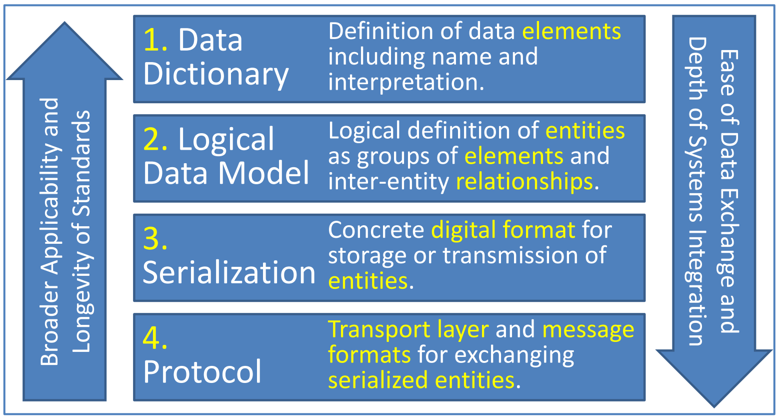
A Four-Layer Framework for Data Standards

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, application interoperability is enhanced and the cost of systems integration decreases.


Broader Applicability and Longevity of Standards
1. Data Dictionary
Definition of data elements including name and interpretation.
2. Logical Data Model
Logical definition of entities as groups of elements and inter-entity relationships.
3. Serialization
Concrete digital format for storage or transmission of entities.
4. Protocol
Transport layer and message formats for exchanging serialized entities.
Ease of Data Exchange and Depth of Systems Integration

1. **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.
2. **Logical 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 include the properties name, birthdate, gender, address, etc. The Student entity type would have a many to many *relationship* with the Class entity type.
3. **Serialization:** This is a concrete format in which entities may be stored or exchanged. Two frameworks for serialization are XML and JSON but custom serializations are also common. 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 particular serialization. There may be (and often are) multiple serializations of the same data model. Synonymous terms include "physical data model," "binary format," "marshaled format," "binding," "storage format," or "encoding."
4. **Protocol:** The infrastructure over which the serialized representations of Data Model Entities are accessed and exchanged. When it is standardized for a particular domain, the entire communication between applications is defined and out-of-the-box interoperability between conforming applications becomes possible. Protocol really contains many layers, hence the term "protocol stack." Typical layers include Messaging Framework (e.g. Publish/Subscribe, Request/Response, Create/Read/Update/Delete, REST, SOAP, Enterprise Service Bus), Transport (e.g. HTTP(S) or FTP) and Network (e.g. TCP/IP).

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 significant and broad-reaching benefits. Because of this, it's important to clearly delineate between the layers even when a single standard or specification addresses more than one.

Previous versions of this document referred to Layer 2 as simply "Data Model." However, the Common Education Data Standards project calls this this the "Logical Data Model" and Layer 3 the "Physical Data Model." This edition compromises by using "Logical Data Model" for Layer 2 but retains "Serialization" for Layer 3 as we feel it's a more accurate description.