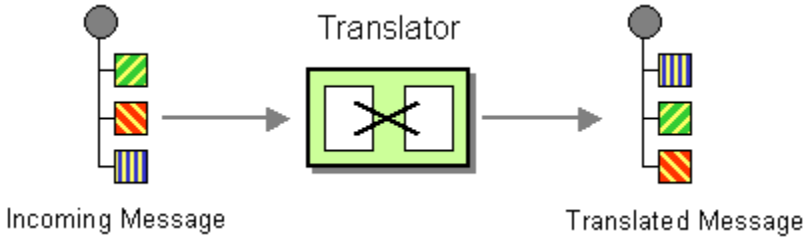
MESSAGE TRANSLATOR

The previous patterns describe how to construct messages and how to route them to the correct destination. In many cases, enterprise integration solutions route messages between existing applications such as legacy systems, packaged applications, homegrown custom applications, or applications operated by external partners. Each of these applications is usually built around a proprietary data model. Each application may have a slightly different notion of the *Customer* entity , the attributes that define a *Customer* and which other entities a *Customer* is related to. For example, the accounting system may be more interested in the customer's tax payer ID numbers while the customer-relationship management (CRM) system stores phone numbers and addresses. The application's underlying data model usually drives the design of the physical database schema, an interface file format or a programming interface (API) -- those entities that an integration solution has to interface with. As a result, the applications expect to receive messages that mimic the application's internal data format.

In addition to the proprietary data models and data formats incorporated in the various applications, integration solutions often times interact with standardized data formats that seek to be independent from specific applications.

There are a number of consortia and standards bodies that define these protocols, such as RosettaNet, ebXML, OAGIS and many other, industry specific consortia. In many cases, the integration solution needs to be able to communicate with external parties using the 'official' data formats while the internal systems are based on proprietary formats.

How can systems using different data formats communicate with each other using messaging?



Use a special filter, a *Message Translator*, between other filters or applications to translate one data format into another.

The *Message Translator* is the messaging equivalent of the *Adapter* pattern described in [GoF]. An adapter converts the interface of a component into a another interface so it can be used in a different context.

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