Providing validated, templated and richer metadata using a bidirectional conversion between JSON and iRODS AVUs.

Paul van Schayck, Ton Smeele, Daniel Theunissen and Lazlo Westerhof
Come up with a generic way within iRODS to provide metadata templates, validation and user interaction
Metadata in iRODS

iRODS object
(Resource, User, Object, Collection)

Attribute VARCHAR(255)
Value VARCHAR(2700)
Unit VARCHAR(255)

AVU: Attribute Value Unit
Why JSON?

- Long dark flowing hair
- Knows C++
- Loves beer
- All around nice guy
Why JSON?

• JSON is flexible and easy
• Features: nesting and arrays
• Human and developer readable
• Validation: JSON-schema
• Linked Data: JSON-LD
Design requirements

• Bijection between JSON <-> AVU
• Lean JSON -> AVU conversion.
• Keep Attribute->Value pairs the same
• Compatible with existing or additional AVUs
• Compatible/aware of JSON-LD
Conversion

```json
{
    "title": "Hello World!",
    "parameters": {
        "size": 42,
        "readOnly": false
    },
    "authors": ["Foo", "Bar"],
    "references": [
        {
            "title": "The Rule Engine",
            "doi": "1234.5678"
        }
    ]
}
```
Conversion: step by step

```json
{
    "title": "Hello World!",
    "parameters": {
        "size": 42,
        "readOnly": false
    },
    "authors": ["Foo", "Bar"],
    "references": [
        {
            "title": "The Rule Engine",
            "doi": "1234.5678"
        }
    ]
}
```
Conversion - Usage of the unit field

root_parent_type#index

- JSON-root [a-z]
- Parent object [0-9]
- Type [osbnze]
- array index [0-9]
Conversion - Implementation

Conversion
irods_avu_json

- Python
- `pip` package
- Standalone from iRODS

`/MaastrichtUniversity/irods_avu_json`

Ruleset
irods_avu_json-ruleset

- Python ruleset (core.py)
- Includes AVU microservices

`/MaastrichtUniversity/irods_avu_json-ruleset`

```python
setJsonToObj(*object, *objectType, *jsonRoot, *json)
getJsonFromObj(*object, *objectType, *jsonRoot)
```
Conversion - Demo
Validation - Overview

Template

Metadata

Validation

Conversion

Metadata AVUs
Validation – JSON-schema

**JSON Schema** is a vocabulary that allows you to **annotate** and **validate** JSON documents.

```json
{
  "title": "Person",
  "type": "object",
  "properties": {
    "firstName": {
      "type": "string"
    },
    "lastName": {
      "type": "string"
    },
    "age": {
      "description": "Age in years",
      "type": "integer",
      "minimum": 0
    }
  },
  "required": ["firstName", "lastName"]
}
```
**Validation – Process**

- **iRODS object**
  - (Resource, User, Object, Collection)

- **A: $id**
  - **V: URI**
  - **U: JSON-root**

- **setJsonToObj()**
  - **imeta/metalnx/etc**

- **PEPs**
  - `pep_database_*_avu_*`

- **Validation**

- **Template**
  - `i:/tempZone/path/to/schema.json`
  - `https://example.com/schema.json`

- **Metadata AVUs**

- **AVUs**
Validation - Implementation

**Ruleset**

*ruleset*

- Python ruleset (core.py)
- Includes AVU microservices
- PEPs

[MaastrichtUniversity/irods_avu_json-ruleset](MaastrichtUniversity/irods_avu_json-ruleset)

- `getJsonSchemaFromObj(*object, *objectType, *jsonRoot)`
- `pep_database_*_avu_*`(*)
Validation - Demo
Overview - Presentation

- Template
- Presentation
- Conversion
- Validation
- Form UI
- Metadata
- Conversion
- Metadata AVUs
Presentation – JSON-schema -> Form

From [https://github.com/networknt/react-schema-form](https://github.com/networknt/react-schema-form)
Our use cases

1. YoDa (Utrecht University)
   • Uses forms defined in JSON-schema (build in ReactJS), outputting JSON

2. DataHub (Maastricht University)
   • Looks at CEDAR, an Angular JSON-schema form generator, outputting JSON-LD
Presenation - Demo
Future

Form Builder → Template → Presentation → Validation → Conversion → Metadata AVUs

Search UI → Form UI → Metadata AVUs
Final thoughts

• Devils advocate:
  • Why not store entire JSON in single AVU?
• JSON-schema:
  • No real standard for UI presentation (yet)
• Implementation:
  • Devs: Microservices for setting AVUs! (iRODS-4185)
  • Devs: PEPs for AVU control are difficult to use
  • Possible race conditions during set (locking?)
Acknowledgements

Daniel Theunissen + rest of team

Lazlo Westerhof

Ton Smeele

Metadata Templates Working Group
Hackaton

Conversion
irods_avu_json

- Python
- pip package
- Standalone from iRODS

/MaastrichtUniversity/irods_avu_json

Ruleset
irods_avu_json-ruleset

- Python ruleset (core.py)
- Includes AVU microservices
- PEPs

/MaastrichtUniversity/irods_avu_json-ruleset

Docker
irods_avu_json-docker

- Single iCAT instance
- Install microservices and ruleset
- Runs tests

/MaastrichtUniversity/irods_avu_json-docker

Ideas:
- Add AVU-unit microservices to core?
- C++ microservice implementation?
- Better PEPs?
- Your applications?
A quick introduction to JSON-LD

Convert any JSON into linked data by providing the @context

Result: Human and developer readable linked data!

JSON input

```json
{
  "id": "http://hdl.handle.net/21.12109/P000000009C000000008",
  "creator": "https://orcid.org/0000-0001-6591-4637",
  "description": "Lorem Ipsum",
  "title": "Foobar"
}
```

@context (also JSON)

```json
{
  "id": "@id",
  "creator": "http://purl.org/dc/terms/creator",
  "description": "http://purl.org/dc/terms/description",
  "title": "http://purl.org/dc/terms/title",
}
```

Result: Linked data (RDF)

```
```