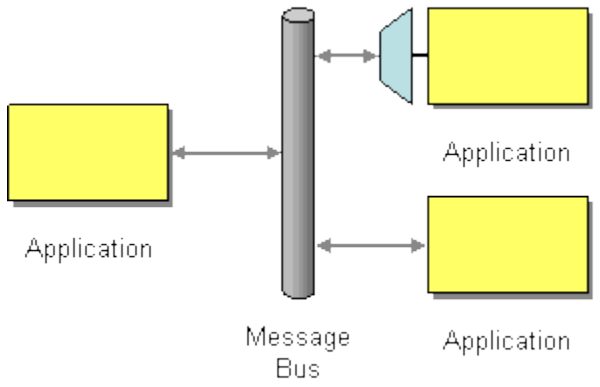
MESSAGE BUS AND COMMAND MESSAGE

Message Bus

An enterprise contains several existing systems that must be able to share data and operate in a unified manner in response to a set of common business requests.

What is an architecture that enables separate applications to work together, but in a decoupled fashion such that applications can be easily added or removed without affecting the others?



Structure the connecting middleware between these applications as a *Message Bus* that enables them to work together using messaging.

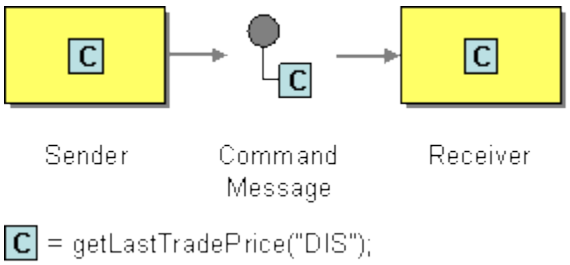
A *Message Bus* is a combination of a common data model, a common command set, and a messaging infrastructure to allow different systems to communicate

through a shared set of interfaces. This is analogous to a communications bus in a computer system, which serves as the focal point for communication between the CPU, main memory, and peripherals. Just as in the hardware analogy, there are a number of pieces that come together to form the message bus

Command Message

An application needs to invoke functionality provided by other applications. It would typically use *Remote Procedure Invocation*, but would like to take advantage of the benefits of using *Messaging*.

How can messaging be used to invoke a procedure in another application?



Use a *Command Message* to reliably invoke a procedure in another application.

There is no specific message type for commands; a *Command Message* is simply a regular message that happens to contain a command.

In JMS, the command message could be any type of message; examples include an ObjectMessage containing a Serializable command object, a TextMessage containing the command in XML form, etc. In .NET, a command message is a Message with a command stored in it. A Simple Object Access Protocol (SOAP) request is a command message.

Source:
http://www.enterpriseintegrationpatterns.com/patterns/messaging/CommandMessage.html