FHIR Airbyte

Author: Steve + Alex from Pentavere Date: 2024-06-06

Documents possible approach to FHIR Airbyte connector, from point of view of Observation.

Our goal is to produce an Airbyte schema for Observation, following the structure defined by https://docs.airbyte.com/understanding-airbyte/supported-data-types/#record-structure

A simple example of an instance of R4 Observation message:


```
{
  "resourceType": "Observation",
  "id": "656",
  "text": {
    "status": "generated",
    "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\"><p><b>Generated Narrative:
Observation</b><a name=\"656\"> </a></p><div style=\"display: inline-block;
background-color: #d9e0e7; padding: 6px; margin: 4px; border: 1px solid #8da1b4;
border-radius: 5px; line-height: 60%\"><p style=\"margin-bottom: 0px\">Resource
Observation &quot;656&quot; </p></div><p><b>identifier</b>: id: patientId-
urn:oid:1.2.4~0000000000000000~152584~20170503155426~820</p><p><b>status</b>:
final</p><p><b>code</b>: MDC_FLOW_AWAY_EXP_FORCED_PEAK <span style=\"background:
LightGoldenRodYellow; margin: 4px; border: 1px solid khaki\"> (<a
href=\"http://terminology.hl7.org/5.1.0/CodeSystem-v3-mdc.html\">ISO 11073-10101
Health informatics - Point-of-care</a>#152584)</span></p><p><b>subject</b>: <a
href=\"broken-link.html\">Patient/PatientId-patientId</a></p><p><b>effective</b>:
2017-05-03T15:54:26-04:00</p><p><b>performer</b>: <a href=\"broken-
link.html\">Patient/PatientId-patientId</a></p><p><b>value</b>: 820 cL/s<span
style=\"background: LightGoldenRodYellow\"> (Details: urn:iso:std:iso:11073:10101 code
265201 = '265201')</span></p></div>"
  },
  "identifier": [
    {
      "value": "patientId-urn:oid:1.2.4~0000000000000000~152584~20170503155426~820"
    }
  ],
  "status": "final",
  "code": {
    "coding": [
      {
        "system": "urn:iso:std:iso:11073:10101",
        "code": "152584",
        "display": "MDC_FLOW_AWAY_EXP_FORCED_PEAK"
      }
    ]
  },
  "subject": {
    "reference": "Patient/PatientId-patientId"
  },
  "effectiveDateTime": "2017-05-03T15:54:26-04:00",
```

```
  "performer": [
    {
      "reference": "Patient/PatientId-patientId"
    }
  ],
  "valueQuantity": {
    "value": 820,
    "unit": "cL/s",
    "system": "urn:iso:std:iso:11073:10101",
    "code": "265201"
  }
}
```

Step 1:

Discover streams from FHIR server capability statement: rest resource type where interaction codes includes "read"

```
{
  "resourceType": "CapabilityStatement",
  "status": "active",
  "date": "2024-06-06T15:45:24.202214+00:00",
  "kind": "instance",
  "fhirVersion": "5.0.0",
  "format": [
    "json"
  ],
  "rest": [
    {
      "mode": "server",
      "resource": [
        {
          "type": "Observation",
          "interaction": [
            {
              "code": "read"
            }
          ]
        },
        {
          "type": "Patient",
          "interaction": [
            {
              "code": "read"
            }
          ]
        }
      ]
    }
  ]
}
```

Since we have read on Observation and Patient, this will expose two streams: Observation and Patient

Step 2:

Use https://pypi.org/project/fhir.resources/ package to help define schema of the stream.

NOTE: This package supports *R5, R4B, R4 and STU3* versions of FHIR. We should aim to support all versions using whichever is the published capability of the server at CapabilityStatement.fhirVersion

```
from fhir.resources.observation import Observation


# Get the top level properties
list(Observation.schema()["properties"].keys())

['resource_type', 'fhir_comments', 'id', 'implicitRules', '_implicitRules',
'language', '_language', 'meta', 'contained', 'extension', 'modifierExtension',
'text', 'basedOn', 'bodySite', 'bodyStructure', 'category', 'code', 'component',
'dataAbsentReason', 'derivedFrom', 'device', 'effectiveDateTime',
'_effectiveDateTime', 'effectiveInstant', '_effectiveInstant', 'effectivePeriod',
'effectiveTiming', 'encounter', 'focus', 'hasMember', 'identifier',
'instantiatesCanonical', '_instantiatesCanonical', 'instantiatesReference',
'interpretation', 'issued', '_issued', 'method', 'note', 'partOf', 'performer',
'referenceRange', 'specimen', 'status', '_status', 'subject', 'triggeredBy',
'valueAttachment', 'valueBoolean', '_valueBoolean', 'valueCodeableConcept',
'valueDateTime', '_valueDateTime', 'valueInteger', '_valueInteger', 'valuePeriod',
'valueQuantity', 'valueRange', 'valueRatio', 'valueReference', 'valueSampledData',
'valueString', '_valueString', 'valueTime', '_valueTime']


# Get the data type of each field
{k: v.get("type") for k, v in sorted(Observation.schema()["properties"].items())}

{'_effectiveDateTime': 'FHIRPrimitiveExtension', '_effectiveInstant':
'FHIRPrimitiveExtension', '_implicitRules': 'FHIRPrimitiveExtension',
'_instantiatesCanonical': 'FHIRPrimitiveExtension', '_issued':
'FHIRPrimitiveExtension', '_language': 'FHIRPrimitiveExtension', '_status':
'FHIRPrimitiveExtension', '_valueBoolean': 'FHIRPrimitiveExtension', '_valueDateTime':
'FHIRPrimitiveExtension', '_valueInteger': 'FHIRPrimitiveExtension', '_valueString':
'FHIRPrimitiveExtension', '_valueTime': 'FHIRPrimitiveExtension', 'basedOn': 'array',
'bodySite': 'CodeableConcept', 'bodyStructure': 'Reference', 'category': 'array',
'code': 'CodeableConcept', 'component': 'array', 'contained': 'array',
'dataAbsentReason': 'CodeableConcept', 'derivedFrom': 'array', 'device': 'Reference',
'effectiveDateTime': 'string', 'effectiveInstant': 'string', 'effectivePeriod':
'Period', 'effectiveTiming': 'Timing', 'encounter': 'Reference', 'extension': 'array',
'fhir_comments': None, 'focus': 'array', 'hasMember': 'array', 'id': 'string',
'identifier': 'array', 'implicitRules': 'string', 'instantiatesCanonical': 'string',
'instantiatesReference': 'Reference', 'interpretation': 'array', 'issued': 'string',
'language': 'string', 'meta': 'Meta', 'method': 'CodeableConcept',
'modifierExtension': 'array', 'note': 'array', 'partOf': 'array', 'performer':
```

```
'array', 'referenceRange': 'array', 'resource_type': 'string', 'specimen':
'Reference', 'status': 'string', 'subject': 'Reference', 'text': 'Narrative',
'triggeredBy': 'array', 'valueAttachment': 'Attachment', 'valueBoolean': 'boolean',
'valueCodeableConcept': 'CodeableConcept', 'valueDateTime': 'string', 'valueInteger':
'integer', 'valuePeriod': 'Period', 'valueQuantity': 'Quantity', 'valueRange':
'Range', 'valueRatio': 'Ratio', 'valueReference': 'Reference', 'valueSampledData':
'SampledData', 'valueString': 'string', 'valueTime': 'string'}


# For a non-primitive data type, (recursively) determine the primitive data types

from fhir.resources.quantity import Quantity

list(Quantity.schema()["properties"].keys())

['resource_type', 'fhir_comments', 'extension', 'id', 'code', '_code', 'comparator',
'_comparator', 'system', '_system', 'unit', '_unit', 'value', '_value']


{k: v.get("type") for k, v in Quantity.schema()["properties"].items()}

{'resource_type': 'string', 'fhir_comments': None, 'extension': 'array', 'id':
'string', 'code': 'string', '_code': 'FHIRPrimitiveExtension', 'comparator': 'string',
'_comparator': 'FHIRPrimitiveExtension', 'system': 'string', '_system':
'FHIRPrimitiveExtension', 'unit': 'string', '_unit': 'FHIRPrimitiveExtension',
'value': 'number', '_value': 'FHIRPrimitiveExtension'}


# For array types, you need to look at the "items" type

Observation.schema()["properties"]["referenceRange"]

{'title': 'Provides guide for interpretation', 'description': 'Guidance on how to
interpret the value by comparison to a normal or recommended range...o
`referenceRange` elements would be used.', 'element_property': True, 'type': 'array',
'items': {'type': 'ObservationReferenceRange'}}


# Get the type for the array
from fhir.resources.observation import ObservationReferenceRange

{k: v.get("type") for k, v in ObservationReferenceRange.schema()
["properties"].items()}

{'resource_type': 'string', 'fhir_comments': None, 'extension': 'array', 'id':
'string', 'modifierExtension': 'array', 'age': 'Range', 'appliesTo': 'array', 'high':
'Quantity', 'low': 'Quantity', 'normalValue': 'CodeableConcept', 'text': 'string',
'_text': 'FHIRPrimitiveExtension', 'type': 'CodeableConcept'}

# Keep going...
from fhir.resources.codeableconcept import CodeableConcept
```

```
{'resource_type': 'string', 'fhir_comments': None, 'extension': 'array', 'id':
'string', 'coding': 'array', 'text': 'string', '_text': 'FHIRPrimitiveExtension'}
```

Step 3

Once everything is resolved to primitive types, we can output the schema for Airbyte.

The types accepted by Airbyte are available here, so we would map the FHIR primitives to Airbyte according to https://docs.airbyte.com/understanding-airbyte/supported-data-types/#the-types

Just a small subset of top level entries and subfields relating to the above is shown here.

id, text, identifier, valueQuantity, referenceRange....

```
{
  "name": "Observation",
  "json_schema": {
    "type": "object",
    "properties": {
      "resourceType": {
        "type": "string"
      },
      "id": {
        "type": "string"
      },
      "text": {
        "type": "object",
        "properties": {
          "status": {
            "type": "string"
          },
          "div": {
            "type": "string"
          }
        }
      },
      "valueQuantity": {
        "type": "object",
        "properties": {
          "value": {
            "type": "number"
          },
          "unit": {
            "type": "string"
          },
          "system": {
            "type": "string"
          },
          "code": {
            "type": "string"
```

```
        }
      }
    },
    "referenceRange": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "type": {
            "text": {
              "type": "string"
            },
            "high": {
              "type": "object",
              "properties": {
                "value": {
                  "type": "number"
                },
                "unit": {
                  "type": "string"
                },
                "system": {
                  "type": "string"
                },
                "code": {
                  "type": "string"
                }
              }
            },
            "low": {
              "type": "object",
              "properties": {
                "value": {
                  "type": "number"
                },
                "unit": {
                  "type": "string"
                },
                "system": {
                  "type": "string"
                },
                "code": {
                  "type": "string"
                }
              }
            },
            "normalValue": {
              "type": "object",
              "properties": {
                "id": "string",
                "text": "string"
              }
```

```
              }
            }
          }
        }
      }
    }
  }
}
```

Note:

1. Schema is known a priori depending on the version of FHIR, as encapsulated in the pydantic objects. We do not need to rediscover it dynamically each time. We *do* have to check the capability statement dynamically to get the readable streams and fhir version.
2. We must of course include all possible fields in the schema.
3. A good implementation strategy is to map each of the FHIR objects in https://pypi.org/project/fhir.resources/ from the bottom of the tree. Each block of json definition is reusable in multiple places as you move up the tree. For example, once the idea of CodeableConcept has a json representation, everywhere that it is used as a type, just drop in the json definition. In the above - valueQuantity, and ref range high/low are exactly the same json rep.
4. If we stick to this approach, it will be possible for downstream uses to use the same class to reconstruct the object rep. of the resource which gives a convenient way to work with it behind the scenes.
5. You can quickly generate a list of available fhir types by running `fhir.resources.fhirtypes.__dict__`