Consulting
Services

TIBCO EMS Routing Design Patterns

This is a TIBCO Best Practices document from TIBCO Professional Services Group.

The purpose of this document is to provide some design patterns for the routing of messages between EMS (Enterprise Message Service) servers.

This document can be used to establish company specific standards, or simply used as a reference.

Document Version: 2007.1

Document Date: August 16, 2007

Author: Richard Lawrence

TIBCO® The Power of Now®

http://www.tibco.com

Global Headquarters
3303 Hillview Avenue
Palo Alto, CA 94304
Tel: +1 650-846-1000
Toll Free: 1 800-420-8450
Fax: +1 650-846-1005

For more information, please contact:

TIBCO Software Inc.
3303 Hillview Avenue
Palo Alto, CA 94304
USA

Table of Contents

Table of Figures

1 Introduction

The purpose of this document is to provide some design patterns for the routing of messages between EMS (Enterprise Message Service) servers. This document should not be taken as a complete set of design patterns, and like all documents of this type should be considered a work in progress. Hopefully, additional patterns will be added to this document over time.

Specifically, this document addresses routing designs for the following message exchange patterns:

- Point to Point
- Publish/Subscribe
- Synchronous Request/Reply
- Asynchronous Request/Reply

2 Overview

2.1 JMS Queues

In point-to-point messaging, each message has one producer and one consumer. In JMS, this style of messaging uses a queue to store messages until they are received. Under normal conditions, the message producer sends the message to the queue and receives an acknowledgement from the EMS server of its successful placement on the queue; the message consumer retrieves the message from the queue and sends an acknowledgement to the EMS server that the message was received. By default, the message will remain in the EMS server's queue until its receipt has been acknowledged.

2.2 JMS Topics

In the publish-subscribe message style, each message is delivered to all the recipients interested in the message. This is accomplished through the EMS server mechanism known as a topic. The producer sends the message to the topic and receives an acknowledgement from the EMS server after the message has been successfully placed in the topic. The consumer retrieves the message from the topic and sends an acknowledgement to the EMS server that the message was received. In contrast to a JMS queue, however, the message remains eligible for delivery to other subscribers. The message will remain in the topic until all subscribers have received and acknowledged the message.

2.3 Destination Bridging

EMS provides a feature to send messages to multiple destinations called destination bridging. A bridge between destinations causes messages that are sent to one destination to also be delivered to the bridged destination at the same time. Bridges can be created between one destination and one or more other destinations of the either the same or different type. That is, a bridge can be created from a topic to a topic, a topic to a queue, a queue to a topic, or a queue to a queue.

In addition, the EMS can export JMS messages to Rendezvous and import messages from Rendezvous. Rendezvous messages can be imported into JMS topics and queues, and JMS topics can be exported to Rendezvous messages. Export of JMS queues is not possible.

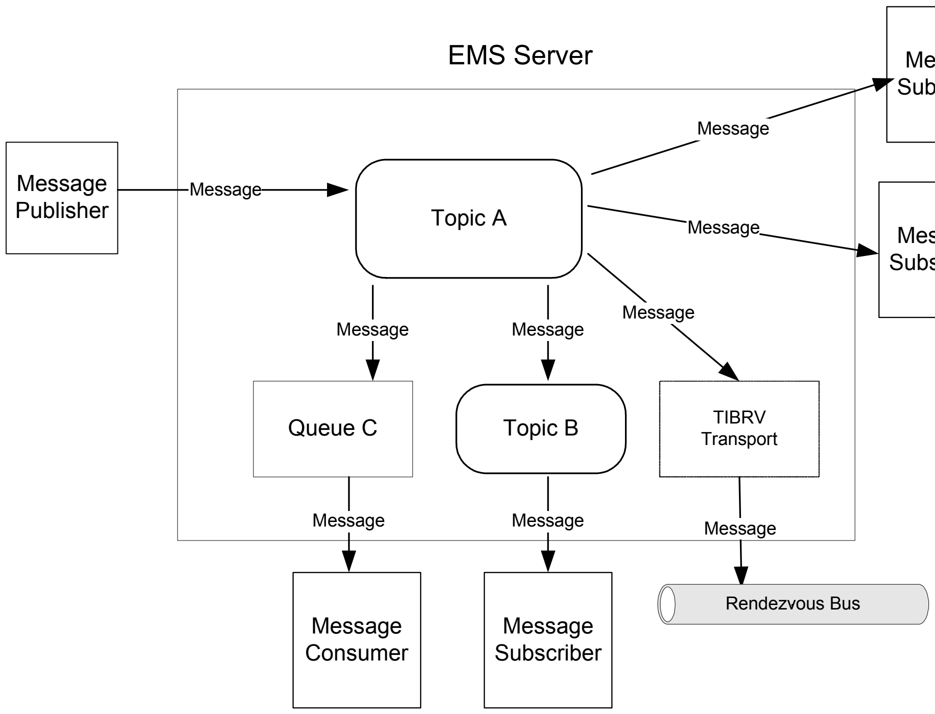

Figure 1. EMS Destination Bridging

When a message producer sends a message within a transaction, all messages sent across a bridge are part of the transaction. Therefore, if the transaction succeeds, all messages are delivered to all bridged destinations. If the transaction fails, no consumers for bridged destinations receive the messages.

2.4 EMS Routing

EMS provides the ability to route messages between servers.

A *route* is a named bi-directional connection between two EMS servers. Each route forwards messages between corresponding destinations in the two servers. Destinations correspond if they are of the same type and have the same name in both servers. Routes are specified via the EMS server administrative interfaces.

Once a route between EMS servers is configured, messages are only eligible for routing between servers if all of the following conditions are met:

- Both destinations have the same name
- Both destinations are of the same type (i.e. both topics or both queues)
- Both destinations have the global property is set to true. This indicates that messages sent to this destination are to be also sent along any configured routes.

A *zone* is a set of named routes, and each route belongs to a zone. The zone and its properties determine the message forwarding behaviour between routes. There are two types of zones: one-hop and multi-hop. In a one-hop zone, messages will only be forwarded once, from one server to another: no further propagation of the message will occur. In a multi-hop zone, messages can be forwarded across as many routes as necessary to reach all destinations.

A server can have routes that belong to several zones. When zones overlap at a server, the routing behaviour within each zone does not limit routing in other zones. That is, when a forwarded message reaches a server with routes in several zones, the message crosses zone boundaries, and its hop count is reset to zero.

For topic-based messages to be routed between servers, the eligibility criteria for routing (above) must first be met. Whether or not messages are actually forwarded to a topic on another server depends on whether or not that server is on the path to a known subscriber to the topic. Unless selectors have been specified, all messages sent to a global topic are forwarded to all routed servers that have subscribers for that topic (and, of necessity, any servers on the route in between).

Configuring queues for routing is somewhat different than for topics. In routing topics, the declaration of the topic is identical on all servers (with the possible exception of a message selector). Queue declarations, on the other hand, make a distinction between the server that owns the queue and other servers with routed queues that reference both the queue name and the owning server.

Routed queues serve as proxies for the real queue. Messages that are published to the proxy queue are forwarded to the real queue, and are not eligible for delivery until they reach the real queue. If the route from the proxy queue to the real queue is broken, the message will remain in the proxy queue until such time as the route is restored. Meanwhile, the message is not eligible for delivery while it sits in the proxy queue. While clients can connect to the proxy queue to receive messages, the message remains in the real queue until the client requests a message. The consequence is that if the route from the proxy queue to the real queue is broken, no clients attached to the proxy queues can receive messages until the route is restored. Note that active and passive routes do not impact the forwarding of messages for queues.

Queue messages can be routed at most one hop from the server that owns the queue.

Sometimes applications on different servers are only interested in a subset of the messages being published on a topic. Routing these messages to all applications can be expensive in terms of network bandwidth between the EMS servers, particularly if this is a WAN connection. In these circumstances it is reasonable to consider the use of a message selector to filter the messages being routed from one server to another. However, it should be recognized that the use of selectors will have a performance impact on the EMS server, as the selector expression will have to be evaluated for every message on the topic.

3 Point to Point Routing Design Pattern

The diagram below shows how a simple point-to-point queue pattern may be extended across multiple EMS servers. A global topic would be created to route messages between EMS servers and destination bridges configured to connect the topic and queue together. The use of routed topics is preferred since the advanced routing capabilities of topics allow scaling the number of servers beyond the one-hop restriction with queues.

The producer could still send messages to the same queue, which would exist on the producer's server, however since messages are copied and not moved across a bridge the queue must be configured appropriately to prevent message build up.

Alternatively the producer may be changed to publish directly to the routed topic, thereby negating the need for the producer side queue.

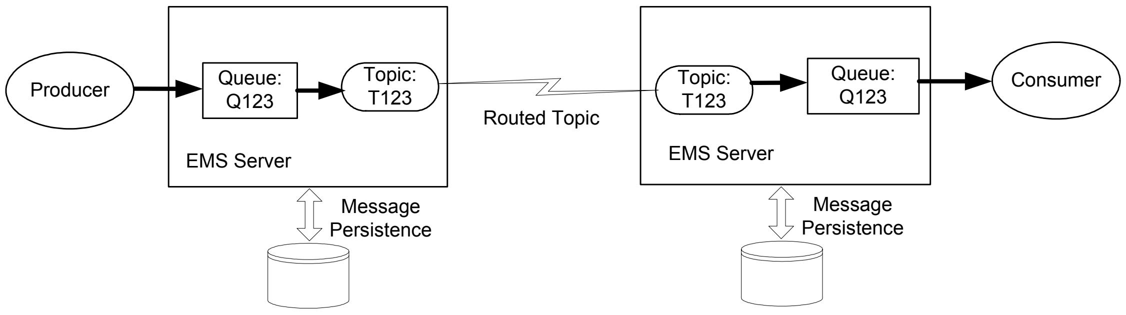

Figure 2. Point-to-Point Routing

Required property configuration for producer-side queue:

- maxmsgs = 1
- overflowPolicy = discardOld

Required property configuration for routed topic (both producer-side and consumer-side):

- global

Note; if authentication and permissions are enabled, it is not necessary to grant the producer or consumer user permission to access the routed topic. The secure property may be enabled on the routed topic to prevent any direct user access, however access must be granted for the EMS server user that the route is connected to.

4 Publish/Subscribe Routing Design Pattern

In a publish/subscribe scenario it is generally preferred for consumers to read from queues rather than use durable topic subscribers, because:

- Queues are easier to manage, for example, queues can be browsed and messages purged for one consumer without affecting other consumers.
- Queues support load-balancing across multiple consumers
- Proxy (or remote) queues can be used for high fan-out scenarios.

The diagram below shows an example publish/subscribe routing pattern where each consumer has a dedicated queue to receive messages. Selectors may be specified on the topic to queue bridges for filtering, so consumers only receive messages of interest. This is preferable than use of selectors at the consumer-side.

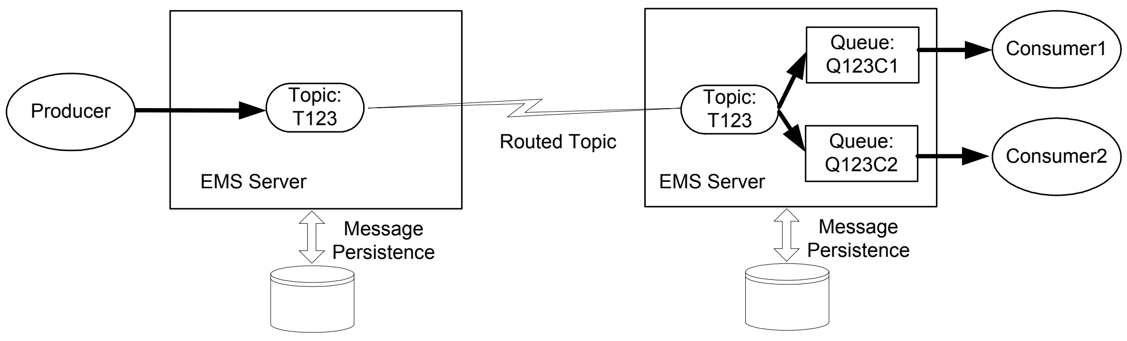

Figure 3. Publish/Subscribe Routing

Required property configuration for routed topic (both producer-side and consumer-side):

- global

For high fan-out scenarios across many consumers, clients can connect to a farm of EMS servers, as shown in the diagram below.

The EMS servers in the server farm have proxies for the primary queues, each EMS server in the farm will have the same configuration of proxy queues and clients may connect to any EMS server in the farm.

The proxies do not actually contain the messages, and no messages are ever stored in the server farm. The proxy queue retrieves messages from the primary queue on demand and delivers them to the client, should a client receive a message but not acknowledge the message the message would remain persisted on the EMS server hosting the primary queue.

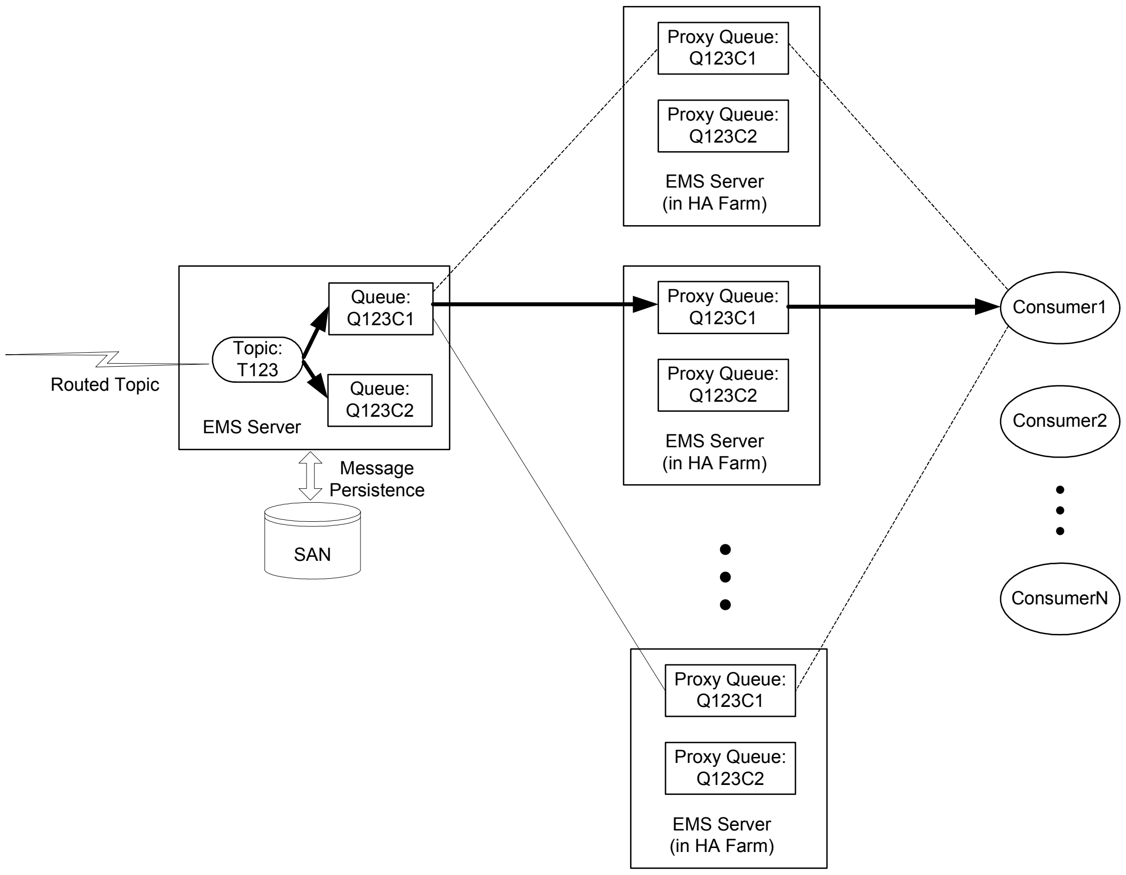

Figure 4. Publish/Subscribe Routing with High Fan-out

Required property configuration for primary (home) queue:

- global

Required property configuration for proxy (remote) queue:

- configuration name = queue_name@primary_server_name
- global

5 Synchronous Request/Reply Routing Design Pattern

The diagram below shows an example synchronous request/reply routing pattern. Again topics are used for routing request and reply messages, for the same reasons as described above.

A temporary topic is created for the reply message, the lifetime of the temporary topic is only for this request/reply operation, and it will automatically be deleted once the reply message has been consumed. It is important that the receiver should set an expiry time on the reply message to ensure the temporary is deleted in case that requester should timeout waiting for the reply.

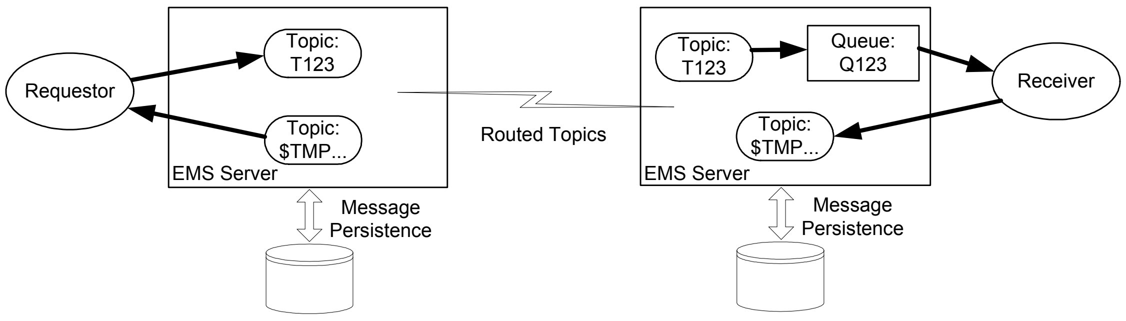

Figure 5. Synchronous Request/Reply Routing

Required property configuration for routed topic (both producer-side and consumer-side):

- global

Note: Temporary topics are automatically routed and no special configuration is required.

Requestor-side processing:

1. Create request topic publisher
2. Create request message
3. Create a new temporary reply topic
4. Create subscriber to the temporary reply topic
5. Set JMSReplyTo property in request message to temporary reply topic
6. Publish request message to request topic
7. Receive reply from temporary reply topic with timeout specified

Note; the subscriber to the temporary reply topic must be created before the request message is sent.

A generic receiver can be designed for both synchronous and asynchronous message patterns. In fact the receiver should not need to know, or care which pattern the producer client is using. The following steps describe an example of receiver-side processing for both synchronous and asynchronous message patterns.

Receiver-side processing:

1. Create request queue consumer
2. Create generic producer
3. Receive message from request queue
4. Process request
5. Create reply message
6. Retrieve the JMSCorrelationID property from request message
7. Propagate the JMSCorrelationID property from request message to the JMSCorrelationID property in the reply message
8. Retrieve the JMSReplyTo property from request message
9. If reply destination name starts with “$TMP.” set expiry time on reply message
10. Determine delivery mode for response (e.g. if delivery mode of request = PERSISTENT and reply destination name does not start with $TMP then use PERSISTENT else use NON_PERSISTENT)
11. Send reply message to destination specified in JMSReplyTo

Note; temporary topics will not survive a server restart. If durability is required a non-temporary topic and a correlation id should be used, as described in the next section for asynchronous request/reply. The requester can synchronously wait for the correct reply by specifying a selector on the JMSCorrelationID property.

6 Asynchronous Request/Reply Routing Design Pattern

The diagram below shows an example asynchronous request/reply routing pattern. Again topics are used for routing request and reply messages, for the same reasons as described above.

A non-temporary topic is created for the reply message. The lifetime of the reply topic is only for this requestor and it will be used for receiving multiple replies, possibly from different receivers. Requests and replies are correlated together using the standard JMSCorrelationID header property. The requestor must create and set the JMSCorrelationID property in the request, and the receiver must copy this into the reply.

For durability of replies a durable topic subscriber may be used or the reply topic can be bridged to a reply queue at the requestor side.

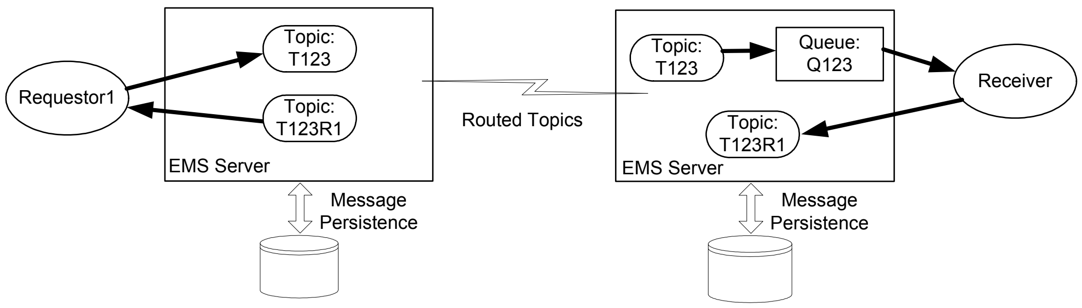

Figure 6. Asynchronous Request/Reply Routing

Required property configuration for routed topic (both producer-side and consumer-side):

- global
- expiry override may be set on reply topic if required

Topics may be created dynamically, as long as the name conforms to a well-defined naming standard, such that the topic inherits the global property setting. This is particularly desirable on the receiver-side, since it would be cumbersome to configure reply topics for every potential requestor.

Requestor-side processing:

1. Create request topic publisher
2. Create response topic subscriber
3. Create request message
4. Set JMSReplyTo property in request message to reply topic

5. Generate a unique identifier for this requestor
6. Set JMSCorrelationID property in the request to the unique identifier for request
7. Publish request message to the request topic
8. Receive reply from reply topic via message listener
9. Use JMSCorrelationID property in reply message to correlated the reply with a previous request.

A generic receiver can be designed for both synchronous and asynchronous message patterns. In fact the receiver should not need to know, or care which pattern the producer client is using. The following steps describe an example of receiver-side processing for both synchronous and asynchronous message patterns.

Receiver-side processing:

1. Create request queue consumer
2. Create generic producer
3. Receive message from request queue
4. Process request
5. Create reply message
6. Retrieve the JMSCorrelationID property from request message
7. Propagate the JMSCorrelationID property from request message to the JMSCorrelationID property in the reply message
8. Retrieve the JMSReplyTo property from request message
9. If reply destination name starts with “$TMP.” set expiry time on reply message
10. Determine delivery mode for response (e.g. if delivery mode of request = PERSISTENT and reply destination name does not start with $TMP then use PERSISTENT else use NON_PERSISTENT)
11. Send reply message to destination specified in JMSReplyTo