FHIR Profiling Language (FPL)

Mark Kramer and Chris Moesel

Sept. 10, 2019

# What is FPL?

**FPL is an author-friendly specification for profiling FHIR using a domain-specific language (DSL) to define profiles, extensions, value sets, and code systems**, paired with a reference implementation of an interpreter/compiler that creates FHIR StructureDefinitions (SD) ready for the FHIR IG Publisher.

# Why is it needed?

**To better support complex clinical modeling/profiling projects and effectively integrate across projects**.

Summary of the rationale:

1. The FHIR community needs scalable, fast, and user-friendly tools for IG creation and maintenance. Profiling projects are difficult and slow, and the resulting IG quality is inconsistent.
2. As a user-facing format, SDs are complex and unwieldy.
3. Available tools (Forge, Trifolia-on-FHIR, Excel spreadsheets) improve this situation. These tools share certain characteristics:
	1. Although the tools provide a friendlier interface, the user must still understand many SD details.
	2. The tools are not particularly agile when it comes to refactoring. Cross-cutting revisions happen all the time in non-trivial profiling projects.
	3. Source code control system (SCCS) features such as differentials and merging changes are not well supported. Excel files cannot be effectively diff’ed, and the other tools can be managed in SCCS only as SDs.
4. It can be difficult make sense of the Profile pages in IGs (see [this example from the September 2019 ballot](https://lightmyfhir.org/2019/09/09/fhir-implementation-guide-presentation/)). FPL is clearer and more compact.
5. Many years of experience has proven that creating and maintaining complex software projects is best approached with textual languages. As a DSL designed for the job of profiling, FPL is concise, understandable, and aligned to user intentions.
6. FPL is ideal for SCCS, with meaningful differentials, support for merging and conflict resolution, and refactoring through global search/replace operations. These features allow FPL to scale in ways that visual editors and spreadsheets cannot.
7. FPL will provide an easy path to migrate forward and backward between FHIR versions

# What might FPL look like?

Here is an example of what FPL *might* look like if we used FPL to define vital sign profiles (imagine that US Core and FHIR vital sign profiles did not exist). The syntax below is roughly based on grammar developed in the [CIMPL project](http://standardhealthrecord.org/cimpl-doc/). This code would be in one or more text files. Highlighting is provided by a Visual Studio Code plug-in. Explanatory comments are presented in Java style (//).

// File header

Canonical: http://example.org/vital

FHIRversion:4.0.0

Version: 1.2.0

Publisher: "Not the HL7 International Patient Care Work Group"

// CodeSystems used in this module. Aliases for well-known code system could be built into FPL to reduce the “boilerplate” aspect.

CodeSystem: LNC = http://loinc.org

CodeSystem: UCUM = http://unitsofmeasure.org

CodeSystem: SCT = http://snomed.info/sct

CodeSystem: CAT = http://terminology.hl7.org/CodeSystem/observation-category

// The parent class for all Vital Sign profiles

Profile: VitalSign

Parent: Observation // corresponds to SD baseDefinition and/or type

Title: "Parent for all vital sign profiles."

Definition: "Sets minimum expectations for the Observation Resource to record, search and fetch the vital signs associated with a patient."

// constraints:

\* value[x] only Quantity

\* category 1..1

\* category = CAT#vital-signs

\* hasMember 0..0

// Define Body Weight

Profile: BodyWeight

Parent: VitalSign

Title: "Body Weight Vital Sign"

Definition: "The mass or heaviness of the individual."

\* code = LNC#29463-7 "Body weight"

\* valueQuantity units from BodyWeightUnitsVS (required)

// The following creates a top-level extension. Extensions at deeper levels could be expressed as referenceRange.extension([http://hl7.org/fhir/us/core/StructureDefinition/us-core-race](http://hl7.org/fhir/us/core/StructureDefinition-us-core-race.html))

\* extension(PreconditionCode) 0..\*

\* method from BodyWeightMethodVS (extensible)

\* PreconditionCode from BodyWeightPreconditionVS (extensible)

\* bodySite 0..0

\* component 0..0

Extension: PreconditionCode

Title: "The circumstance of an observation."

Definition: "Conditions or context of an observation, for example, under sedation, fasting or post-exercise."

\* value[x] only CodeableConcept

// Define Blood Pressure

Profile:    BloodPressure

Parent: VitalSign

Title: "Blood pressure measurement."

Definition: "Records blood pressure measurements, defined as the force of circulating blood on the walls of the arteries."

Extension:   PreconditionCode 0..\*

\* code = LNC#85354-9 "Blood pressure panel with all children optional"

\* value 0..0

\* method from BloodPressureMethodVS (extensible)

// Binding (“from”) can refer internally or externally defined value sets

\* bodySite from BloodPressureBodyLocationVS (extensible)

\* PreconditionCode from CardiopulmonaryPreconditionVS (extensible)

\* component discriminates on value at code.coding.code

\* component.slice(SystolicBP) 1..1

\* component.slide(DiastolicBP) 1..1

Slice: SystolicBP

Slices: Observation.component

Title: "Systolic Blood Pressure"

Definition: "The blood pressure during the contraction of the left ventricle of the heart."

\* code = LNC#8480-6 "Systolic blood pressure"

\* value[x] only Quantity

\* valueQuantity units = UCUM#mm[Hg] "mmHg"

Slice: DiastolicBP

Slices: Observation.component

Title: "Diastolic Blood Pressure"

Definition: "The blood pressure after the contraction of the heart while the chambers of the heart refill with blood, when the pressure is lowest."

\* code = LNC# 8462-4 "Diastolic blood pressure"

\* value[x] only Quantity

\* valueQuantity units = UCUM#mm[Hg] "mmHg"

// Value set definitions... could be in the same file or another one

ValueSet: BodyWeightUnitsVS

Title: "Body weight units."

Definition: "Acceptable units for body weight measurement."

UCUM#kg "kg"

UCUM#[lb\_av] "pounds"

ValueSet: BodyWeightMethodVS

Title: "Body weight measurement methods."

Definition: "Method used to determine body weight."

SCT#466289007 "Bed scale"

SCT#720689000 "Chair scale"

SCT#462242008 "Patient sling scale"

SCT#58514003 "Infant scale"

SCT#469204003 "Floor scale, electronic"

SCT#469787007 "Floor scale, mechanical"

// truncated

ValueSet: BodyWeightPreconditionVS

Title: "Body weight preconditions."

Definition: "Circumstances for body weight measurement."

SCT#971000205103 "Wearing street clothes with shoes"

SCT#961000205106 "Wearing street clothes, no shoes"

SCT#951000205108 "Wearing underwear or less"

// An unrelated example showing potential grammar for intensional value sets

ValueSet: PrimaryCancerDisorderVS

Description: "Types of primary malignant neoplastic disease."

SCT#363346000 "Malignant neoplastic disease (disorder)"

\* include descendent-of SCT#363346000 "Malignant neoplastic disease (disorder)"

\* exclude descendent-of SCT#128462008 "Secondary malignant neoplastic disease (disorder)

# Benefits of FPL

* Lowers the barrier of entry to profiling
* Increased productivity
* Supports distributed development
* Easier to understand
* Makes the author’s intent clear
* Increases readability
* Reduces errors
* Round tripping to/from SDs provides pathway for existing IGs and tooling
* Enforces consistency in SDs (by compiling FPL into SD using consistent patterns)
* Works well with SCCS
* Provides meaningful differentials
* Any text editor can be used to modify an FPL file
* Existing editing environments can provide text colorization, look-ahead syntax, go-to-definition, etc.

# After FPL is authored, then what?

* User will run an FPL compiler
* Syntactic and logical errors will be flagged at “compile time”
* FPL will be translated automatically into SDs that could populate an IG
* Formats such as Mind Maps, “Java doc” and data dictionaries can be generated from FPL
* Profiles created in FPL and the FPL itself can be released and used by other projects

# Why should FPL be standardized?

* Although FPL does not provide a new capability, it plays a role similar to Clinical Quality Language (CQL), a user-friendly interface on Expression Logical Model (ELM), the underlying machine-exchangeable format
* Standardization will build a community of users
* Promotes sharing of profiles in readable form
* Provides an exchange format between profiling tools
* Sharing of tools built to FPL (such as exporters to Mind Maps, “Java doc” and data dictionaries)