Google Cloud

Cloud Healthcare Data Harmonization

HL7v2 to FHIR Mapping

Sajjad Hussain
Yatish Gupta

Presented at FHIR Connectathon 25

Agenda

- Healthcare Data Harmonization
- Whistle Data Transformation Language
- HL7v2 To FHIR: Our Mapping Approach
  - Mapping Gap Analysis
  - Mapping Configuration
  - Datatype Transformation
  - Vocabulary Transformation
  - Data Element (Segment) Transformation
- Pipelines
  - Cloud Dataflow + Demo
  - Cloud DataFusion
- Questions

Healthcare Data Harmonization

Achieving healthcare data harmonization and interoperability is deeply dependent on

- use of standard healthcare information models
- controlled terminologies

Healthcare data harmonization enables:

- sharing and re-use of clinical data for predictive data analytics
- clinical decision making

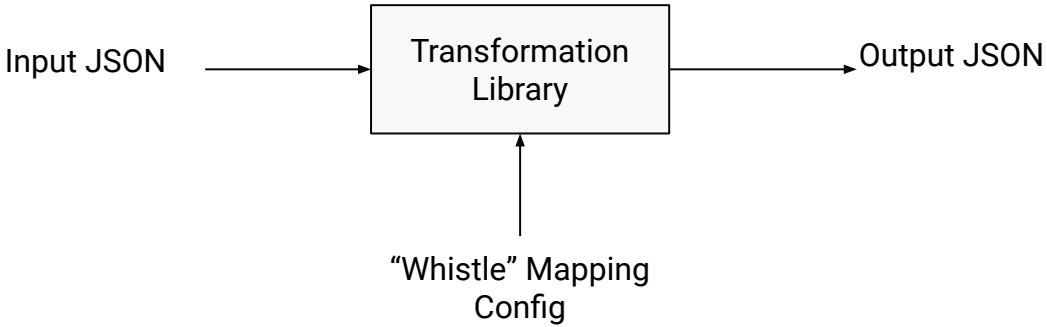
Healthcare information models and terminologies

- HL7v2
- FHIR
- OMOP
- DICOM
- VCF
- CDISC
- ...

- SNOMED-CT
- LOINC
- RxNOM
- UCUM
- MEDDRA
- ICD-10
- ...

Whistle Data Transformation Language

Define Mapping Configuration using Whistle Data Transformation Language

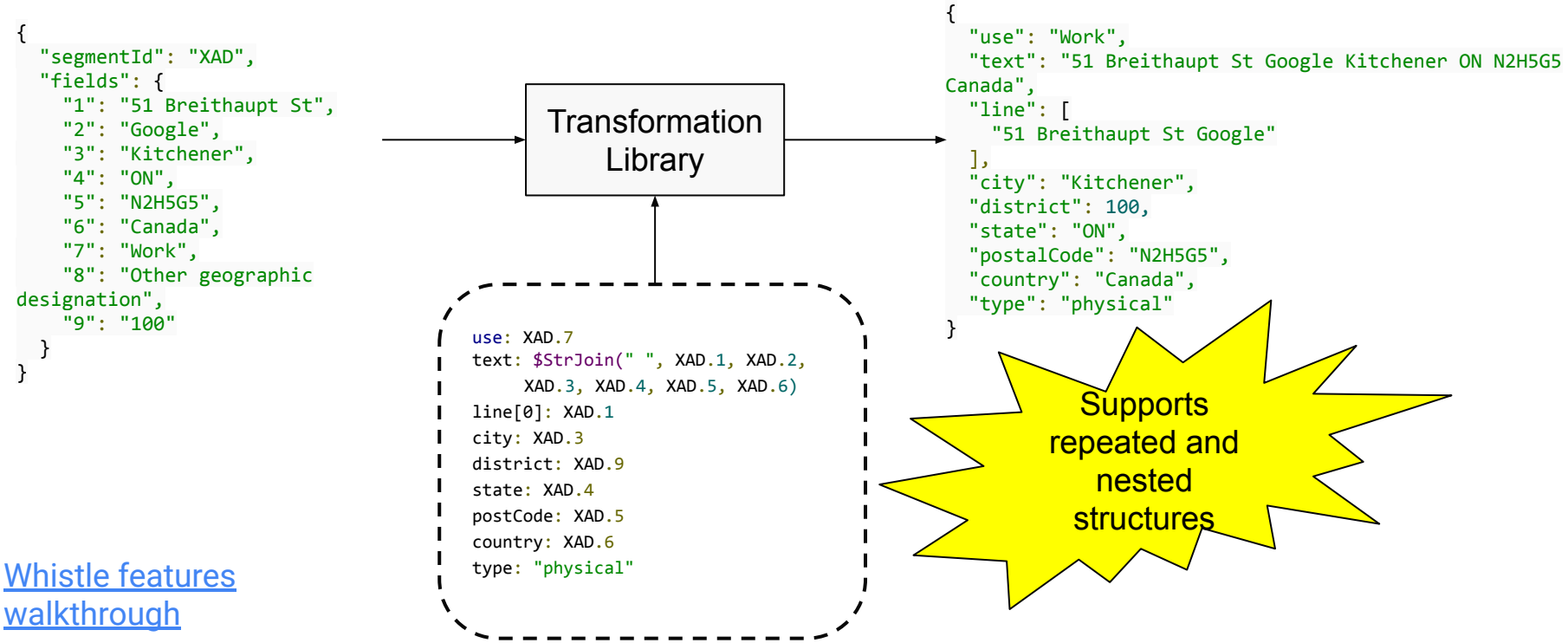


“Whistle” is the name of core mapping config language; the associated transformation library is implemented in Go.

Behind the scene whistle configs are translated into “Whistler” protos for execution.

GitHub

Define Mapping Configuration using Whistle



[Whistle features walkthrough]

HL7v2 to FHIR: Mapping Approach

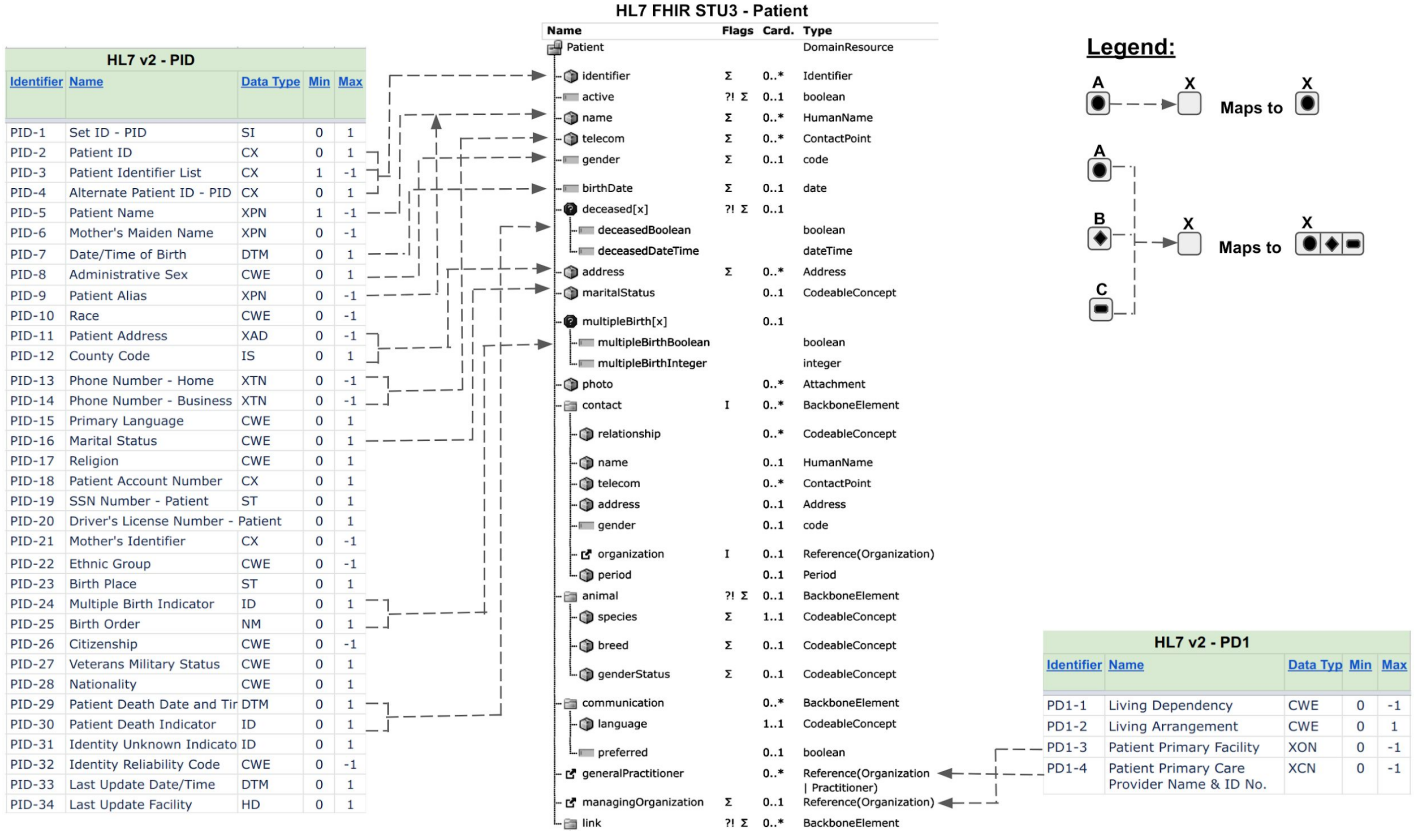
Healthcare Data Harmonization: Mapping Process

- A mapping process aims towards
  - identifying alignments between source and target schemas
  - achieving data transformation
  - mediation between two or more data sources

- Step 1: Mapping Gap Analysis
  - Finding candidate mappings between source and target concepts
  - Finding candidate mappings between source and target attributes
  - Ensuring mandatory target concepts and attributes are mapped
  - Documenting unmapped concepts and attributes for further considerations

- Step 2: Define Mapping Configuration
  - Datatype Transformation
  - Vocabulary Transformation
  - Data Element Transformation

v2 To FHIR: Mapping Gap Analysis



Datatype Transformation

HL7 v2 - PID

Identifier	Name	Data Type	Min	Max
PID-1	Set ID - PID	SI	0	1
PID-2	Patient ID	CX	0	1
PID-3	Patient Identifier List	CX	1	-1
PID-4	Alternate Patient ID - PID	CX	0	1
PID-19	SSN Number - Patient	ST	0	1
PID-20	Driver's License Number -	DLN	0	1

HL7 FHIR STU3 - Patient

Name	Flags	Card.	Type
Patient			DomainResource
identifier	Σ	0..*	Identifier
active	?! Σ	0..1	boolean
name	Σ	0..*	HumanName
telecom	Σ	0..*	ContactPoint
gender	Σ	0..1	code

HL7 v2 - CX Datatype

Identifier	Name	Data Type	Min	Max
CX.1	ID Number	ST	1	1
CX.2	Identifier Check Digit	ST	0	1
CX.3	Check Digit Scheme	ID	0	1
CX.4	Assigning Authority	HD	0	1
CX.5	Identifier Type Code	ID	1	1
CX.6	Assigning Facility	HD	0	1
CX.7	Effective Date	DT	0	1
CX.8	Expiration Date	DT	0	1
CX.9	Assigning Jurisdiction	CWE	0	1
CX.10	Assigning Agency or Department	CWE	0	1
CX.11	Security Check	ST	0	1
CX.12	Security Check Scheme	ID	0	1

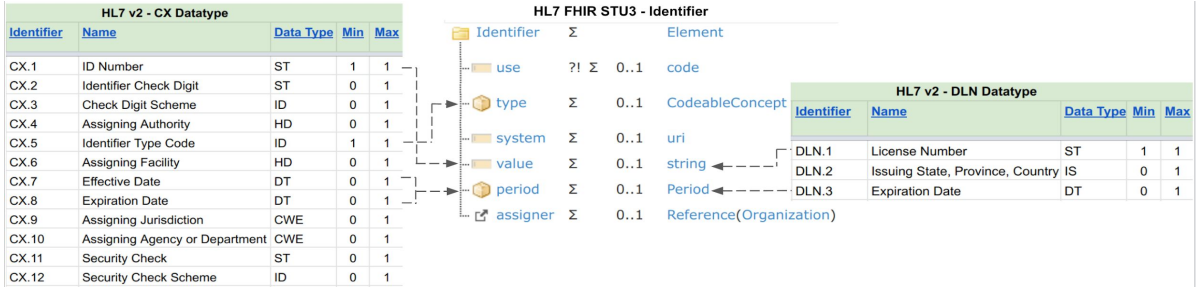
HL7 FHIR STU3 - Identifier

Identifier	Σ		Element
use	?! Σ	0..1	code
type	Σ	0..1	CodeableConcept
system	Σ	0..1	uri
value	Σ	0..1	string
period	Σ	0..1	Period
assigner	Σ	0..1	Reference(Organization)

HL7 v2 - DLN Datatype

Identifier	Name	Data Type	Min	Max
DLN.1	License Number	ST	1	1
DLN.2	Issuing State, Province, Country	IS	0	1
DLN.3	Expiration Date	DT	0	1

Datatype Transformation



HL7 v2 - CX Datatype

Identifier	Name	Data Type	Min	Max
CX.1	ID Number	ST	1	1
CX.2	Identifier Check Digit	ST	0	1
CX.3	Check Digit Scheme	ID	0	1
CX.4	Assigning Authority	HD	0	1
CX.5	Identifier Type Code	ID	1	1
CX.6	Assigning Facility	HD	0	1
CX.7	Effective Date	DT	0	1
CX.8	Expiration Date	DT	0	1
CX.9	Assigning Jurisdiction	CWE	0	1
CX.10	Assigning Agency or Department	CWE	0	1
CX.11	Security Check	ST	0	1
CX.12	Security Check Scheme	ID	0	1

HL7 v2 - DLN Datatype

Identifier	Name	Data Type	Min	Max
DLN.1	License Number	ST	1	1
DLN.2	Issuing State, Province, Country	IS	0	1
DLN.3	Expiration Date	DT	0	1

```
// Description: Constructs Identifier datatype
//
// Argument(s):
//   CX datatype
//
// Output(s):
//   Identifier datatype :
https://www.hl7.org/fhir/stu3/datatypes.html#Identifier
//
def CX_Identifier(CX) {
  value : CX.1;
  type.coding[].code : CX.5;
  period.start : Convert_DateTime(CX.7);
  period.end : Convert_DateTime(CX.8);
  }
```

```
// Description: Constructs Identifier datatype
//
// Argument(s):
//   DLN datatype
//   varCode: a code string to be mapped with
Identifier.type code
// Output(s):
//   Identifier datatype :
https://www.hl7.org/fhir/stu3/datatypes.html#Identifier
//
def DLN_Identifier(DLN, varCode) {
  value : DLN.1;
  type.coding[].code: varCode;
  system : DLN.2.1;
  period.end : DLN.3;
  }
```

Datatype Transformation

```
def CWE_CodeableConcept(CWE) {
  coding[0].code : CWE.1;
  coding[0].display : CWE.2;
  coding[0].system : CWE.3;
  coding[1].code : CWE.4;
  coding[1].display : CWE.5;
  coding[1].system : CWE.6;
  coding[0].version : CWE.7;
  coding[1].version : CWE.8;
  text : CWE.9;
  coding[2].code : CWE.10;
  coding[2].display : CWE.11;
  coding[2].system : CWE.12;
  coding[2].version : CWE.13;
}
```

```
def XPN_HumanName(XPN) {
  family : FN_Family(XPN.1);
  given[] : XPN.2;
  given[] : XPN.3;
  suffix[] : XPN.4;
  prefix[] : XPN.5;
  suffix[] : XPN.6;
  use : Convert_TerminologyCode(XPN.7, "Name_Type");
  if (~XPN.12? and ~XPN.13?) {
    period : DR_Period(XPN.10);
  }
  period.start : Convert_DateTime(XPN.12);
  period.end : Convert_DateTime(XPN.13);
  suffix[] : XPN.14;
}
```

Vocabulary Transformation

HL7 v2			HL7 FHIR		
Code	**Text**	**Code System**	**Code**	**Display**	**Code System**
F	Female	HL70001	female	Female	http://hl7.org/fhir/administrative-gender
M	Male	HL70001	male	Male	http://hl7.org/fhir/administrative-gender
O	Other	HL70001	other	Other	http://hl7.org/fhir/administrative-gender
U	Unknown	HL70001	unknow	Unknown	http://hl7.org/fhir/administrative-gender
A	Ambiguous	HL70001	other	Other	http://hl7.org/fhir/administrative-gender
N	Not applicable	HL70001	other	Other	http://hl7.org/fhir/administrative-gender

```
// Description: Converts HL7 v2 terminology into
FHIR terminology

def Convert_TerminologyCode(Code, ConceptMapID){
  var mapping : $HarmonizeCode("$Local", Code, "",
ConceptMapID);
  $this : mapping[0].code;
}
```


```
{
  "group": [
    {
      "element": [
        {
          "code": "F",
          "display": "Female",
          "target": [
            {
              "code": "female",
              "display": "Female",
              "equivalence": "equivalent"
            }
          ]
        },
 ....  ....  ....
      ],
      "source": "HL70001",
      "target": "http://hl7.org/fhir/administrative-gender",
      "unmapped": {
        "mode": "fixed",
        "code": "unknown",
        "display": "Unknown"
      }
    }
  ],
  "id": "Gender",
  "resourceType": "ConceptMap",
  "version": "v1"
}
```

Data Element (Segment) Transformation

```
def PID_PD1_Patient(PID, PD1) {
  identifier[] : CX_Identifier(PID.2);
  identifier[] : CX_Identifier(PID.3[]);
  identifier[] : CX_Identifier(PID.4[]);
  name[] : XPN_HumanName(PID.5[]);
  birthDate : Extract_Date(PID.7);
  gender : Convert_TerminologyCode(CWE_Code(PID.8), "Gender");
  name[] : XPN_HumanName(PID.9[]);
  address[] : XAD_Address(PID.11[]);
  telecom[] : XTN_ContactPoint(PID.13[], "home");
  telecom[] : XTN_ContactPoint(PID.14[], "work");
  maritalStatus : CWE_CodeableConcept(PID.16);
  identifier[] : ST_Identifier(PID.19);
  identifier[] : DLN_Identifier(PID.20, "DL");
  if (~PID.25?) {
    multipleBirthBoolean : PID.24;
  }
  multipleBirthInteger : PID.25;
  deceasedDateTime : Convert_DateTime(PID.29);
  if (~PID.29?) {
     deceasedBoolean : PID.30;
  }
  meta.lastUpdated : Convert_DateTime(PID.33);
  telecom[] : XTN_ContactPoint(PID.40[], "home");
  generalPractitioner[] :
  Build_Reference(Output_Resource[](XON_Organization[](PD1.3[])));
  generalPractitioner[] :
  Build_Reference(Output_Resource[](XCN_Practitioner[](PD1.4[])));
  active : true;
  id : $UUID();
  resourceType : "Patient";
}
```

Google Cloud Healthcare: Mappings and APIs

HL7v2 API

- Store and retrieve HL7v2 messages
- Parse HL7v2 messages
- Sample message
- Documentation

Other APIs

- DICOM API
- De-ID
- Full documentation

FHIR API

- Mostly implements the external FHIR spec
- Custom methods for transformations
- Sample resource
- Documentation

Standard Mappings

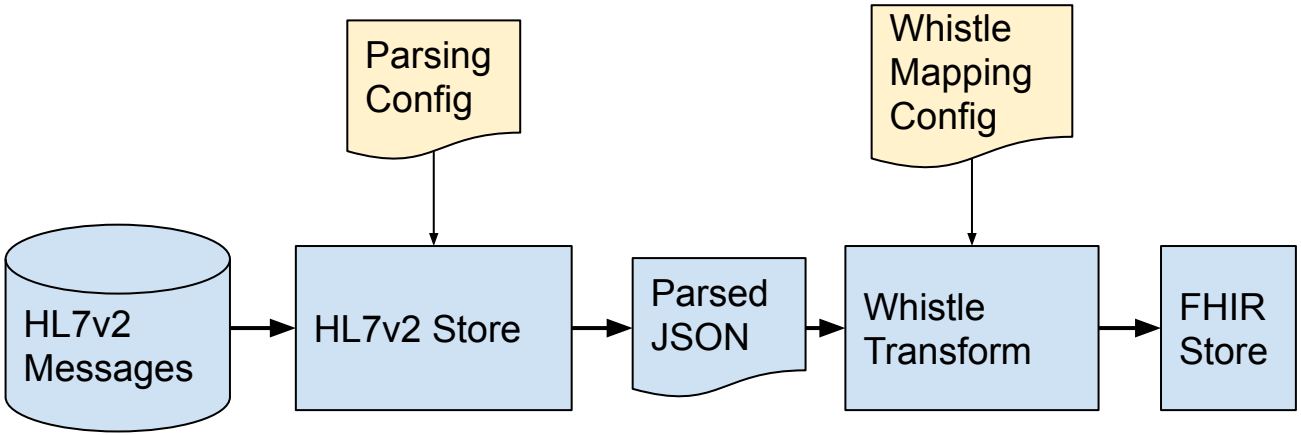
- Mapping Language: GitHub
- Mapping Engine: GitHub
- HL7v2-FHIR Mapping Configs: GitHub
- FHIR-OMOP Mapping Configs: GitHub
- Sample Streaming V2 to FHIR pipeline: GitHub
- Jupyter tooling for creating mappings: GitHub

Pipelines

Cloud DataFlow

- Batch and streaming data processing pipeline
- Runs Apache Beam pipelines in the cloud
- Sample v2 to FHIR streaming pipeline
  - Listens on Cloud PubSub for incoming v2 messages
  - Reads v2 message from Cloud HL7v2 Store
  - Transforms v2 to FHIR using mapping Whistle config
  - Writes FHIR bundle to Cloud FHIR Store
  - Code: Github
- Performance
  - Streaming: 15qps on 75% utilization per worker on a standard worker
  - Batch: 500+ records/second per worker on a standard worker
  - Linearly scales to 1000s of workers

End-to-End Demo Pipeline



Google Cloud DataFusion

- ELT/ETL tool for creating pipelines
- Has a GUI for setting up a pipeline with plugins
- Healthcare Plugins for CDF
  - HL7v2 Store connector (streaming)
  - FHIR Store connector (bundle sink/batch sink)
  - Whistle Mapping Transform

Questions