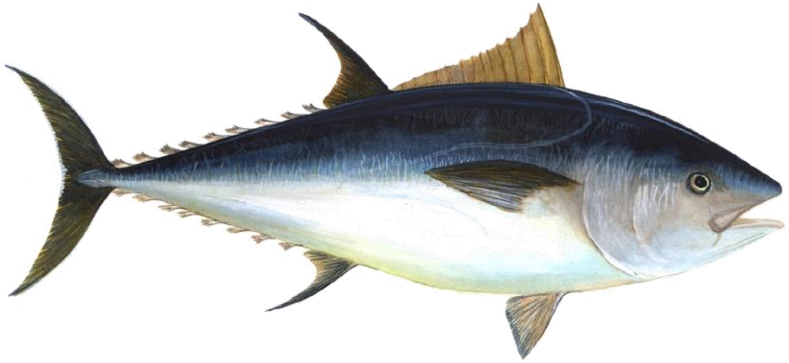
Advanced FHIR Shorthand



Mark Kramer
Chief Engineer, Health Innovation Center
MITRE Corporation

Chris Moesel
Principal Software Systems Engineer
MITRE Corporation

September, 2020

MITRE

Tuna Topics

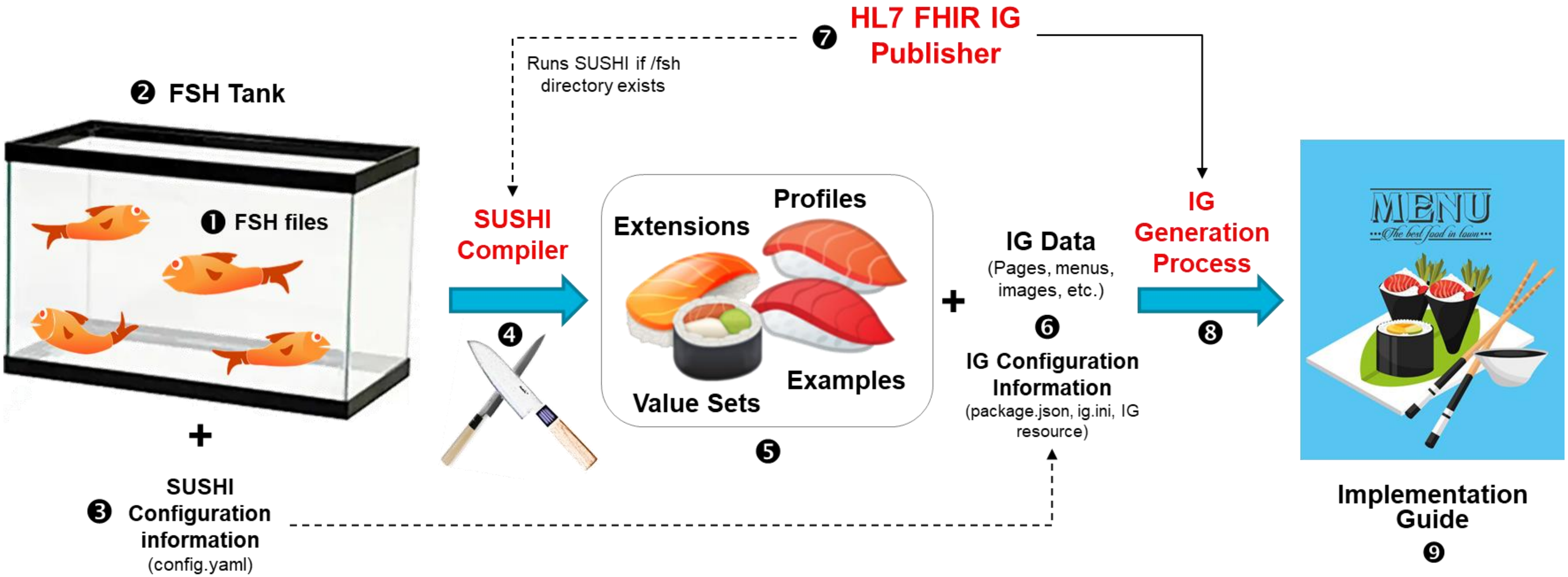
- **SUSHI**

  +

- **FSH Features:**
  - ValueSets
  - Extensions
  - Caret Paths
  - Slicing
  - Instances
  - Rule Sets, Mixins
  - Invariants
  - Mapping
  - Exact Equality

Using SUSHI to Produce FHIR from FSH

Workflow with FSH, SUSHI, and IG Publisher



❶ FSH files

❷ FSH Tank

❸ SUSHI Configuration information (config.yaml)

❹ SUSHI Compiler

❺ Profiles Extensions Value Sets Examples

❻ IG Data (Pages, menus, images, etc.) + IG Configuration Information (package.json, ig.ini, IG resource)

❼ HL7 FHIR IG Publisher

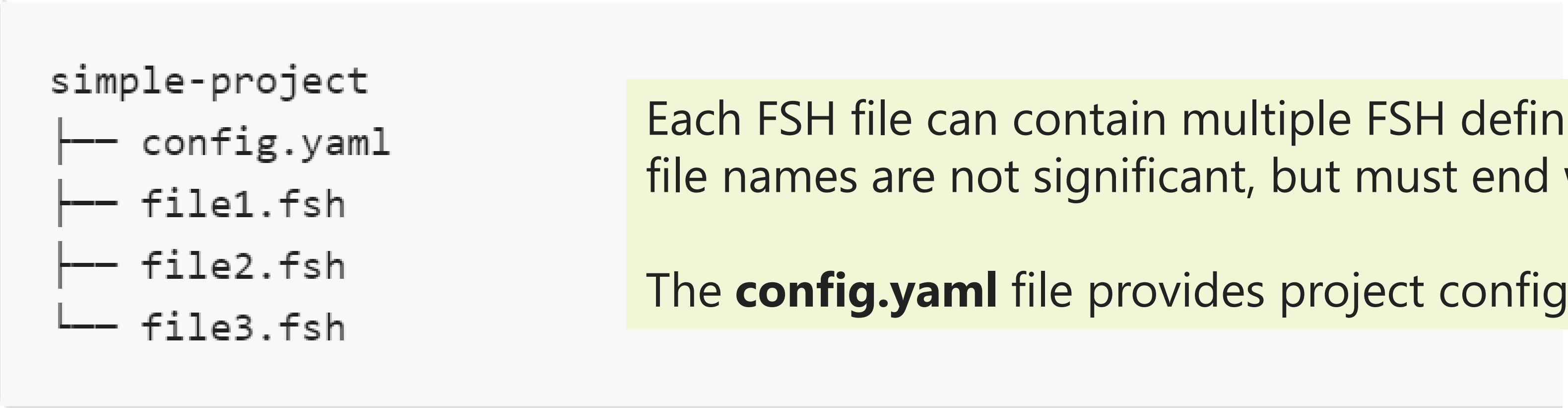
Runs SUSHI if /fsh directory exists

❽ IG Generation Process

❾ Implementation Guide

Credits: Sushi clipart from Google and WhatsApp rendering of Unicode 6.0 sushi emoji, Sushi menu from PNGWave, Non-Commercial Use, no attribution required (https://www.pngwave.com/png-clip-art-oxcer)

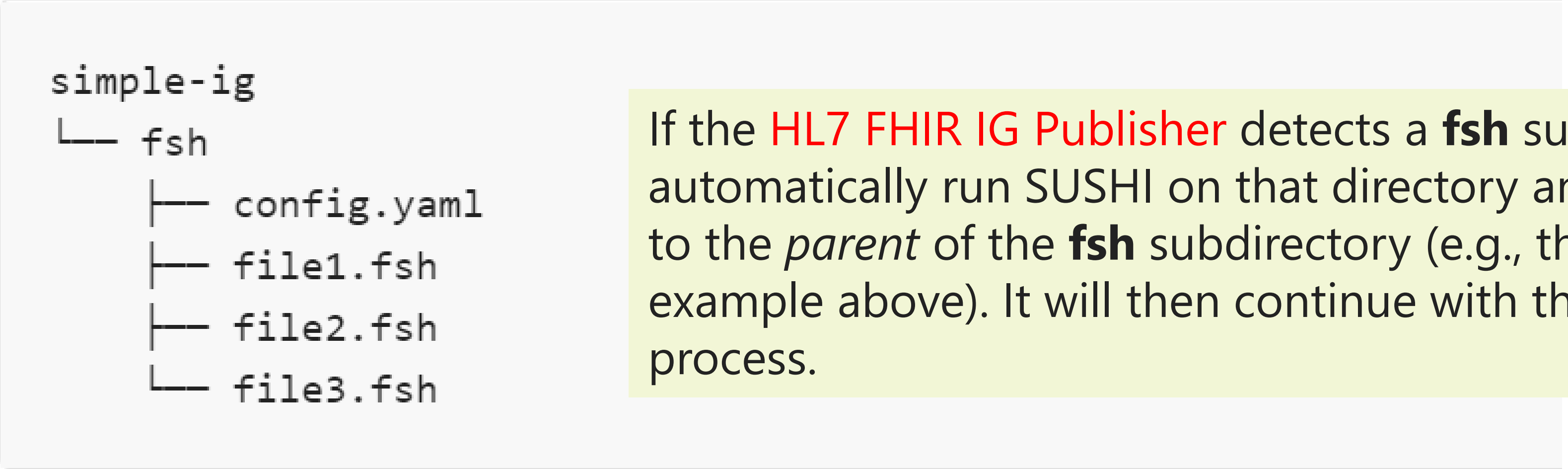
Project (FSH Tank) Structure



```
simple-project
├── config.yaml
├── file1.fsh
├── file2.fsh
└── file3.fsh
```

Each FSH file can contain multiple FSH definitions of varying types. FSH file names are not significant, but must end with the **.fsh** extension.

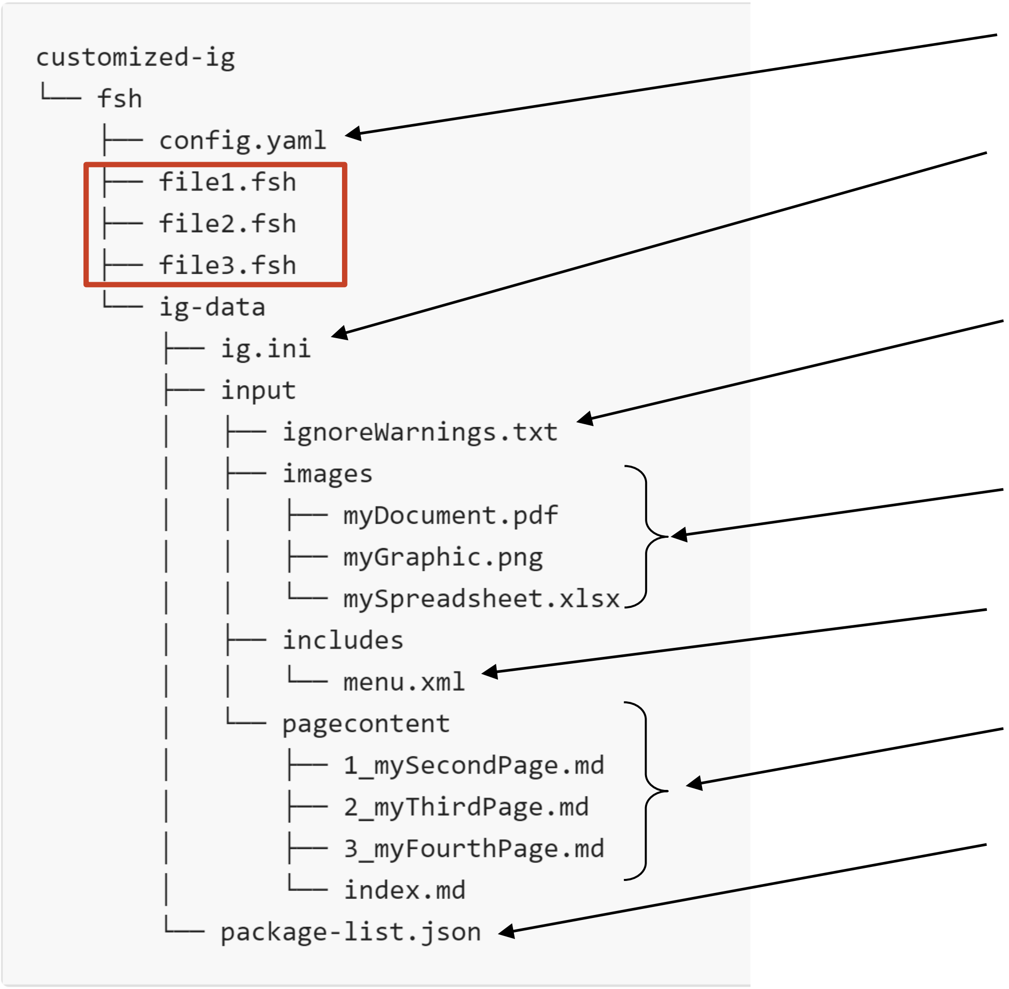
The **config.yaml** file provides project configuration data to SUSHI.



```
simple-ig
└── fsh
    ├── config.yaml
    ├── file1.fsh
    ├── file2.fsh
    └── file3.fsh
```

If the HL7 FHIR IG Publisher detects a **fsh** subdirectory, it will automatically run SUSHI on that directory and output the SUSHI results to the *parent* of the **fsh** subdirectory (e.g., the **simple-ig** directory in the example above). It will then continue with the normal IG Publisher process.

Project Structure for IGs



```
customized-ig
└── fsh
    ├── config.yaml
    ├── file1.fsh
    ├── file2.fsh
    ├── file3.fsh
    └── ig-data
        ├── ig.ini
        ├── input
        │   ├── ignoreWarnings.txt
        │   ├── images
        │   │   ├── myDocument.pdf
        │   │   ├── myGraphic.png
        │   │   └── mySpreadsheet.xlsx
        │   ├── includes
        │   │   └── menu.xml
        │   └── pagecontent
        │       ├── 1_mySecondPage.md
        │       ├── 2_myThirdPage.md
        │       ├── 3_myFourthPage.md
        │       └── index.md
        └── package-list.json
```

config.yaml: This required file provides project configuration data to SUSHI.

ig-data/ig.ini: If present and no template property is specified in config.yaml, the user-provided file will be used instead of a generated one.

ig-data/input/ignoreWarnings.txt: To suppress specific QA warnings and information messages during the FHIR IG publication process.

ig-data/input/images/: Contains non-page content (images, spreadsheets, etc.) to include in the IG.

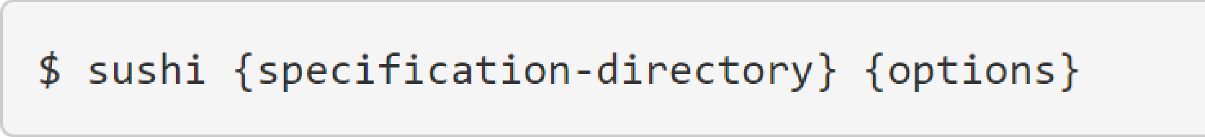
ig-data/includes/menu.xml: If present and no menu property is specified in config.yaml, it will be used for the IG's menu layout.

ig-data/input/pagecontent/: Contains markup (.xml) or markdown (.md) files with the narrative content of your IG.

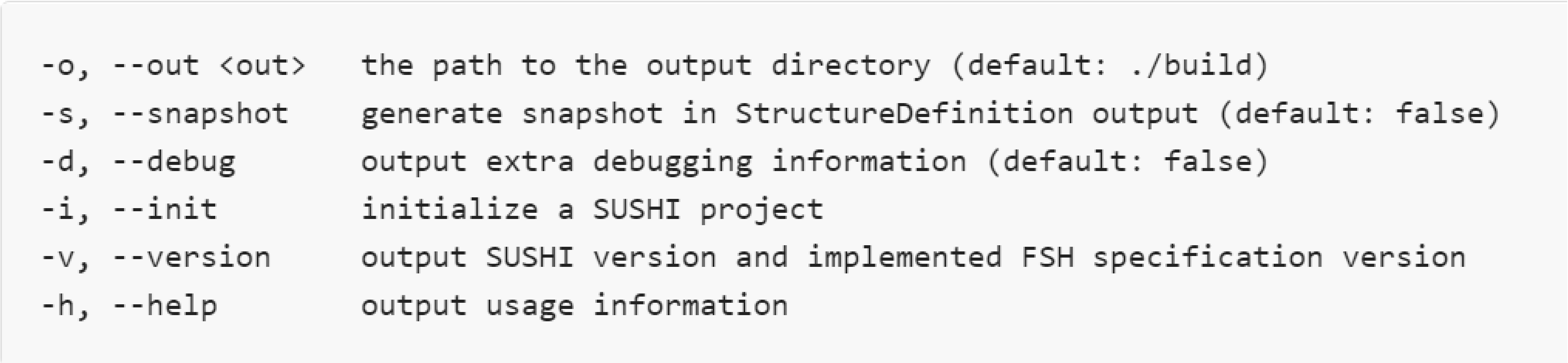
ig-data/package-list.json: If present and no history property is specified in config.yaml, it will be used instead of a generated package-list.json.

Executing SUSHI

- **Sushi translates FSH files into FHIR artifacts (profiles, extensions, value sets, instances, code systems)**
- **SUSHI runs from a command prompt ($)**
  - For installation, see https://fshschool.org/docs/sushi/installation/



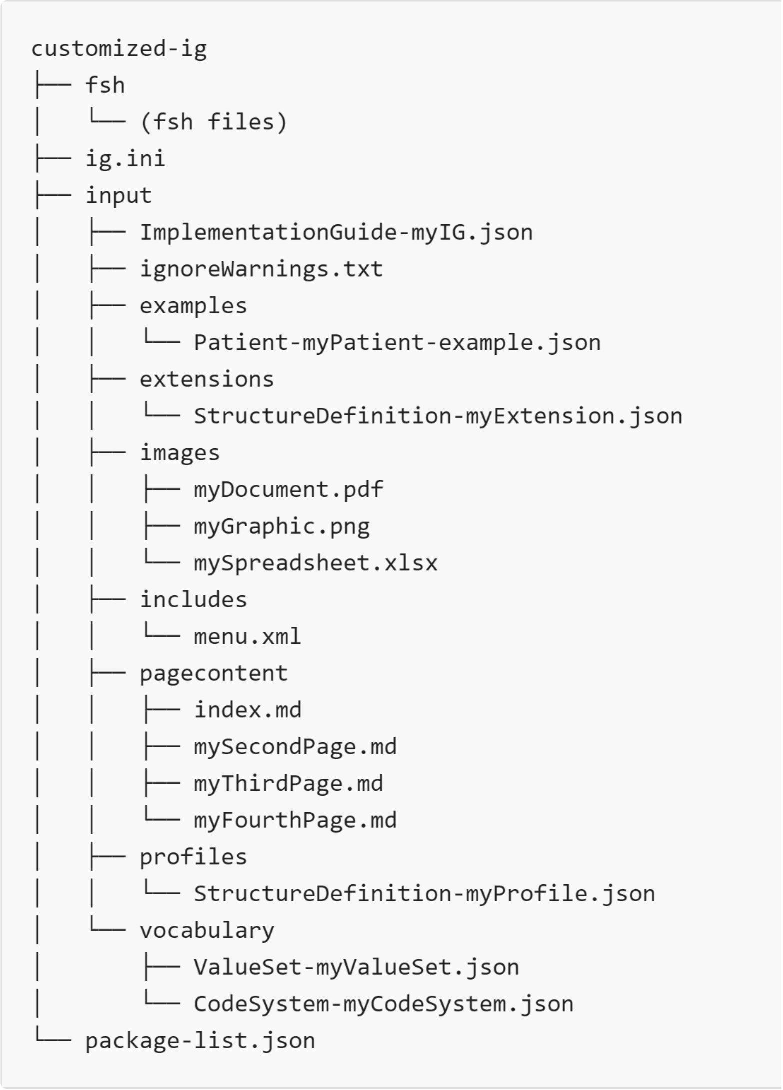
```
$ sushi {specification-directory} {options}
```



```
-o, --out <out>   the path to the output directory (default: ./build)
-s, --snapshot    generate snapshot in StructureDefinition output (default: false)
-d, --debug       output extra debugging information (default: false)
-i, --init        initialize a SUSHI project
-v, --version     output SUSHI version and implemented FSH specification version
-h, --help        output usage information
```

https://fshschool.org/docs/sushi/running/#running-sushi

SUSHI Outputs



```
customized-ig
├── fsh
│   └── (fsh files)
├── ig.ini
├── input
│   ├── ImplementationGuide-myIG.json
│   ├── ignoreWarnings.txt
│   ├── examples
│   │   └── Patient-myPatient-example.json
│   ├── extensions
│   │   └── StructureDefinition-myExtension.json
│   ├── images
│   │   ├── myDocument.pdf
│   │   ├── myGraphic.png
│   │   └── mySpreadsheet.xlsx
│   ├── includes
│   │   └── menu.xml
│   ├── pagecontent
│   │   ├── index.md
│   │   ├── mySecondPage.md
│   │   ├── myThirdPage.md
│   │   └── myFourthPage.md
│   ├── profiles
│   │   └── StructureDefinition-myProfile.json
│   └── vocabulary
│       ├── ValueSet-myValueSet.json
│       └── CodeSystem-myCodeSystem.json
└── package-list.json
```

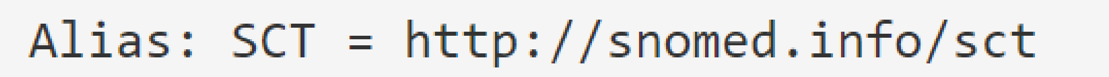
- Default output is /build
- Everything is where the IG Publisher expects to find them
- The /build/input directory is actually an *output* of SUSHI, but so named because it is an input to the IG Publisher.

Value Sets

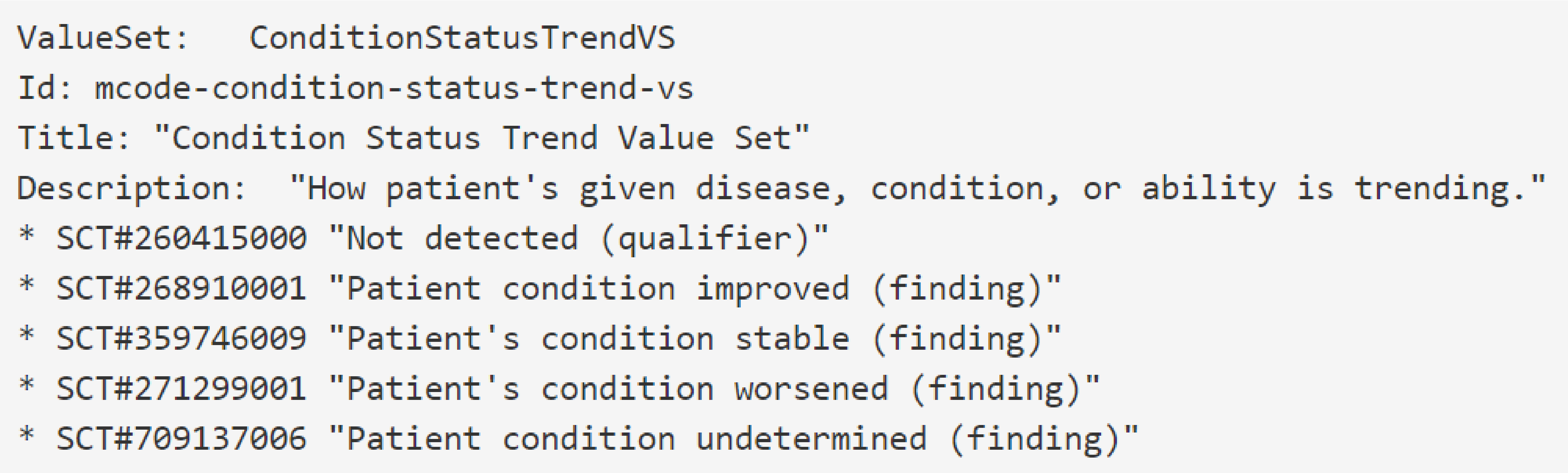
Defining Value Sets in FSH

An extensional value set contains an explicit list of codes

The extensional form is very simple: * {coding}



```
Alias: SCT = http://snomed.info/sct
```



```
ValueSet:   ConditionStatusTrendVS
Id: mcode-condition-status-trend-vs
Title: "Condition Status Trend Value Set"
Description:  "How patient's given disease, condition, or ability is trending."
* SCT#260415000 "Not detected (qualifier)"
* SCT#268910001 "Patient condition improved (finding)"
* SCT#359746009 "Patient's condition stable (finding)"
* SCT#271299001 "Patient's condition worsened (finding)"
* SCT#709137006 "Patient condition undetermined (finding)"
```

Value Set Rules

- **Rule to include or exclude a single code:**

```
* SCT#54102005  "G1 grade (finding)"
* exclude SCT#54102005  "G1 grade (finding)"
```

- **Rule to include/exclude an entire value set:**

```
* codes from valueset http://hl7.org/fhir/ValueSet/bodysite-laterality
* exclude codes from valueset http://hl7.org/fhir/ValueSet/bodysite-laterality
```

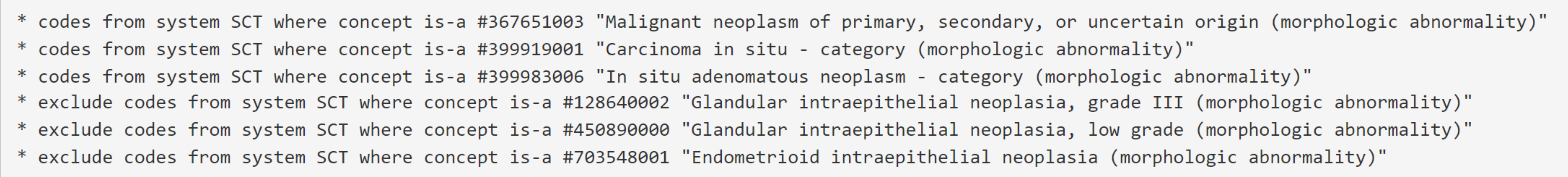
- **Rule to include/exclude an entire code system:**

```
* codes from system http://hl7.org/fhir/ndfrt
* exclude codes from system http://hl7.org/fhir/ndfrt
```

Value Set Filtering Rules

- **Rules can contain filter expressions that modify the codes to be included/excluded**
- **Syntax of filters depends on the particular vocabulary**
    - **e.g., ICD-10 filters are not the same as SNOMED-CT filters**

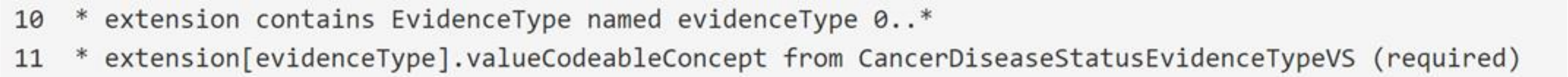
Here are examples for SNOMED-CT (aliased to SCT):



```
* codes from system SCT where concept is-a #367651003 "Malignant neoplasm of primary, secondary, or uncertain origin (morphologic abnormality)"
* codes from system SCT where concept is-a #399919001 "Carcinoma in situ - category (morphologic abnormality)"
* codes from system SCT where concept is-a #399983006 "In situ adenomatous neoplasm - category (morphologic abnormality)"
* exclude codes from system SCT where concept is-a #128640002 "Glandular intraepithelial neoplasia, grade III (morphologic abnormality)"
* exclude codes from system SCT where concept is-a #450890000 "Glandular intraepithelial neoplasia, low grade (morphologic abnormality)"
* exclude codes from system SCT where concept is-a #703548001 "Endometrioid intraepithelial neoplasia (morphologic abnormality)"
```

Extensions, Caret Rules, and Slicing

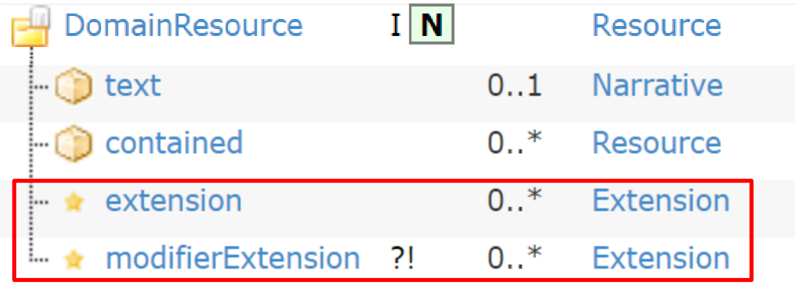
Walkthrough (Continued from Basic Tutorial)



```
* extension contains EvidenceType named evidenceType 0..*
* extension[evidenceType].valueCodeableConcept from CancerDiseaseStatusEvidenceTypeVS (required)
```

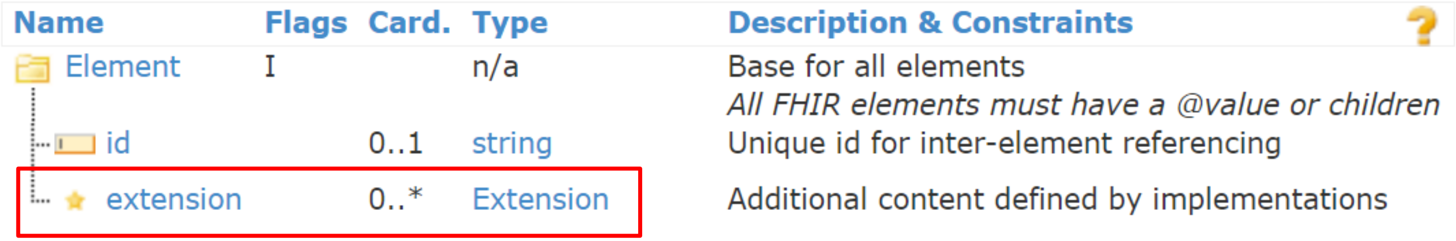
- **Use "contains" rule both for extensions and slicing**

What is an Extension?



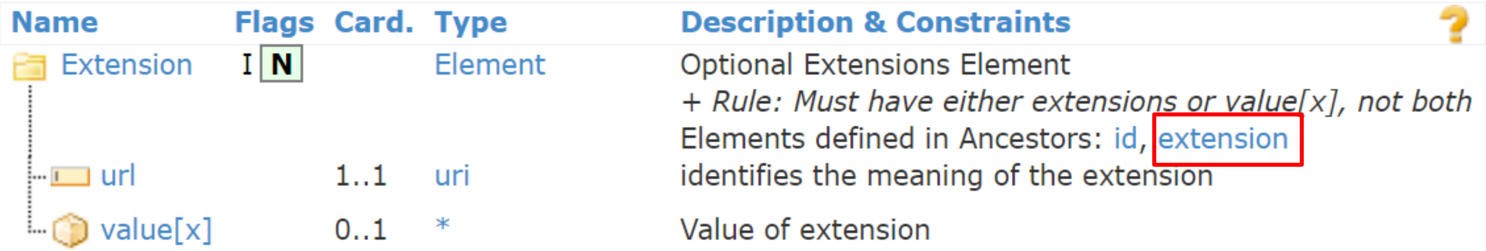
DomainResource	I N		Resource
text		0..1	Narrative
contained		0..*	Resource
extension		0..*	Extension
modifierExtension	?!	0..*	Extension

Every Resource has an extension array at the top level.



Name	Flags	Card.	Type	Description & Constraints
Element	I		n/a	Base for all elements *All FHIR elements must have a @value or children*
id		0..1	string	Unique id for inter-element referencing
extension		0..*	Extension	Additional content defined by implementations

Every element has an extension array



Name	Flags	Card.	Type	Description & Constraints
Extension	I N		Element	Optional Extensions Element + *Rule: Must have either extensions or value[x], not both* Elements defined in Ancestors: id, extension
url		1..1	uri	identifies the meaning of the extension
value[x]		0..1	*	Value of extension

Extension arrays contain Extension elements.

An Extension either has a value[x] or further extensions

"Adding an extension" really means constraining an extension array to contain a certain type of extension.

Extensions Rules (Two Types)

Inline Extensions:

* {extension-path} contains

{extension1} {card1}. *{flags1}* and

{extension2} {card2} *{flags2}* ...

Example:

```
* extension contains
    treatmentIntent 0..1 MS and
    terminationReason 0..* MS
```

choose names

Stand-Alone (Existing) Extensions:

* {extension-path} contains

{extension1} named {name1} {card1} *{flags1}* and

{extension2} named {name2} {card2} *{flags2}* ...

Example:

local name inside profile

```
* extension contains
    RadiationDosePerFraction named dosePerFraction 0..1 and
    RadiationFractionsDelivered named fractionsDelivered 0..1 MS and
    TotalRadiationDoseDelivered named totalDose 0..1
```

existing extension names

Defining In-Line Extensions

Defining the extension in-line does not require an "Extension" structure.

The resulting extension will not have a separate StructureDefinition.

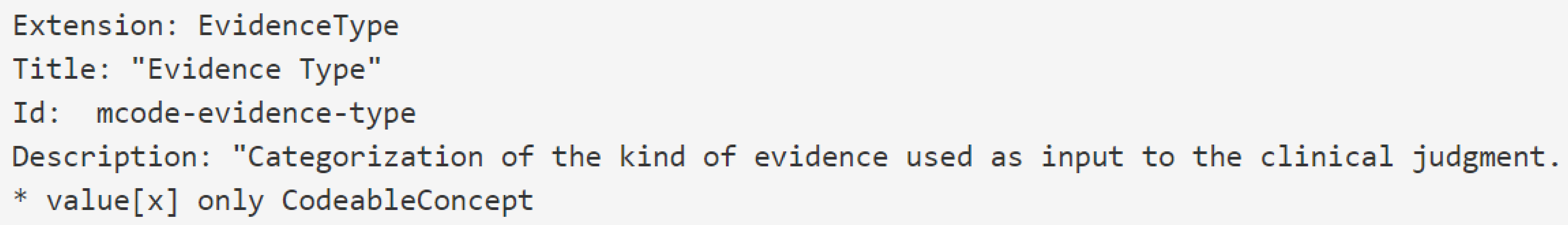
The "contains" statement is similar but does not name an extension.

```
* extension contains evidenceType 0..*

* extension[evidenceType].value[x] only CodeableConcept
* extension[evidenceType].valueCodeableConcept from CancerDiseaseStatusEvidenceTypeVS (required)
```

Defining Stand-Alone Extensions

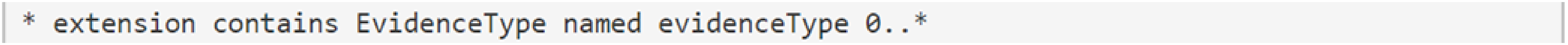
Define the extension using the "Extension" keyword. No parent is needed because FSH knows it is an Extension.



```
Extension: EvidenceType
Title: "Evidence Type"
Id:  mcode-evidence-type
Description: "Categorization of the kind of evidence used as input to the clinical judgment.
* value[x] only CodeableConcept
```

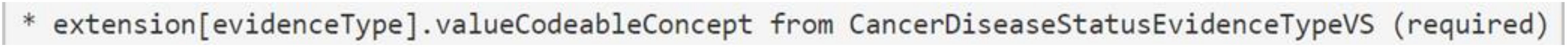
Now, in the profile, add it to an extension array using "contains".

This grammar also applies to an extension defined in another IG (use its URL).



```
* extension contains EvidenceType named evidenceType 0..*
```

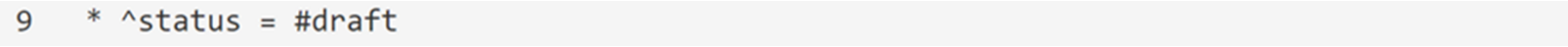
Once added, the extension can be further constrained by referring to the element in the extension array by name:



```
* extension[evidenceType].valueCodeableConcept from CancerDiseaseStatusEvidenceTypeVS (required)
```

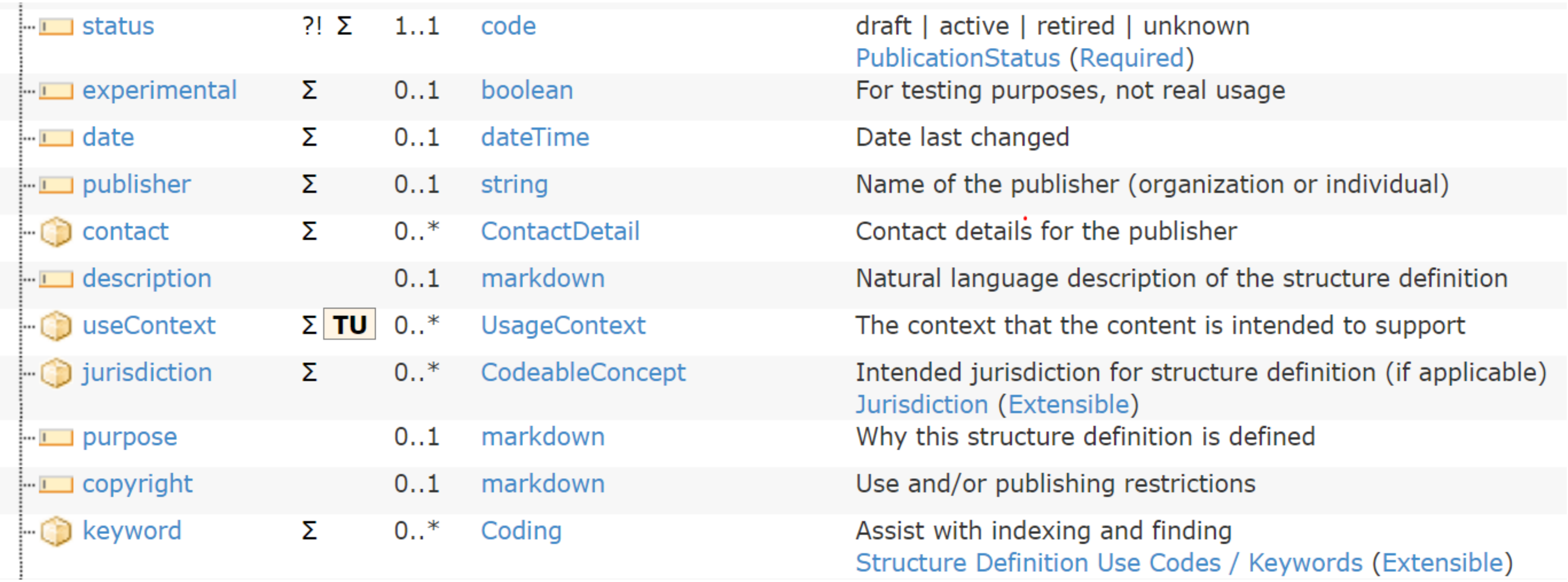
Caret Paths for StructureDefinitions

- **Caret (^) gives direct access to elements in StructureDefinition**



```
* ^status = #draft
```

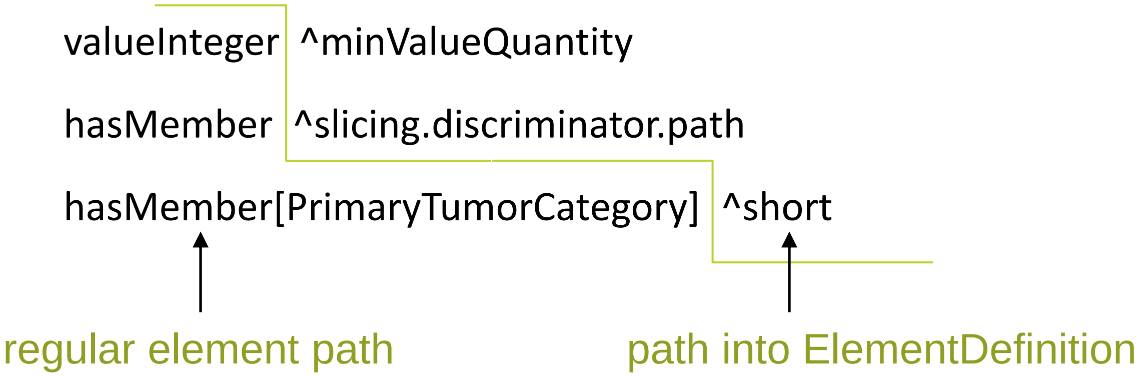
- **Useful for setting or overriding metadata elements:**



status	?! Σ	1..1	code	draft | active | retired | unknown PublicationStatus (Required)
experimental	Σ	0..1	boolean	For testing purposes, not real usage
date	Σ	0..1	dateTime	Date last changed
publisher	Σ	0..1	string	Name of the publisher (organization or individual)
contact	Σ	0..*	ContactDetail	Contact details for the publisher
description		0..1	markdown	Natural language description of the structure definition
useContext	Σ TU	0..*	UsageContext	The context that the content is intended to support
jurisdiction	Σ	0..*	CodeableConcept	Intended jurisdiction for structure definition (if applicable) Jurisdiction (Extensible)
purpose		0..1	markdown	Why this structure definition is defined
copyright		0..1	markdown	Use and/or publishing restrictions
keyword	Σ	0..*	Coding	Assist with indexing and finding Structure Definition Use Codes / Keywords (Extensible)

Caret Paths for ElementDefinitions

- **A StructureDefinition contains one ElementDefinition for every element and subelement**
- **Use the element name followed by caret path into the ElementDefinition**
- **Path examples:**



valueInteger ^minValueQuantity

hasMember ^slicing.discriminator.path

hasMember[PrimaryTumorCategory] ^short

regular element path

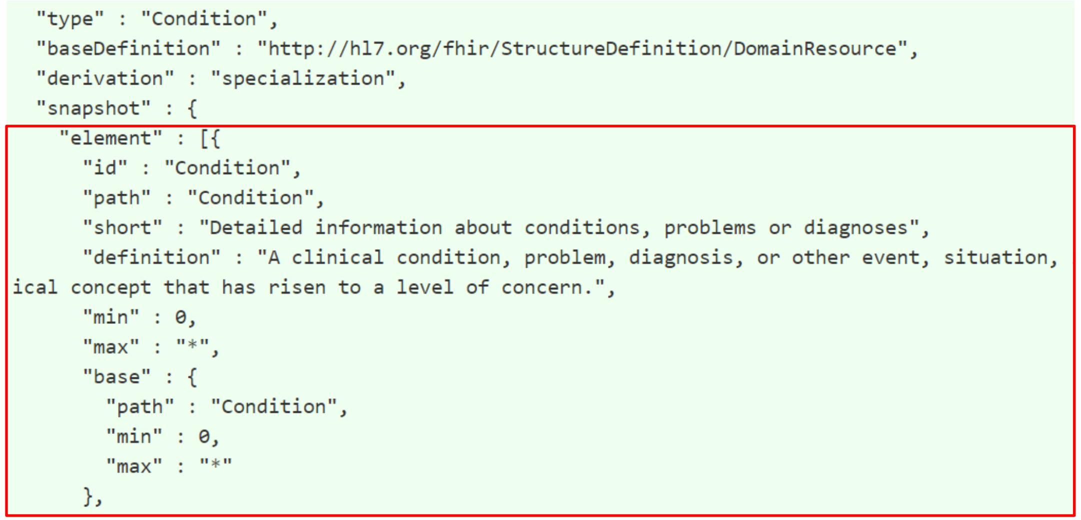
path into ElementDefinition

- **Example:**

```
* communication.language ^binding.description = "This binding is dictated by US FDA regulations."
```

The Oddball Dot Caret Path

- **The first ElementDefinition in any StructureDefinition refers to entire item**



```
  "type" : "Condition",
  "baseDefinition" : "http://hl7.org/fhir/StructureDefinition/DomainResource",
  "derivation" : "specialization",
  "snapshot" : {
    "element" : [{
      "id" : "Condition",
      "path" : "Condition",
      "short" : "Detailed information about conditions, problems or diagnoses",
      "definition" : "A clinical condition, problem, diagnosis, or other event, situation,
ical concept that has risen to a level of concern.",
      "min" : 0,
      "max" : "*",
      "base" : {
        "path" : "Condition",
        "min" : 0,
        "max" : "*"
      },
```

- **To refer to properties of this particular "self" element, use dot (.) as the element path**

Example: Provide a short description for an extension (defined in the “self” ElementDefinition):

```
* . ^short = "US Core Race Extension"
```

Slicing

- **Similar to extensions -- the objective is to say what can go into an array**
- **The array elements will not be Extensions**
- **Arrays we typically want to slice:**
  - Backbone elements, such as Observation.component
  - Arrays of complex data types, such as Identifier or Address, such as Practitioner.identifier
  - Arrays of references to resources, such as Observation.hasMember

- **Divide slicing into three steps:**
  1. Specify the slicing logic
  2. Identify the slices
  3. Define each slice

Slicing Step 1: Define Slicing Logic

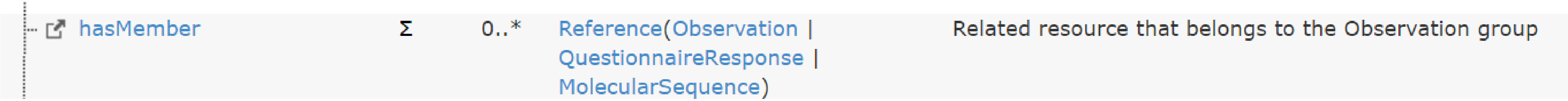
- **There has to be something that uniquely and reliably distinguishes the slices**
  - Given an instance assigned to the array, how do we know what slice it belongs to?
  - The "discriminator" -- comprised of a **type** and **path**
- **Slicing logic is specified in the ElementDefinition part of the StructureDefinition**
  - Use caret paths to specify the slicing logic

Example: Slice Observation.component on Observation.component.code

```
* component ^slicing.discriminator.type = #pattern     // or #value, #profile
* component ^slicing.discriminator.path = "code"    // any FHIRPath expression
* component ^slicing.rules = #open    // additional elements are ok
* component ^slicing.ordered = false    // by default, array elements in any order
* component ^slicing.description = “Slice pattern for component.code"  // optional
```

Slicing Logic: Another Example

Example: Slice Observation.hasMember



hasMember	Σ	0..*	Reference(Observation | QuestionnaireResponse | MolecularSequence)	Related resource that belongs to the Observation group

```
* hasMember ^slicing.discriminator.type = #profile
* hasMember ^slicing.discriminator.path =  "$this.resolve()"
* hasMember ^slicing.rules = #open
```

Slicing Step 2: Identify the slices ("contains")

*** array-element-path contains**

slice-name1 card1 flags1 and

slice-name2 card2 flag s2 ...

Each element must match
the datatype of the array

```
* component contains
    systolicBP 1..1 and
    diastolicBP 1..1
```

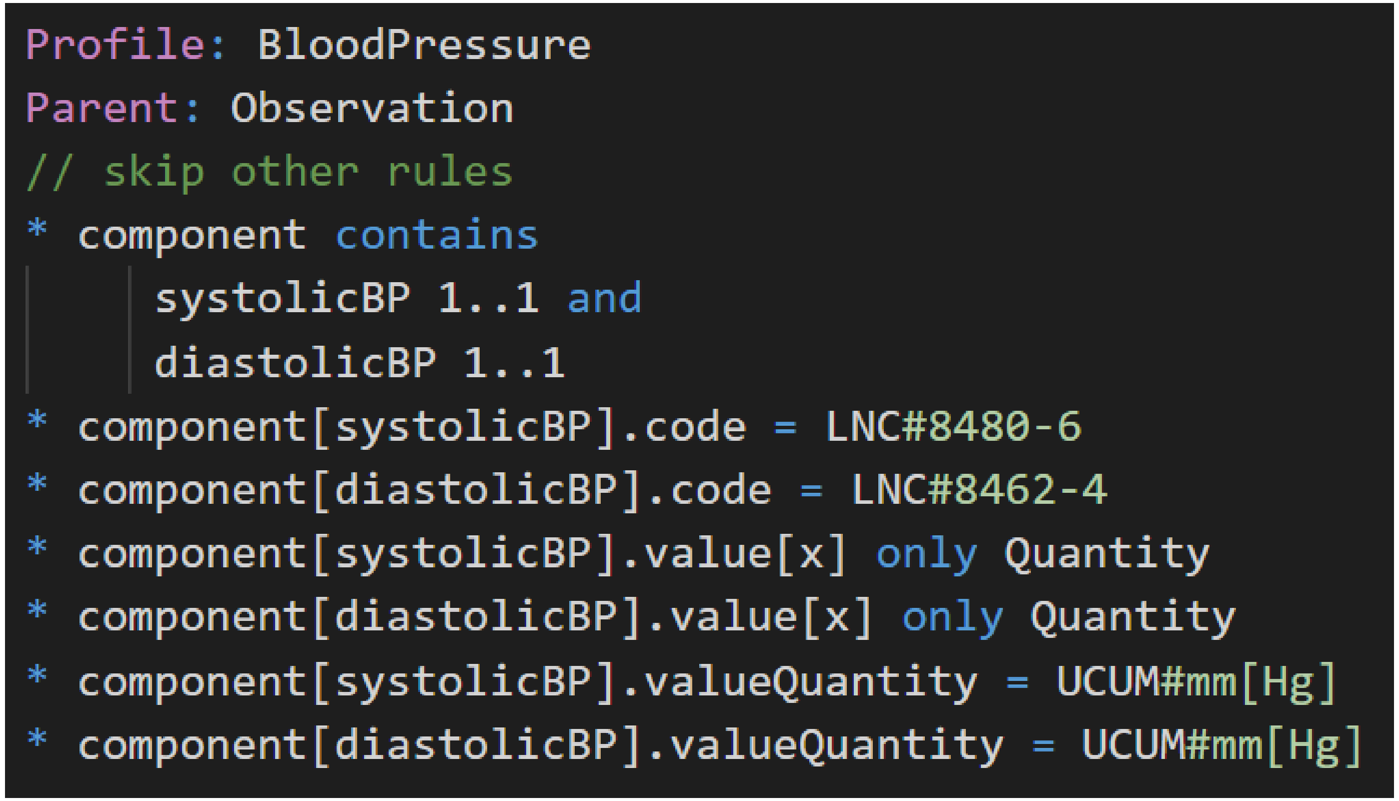
components

```
* hasMember contains
      PrimaryTumorCategory 0..1 and
      RegionalNodesCategory 0..1 and
      DistantMetastasesCategory 0..1
```

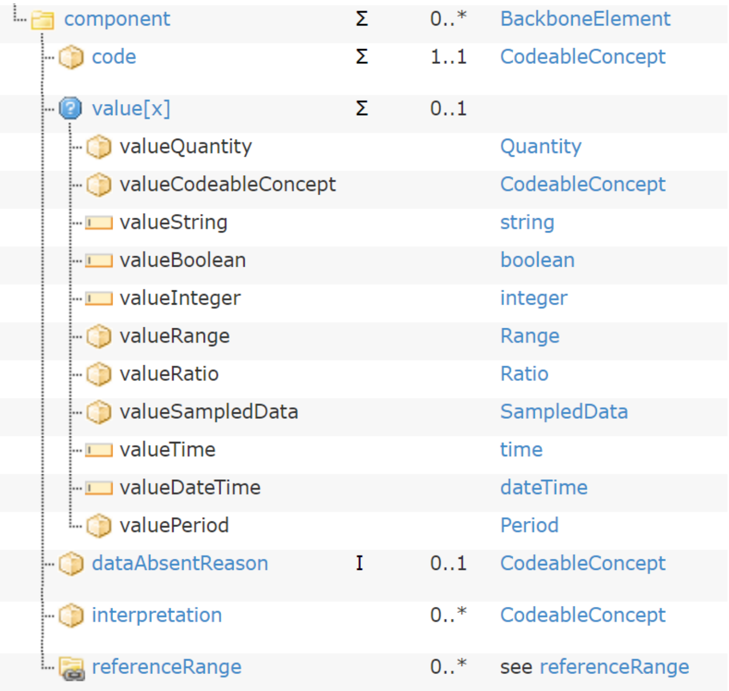
Profiled Observations

Slicing Step 3: Define Properties Each Slice

- **If the array type is resource reference(s), then the slices are defined either in an existing resource profile, or any one you define in your project (similar to "stand-alone" extensions)**
- **Slices are only defined in-line**



```
Profile: BloodPressure
Parent: Observation
// skip other rules
* component contains
      systolicBP 1..1 and
      diastolicBP 1..1
* component[systolicBP].code = LNC#8480-6
* component[diastolicBP].code = LNC#8462-4
* component[systolicBP].value[x] only Quantity
* component[diastolicBP].value[x] only Quantity
* component[systolicBP].valueQuantity = UCUM#mm[Hg]
* component[diastolicBP].valueQuantity = UCUM#mm[Hg]
```



component	Σ	0..*	BackboneElement
code	Σ	1..1	CodeableConcept
value[x]	Σ	0..1	
valueQuantity			Quantity
valueCodeableConcept			CodeableConcept
valueString			string
valueBoolean			boolean
valueInteger			integer
valueRange			Range
valueRatio			Ratio
valueSampledData			SampledData
valueTime			time
valueDateTime			dateTime
valuePeriod			Period
dataAbsentReason	I	0..1	CodeableConcept
interpretation		0..*	CodeableConcept
referenceRange		0..*	see referenceRange

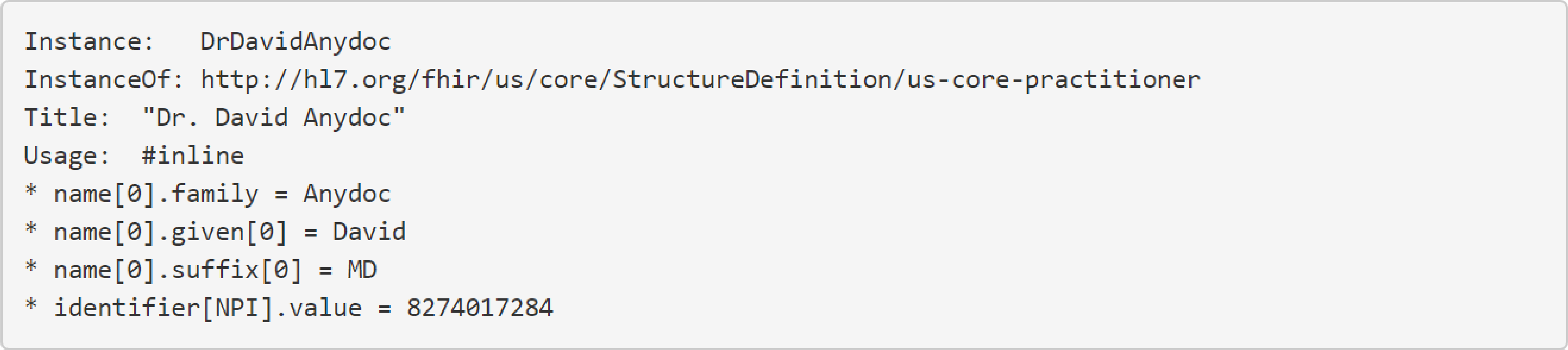
Defining Instances in FSH

Instances in IGs

- **Examples**
  - Instances that illustrate how to use a profile, presented on the Examples tab for the corresponding profile. You must have at least one example of each profile and extension in the IG.
- **Definitions**
  - Conformance items that are instances of resources such as search parameter, operation definition, or questionnaire
- **Inline**
  - Instances that should not be instantiated as an independent resource, but appears as part of another instance (for example, in a composition or bundle)

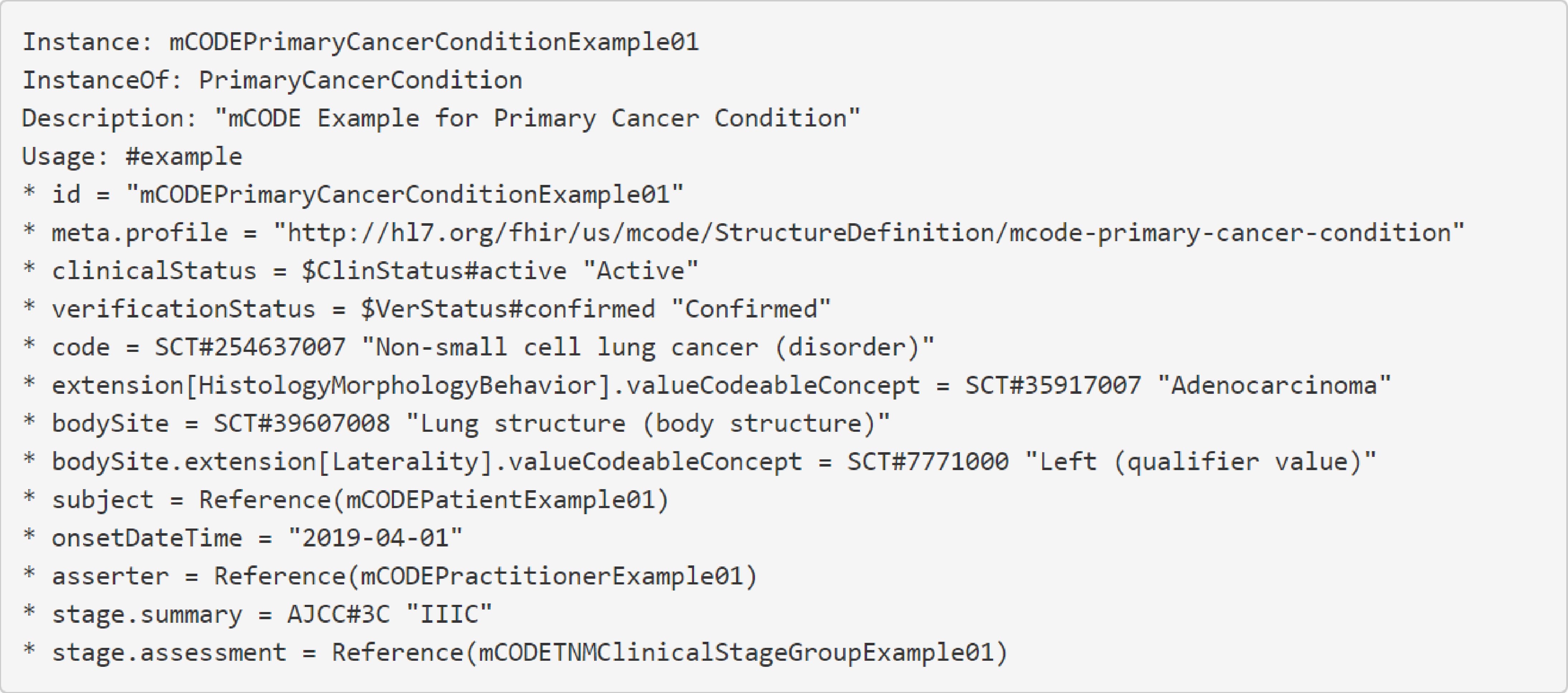
Defining Instances in FSH

- **Instances are defined in FSH using the "Instance" keyword**
- **"InstanceOf" instead of "Parent"**
- **All structures and values are inherited from the StructureDefinition (i.e. fixed codes, extensions) -- don't have to be repeated**
- **Instances only have fixed value rules, because instances have specific values**



```
Instance:   DrDavidAnydoc
InstanceOf: http://hl7.org/fhir/us/core/StructureDefinition/us-core-practitioner
Title:  "Dr. David Anydoc"
Usage:  #inline
* name[0].family = Anydoc
* name[0].given[0] = David
* name[0].suffix[0] = MD
* identifier[NPI].value = 8274017284
```

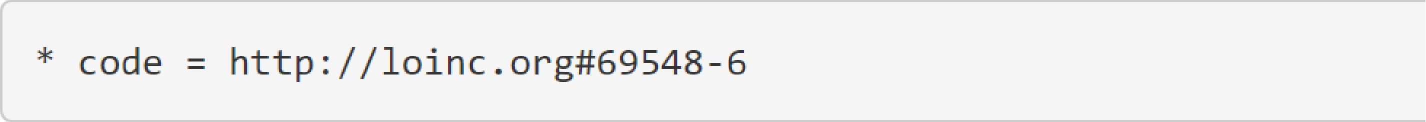
More Complex Instance Example



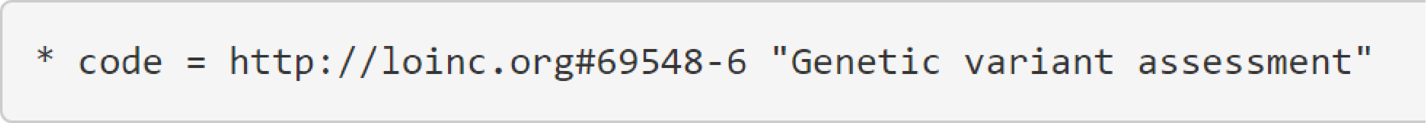
```
Instance: mCODEPrimaryCancerConditionExample01
InstanceOf: PrimaryCancerCondition
Description: "mCODE Example for Primary Cancer Condition"
Usage: #example
* id = "mCODEPrimaryCancerConditionExample01"
* meta.profile = "http://hl7.org/fhir/us/mcode/StructureDefinition/mcode-primary-cancer-condition"
* clinicalStatus = $ClinStatus#active "Active"
* verificationStatus = $VerStatus#confirmed "Confirmed"
* code = SCT#254637007 "Non-small cell lung cancer (disorder)"
* extension[HistologyMorphologyBehavior].valueCodeableConcept = SCT#35917007 "Adenocarcinoma"
* bodySite = SCT#39607008 "Lung structure (body structure)"
* bodySite.extension[Laterality].valueCodeableConcept = SCT#7771000 "Left (qualifier value)"
* subject = Reference(mCODEPatientExample01)
* onsetDateTime = "2019-04-01"
* asserter = Reference(mCODEPractitionerExample01)
* stage.summary = AJCC#3C "IIIC"
* stage.assessment = Reference(mCODETNMClinicalStageGroupExample01)
```

Assignment Statements in Profiles versus Instances

- In profiles and extensions, values represent the **minimum criteria** for conformance



```
* code = http://loinc.org#69548-6
```



```
* code = http://loinc.org#69548-6 "Genetic variant assessment"
```

- In the context of a **profile**, the first statement signifies an instance must have (1) the system http://loinc.org and (2) the code 69548-6 to pass validation.
- The second statement says that an instance must have (1) the system http://loinc.org, (2) the code 69548-6, **and (3)** the display text “Genetic variant assessment” to pass validation.

Typically, only the system and code are important conformance criteria, so the first statement (without the display text) is preferred in a profiling context.

In an **instance**, however, the display text conveys additional information useful to the information receiver, so the second statement would be preferred.

Forcing an Exact Match (Profiles and Extensions)

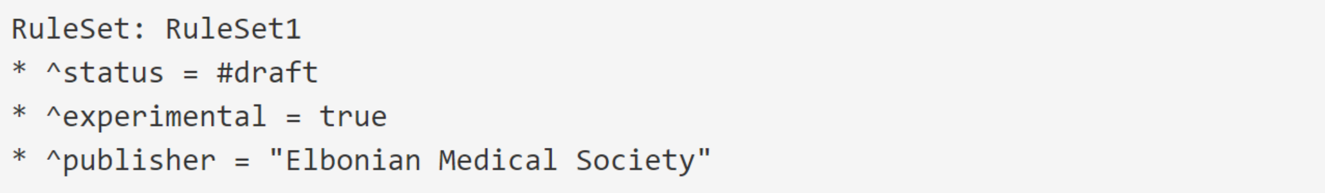
*** {path} = {value} (exactly)**

- **"(exactly)" indicates conformance to the profile requires a precise match to the specification, no more or less**
  - NO additional extensions, array elements, codings in CodeableConcept, etc.
- **Without "(exactly)" any instance that fulfills the pattern is valid -- i.e., no less but possibly more**

Additional Rules

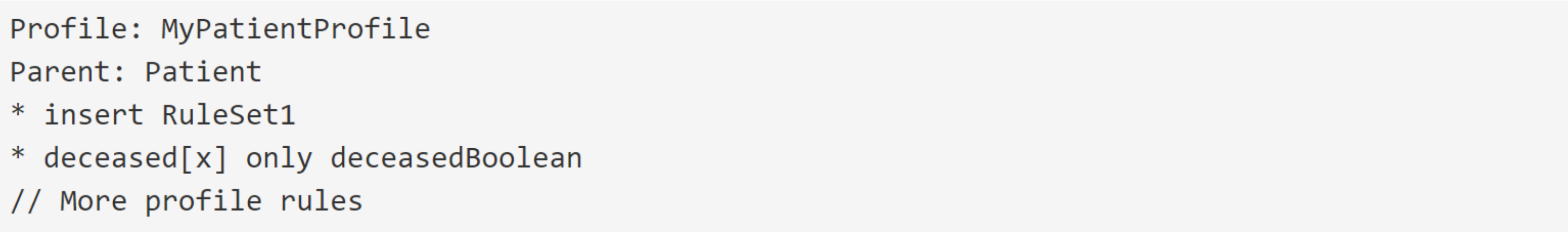
Rule Sets and Insert Rules

- **Provides ability to define free-floating rules and apply them to a compatible target**
- **The same rule set can be used in multiple places**
  – An example could be to set the same metadata on every StructureDefinition
- **A rule set can contain other rule sets**



```
RuleSet: RuleSet1
* ^status = #draft
* ^experimental = true
* ^publisher = "Elbonian Medical Society"
```

Defining a RuleSet

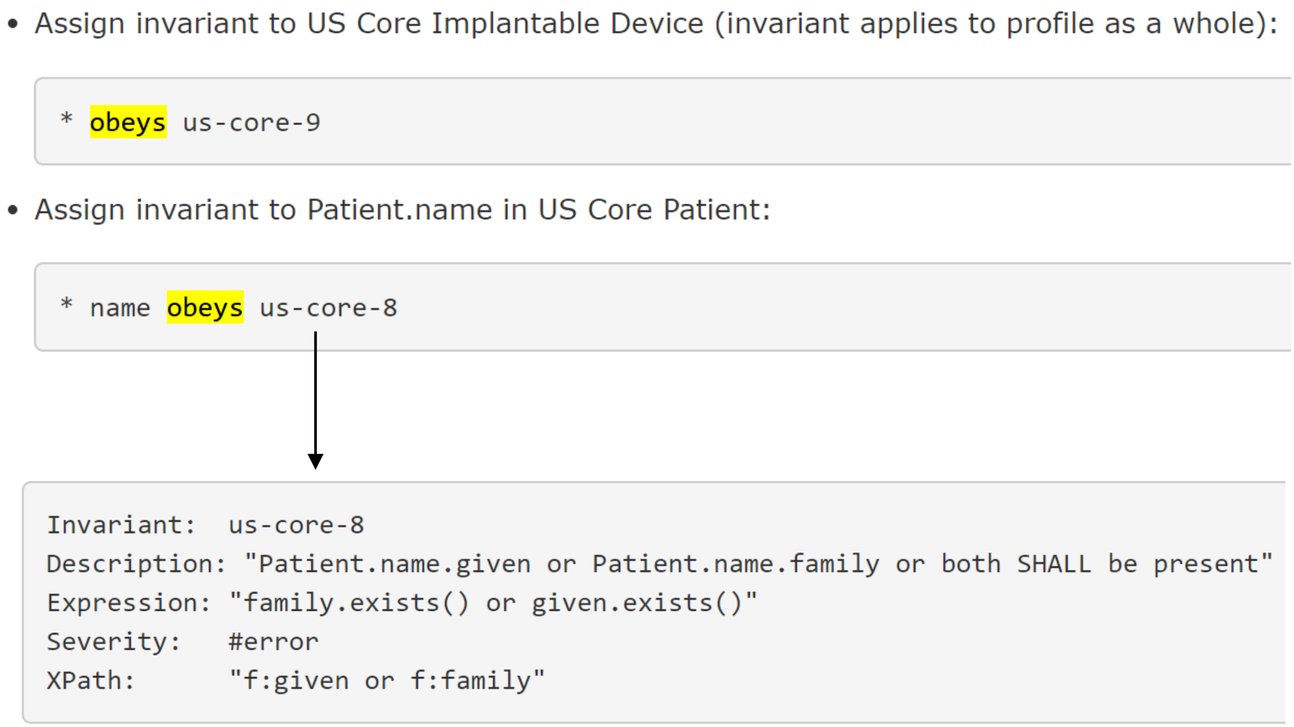


```
Profile: MyPatientProfile
Parent: Patient
* insert RuleSet1
* deceased[x] only deceasedBoolean
// More profile rules
```

Inserting a RuleSet

Invariants and "obeys"

- **Invariants represent logical constraints on values in a resource**
  - "obeys" rule populates ElementDefinition.constraint



- Assign invariant to US Core Implantable Device (invariant applies to profile as a whole):

```
* obeys us-core-9
```

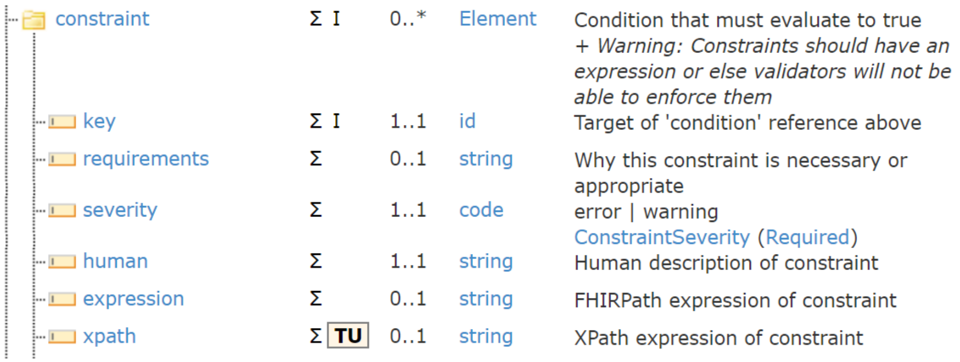
- Assign invariant to Patient.name in US Core Patient:

```
* name obeys us-core-8
```

```
Invariant:  us-core-8
Description: "Patient.name.given or Patient.name.family or both SHALL be present"
Expression: "family.exists() or given.exists()"
Severity:   #error
XPath:      "f:given or f:family"
```

adds constraint to "self" ElementDefinition (remember dot caret?)

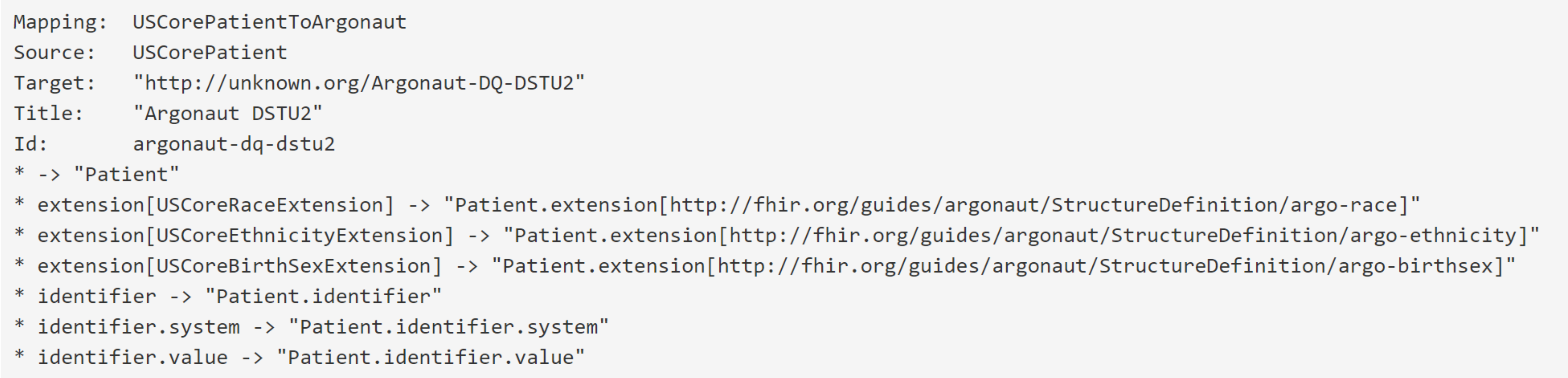
adds constraint to "name" ElementDefinition



constraint	Σ I	0..*	Element	Condition that must evaluate to true + *Warning: Constraints should have an expression or else validators will not be able to enforce them*
key	Σ I	1..1	id	Target of 'condition' reference above
requirements	Σ	0..1	string	Why this constraint is necessary or appropriate
severity	Σ	1..1	code	error | warning ConstraintSeverity (Required)
human	Σ	1..1	string	Human description of constraint
expression	Σ	0..1	string	FHIRPath expression of constraint
xpath	Σ TU	0..1	string	XPath expression of constraint

Mapping

- Mappings are an optional part of SDs that can be provided to help implementers understand the content and use resources correctly
- Mappings are informative and are not to be confused with computable mappings provided by FHIR Mapping Language or the StructureMap resource
- In FSH, mapping rules are part of a separate Mapping definition



```
Mapping:  USCorePatientToArgonaut
Source:   USCorePatient
Target:   "http://unknown.org/Argonaut-DQ-DSTU2"
Title:    "Argonaut DSTU2"
Id:       argonaut-dq-dstu2
* -> "Patient"
* extension[USCoreRaceExtension] -> "Patient.extension[http://fhir.org/guides/argonaut/StructureDefinition/argo-race]"
* extension[USCoreEthnicityExtension] -> "Patient.extension[http://fhir.org/guides/argonaut/StructureDefinition/argo-ethnicity]"
* extension[USCoreBirthSexExtension] -> "Patient.extension[http://fhir.org/guides/argonaut/StructureDefinition/argo-birthsex]"
* identifier -> "Patient.identifier"
* identifier.system -> "Patient.identifier.system"
* identifier.value -> "Patient.identifier.value"
```