This time is called the **tick-to-trade** or **tick-to-order**.

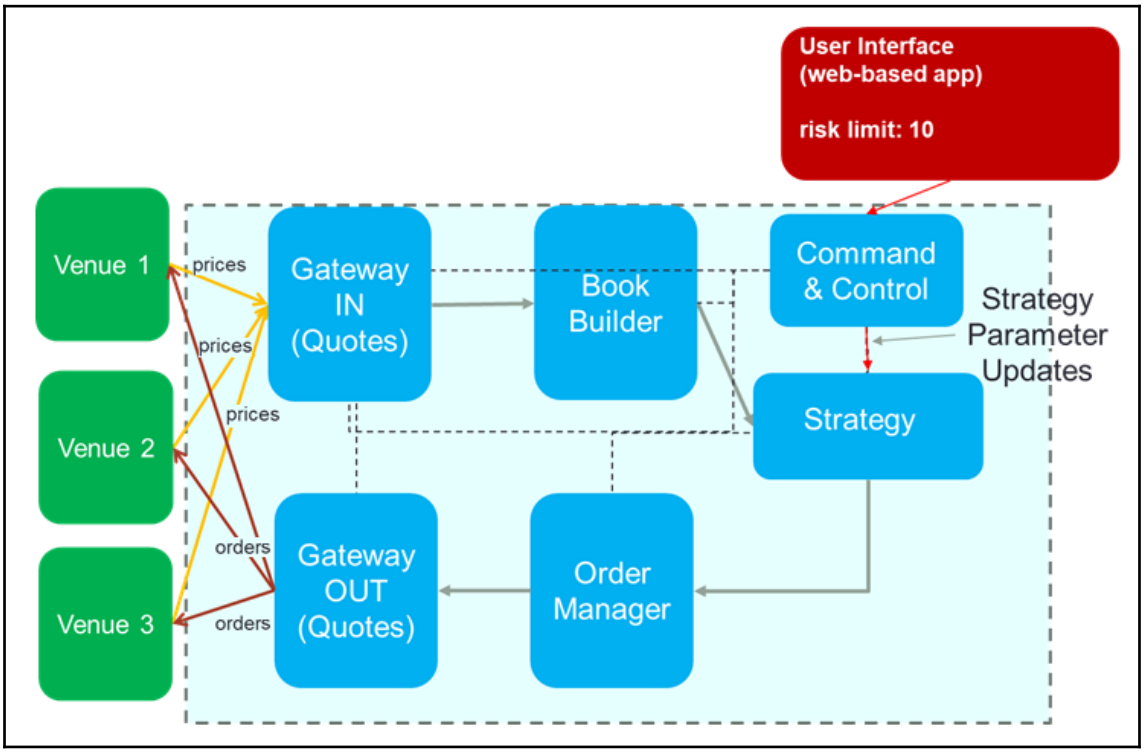
In the most recent systems, this time is in the order of microseconds (around 10 microseconds). When optimized with special hardware and software programming, this time can even be reduced to nanoseconds (around 300 nanoseconds). Because we choose to use Python to implement our trading system, the tick-to-trade of this Python system will be in the order of milliseconds.

Non-critical components

The non-critical components are the components not directly linked with the decision to send an order. They modify parameters, report data, and gather data. For instance, when you design a strategy, you will have a set of parameters that you need to adjust in real time. You need a component capable of conveying the information to the trading strategy component. For that, we will have a component called **command and control**.

Command and control

Command and control is an interface between traders and the trading system. It can be a command-line system or a user interface receiving the commands from the traders and sending the messages to the appropriate components. Let's have a look at the following diagram:



As shown in the diagram, if we need to update the trading strategy parameters, the trader can use a text field on a web-based application to specify the risk tolerance the trading strategy can take. The number (corresponding to the tolerance limit) will be sent to the appropriate trading strategy.

Services

Additional components may be added to the trading system. We will talk about the following components (it is not an exhaustive list):

- **Position server**: This keeps track of all the trades. It updates the positions for all the traded financial assets. For instance, if a trade is made for 100,000 EUR/USD at a price of $1.2, the notional position will be $120,000. If a trading system component needs the position amount for EUR/USD, it will subscribe the position server for getting position updates. The order manager or the trading strategy may want to know this information before allowing an order to go out. If we want to limit the position to $200,000 for a given asset, another order to get 100,000 EUR/USD will be rejected.
- **Logging system**: This gathers all the logs from the components and will write a file or modify a database. A logging system helps with debugging, figuring out causes of issues, and also just reports.
- **Viewers** (read-only user interface view): These display the views for trading (positions, orders, trades, task monitoring, and so on).
- **Control viewers** (interactive user interface): These provide a way to modify parameters and start/stop components of the trading system.
- **News server**: This gathers news from many news companies (such as Bloomberg, Reuters, and Ravenpack) and provides this news in real time or on demand to the trading system.