BASIC MESSAGING CONCEPTS

Like most technologies, *Messaging* involves certain basic concepts. Once you understand these concepts, you can make sense of the technology even before you understand all of the details about how to use it. These basic messaging concepts are:

Channels — Messaging applications transmit data through a *Message Channel*, a virtual pipe that connects a sender to a receiver. A newly installed messaging system doesn't contain any channels; you must determine how your applications need to communicate and then create the channels to facilitate it.

Messages — A *Message* is an atomic packet of data that can be transmitted on a channel. Thus to transmit data, an application must break the data into one or more packets, wrap each packet as a message, and then send the message on a channel. Likewise, a receiver application receives a message and must extract the data from the message to process it. The message system will try repeatedly to deliver the message (e.g., transmit it from the sender to the receiver) until it succeeds.

Multi-step delivery — In the simplest case, the message system delivers a message directly from the sender's computer to the receiver's computer.

However, actions often need to be performed on the message after it is sent by its original sender but before it is received by its final receiver. For example, the message may have to be validated or transformed because the receiver expects a different message format than the sender. A *Pipes and Filters* architecture describes how multiple processing steps can be chained together using channels.

Routing — In a large enterprise with numerous applications and channels to connect them, a message may have to go through several channels to reach its final destination. The route a message must follow may be so complex that the original sender does not know what channel will get the message to the final receiver. Instead, the original sender sends the message to a *Message Router*, an application component and filter in the pipes-and-filters architecture, which will determine how to navigate the channel topology and direct the message to the final receiver, or at least to the next router.

Transformation — Various applications may not agree on the format for the same conceptual data; the sender formats the message one way, yet the receiver expects it to be formatted another way. To reconcile this, the message must go through an intermediate filter, a *Message Translator*, that converts the message from one format to another.

Endpoints — An application does not have some built-in capability to interface with a messaging system. Rather, it must contain a layer of code that knows both how the application works and how the messaging system works, bridging the two so that they work together. This bridge code is a set of coordinated *Message Endpoint*s that enable the application to send and receive messages.

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