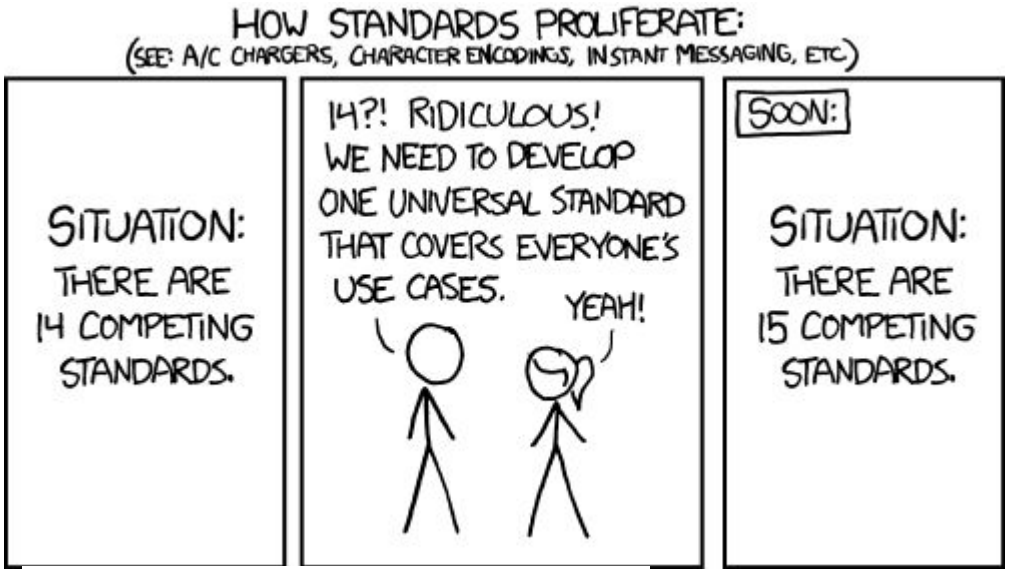
Semantic Interoperability with Layered Schemas and Linked Data

Burak Serdar
bserdar@cloudprivacylabs.com



https://imgs.xkcd.com/comics/standards.png

Semantic Harmonization

Transform to a standard model?

FHIR Patient

```
{
  "birthDate": "1946-12-03"
}
```

Standard model

```
"birthDate": "1946-12-03"
```

CCDA Patient

```
<birthTime value="19461203081200"/>
```

Semantic Harmonization

Meaning is context dependent.
Map to linked data, annotate and define equivalences.
Interpret when used.

FHIR Patient

```
{
  "birthDate": "1946-12-03"
}
```

```
http://hl7.org/fhir/Patient.birthDate: 1946-12-03
type: date
format: yyyy-mm-dd
```

equal

CCDA Patient

```
<birthTime value="19461203081200"/>
```

```
http://hl7.org/ccda/patientRole.birthTime: 19461203081200
type: date-time
format: yyyymmddHHMMSS
```

Schemas

Schemas describe structure of data

JSON schema

```
{
  "$id": "https://example.com/person.schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "Person",
  "type": "object",
  "properties": {
    "firstName": {
      "type": "string"
    },
    "lastName": {
      "type": "string"
    },
    "age": {
      "type": "integer",
      "minimum": 0
    }
  }
}
```

Schema instance

```
{
  "firstName": "John",
  "lastName": "Doe",
  "age": 20
}
```

- Validation
- Code generation
- Limited semantics

Layered Schemas

Layered schemas describe an abstract data model as linked data

```
...
"firstName": {
  "type": "string"
}
...
```

```
...
<xsd:element
   name="firstName"
   type="xsd:string"/>
...
```

```
firstName,lastName,...
John,Doe,...
```

Linked data node

```
{
  "@id": "http://example.org/Person.firstName",
  "type": "xsd:string",
  "privacyClassifications": "PII",
  "retentionPolicy": "http://link-to-policy",
  ...
}
```

Open-ended annotations

Layered Schemas

JSON

```
...
"firstName": {
  "type": "string"
}
...
```

```
{
  "firstName": "John",
   ...
}
```

Layered Schema

```
{
  "@id": "http://example.org/Person.firstName",
  "type": "xsd:string",
  "privacyClassifications": "PII",
  "retentionPolicy": "http://link-to-policy",
  ...
}
```

```
{
  "@id": "http://mycompany.com/Person.firstName",
  "@type": "Value",
  "type": "xsd:string",
  "privacyClassifications": "PII",
  "retentionPolicy": "http://link-to-policy",
  "name": "firstName",
  "value": "John",
  ...
}
```

Schema Slicing/Composition

Schema

```
"attributes": [
 {
   "@id": "http://example.org/firstName",
   "@type": "Value",
   "name": "firstName",
   "type": "string",
   "privacyClassifications": [ "PII"]
},
...
```

Slice

Compose

Layers

```
"attributes": [
 {
   "@id": "http://example.org/firstName"
   "@type": "Value"
}
```

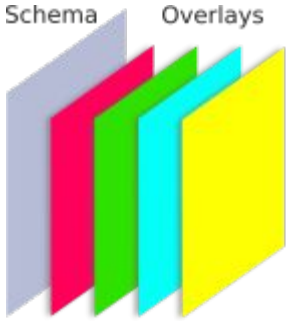
```
"attributes": [
 {
   "@id": "http://example.org/firstName",
   "name": "firstName"
  }
```

```
"attributes": [
 {
   "@id": "http://example.org/firstName",
   "type": "string"
  }
```

```
"attributes": [
 {
   "@id": "http://example.org/firstName",
   "privacyClassifications": [ "PII"]
 }
```

Schema + Overlay -> Schema
Overlay + Overlay -> Overlay

Layered Schemas



Schema base defines structure

Interchangeable overlays add

- Constraints (min length, max value, required, etc.)
- Format (phone number, date/time, etc.)
- Language (English, Spanish, etc.)
- Privacy classifications (PII, Sensitive, etc.)
- Security flags (encryption, context, etc.)
- Retention policies,
- ...

Schema = Schema base + overlays

Schema

```
"@type": "Schema",
"objectType": "http://example.org/Person",
"attributes": {
  "http://example.org/name": {
   "@type": "Value",
   "name": "name"
  },
  "http://example.org/work": {
   "@type": "Object"
   "attributes": {
      "http://example.org/jobTitle": {
         "@type": "Value"
      },
      "http://example.org/department": {
         "@type": "Value"
      }
  },
  "http://example.org/accountId": {
   "@type": "Reference"
   "reference": "http://example.org/Account"   },
  "http://example.org/contact": {
   "name": "contact",
   "@type": "Array",
   "arrayItems": {
      "@type": "Reference"
      "reference": "http://example.org/Contact"
   }
  },
```

Object defined by this schema layer

Attribute ID

Node type, not data type

Attribute Name (can be in an overlay)

Nested object

Reference to another object

Array attribute

Overlay

```
"@type": "Overlay",
"objectType": "http://example.org/Person",
"attributes": {
  "http://example.org/name": {
   "@type": "Value",
   "type": "string"
   "privacyClassification": "http://example.org#BIT"
  },
 "http://example.org/jobTitle": {
    "@type": "Value",
   "type": "string"
  },
 "http://example.org/department": {
    "@type": "Value",
   "type": "string"
 },
 "http://example.org/newAttribute": {
    "@type": "Value",
    "type": "string"
 },
 "http://example.org/accountId": {
    "@type": "Reference",
    "required": true
 }
 ...
}
```

The object annotated by this layer

Refer to attributes using @id

Metadata for the attribute

Overlay can define new attributes

Overlay can add/remove constraints

Composition

```
"@type": "Schema",
"objectType": "http://example.org/Person",
"attributes": {
  "http://example.org/USAddress": {
      "@type": "Composite",
      "allOf": [
         {
           "reference": "http://example.org/basicAddress"
         },
         {
           "attributes": {
             "state": {}
           }
         },
          ...
      ]
    }
  },
 ...
}
```

Polymorphism

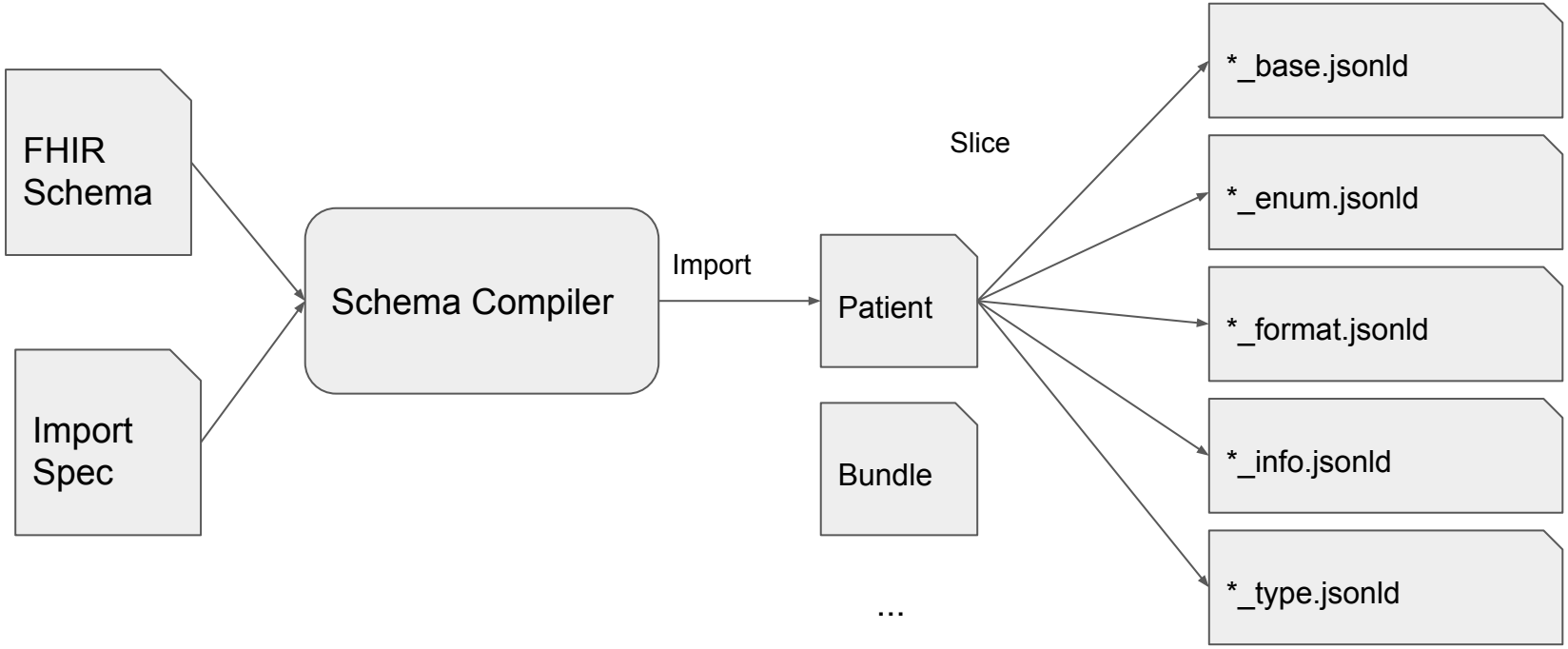
```
"@type": "Schema",
"objectType": "http://hl7.org/fhir/Bundle",
"attributes": {
  "http://hl7.org/fhir/Bundle.entries": {
    "arrayItems": {
      "@type": "Polymorphic",
      "oneOf": [
         {
           "reference": "http://hl7.org/fhir/Patient"
         },
         {
           "reference": "http://hl7.org/fhir/Encounter"
         },
          ...
      ]
   }
  },
 ...
}
```

Schema Manifest

The schema manifest links a schema and a set of overlays to create a new schema that is localized, adopted to a particular context/jurisdiction, and versioned. An object may have many variations for different contexts.

```
{
  "@context": "http://schemas.cloudprivacylabs.com/schema.jsonld",
  "@type": "SchemaManifest",
  "@id": "schema Id",
  "objectType": "http://example.org/Person",
  "base": "http://example.org/Person/base",
  "overlays": [
     "http://example.org/Person/ovl/info",
     "http://example.org/Person/ovl/BIT" ,
     ...
   ]
}
```

Example: FHIR



Example: FHIR Bundle

```
{
    "resourceType": "Bundle",
    "entry": [
        {
            "resource":{
                "resourceType": "Immunization",
                ...
            }
        },
        {
            "resource":{
                "resourceType": "Patient",
               ....
            }
        },
       ...
    ]
}
```

Polymorphic array

Each “resource” has a different schema

Example: FHIR Bundle

```
{
    "resourceType": "Bundle",
    "entry": [
        {
            "resource":{
                "resourceType": "Immunization",
                ...
            }
        },
        {
            "resource":{
                "resourceType": "Patient",
               ....
            }
        }
    ]
}
```

```
{
    "http://layeredschemas.org/v1.0/attr/name":"resourceType",
    "http://layeredschemas.org/v1.0/doc/attributeId":
         "http://hl7.org/fhir/Immunization.resourceType",
    …
}
```

```
{
    "http://layeredschemas.org/v1.0/attr/name":"resourceType",
    "http://layeredschemas.org/v1.0/doc/attributeId":
         "http://hl7.org/fhir/Patient.resourceType",
    …
}
```

Use Case: Data Warehouse

JSON

```
height: 5’8”
```

XML

```
<Height unit=”inch”>
  68
</Height>
```

CSV

```
HEIGHT
173
```

Use Case: Data Warehouse

Schema base

```
{
  "@id": "http://example.org/height",
  "@type": "Value"
}
```

Overlays

```
{
  "@id": "http://example.org/height",
  "format": "feet_inch",
  "name": "height"
}
```

```
{
  "@id": "http://example.org/height",
  "format": "with_unit",
  "unit": "http://example.org/unit",
  "name": "Height"
},{
  "@id": "http://example.org/unit",
  "name": "Height#unit"
},
```

```
{
  "@id": "http://example.org/height",
  "format": "centimeters",
  "name": "HEIGHT"
}
```

JSON

```
height: 5’8”
```

XML

```
<Height unit=”inch”>
  68
</Height>
```

CSV

```
HEIGHT
173
```

Use Case: Data Warehouse

```
height: 5’8”
```

```
{
  "@id": "http://example.org/height",
  "format": "feet_inch",
  "name": "height",
  "value": "5’8”"
}
```

```
<Height unit=”inch”>
  68
</Height>
```

```
{
  "@id": "http://example.org/height",
  "format": "with_unit",
  "name": "Height",
  "unit": "http://example.org/unit",
  "value": "68"
},{
  "@id": "http://example.org/unit",
  "name": "Height#unit",
  "value": "inch"
}
```

```
HEIGHT
173
```

```
{
  "@id": "http://example.org/height",
  "format": "centimeters",
  "name": "HEIGHT",
  "value": "173"
}
```

Use Case: Data Warehouse

```
{
  "@id": "http://example.org/height",
  "format": "feet_inch",
  "name": "height",
  "value": "5’8”"
}
```

```
{
  "@id": "http://example.org/height",
  "format": "with_unit",
  "name": "Height",
  "unit": "http://example.org/unit",
  "value": "68"
},{
  "@id": "http://example.org/unit",
  "name": "Height#unit",
  "value": "inch"
}
```

```
{
  "@id": "http://example.org/height",
  "format": "centimeters",
  "name": "HEIGHT",
  "value": "173"
}
```

Transformation Library

`feet_inch(5’8)=173`

`with_unit(68,”inch”)=173`

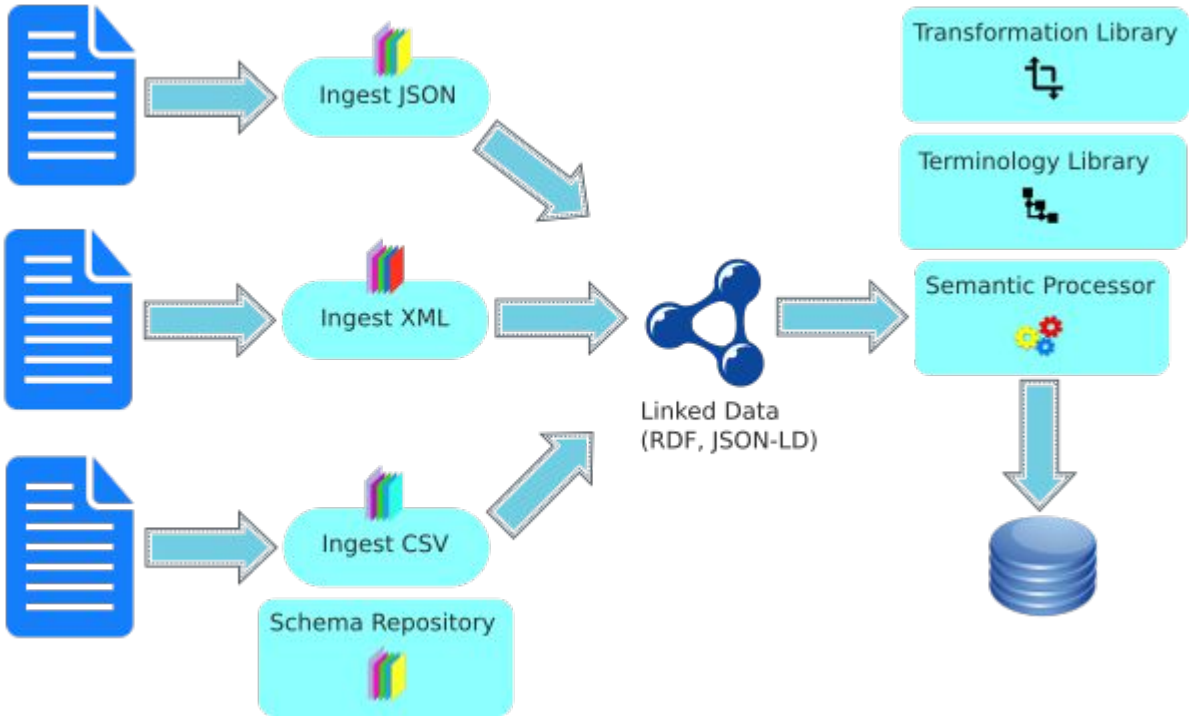
`centimeters(173)=173`

```
{
  "@id": "http://example.org/height",
  "format": "feet_inch",
  "name": "height",
  "value": "5’8”",
  "cm": "173"
}
```

```
{
  "@id": "http://example.org/height",
  "format": "with_unit",
  "name": "Height",
  "unit": "http://example.org/unit",
  "value": "68",
  "cm": "173"
}
```

```
{
  "@id": "http://example.org/height",
  "format": "centimeters",
  "name": "HEIGHT",
  "value": "173",
  "cm": "173"
}
```

Architecture: Data Warehouse



Use Case: Data Use Agreements as Overlays

```
{
  "@id": "http://example.org/height",
  "DUA": link to data use agreement,
  "useAllowed": true
}
```

```
name: John Doe
height: 5’8”
postalCode: 12345
```

Select nodes where useAllowed=true

```
height: 5’8”
```

```
{
  "@id": "http://example.org/name",
  "@type": "Value"
},
{
  "@id": "http://example.org/height",
  "@type": "Value"
},
{
  "@id": "http://example.org/postalCode",
  "@type": "Value"
}
```

Use Case: Granular Consent as Overlays

```
{
  "@id": "http://example.org/name",
  "consent": link to consent,
  "useAllowed": true
}
```

```
{
  "@id": "http://example.org/postalCode"
  "consent": link to consent,
  "useAllowed": true
},
{
  "@id": "http://example.org/height",
  "consent": link to consent,
  "useAllowed": true
}
```

```
name: John Doe
height: 5’8”
postalCode: 12345
```

Select nodes where useAllowed=”true”

```
height: 5’8”
postalCode: 12345
```

```
{
  "@id": "http://example.org/name",
  "@type": "Value"
},
{
  "@id": "http://example.org/height",
  "@type": "Value"
},
{
  "@id": "http://example.org/postalCode",
  "@type": "Value"
}
```

Use Case: Deidentification

```
{
  "@id": "http://example.org/name",
  "@type": "Value",
  "privacyClassifications": "PII"
}
```

```
name: John Doe
height: 5’8”
postalCode: 12345
```

Select nodes where
privacyClassifications!=”PII”

```
height: 5’8”
postalCode: 12345
```

```
{
  "@id": "http://example.org/name",
  "@type": "Value"
},
{
  "@id": "http://example.org/height",
  "@type": "Value"
},
{
  "@id": "http://example.org/postalCode",
  "@type": "Value"
}
```

Use Case: Transformation

```
{
  "@id": "creds:credentialSubject",
  "@type": "Value"
}
```

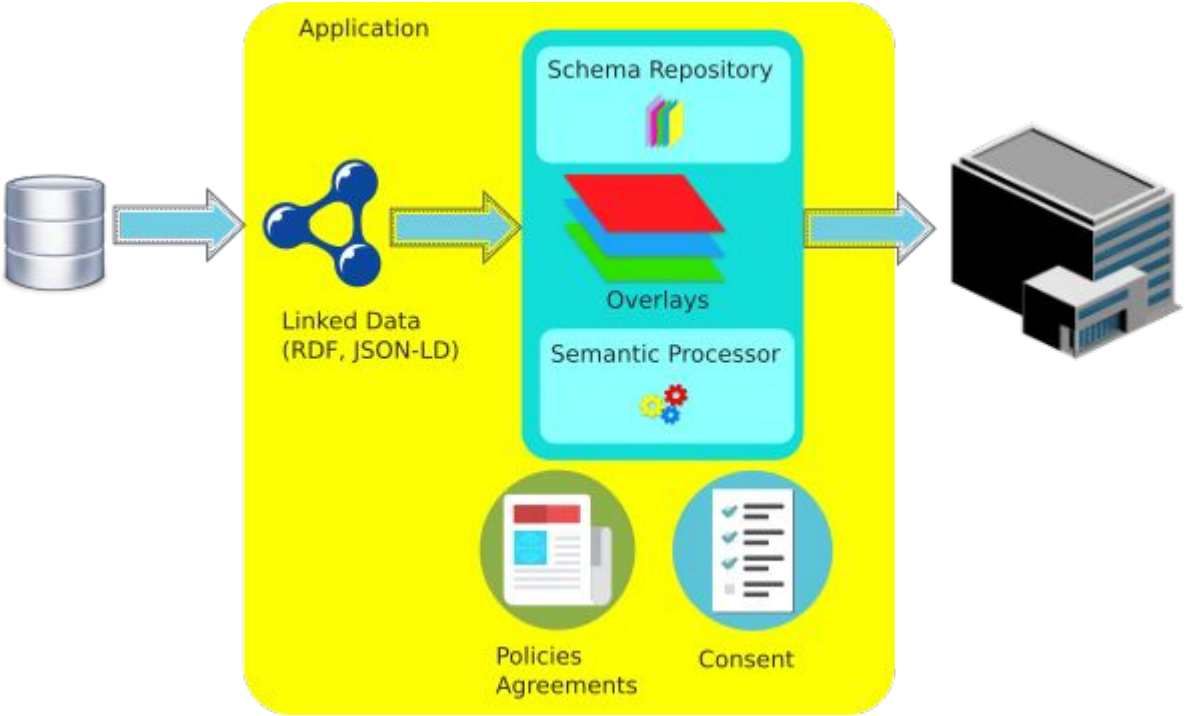
```
patient: {
  id: “abc”,
  …
}
```

```
Select nodes (@id=http://hl7.org/fhir/Patient.id)
project (http://layeredscheams.org/doc/value,...)
link creds:credentialSubject
```

```
verifiableCredential {
  credentialSubject: “abc”
  …
}
```

```
{
  "@id": "http://hl7.org/fhir/Patient.id",
  "@type": "Value"
},
```

Architecture: Data Exchange



Use Case: Data Capture

```
{
  "@id": "http://example.org/postalCode",
  "@type": "Value",
  "type": "xsd:string"
}
```

```
{
  "@id": "http://example.org/postalCode",
  "@type": "Value",
  "label": "ZIP code",
  "pattern": "(^\d{5}$)|(^\d{9}$)|(^\d{5}-\d{4}$)"
}
```

ZIP code: 12345-3456

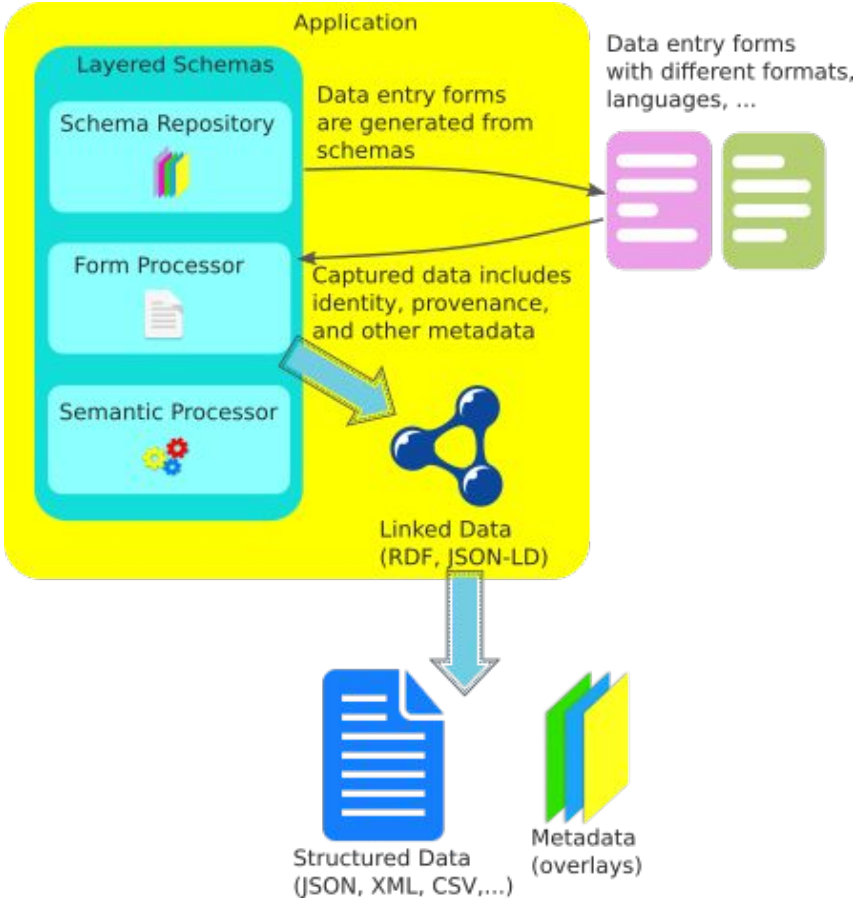
```
"http://example.org/postalCode": "12345-3456"
```

```
{
  "@id": "http://example.org/postalCode",
  "@type": "Value",
  "label": "Postal code",
  "pattern": "^[A-Za-z]\d[A-Za-z][ -]?\d[A-Za-z]\d$"
}
```

Postal code: A1A A1A

```
"http://example.org/postalCode": "A1A-A1A"
```

Architecture: Data Capture



Schema defines attributes to capture

Overlays for

- Different languages
- Different locale
- Entry variations

Schema annotations are used to build forms:

- CSS class
- Input type/format
- User prompts

Components

- Schema compiler (open source): Import schema, slice, compose, ingest data
- Schema repository:
  - GetSchema(http://hl7.org/fhir/Patient)
  - GetSchema(sha256:123467890abcdef...)
- Ingest (json/csv open source): Input data + Schema -> Annotated linked data
- Form processor: Generate data entry forms using schema annotations
- Semantic processor:
  - Select nodes where…
  - Select nodes and project...
  - Select nodes and link...