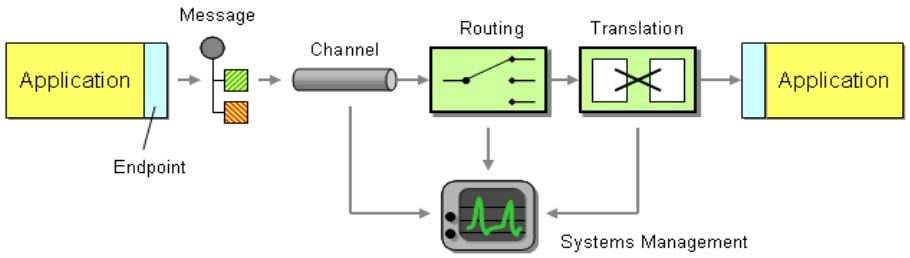
A LOOSELY COUPLED INTEGRATION SOLUTION

In order to connect two systems via an integration solution, a number of things have to happen. These things make up what we call middleware – the things that sit between applications.

Invariably, some data has to be transported from one application to the next. This data could be an address record that needs to be replicated, a call to a remote service or a snippet of HTML headed for a portal display. Regardless of the payload, this piece of data needs to be understood by both ends and needs to be transported, usually across a network. Two elements provide this basic function. We need a communications *channel* that can move information from one application to the other. This channel could be a series of TCP/IP connections, a shared file, a shared database or a floppy disk being carried from one computer to the next (the infamous ‘sneakernet’). Inside this channel we place a message – a snippet of data that has an agreed-upon meaning to both applications that are to be integrated. This piece of data can be very small, such as the phone number of a single customer that has changed, or very large, such as the complete list of all customers and their associated addresses. We call this piece of data a *message*.



Basic Elements of an Integration Solution

Now that we can send messages across channels we can establish a very basic form of integration. However, we promised that simple integration is an oxymoron, so let's see what is missing. We mentioned before that integration solutions often have limited control over the applications they are integrating, such as the internal data formats used by the applications. For example, one data format may store the customer name in two fields, called FIRST_NAME and LAST_NAME, while the other system may use a single field called Customer_Name. Likewise, one system may support multiple customer addresses while the other system only supports a single address. Because the internal data format of an application can often not be changed the middleware needs to provide some mechanism to convert one application's data format in the other's. We call this step *translation*.

So far we can send data from one system to another and accommodate differences in data formats. What happens if we integrate more than two systems? Where does the data have to be moved? We could expect each application to specify the target system(s) for the data it is sending over the channel.

For example, if the customer address changes in the customer care system we could make that system responsible for sending the data to all other systems that store copies of the customer address. As the number of systems increases this becomes very tedious and requires the sending system to have knowledge about all other systems. Every time a new system is added, the customer care system would have to be adjusted to the new environment. Things would be a lot easier of the middleware could take care of sending messages to the correct places. This is the role of a *routing* component such as a message broker.

Integration solutions can quickly become complex because they deal with multiple applications, data formats, channels, routing and transformation. All these elements may be spread across multiple operating platforms and geographic locations. In order to have any idea what is going on inside the system we need a *systems management* function. This subsystem monitors the flow of data, makes sure that all applications and components are available and reports error conditions to a central location.

Our integration solution is now almost complete. We can move data from one system from another, accommodate differences in the data format, route the data to the required systems and monitor the performance of the solution. So far we assumed that an application sends data as a message to the channel.

However, most packaged and legacy applications and many custom applications are not prepared to participate in an integration solution. We need a message *endpoint* to connect the system explicitly to the integration solution. The endpoint can be a special piece of code or a *Channel Adapter* provided by an integration software vendor.

Source:
http://www.enterpriseintegrationpatterns.com/patterns/messaging/Chapter1.html