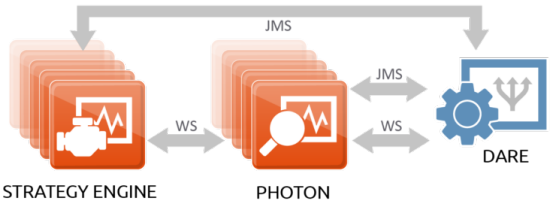
Strategy Engines

STRATEGY ENGINES

Strategy Engines

Strategy Engine modules are responsible for running strategies. They can be run in either your local environment or collocated at your broker's exchange, providing tremendous deployment flexibility and the ability to take advantage of the processing capabilities of server class hardware available in commercial datacentres.

Strategy Engines start as empty containers that can include the modules that participate in event driven data flows. Each event flow is comprised of multiple modules and each module in the framework can be an event emitter, receiver or both. Each module, such as the Complex Event Processing Module, Market Data Module, Threading Model Policy module, etc., implements distinct functionality. A module can be a singleton (i.e., only one instance of this module can exist in MEFP) or can have multiple instances. Strategy Engine modules can support multiple run-time event flows in which different modules participate in each flow (see Figure 2). This threading model policy can be dictated by the first emitter in a data flow and can be changed in any emitter module in the flow.



Strategy Engines provide a framework that allows strategy developers to scale their solution as strategies are being developed and evolved – from concept to production or when additional iterations are needed. Multiple factors affect the feasibility of implementing a new strategy idea — the dataset, rate of updates, functionality, performance (latency and throughput), capacity and scalability. Marketcetera provides the necessary building blocks to overcome these challenges, including a powerful, flexible module framework that supports connectivity between diverse functionality modules. The inherent flexibility of this design means that platform functionality can be developed and evolved over time as new functionality is added and the performance of the data flow critical paths can be optimized and scaled up as needed

The basic data flow is single threaded to ensure the sequence of events in the flow (for instance, market data updates should be delivered in order). Strategy Engines can also support a defined multi-threading policy since the threading model policy can be dictated by the first emitter in a data flow and can be changed in any emitter module in the flow. Thus, any data flow can consist of modules being called in the context of a single thread or multiple threads. The combination of multiple data flows, with the ability to dictate whether some or all of the modules participating in each of those flows is single threaded or multi-threaded provides a powerful, almost limitless scalability model that is bound only by the hardware configuration being used.

For example, let's assume we have one strategy moving approximately 3,000 symbols over 10 seconds. How we can implement that strategy depends on the particular strategy algorithm, but a few options are:

- ***SINGLE DATA FLOW:*** Create a single data flow that includes a market data adapter, CEP (to calculate moving averages) and the strategy execution engine module.
- ***MULTIPLE DATA FLOWS:*** Create 10 data flows, each with a different CEP module and using the symbol affinity threading policy, i.e., map 300 symbols to each CEP module.
- ***GROUPED DATA FLOWS:*** Split the symbols into groups, create multiple requests to the market data provider for each symbol group and create one data flow for each such request. In each flow, a different instance of the market data adapter, CEP and strategy execution engine module is used.

As you can see, Strategy Engines offer a scalability framework that can accommodate the functional requirements of the strategy while maximizing hardware utilization. Moreover, each active strategy can use a different scalability approach, and the functional requirements of each strategy affect only its unique the data flow.