vlingo/schemata Specification

This component is a schema registry. It provides the means for services built using the vlingo/platform to publish standard types that are made available to client services. The published standard types are known as schemas, and the registry hosts the schemas for given organizations and the services within. This is the basic hierarchy:

```
Organization
  Unit
    Context
      Commands
        Schema
        ...
      Data
        Schema
        ...
      Documents
        Schema
        ...
      Envelopes
        Schema
        ...
      Events
        Schema
        ...
      Metadata
        Schema
        ...
```

From the top of the hierarchy the nodes are defined as follows:

- **Organization:** The top-level division. This may be the name of a company or the name of a prominent business division within a company. If there is only one company using this registry then the Organization could be a major division within the implied company. There may be any number of Organizations defined, but there must be at least one.
- **Unit:** The second-level division. This may be the name of a business division within the Organization, or if the Organization is a business division then the Unit may be a department or team within a business division.
- **Context:** The logical application or (micro)service within which schemas are to be defined and for which the schemas are published to potential consumers. You may think of this as the name of the *Bounded Context,* and it may even be appropriate to name it the top-level namespace used by the Context, e.g. `com.saasovation.agilepm`. Within each Context there may be a number of parts used to describe its *Published Language* served by its *Open Host Service.* Currently these include: Commands, Data, Documents, Envelopes, Events, and Metadata. Some of the parts are meant to help defined other parts, and so are building blocks. Other parts are the highest level of the *Published Language.* These are called out in the following definitions.

- **Commands:** This is a top-level schema type where Command operations, such as those supporting CQRS, are defined by schemas. If the Context's *Open Host Service* is REST-based, these would define the payload schema submitted as the HTTP Request body of POST, PATCH, and DELETE methods. If the *Open Host Service* is asynchronous-message-based mechanism (e.g. RabbitMQ or Kafka), these would define the payload of Command messages sent through the messaging mechanism.
- **Data:** This is a building-block schema type where general-purpose data records, structures, or objects are defined and that may be used inside any of the other schema types.
- **Documents:** This is a top-level schema type that defines the full payload of document-based operations, such as the query results of CQRS queries. These documents are suitable for use as REST Response bodies and messaging mechanism payloads.
- **Envelopes:** This is a building-block schema type meant to define the few number of message envelopes that wrap message-based schemas. When sending any kind of message, such as Command messages and Event messages, it is common to wrap these in an Envelope that defines some high-level metadata about the messages being sent by a sender and being received by a receiver.
- **Events:** This is a top-level schema type that conveys the facts about happenings within the Context that are important for other Context's to consume. These are known as *Domain Events* but may also be named *Business Events.* The reason for the distinction is that some viewpoints consider Domain Events to be internal-only events; that is, those events only of interest to the owning Context. Those holding that viewpoint think of events of interest outside the owning Context as Business Events. To avoid any confusion the term Event is used for this schema type and may be used to define any event that is of interest either inside or outside the owning Context, or both inside and outside the owning Context.
- **Metadata:** This is a building-block schema type that is used to define metadata data records, structures, or objects that are used inside any of the other schema types.

More to come...