DARE - Deliver Anywhere Order Routing Engine

DARE is Marketcetera's powerful order routing engine. It is built upon the QuickFIX/J framework and is available as a standalone component or as part of our algorithmic trading platform. DARE provides bi-directional message handling, receiving orders from multiple strategies and routing them to broker and execution destinations using FIX protocol. DARE was designed to address two issues confronting our sell-side and buy-side clients.

1. Control - At Marketcetera our objective is to provide our clients with greater control over their trading operations. Our clients face a myriad of challenges to their independence. Buy-side firms want to control their order flow so they can route their orders to brokers with the lowest transaction costs. Large financial firms want greater control over their infrastructure. They want to remove artificial technical and licensing barriers that interfere with their mandate to build the most reliable, efficient trading ecosystem for their firms. They demand the ability to leverage legacy applications, systems they have written in-house and best-of-breed ISV tools into a single, functional ecosystem. Sell-side firms want to control the solution they offer their clients a single tool that supports multiple asset classes, custom algorithms or even click-trading if that is what their clients prefer.
2. Resilience - The second order routing issue we address is the goal of resilience, defined as the ability of a trading system to maintain a desired level of performance or normal operation without irrecoverable consequences. Increased automation by both investors and exchanges has created the well-documented surge in trading volumes. These volumes provide a challenge not only for the order routing engine itself, but also for the surrounding infrastructure like network bandwidth. Performance is a critical attribute in this trading era and building high performance, low latency systems receives a lot of attention, but as important is the issue of resilient performance, that ability to reliably run these mission critical systems.

To address these two issues of control and resilience for our clients, Marketcetera engineered DARE to deliver the following benefits:

- Scalable, robust and globally proven FIX-enabling toolbox
- Low latency, low footprint
- Easy counterparty connectivity
- Easy integration with (and within) internal systems
- FIX dialects for non-standard FIX connections
- An open component model for tailored message processing

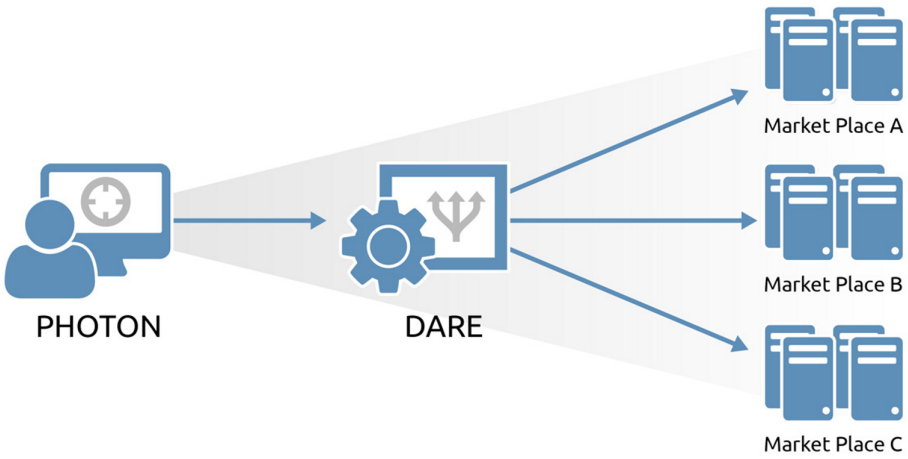
Control

Routing Possibilities

The competition for order flow amongst sell-side institutions is intense. One of the preferred techniques to "capture" order flow is to offer buy-side clients "free" trading tools - of course these tools won't let their clients direct their orders to any competing brokers.

The Marketcetera DARE order routing engine puts control over order flow back in the hands of the trader, where they are free to seek out the most competitive rates for their business. DARE incorporates a FIX engine which greatly simplifies connecting to exchanges or brokers. This FIX Engine is a class library which implements all important aspects of the FIX protocol in all of its versions, from 4.0 to 4.4. Buy-side firms can take advantage of the connectivity DARE provides in two different ways: direct order routing, where your firm connects directly to your counterparty; or via one of the many order routing hubs that have emerged as gateways to networks of sell-side firms, exchanges, ECN’s and ATS’s worldwide. These two models can be used separately or in tandem – enabling buy-side firms to start trading the way they want to immediately.

The DARE order routing engine provides centralized certification of connected parties in compliance with FIX. All counterparties are certified in a consistent, reliable format. DARE handles all FIX version translations, including translation to/from non-FIX messaging standards, which eliminates the need to support multiple systems for each counterparty and minimizes system changes as a result of on-going advances in technology. DARE provides full support for customized FIX messaging, including support for all algorithms.



Integration

Trading firms manage a complex ecosystem of high performance systems created by both in-house teams and independent software vendors. To help clients control this ecosystem, Marketcetera, an open-source company, subscribes to open standards and open APIs that make the integration task easier. The Financial Information eXchange ("FIX") Protocol, which consists of a series of messaging specifications for electronically communicating trade-related messages, is one of these standards that was developed to improve the global trading process. Sell-side and Buy-side firms looking for a pluggable, stand- alone FIX engine use DARE to “FIX-enable” internal systems such as an order management system and trading applications, and to assert greater control over their trading ecosystem.

Messaging

We believe that our role is to provide a flexible, modular architecture that can be customized to meet our client's specifications. DARE order router is built on the concept of a “pluggable transport” which enables clients to deploy their preferred message middleware. We ship with ActiveMQ, but our clients have the control to swap out our message bus for their preferred solution. DARE offers a separation between the message formats, session handling and the transport middleware. Implementation of adapters for specific middleware may be developed and injected transparently inside the infrastructure. Several adapters have been developed including pure socket, JMS, RMDS, RV, SOAP, FIX and others.

Resilience

Trading systems are challenged by any number of events, a spike in orders, network outages, flooding. DARE is designed to operate within this dynamic environment and insulate our clients from any "irrecoverable consequences". Three examples of our resilient system design include high availability, failover and scalability.

High Availability and Fail-over

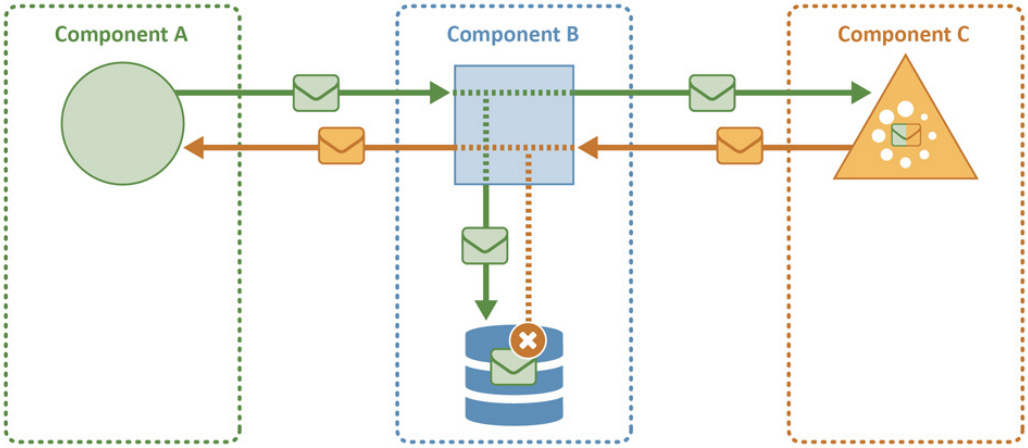
High availability and support for fail-over is a constant and critical requirement in today’s highly volatile trading environments. Marketcetera has developed two complimentary strategies for HA/FO, each builds upon the other:

1. The first strategy is for components that are singleton in nature. For various reasons, these components can have a single active instance at a time. Reasons for this include maintaining a socket connection to a FIX gateway, which, by their nature, require a single host. An active, or “hot” instance is running while a “warm” instance monitors the active instance. The warm instance is already running, but is not actively performing its role. More than one warm instance can run at the same time across multiple nodes. The warm instances each monitor the active instance and a common queue. If the active instance fails, one of the warm instances is promoted to active. Additional warm instances can be added at any time.
2. The second strategy for HA/FO is to maintain multiple concurrent instances. In this strategy, multiple agents each capable of performing a given task are created as needed. These multiple agents are controlled by a singleton controller that follows the first strategy. The controller distributes work to the agents as necessary. The key to this strategy is that each agent is capable of processing any given task without referring to any internal state. This allows the number of available agents to expand and contract to meet the needs of the system. DARE maintains a single active Order Routing Service for a given external FIX gateway at a given time. Multiple warm backup Order Routing Service processes monitor the active Order Routing Service. If the active Order Routing Service fails, one of the warm backup Order Routing Services is nominated to take over as the active Order Routing Service. All Order Routing Services share the same persistent store, both for messages and for FIX session data.

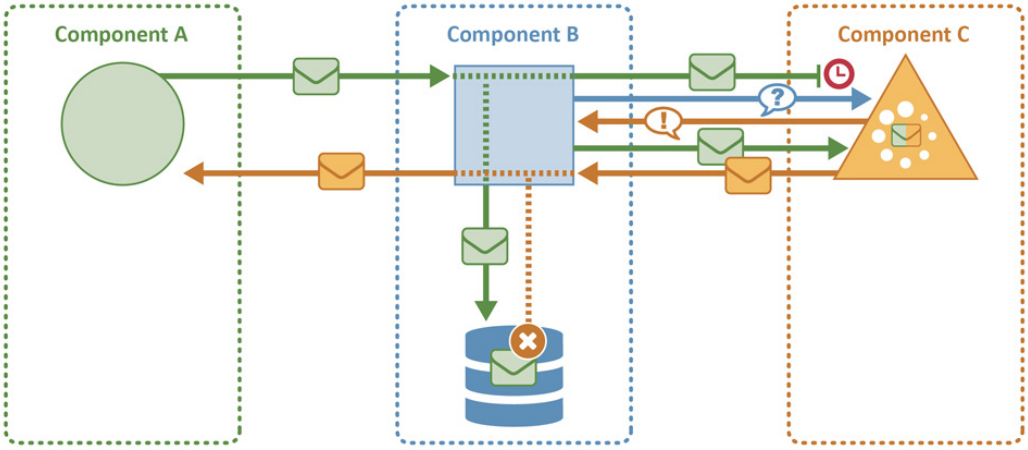
Scalability

DARE maintains agents that are responsible for the successful routing and transmission of an outgoing order. The work of the “send” order agent represents the bulk of the computational load associated with sending an outgoing order which means that horizontal scalability is most effectively applied at this point.

Guaranteed Message Delivery



The above diagram demonstrates how messages are passed from one component to another. Component A passes the message to Component B, which records the message in a persistent, shared store and acknowledges receipt of the message to Component A. Component B processes the message and hands it to Component C. When Component C acknowledges the message, Component B marks the message as processed in its persistent store, or removes it, as appropriate.



Guaranteed Message Delivery Resend

The above diagram shows the interaction between components if a receiving component does not acknowledge receipt of the message within a given, configurable interval. In this diagram, Component C initially does not acknowledge receipt of the message from Component B. After the appropriate interval, Component B sends a status inquiry to Component C. Based on the response from Component C, Component B resends the message. The normal message flow resumes from there.

Each component must be able to receive messages of the appropriate type and respond to appropriate status inquiries. Each component must track transmission of messages to the next component and corresponding acknowledgements. If the message is not acknowledged, the message can either be resent, an alert raised to the operator, or both, as appropriate for the component.

DARE Snapshot

Functional Benefits	Technical Benefits
Full FIX protocol connectivity support, including the latest versions	Supports FIX versions 4.0 - 4.4, 5.0/FIXT1.1.
Automatic header/trailer construction	Runs on any hardware and operating system supported by 1.4+ Java SE or compatible VM.
Acts as both Initiator and Acceptor	Compatibility with QuickFIX C++ Java Native Wrapper API (easy to upgrade)
Repeating groups, infinite loop detection	Java NIO asynchronous network communications for scalability (using Apache MINA)
ResetSeqNumFlag support	Supports embedded SSL with Java 5+
Multiple validation: CompID, Checksum, Sending Time accuracy	Provides standard JMX MBeans for FIX engine management
Heartbeat timeout	Scheduling of session connections.
Flexible: easily adaptable to different FIX dialects, including non-standard implementations	Choice of message processing threading strategies
Comfortable: object-oriented abstractions of FIX messages makes FIX programming trivial	Communication transports for TCP sockets and VM pipes.
Support for plug-in encryption	Unlimited message size
Support for socket, messaging, or custom transport layer	Modular persistence
Messages stored persistently to file or database	Modular transport layer
	Support for protocol customizations (new messages, fields, constraints).
	Session state storage plugins: JDBC, File, SleepyCat/JE, In memory
	Logging plugins: JDBC, File, SFL4J (supports JDK1.4 logging, Log4J, Commons Logging), Console, Composite
	Raw FIX message listener