Allotrope Simple Models (ASM)
Validation and Hands-On Demonstration

Allotrope Virtual Connect, Fall 2021
September 2021

Rethinking Scientific Data
JSON Schema Primer
JSON Schema Primer

- **JSON Schema** is a vocabulary that allows the annotation and validation of JSON documents
  - It provides a detailed description for the data format
  - It provides clear human & machine-readable documentation.
  - It validates the data which is useful for:
    - Automated testing.
    - Ensuring the quality of the submitted data in a client-server architecture.
JSON Schema Primer

- **JSON Schema** is a vocabulary that allows the annotation and validation of JSON documents
  - Why do we need to formalize the description?... For content producers:
    - get clear guidelines
    - distributed content structure is unified and interoperable
    - content distribution can be associated with its structure
JSON Schema Primer

- **JSON Schema** is a vocabulary that allows the annotation and validation of **JSON** documents
  - Why do we need to formalize the validation?... For content consumers:
    - get clear guidelines on the expected structure
    - can verify it prior to processing
    - expected content structure is unified and interoperable

©2021 Allotrope Foundation
Some basic JSON schema definition:

- **JSON schema has properties called keywords**
- **keywords** are expressed as JSON keys
- **Schema keywords:**
  - The schema: `$schema` specifies the standard version used to draft this document and it provides version control.
  - The identifier: `$id` defines the schema URI. It also serves as the base URI so relative URI-references in keywords within the schema can be resolved against.
- **Annotation keywords:**
  - The title: `title` annotation keywords is descriptive only and do not add constraints to the validated data being validated. The intent is stated with the keyword.
  - The description: `description` annotation keywords is descriptive only and do not add constraints to the validated data being validated. The intent is stated with the keyword.
Validation keywords:

- The type: `type` validation keyword defines a constraint on the JSON data. In this case it must be a JSON Object.
- The properties: `properties` validation keyword defines a constraint on the JSON data properties.
- The required: `required` validation keyword defines a list where every item in the array is the name of a property in the data instance that needs to be presence.

```json
{
    "instrumentId": 45,
    "instrumentName": "bench top refrigerator",
    "opStatus": true,
    "tags": ["research", "clinical"]
}
```

```json
{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "$id": "https://example.com/instrument.schema.json",
    "title": "Instrument",
    "description": "An instrument in the laboratory",
    "type": "object",
    "properties": {
        "instrumentId": {
            "description": "A unique identifier of a lab instrument",
            "type": "integer"
        }
    },
    "required": ["instrumentId"]
}
```
JSON object property definition

- **instrumentId** JSON key has a numeric value that uniquely identifies an instrument. Given that it is instrument identifier for an instrument, it make sense to mandate it by its listing with the *required* validation.
- **instrumentName** JSON key has a string value that names the instrument. Given that it is instrument name, it make sense to mandate it by its listing with the *required* validation.
- **opStatus** JSON key has a boolean value that indicate the instrument operational status. It make sense to mandate it by its listing with the *required* validation.
- The *required* validation keyword is a list of strings where multiple keys can be noted as required;
  - **instrumentName** and **opStatus** are added to the list.
Additional constraints on the properties:

- It makes sense that the instrument identifier is a positive number
- We specify that value of `instrumentId` must be an integer greater than zero using the `exclusiveMinimum` validation keyword.

Other validation keywords for numeric (number or integer) instances:
- multipleOf
- Maximum
- exclusiveMaximum
- minimum
Additional constraints in the `tags` key:

Let’s assume the following requirements on the instrumentation tagging:

- If there are `tags` there must be at least one `tag`,
- `tags` must be unique; meaning no duplication for a single instrument.
- `tags` must be in a text format.
- `tags` are nice to have but not required to be present.
Those requirements on `tags` translate to:

- The `tags` key is added with the annotations and keywords.
- The type validation keyword is **array**.
- `items` validation keyword is added so we can define what appears in the array.
  - In this case: `string` values via the **type** validation keyword.
- A `minItems` validation keyword is used to make sure the existence of at least one item in the **array**.
- The `uniqueItems` validation keyword means that each one of the **items** in the **array** must be unique.
- The `tags` key was not added to the **required** validation keyword array since it is optional.
Adding a nested JSON data structure:

- A `dimensions` key is added to the data instance.

Therefore:

- The `dimensions` key is added to the schema with the `type` validation keyword and an `object` value.
- The `properties` validation keyword is used to define a nested data structure.
- To prevent verbosity, the `description` annotation keyword is omitted.
- A `required` validation keyword is added and is applicable to the nested dimensions key only!
An outside JSON schema can be referenced:

- Decomposition and modularization of a JSON schema:
  - minimizes verbosity,
  - enhances readability,
  - reduces maintainability
  - increases reusability

- It is a good practice to share a JSON schema across data structures.

- For example; a reuse of a common geo location JSON schema by referencing JSON Schema Primer ©2021 Allotrope Foundation
Reference an outside **JSON schema**

- The geo location **JSON Schema** includes the min max constraints to perform a range validation:
  - `minimum` validation keyword
  - `maximum` validation keyword
The complete data instance for validation by the JSON schema

```
{
  "instrumentId": 45,
  "instrumentName": "bench top refrigerator",
  "opStatus": true,
  "tags": ["research", "clinical"],
  "dimensions": {
    "length": 14.1,
    "width": 22.5,
    "height": 15.0
  },
  "labLocation": {
    "latitude": 67.44,
    "longitude": -55.34
  }
}
```

Referenced JSON Schema

```
{
  "$id": "https://example.com/geo-location.schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "Longitude and Latitude",
  "description": "Geographical coordinates",
  "required": ["latitude", "longitude"],
  "type": "object",
  "properties": {
    "latitude": {
      "type": "number",
      "minimum": -90,
      "maximum": 90
    },
    "longitude": {
      "type": "number",
      "minimum": -180,
      "maximum": 180
    }
  }
}
```
The Allotrope Simple Model (ASM)

- ASM is a simple text representation of the Allotrope tabular models using JSON.
- It uses terms from the Allotrope Foundation Ontology (AFO), and it leverages the Allotrope Data Model (ADM) already defined and governed by Allotrope and SMEs.
- Allotrope tabular models apply to domains where there is a single business object being measured and all measurements directly relate to this object.
- ASM utilizes JSON Schema standard for validation
ASM JSON Model Example: Conductivity ASM

- The Conductivity Tabular Model – **unique** AFO Parameter prefLabel
- “The names within an object SHOULD be **unique**” (JSON specifications: [RFC 4627](https://tools.ietf.org/html/rfc4627))

<table>
<thead>
<tr>
<th>Parameter prefLabel</th>
<th>Example Parameter Value</th>
<th>Parameter Unit Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>measurement identifier</td>
<td>413befdd</td>
<td></td>
</tr>
<tr>
<td>measurement time</td>
<td>2015-09-24T03:47:13.001Z</td>
<td></td>
</tr>
<tr>
<td>analyst</td>
<td>Amgentoaks1</td>
<td></td>
</tr>
<tr>
<td>sample identifier</td>
<td>unknown-10</td>
<td></td>
</tr>
<tr>
<td>equipment serial number</td>
<td>serial-number</td>
<td></td>
</tr>
<tr>
<td>batch identifier</td>
<td>batch-number</td>
<td></td>
</tr>
<tr>
<td>conductivity</td>
<td>273000 S/m</td>
<td></td>
</tr>
<tr>
<td>temperature</td>
<td>28.6 degC</td>
<td></td>
</tr>
</tbody>
</table>

©2021 Allotrope Foundation
ASM JSON Model Example: Conductivity ASM

- The Conductivity Tabular Model in an ASM JSON format: key-value pairs
- Keys are unique prefLabels in the AF Ontology (AFO).
ASM JSON Model Example: Conductivity ASM

- Each simple model JSON file MUST have a single reference to an ADM manifest (resolvable by the Allotrope PURL server)

```json
{
    "$comment": "Conductivity ASM",
    "measurement identifier": "413befdd",
    "measurement_time": "2015-09-24T03:47:13.001Z",
    "analyst": "Amgentoaks1"
}
```

The manifest provides:
- model identifier
- type
- ref to the ADM
- vocabulary
- ref to the JSON schema validation
ASM JSON Schema

- Allotrope Simple Models are referencing **JSON schemas** for validation.
- These schemas are standard **JSON schema** following the latest specification (2020-12). [https://json-schema.org/specification.html](https://json-schema.org/specification.html)
- While **SHACL** (Shapes Constraint Language) is used to validate an ADM instance data, **JSON schema** is used to validate an ASM instance data.
- ASM JSON schema are generated using a transformation tool from **SHACL** to **JSON schema**
ASM JSON Schema

• Unlike **SHACL**, **JSON schema** is not semantically aware
• In order to maintain the **SHACL** semantic constraints available in the **JSON schema** and to help with transformations, the simple models **JSON schema** contain some Allotrope specific annotations, starting with a "\$asm.*" prefix.
• Generic **JSON schema** tools can and will ignore these annotations.
**ASM JSON Schema**

*JSON schemas* allow for modularization and factoring out commonly used rules by utilizing references to other *JSON schema* files. The simple model schemas make use of this modular approach. The ASM Schema is defined using:

- **Core schema**: a JSON Schema that contains re-usable, domain independent rules.
  - The core schema defines value types for all possible values that may be used in tabular models.

- **Technique specific schema**: a JSON Schema that contains the domain specific rules.
  - It references the core declarations instead of each technique defining its own

- Having the basic rules factored out in a *core schema*, allows for later extensions without changing each *technique specific schema*
ASM Generation:
ADM to ASM Model Transformation Tool
ADM to ASM Transformation

ADM and ASM are produced during the governance process.
ADM ↔ ASM Semantic Compatibility

- **Semantically Rich Model (ADM)**
- **Simplified Model (ASM)**
- **ADM to ASM Transformation**
- **Bidirectional model compatibility**
- **Model Validation**
- **Model Consistency: Data and semantic context**

**Using SHACL to describe and validate ADM/RDF instance data**

**Using JSON Schema to describe and validate ASM/JSON instance data**
ADM to ASM Transformation Tool

Transformation Architecture:
- Written in Java
- The transformation tool has a common pattern model representation
- **ADM** (described by **SHACL**) and **ASM** (described by **JSON Schema**) can be read from and written into the pattern model
- It enables a decomposed, bi-directional transformation between the ADM and the ASM.
ASM
Hands-On Demonstration
The Aggregation Model is an extension of the Tabular Model. It allows aggregation of data attributes under a unified context and/or an indexed collection of similar data pattern.

The main characteristics of an Aggregation Model:

- A set of **key/value pairs** (Table)
- Keys are terms in the AFO providing a bridge to semantic usage.
- The context of the key/value pairs is represented only by the container they are defined in.
- Concepts can be aggregated as:
  - a **collection** (e.g. a peak list contains peaks).
  - **facets** (attributes) of another concept (e.g. retention time and description).

The Modeling WG is using Excel as a tool to create the Aggregation models.
Aggregation Model Reminder

- Modeling an Aggregation model using Excel
ASM demonstration

- ASM Tabular/Aggregation Data Instances Walkthrough
- ASM Schema Tabular/Aggregation Walkthrough
- ASM Validator in Action

*A list of many off-the-shelf validators, written in different languages, is available at [https://json-schema.org/implementations.html](https://json-schema.org/implementations.html)
Thanks for your attention!
Allotrope Foundation Product Team

- Ben Woolford benjamin.woolford-lim@allotrope.org
- Amnon Ptashek amnon.ptashek@allotrope.org