Configuring Managed Queue

Overview

FIXEdge routes messages (e.g. orders, cancels) from source systems to destination systems and it is a common case that a destination system accepts messages at the time interval when session is established. But in certain buisness configurations destination system may work as a broker that owns gateway to exchanges, that are ultimate destinations for messages. Ultimate destination may be closed (work on schedule base) while a broker is running. In such cases messages will be rejected by broker. Thus there is a need to store them in a queue and release at a specific time, according to an exchange schedule.

To meet these needs Managed Queue solution provides a way:

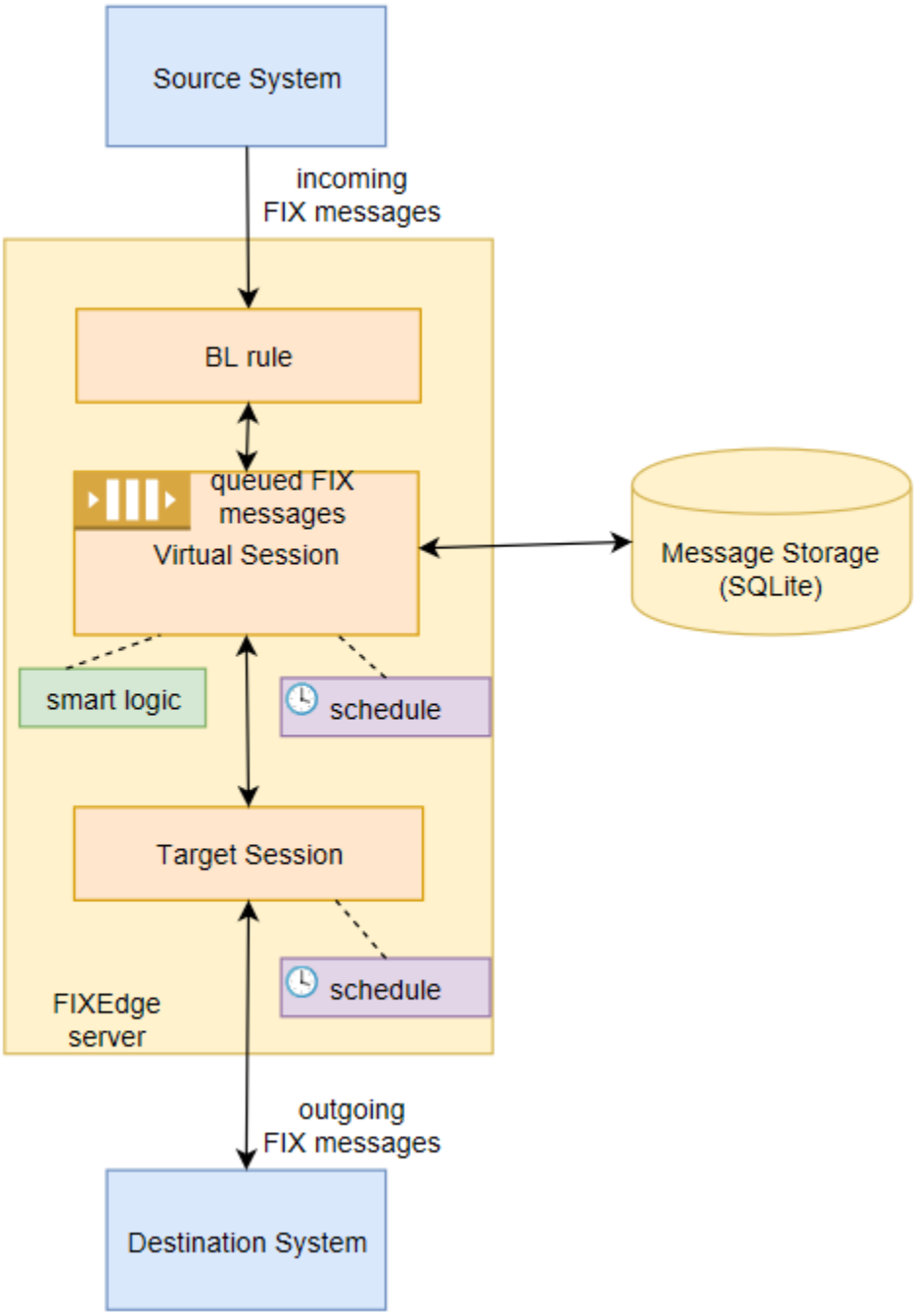
- To store messages in the queue during off hours (market is closed)
- To release messages stored in the queue at the time market is open
- To pass through messages during “market is open” hours
- To define multiple queues for different order types
- To distribute incoming orders among the queues relating on order type - to route only messages that can be accepted by certain destination system.
- To apply smart logic for orders stored in the queue (cancel and cancel/replace)

Manage Queue feature description

Design

Logically FIXEdge server is a router that routes messages from a source system to a destination system. The following diagram demonstrates data flow for the case managed queue is used. Messages from a source system are parsed and put to business layer. BL rule is executed and messages that fit the rule are moved to a certain virtual session. Each virtual session is actually a queue. If virtual session is open (connected to a target session) the message is moved to target session immediately. If virtual session is closed (disconnected from a target session) then the message is stored in the queue. The smart logic is applied. The virtual session connects to a target session or disconnects from a target session according to a schedule configured for a virtual session. Target session may also have a schedule. If target session is not established then messages stay in the queue and are released when target session is established.

All messages are persistently stored in SQLite database.



SQLite database schema

SQLite database is used to store queued messages. Default DB name is fe.db. Messages of all queues, sent to all sessions, are stored within single table – “QueueStorage”. Messages that are delivered or removed are marked in corresponding columns.

Factual removal of messages from table can be done manually or by a script using SQLite utilities.

Table “QueueStorage” is described below:

Table Column Name	Data Type	Description
`GlobalMessageID`	INTEGER	Unique identifier of a message in the table
`QueueID`	TEXT	Manager Queue Name (the same as virtual session name)

MsgType	TEXT	Message Type - taken from tag 35 of a queued message
ClientID	TEXT	Source System identity (e.g. SAP) – taken from tag 49 of a queued message
ClOrdID	TEXT	Client order ID - taken from tag 11 of a queued message
RawMessageText	TEXT	Raw FIX message
TimeStampEnqueued	INTEGER	Time when incoming message was stored in the queue
IsDone	INTEGER	0 – message was not released (must stay in the queue) 1 – message was released (is sent to a target session)
TimeStampDone	INTEGER	Time when message was released
IsRemoved	INTEGER	0 - message was not removed from a queue by Order Cancel 1 - message was removed from a queue by Order Cancel
TimeStampRemoved	INTEGER	Time when message was removed by Order Cancel

Database schema:

```
CREATE TABLE QueueStorage (GlobalMessageID INTEGER PRIMARY KEY, QueueID TEXT, MsgType TEXT, ClientID TEXT,
ClOrdID TEXT, RawMessageText TEXT, TimestampEnqueued INTEGER, IsDone INTEGER, TimeStampDone INTEGER, IsRemoved
INTEGER, TimestampRemoved INTEGER);

CREATE INDEX idx_QueueStorage_QueueID ON QueueStorage (QueueID, IsDone, IsRemoved);

CREATE INDEX idx_QueueStorage_ClOrdID ON QueueStorage (QueueID, ClientID, ClOrdID, IsDone, IsRemoved);
```

Queue releasing process

Queued messages are released when virtual session is connected to a target session and target session is established. Messages are released according to FIFO (First In First Out) logic. If some non-queued messages from the source system are passed to a target session simultaneously with queue release first go the queued messages and then other messages according to FIFO principle.

Configuration

To use managed queue functionality user must configure BL rule in BL_Config.xml and managed queue and session properties in FIXEdge.properties files.

Configuring BL

From the routing perspective BL rule for routing to a virtual session is similar to ordinary BL routing configuration.

The example of BL rule for routing from RestAcceptorClient_Queueing source to MQDEST-XMIC-PreOpen virtual session:

```
<Rule Description="Route XMIC Limit orders to XMIC preopen (MQDEST) ">
      <Source>
            <Client Name="RestAcceptorClient_Queueing"></Client>
      </Source>
      <Condition>
            <MatchField Field="35" Value="D|F|G"></MatchField>
            <MatchField Field="207" Value="XMIC" ></MatchField>
            <MatchField Field="40" Value="2" ></MatchField>
       </Condition>
       <Action>
            <Send Name="MQDEST-XMIC-PreOpen"></Send>
       </Action>
  </Rule>
```

Configuring managed queue

Manages queue properties description:

Property	Description
`<managed_queue_name>.Storage.Type`	Type of storage where incoming messages are stored.
`<managed_queue_name>.Storage.SQLite.FileName`	File name of a storage

Managed Queue properties example:

```
FixLayer.ManagedQueue.Storage.Type = SQLite
FixLayer.ManagedQueue.Storage.SQLite.FileName = fe.db
```

Configuring virtual session

Message is routed from source connection to virtual session.

Virtual session properties description:

Property	Description
`FixLayer.ManagedQueue.Sessions = <session_name>`	Name of a virtual session
`<session_name>.TargetSession`	Destination FIX session for routing a message from virtual session or any other source.
`<session_name>.Schedule`	Schedule name for virtual session schedule

Virtual session schedule properties description:

Property	Description
`<schedule_name>.TimeZone`	Time Zone
`<schedule_name>.ConnectTime`	Time when messages stored in the queue are released
`<schedule_name>.DisconnectTime`	Time when session stops sending messages to a target session and starts to store messages in the queue.

Virtual session properties example (source= RestAcceptorClient_Queueing, virtual session= MQDEST-XMIC-PreOpen):

```
FixLayer.ManagedQueue.Sessions = MQDEST-XMIC-PreOpen
...
FixLayer.ManagedQueue.Session.MQDEST-XMIC-PreOpen.TargetSession = MQTARGET
FixLayer.ManagedQueue.Session.MQDEST-XMIC-PreOpen.Schedule = MQTEST-XMIC-PRE
```

Virtual session schedule properties example:

```
FixLayer.ManagedQueue.Schedules.MQTEST-XMIC-PRE.TimeZone = America/New_York
FixLayer.ManagedQueue.Schedules.MQTEST-XMIC-PRE.ConnectTime = 0 20 18 * * *
FixLayer.ManagedQueue.Schedules.MQTEST-XMIC-PRE.DisconnectTime = 0 25 18 * * *
```

When ConnectTime trigger is on a message is routed to the target session MQTARGET.

Configuring target session

Target session is configured as ordinary FIX session.

Target session properties example:

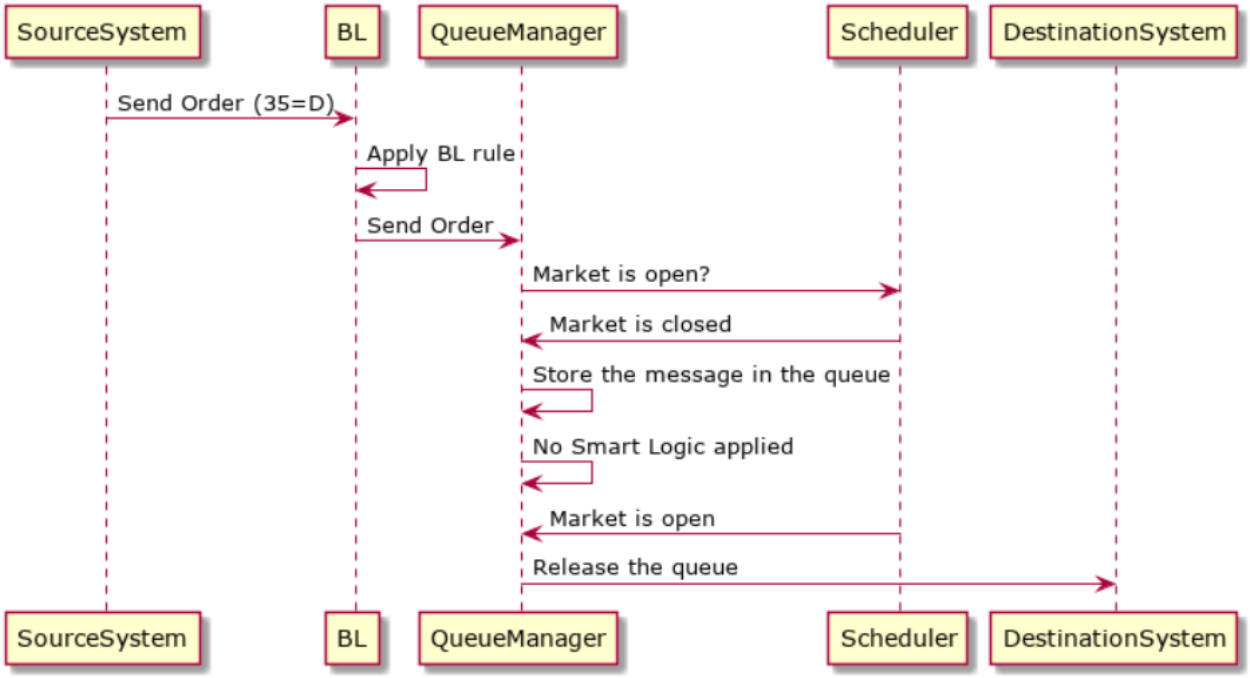
```
FixLayer.FixEngine.Sessions = MQTARGET
...
FixLayer.FixEngine.Session.MQTARGET.Version = FIX44
FixLayer.FixEngine.Session.MQTARGET.SenderCompID = FIXEDGE
FixLayer.FixEngine.Session.MQTARGET.TargetCompID = MQDEST
FixLayer.FixEngine.Session.MQTARGET.Role = Initiator
FixLayer.FixEngine.Session.MQTARGET.StorageType = persistent
FixLayer.FixEngine.Session.MQTARGET.IntradayLogoutTolerance = true
FixLayer.FixEngine.Session.MQTARGET.RecreateOnLogout = true
FixLayer.FixEngine.Session.MQTARGET.TerminateOnLogout = false
FixLayer.FixEngine.Session.MQTARGET.IgnoreSeqNumTooLowAtLogon = true
FixLayer.FixEngine.Session.MQTARGET.ForceSeqNumReset = 0
FixLayer.FixEngine.Session.MQTARGET.Schedule = MQTEST
FixLayer.FixEngine.Session.MQTARGET.HBI = 60
FixLayer.FixEngine.Session.MQTARGET.Host = 10.11.11.111
FixLayer.FixEngine.Session.MQTARGET.Port = 8901
```

Target session schedule example:

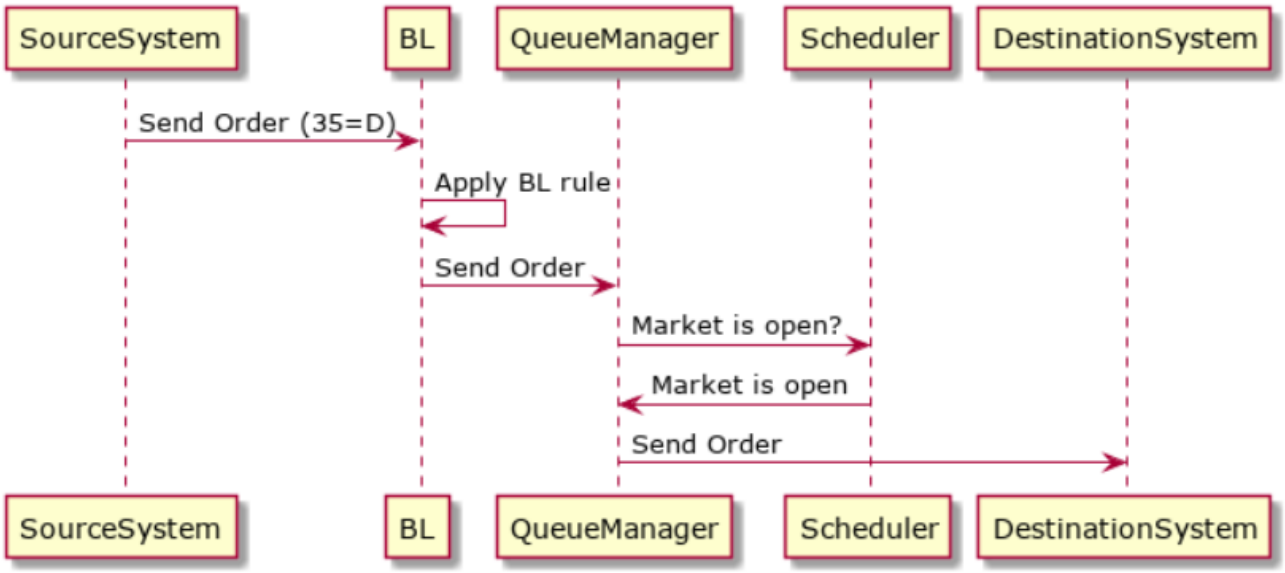
```
Schedules.MQTEST.StartTime = 0 0 18 * * *
Schedules.MQTEST.ConnectTime = 0 05 18 * * *
Schedules.MQTEST.DisconnectTime = 0 55 19 * * *
Schedules.MQTEST.TerminateTime = 0 59 19 * * *
```

Order flow scenarios

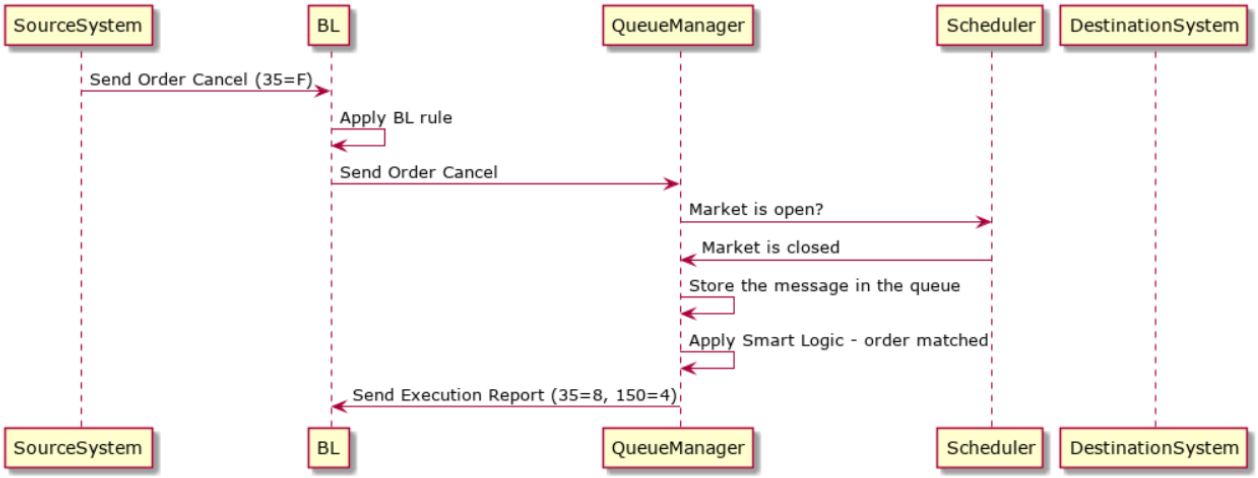
Order is queued in trade-off hours, released when market is open



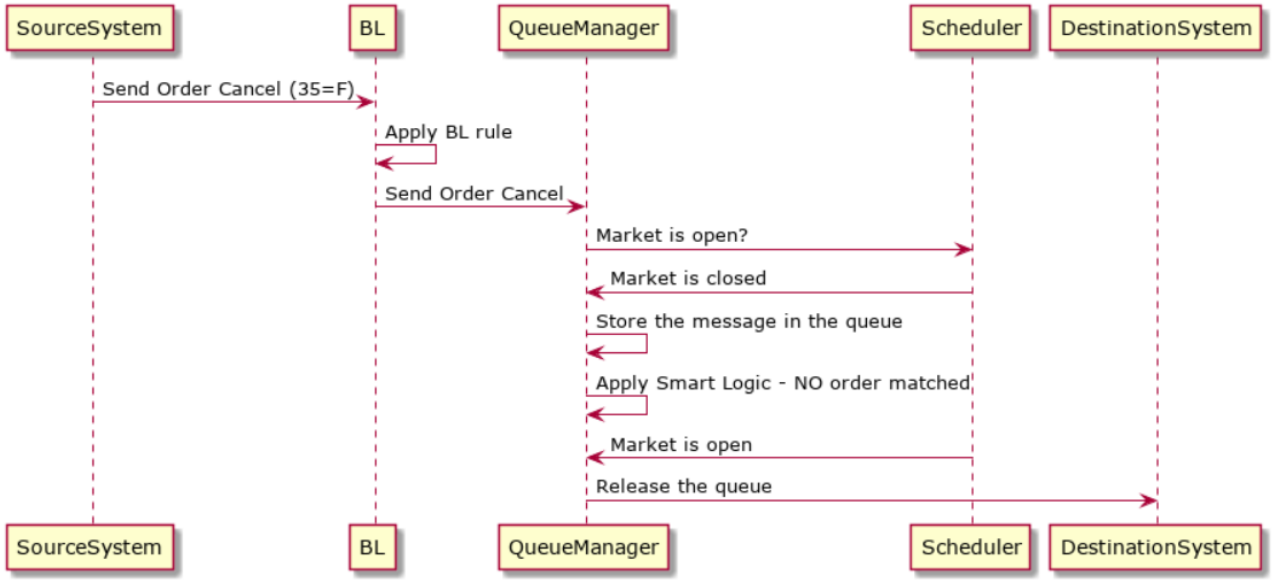
Order goes through without queueing



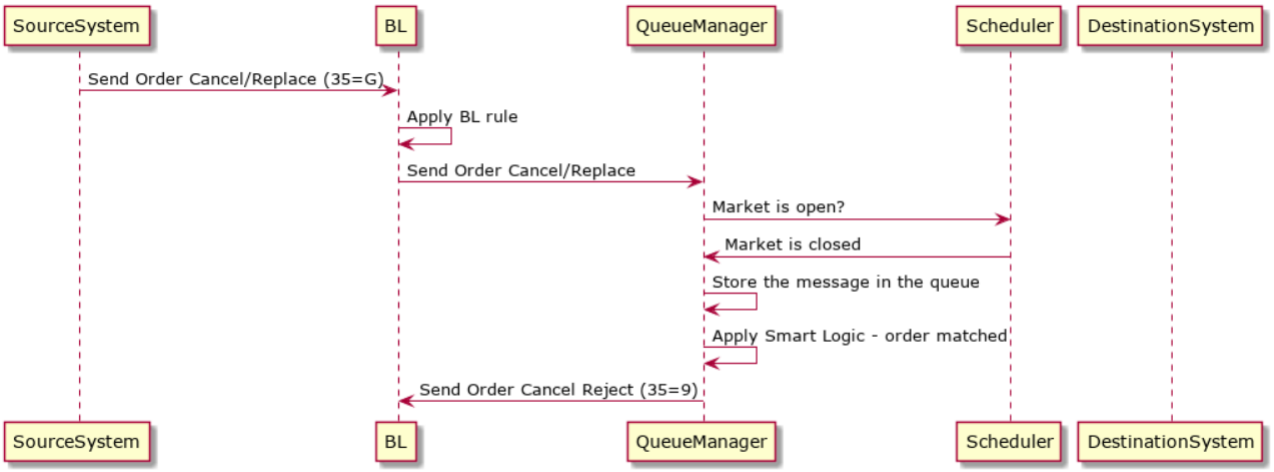
Cancel eliminates order in the queue



Cancel is queued (no corresponding order in the queue)



Cancel/replace is received, Cancel Reject generated, Order stays in the queue



Smart logic features

Smart Logic is a handler, that can play specialized logic for incoming messages, like canceling orders, sending notifications and so on.

Current implementation includes:

- Matching incoming Cancel with queued Order, deleting both and sending confirmation to BL.
- Cancel/Replace queued Order – send Cancel Reject message, order stays in the queue

For details see order flow scenarios.

Logic in handler is hardcoded, but handler itself can be replaced via FIXEdge plugin system. Smart logic can be turned on/off for all sessions simultaneously.

Property	Description
`Components.Component.QueueObserver = <ServiceName>:QueueObserver`	Turn on smart logic plug-in (currently for order cancel and cancel/replace) <ServiceName> is given by user To turn off plug-in remove or comment this property string