**Data Types Data Model – Revision 2.0**

**Data Model**

1. Types characterize data.
2. Some types of data are so fundamental that we define them implicitly. E.g., integer, boolean, string, ID, etc. We call them ‘primitive' types.
3. Types are built from other types. We call types that are built from other types ‘derived’ types.
4. Types use properties for such building. Properties are names (potentially identifiers) that establish the context in which types are aggregated to form derived types.

For example, a ‘Person’ type can be defined using three properties - height, weight, and age. Here, the three properties are just names to indicate what the two floating-point values (height and weight) and one integer value (age) are for. For now, properties are primarily intended for machine reading and human reasoning; future versions of the data model may define properties for machine reasoning as well. Wherever applicable, we will reuse existing standards.

1. Formally speaking, a property has a domain (the type that is being built) and a range (the type used for building).
2. Types contain at least one property: ‘description’ with a range ’string’ (a primitive type).
3. We recommend that types also contain the following properties; ranges of those properties are specified after the colon.
	1. provenance: Contact

Provenance property is intended to capture the person or organization that created this type. Here, Contact is a derived type also pre-defined in the Type Registry to capture point of contact details.

* 1. intent: String

Intent property is intended to capture the intent for creating data of this type.

* 1. explanation-of-use: String

Explanation-of-use property is intended to capture how the creator (of this type) intends others to use this type.

* 1. mime-types: List of Strings

Mime-types property is intended to capture any mime-types that are applicable or useful for describing the property. For now, only a list is allowed. In the future, how mime-types are related to each other may be captured.

* 1. semantic-web-links: List of IDs

Semantic-web-links property is intended to capture any Semantic Web or Linked Data URLs that are applicable or useful for describing the property. Like mime-types property, this is only a list. In the future, relationships between these links may be captured.

1. Types when registered in a Type Registry will be assigned a unique ID. Alternatively, type definitions may suggest unique IDs at the time of registration.

**Tabled Complexities**

1. Note that the words used for naming properties, e.g., description, age, weight, etc., have some hidden semantics. For example, age, weight, and height are measurements, and have units associated with them. Also, in spite of weight and height having the same range (Float), they mean different things. Introducing a notion of ‘attributes’ to further capture the semantics of the properties, such as measurement units, formatting conventions (think dates and timestamps), etc. might be useful. ‘Attributes’ are specifically avoided in this version of the data model for reducing the entry barrier for uptake.
2. Properties cannot be just English words. They should be unique IDs to encourage reuse. Reuse will indicate the correlation between unrelated datasets. For example, if both Wine Type and Person Type use the same property ‘age’, it indicates some synergistic relationship between Wine and Person type with time. Not to mention multi-lingual issues if natural languages are adopted to identify a property. Assigning unique IDs to properties seems like a good long-term solution to these problems. To keep the barrier to entry low, and for immediate type registry release, properties are not unique IDs in this version of the data model.
3. We realize that the constructs presented here to define types can be used in multiple ways. For example, a community might define ‘temperature’ to be a property with a range ‘Float’. Another community might define ‘temperature’ as a type and include ‘energy-level’ as one of its properties to capture the heat-energy. There isn’t anything we can do about the discrepancy (even if you want to term it like that), other than maybe produce best practices and guidelines wherever applicable.
4. We also realize that there is a room or role for profiles (or whatever they are called). One reason to introduce the notion of a profile is to allow for inter-community differences in how the types are used: one community might define temperatures in Fahrenheit and the other in Centigrade. Another reason for defining profiles is mandatory versus optional use of certain properties within a type. The third reason could be to capture the variations in controlled vocabulary enforcements. Profiles are also tabled in this version of the data model.
5. While derived types are constructed from other types allowing for type-aggregation, type-inheritance is not supported in this version of the data model. For example, specifying that ‘Type Y is all that Type X is and more’ cannot be defined now. Future version of the data model will venture into these nuances.
6. Once types are defined, services or systems that process, evaluate, visualize, or transform data based on its type can be created and be linked back in the type record. Such uses of type records are anticipated.