High Availability and Failover

- Product
- Market Data Nexus
- CEP
- Smart Order Routing
- Strategy Engines
- DARE
- Photon
- System Administration
- Database Access Layer and Persistence
- Threading Model
- High Availability and Failover

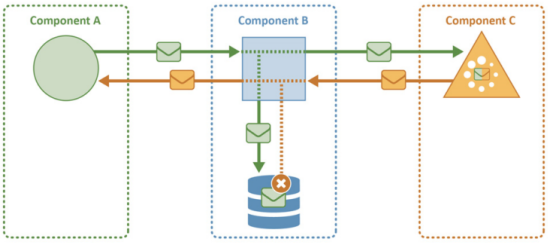
HIGH AVAILABILITY AND FAILOVER

High Availability and Failover

Overview

High-availability and failover refers to the capability of the Marketcetera Automated Trading Platform to detect and recover from transient datacenter failures. Failures may be related to software (failure of a Marketcetera node), hardware (failure of an Marketcetera host), or datacenter (loss of connectivity to /from a datacenter).

The goal of Marketcetera is to guarantee that any message it receives is delivered to an appropriate recipient. In most cases, this will be the intended recipient, e.g. the broker or the client's OMS. However, under certain circumstances, like if a broker connection is unavailable, Marketcetera will reject the message to the OMS. In all cases, the goal is that no message is lost or undelivered to some recipient.



High-availability is more than just message delivery. In order to achieve HA/FO, Marketcetera as a conceptual application needs to be able to survive the transient loss of one or more nodes. There are five essential components that needs to be considered:

1. Marketcetera FIX Gateway (from the OMS)
2. Marketcetera message/report handlers
3. Marketcetera FIX Sender (to brokers)
4. Database

Let's consider them one at a time to understand how HA/FO is to be achieved.

Marketcetera FIX Gateway (from OMS)

The Marketcetera FIX Gateway accepts incoming FIX messages from an OMS and maintains the FIX sessions for that connection. One of the key components of the FIX Gateway is the network socket over which incoming orders are received. The loss of a datacenter or hardware host (which includes a software failure, of course) means that the socket is lost, too. Since we cannot guarantee that the OMS is capable of managing multiple socket addresses (primary, secondary, etc), Marketcetera implements a routing-based failover technique. Using this technique, the IP address supplied to the OMS is a virtual address that initially maps to host 1, then fails over to map to host 2 (and then host 3, etc). The session details are kept in a common database. Any Marketcetera host is capable of managing the sessions, however, only one can be "hot" at once, since only one host can actively manage the real socket connection. Marketcetera will keep "warm" FIX Gateway components and a shared, in-memory collection that indicates which is the hot host. Upon failure of a host, the designated next warm host will become hot. The priority is configured in Marketcetera.xml and must match the network mappings established by our client's IT team. Any Marketcetera FIX Gateway can process messages and move them to the next destination. Incoming messages are stored to an in-memory queue shared by all FIX Gateways and written to the database. Upon receipt, a message tracker is started, which is also shared by all Marketcetera nodes. It is the responsibility for each tracker to verify that the message was handled or take action. Action involves rejecting the message back to the OMS, sending a notification, or both. Upon restart, all FIX Gateway components will seek each other out and sync up.

Marketcetera Message/Report Handlers

The Marketcetera Message and Report handlers are responsible for taking an incoming message, applying business logic to it, and shepherding it to its final destination up to but excluding the actual FIX session and socket. The handlers do not maintain a specific socket on which they're connected, so they are more free to assemble at will. All handlers share an in-memory, trans-node queue of work to be done and tracking tasks. The queue is backed by the database in case all nodes go down at the same time. Each node knows about every other node and shares information about tasks to be done.

Each task, which is either a message from an OMS or a message from a broker, has an order ID associated with it. Nominally, any Marketcetera node can process any message, however, one of the business requirements is that messages for a particular order be processed sequentially. In order to maximize overall order throughput, the optimal case would be to span messages to be processed across all nodes. In order to guarantee sequential processing for messages of the same order, however, it is necessary to affine an order chain to a node. In this context, an order chain refers to all orders that are conceptually or logically grouped. An order chain may consist of a new order single, followed by a cancel /replace for that order, followed by a cancel for the cancel/replace, for example. In practice, these messages do not all share a common field. In concept, they are related and are all members of the same order chain. For convenience, Marketcetera refers to all members of the same order chain as having the same root order ID, which is the order ID of the original order. All members of the same order chain will be handled by the same Marketcetera node. If a Marketcetera node dies, its affined order chains will be orphaned and assigned to a new node to handle the message that was in process, if any, and any subsequent messages of that chain. The new node inherits the order chain from the terminated node.

Marketcetera FIX Sender

The Marketcetera FIX Sender sends outgoing FIX messages to brokers and maintains the FIX sessions. Like the Marketcetera FIX Gateway, the Sender maintains a number of physical socket connections. Therefore, there cannot be multiple hot FIX Sender nodes. In a fashion very similar to the Marketcetera FIX Gateway components, there will be one hot and multiple warm nodes. At such time as a hot node goes off line, one of the warm nodes is nominated to be the new hot node and it starts up. Unlike the FIX Gateway nodes, it does not matter which warm node starts up as the FIX Sender initiates socket connections, not receives them. Like the other nodes, the FIX Sender maintains an in-memory, trans-node queue backed by the database. FIX session information is also stored in the database.

Database

The database Marketcetera uses is administered by our clients. In the absence of viable database service, the Marketcetera nodes will not start. If database service is lost while iRouter nodes are running, message delivery will continue as will the in-memory, trans-nodes shared queues, but the ability for nodes to recover from total shutdown (all nodes go offline) will be absent. In the total shutdown scenario, upon restart, Marketcetera will connect with the external FIX connections and negotiate last known messages. If messages are determined to be missing by either party, they will be resent. Since session information is stored in the database in order to facilitate loss-of-node recovery, it is very important that Marketcetera works closely with our clients to maintain suitable and standard maintenance and cluster techniques and practices for the application database.