WHAT IS MESSAGING?

This article is about how to use messaging to integrate applications. A simple way to understand what messaging does is to consider the telephone system. A telephone call is a synchronous form of communication. I can only communicate with the other party if the other party is available at the time I place the call. Voice mail on the other hand, allows asynchronous communication. With voice mail, when the receiver does not answer, the caller can leave him a message; later the receiver (at his convenience) can listen to the messages queued in his mailbox. Voice mail enables the caller to leave a message now so that the receiver can listen to it later, which is lot easier than trying to get the caller and the receiver on the phone at the same time. Voice mail bundles (at least part of) a phone call into a message and queues it for later consumption; this is essentially how messaging works.

Messaging is a technology that enables high-speed, asynchronous, program-to-program communication with reliable delivery. Programs communicate by sending packets of data called *message*s to each other. *Channel*s, also known as queues, are logical pathways that connect the programs and convey messages. A channel behaves like a collection or array of messages, but one that is magically shared

across multiple computers and can be used concurrently by multiple applications. A *sender* or *producer* is a program that sends a message by writing the message to a channel. A *receiver* or *consumer* is a program that receives a message by reading (and deleting) it from a channel.

The message itself is simply some sort of data structure—such as a string, a byte array, a record, or an object. It can be interpreted simply as data, as the description of a command to be invoked on the receiver, or as the description of an event that occurred in the sender. A message actually contains two parts, a header and a body. The *header* contains meta-information about the message—who sent it, where it's going, etc.; this information is used by the messaging system and is mostly (but not always) ignored by the applications using the messages. The *body* contains the data being transmitted and is ignored by the messaging system. In conversation, when an application developer who is using messaging talks about a message, he's usually referring to the data in the body of the message. Asynchronous messaging architectures are powerful, but require us to rethink our development approach. As compared to the other three integration approaches, relatively few developers have had exposure to messaging and message systems. As a result, application developers in general are not as familiar with the idioms and peculiarities of this communications platform.

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