



# eMBus Messaging – Summary

## 1 About eMBus

Our flagship middleware product, eMBus is an enterprise-level message-oriented middleware application, supporting the deployment of Service Oriented Architecture solutions. It can carry any payload, commonly XML, FIX, and FIXML.

It offers a unique combination of point-to-point message delivery, and publish and subscribe multiple delivery models, with persistent and non-persistent delivery, especially tailored for the delivery of transactions, and market data respectively.

eMBus is highly scalable. eMBus services may be combined together into one network, using no additional components.

It is multiplatform, supporting the following:

- Windows
  - ActiveX Control, .NET assembly, C library
- Linux
  - C Library
- Solaris
  - C Library
- Java
  - Java Bean for MIDP2, Standard, and J2EE support

Samples are provided for each platform.

Additionally, multiple adapters to other technologies also exist. For example: Oracle AQ, MQ Series, SQL Server.

TraderTeam Systems also have message broker components based on eMBus that support FIX, FIXML, various trading systems, market data vendors and middle/back office systems.

## 2 How eMBus Works

Embus allows messages to be sent containing any content between multiple endpoints.

It has two models for message delivery:

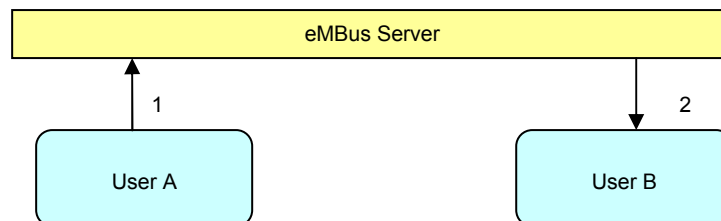
- It allows for simple point to point communication by addressing messages to specific users.

The diagram below shows User A sending to User B.

Logical Message Flow:



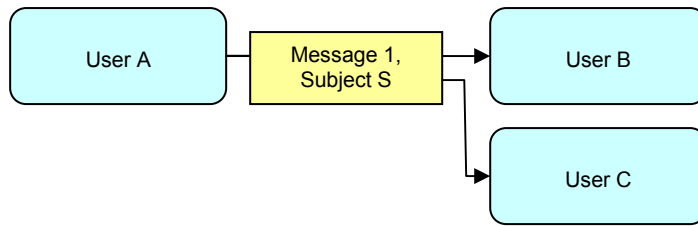
Physical Message Flow:



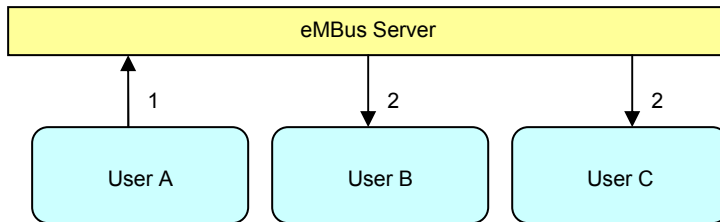
- It allows for multiple endpoint message delivery, using a publish and subscribe model. Subscribers subscribe to subjects, which can be dot-delimited in hierarchy. Publishers publish to a subject, and all subscribers to that subject, or that subject's hierarchy will receive the message.

The diagram below shows User A publishing a message (Message 1) on Subject S, to which Users B and C are subscribed. It then shows Users B and C receiving the published message.

Logical Message Flow:



Physical Message Flow:



## 2.1 Sticky Messaging

Additionally, eMBus supports a store-and-forward model for both point-to-point, and publish-and-subscribe messaging models. Store-and-forward essentially means that eMBus will hold a copy of the message until the user has acknowledged receipt. Messages are stored for delivery, even if the destination user is not currently connected. They are delivered once the user connects.

- For a point-to-point message, the sender must set the sticky message option when sending the message to the destination user. The destination user must acknowledge receipt of the message.
- For publish-and-subscribe messages, the recipient must first subscribe to the subject specifying the sticky subscription option. The recipient is then subscribed to this subject, even if they are no longer connected to the eMBus. Messages will be queued and delivered to them once they reconnect. Again, the recipient must call `Delivered` to acknowledge the messages.

## 2.2 Resilience

Messages are persisted to disk, and replicated to linked eMBus services to provide robust and resilient enterprise scale architecture.

### 3 Embus Features

Features of the platform include:

- Supports any payload  
Message content may be binary, ascii, unicode, xml, fix etc
- Publish and subscribe over subject structure  
One published message can be received by tens of thousands of subscribers.
- Subjects can be hierarchical  
e.g. With a message published on "UK.Equities.FTSE100.VOD", anyone subscribed to "UK.Equities.FTSE100", "UK.Equities" or just "UK" would receive it.
- Point to point messaging  
Messages can be addressed and delivered to specific users
- Store-and-forward mechanism  
Messages are stored to disk until acknowledged. This provides failure recovery without missed messages.
- Encryption  
Using the AES standard and X9.17 and X9.19 for key exchange. Session keys are used to prevent brute force attack on decrypting data in transit.
- Link.  
Link embuses together to create fault tolerant and scalable topologies.
- Multi platform.
- Supports the following technologies: C++, linux, solaris, java, Active/x, .net
- Easy, simple, straightforward api.  
Connect, subscribe, publish, and you are going. Samples provided
- Administrative interface provided.  
Admin api via messages Statistics and monitoring
- Permissions  
User access control levels supported through administrative API
- Audit trail  
Messages are audited and extensive logging available