Messaging Formats

Building on the simple messaging format, service broker, listeners and senders prototyped in DataAdminServices. Thinking about adding Avro as a fundamental part of messaging architecture.

Schema Registry

- Each type of message has an Avro schema associated with it.
- Register each schema in a Schema Registry, hosted on **Redis**, keyed by a unique name and version number.
- Use the Schema Registry to lookup a schema by ID and version, retrieve the schema, and validate a message against the schema.
- Access to Schema Registry is handled via **IOServices**.

Schema in the Messaging Protocol

- Use the schema to encode and decode message bodies.
- Provide the schema ID and version as part of the message protocol.
- Extend the asyncio prototype... Each message would have 3 parts:
    - The topic to subscribe to (or \null)
    - The schema ID and version
    - The message body, encoded per the schema

Tools for Message Schema Design and Build

- Use the "avro" package in PyPI
- Start out simple, and then add more complexity as needed.
- Use plaintext, simple JSON at first, then look at binary, encrypted, compressed options.
- Think, longer-term, about including these schema definitions as one of the things that can be edited in the `Saskan Eyes` editor. Would like to automate storing, versioning, monitoring, reporting on them, as with other resoures.

Basic Schema Design

- An Avro Schema minimally has `type` and an object which describes its `form`.
    - The type is one of the Avro complex types: `record`, enum, array, map, union, fixed.
    - Assume all of my schemas will be records to start out.
- Usually has a name too.
    - Use the `name` to identify what Message uses the schema.
    - A `namespace` is optional, helps to qualify the name.
        - Maybe it identifies what component of the system "owns" this schema.
- The object is a dictionary of `fields`.
    - Each field has a `name` and `type` and may have optional attributes too, like fixed values or enums.
    - The type is one of the Avro primitive types: int, long, float, double, string, boolean, bytes, null.
    - Optional attributes include:

        - "default": default value for field
        - "order": order of field in record
        - "aliases": list of aliases for field
        - "doc": documentation for a speicific field
- Everything is JSON.
- Thinking to store a hash of the schema to assist with automation of versioning.
- The doc field could be a URL that points to a wiki or something.
- In this example,, the response also identifies what SaskanConcept, what CodeTypes were created/retrieved. A JSON map is a hash, an associative array, a dict. The keys are always string. So we'd name the concept, then provide the underlying ontology or link to it. Then name the code-type, and either provide the code in the message, or provide a link to it.
- I am thinking it would be better to use Redis to store the contents, at least temporarily. So the value part of the maps might just be a UUID kind of thing, or maybe a named mnemonic key, or possibly the hash of the value would be best. Not sure yet. Thinking ahead to optimizing the BowDataSchema routines -- can probably check Redis first to see if the object we want has already been generated & whether or not it is fresh.

```
{
  "type": "record",
  "name": "GetSaskanDataObjectRequest",
  "namespace": "net.genuinemerit.data",
  "aliases": ["queue_GetSaskanDataObjectRequest",
              "get_saskan_data_object_request",
              "queue_get_saskan_data_object_request"],
  "fields": [
    {"name": "topics", "type": "array"},
    {"name": "version", "type": "string"},
    {"name": "hash", "type": "string"},
    {"name": "doc", "type": "string"},
    {"name": "SaskanConcepts", "type": "list"},
    {"name": "CodeTypes", "type": "list"}
  ]
}

{
  "type": "record",
  "name": "GetSaskanDataObjectResponse",
  "namespace": "net.genuinemerit.data",
  "aliases": ["queue_GetSaskanDataObjectResponse",
              "get_saskan_data_object_response",
              "queue_get_saskan_data_object_response"],
  "fields": [
    {"name": "topics", "type": "array"},
    {"name": "version", "type": "string"},
    {"name": "hash", "type": "string"},
    {"name": "doc", "type": "string"},
    {"name": "SaskanConcepts", "type": "map"},
    {"name": "CodeTypes", "type": "map"}
  ]
}
```