Discussion Document – Layers of Data Standards

This is a brief and general description of four "layers" of work that build upon each other in a data standard design effort. All four layers do not have to exist to constitute a data standard. However, as more layers are added, the cost of systems integration decreases.

- **Data Dictionary:** This is simply a list of data *elements* each with a title, definition and sometimes a format. For example: Title: Birth Date; Definition: Day the individual was born; Format: year-month-day.
- **Data Model:** Defines *entities* as collections of *properties*. Each *property* is an *element* in the data dictionary. In other words, an element becomes a property when it's assigned to an entity. Also defines *relationships* between entities. For example, a student entity might have the name, birthdate, gender, address, etc. The student entity type would have a many to many *relationship* with the class entity type.
- **Serialization:** This may also be called "binary format," "storage format," or "encoding." It's a concrete format that data may be stored in. Usually this is an interchange format rather than the format that's used by a particular product or system. Examples are XML and JSON. The conversion from a Data Model to a particular Serialization such as XML isn't automatic. There still needs to be a specification that says exactly how the data model is rendered into a specific serialization. There may be (and often are) multiple serializations of the same data model.
- **Protocol:** A specific way to access, transport or exchange data records. HTTP, SOAP, and EDI are general purpose protocols that may be used for a variety of data models. Oftentimes, a protocol will be associated with a particular serialization. For example SOAP is best suited to XML serialization whereas HTTP can handle any serialization or binary format. REST is a design philosophy for protocols but not a protocol in and of itself.

The task of a systems integrator becomes easier and less expensive as more layers are standardized. When all four layers are addressed, systems integration should be a matter of proper configuration settings with no custom programming required. On the other hand, standards (or portions thereof) that are limited to the higher levels of the stack have broader applicability. For example, a principal benefit of a standardized Data Dictionary is reducing the risk that data may be interpreted differently by different systems. So, even this first step yields tremendous benefits. Because of this, it's important to clearly delineate between the layers even when a single standard or specification addresses more than one.