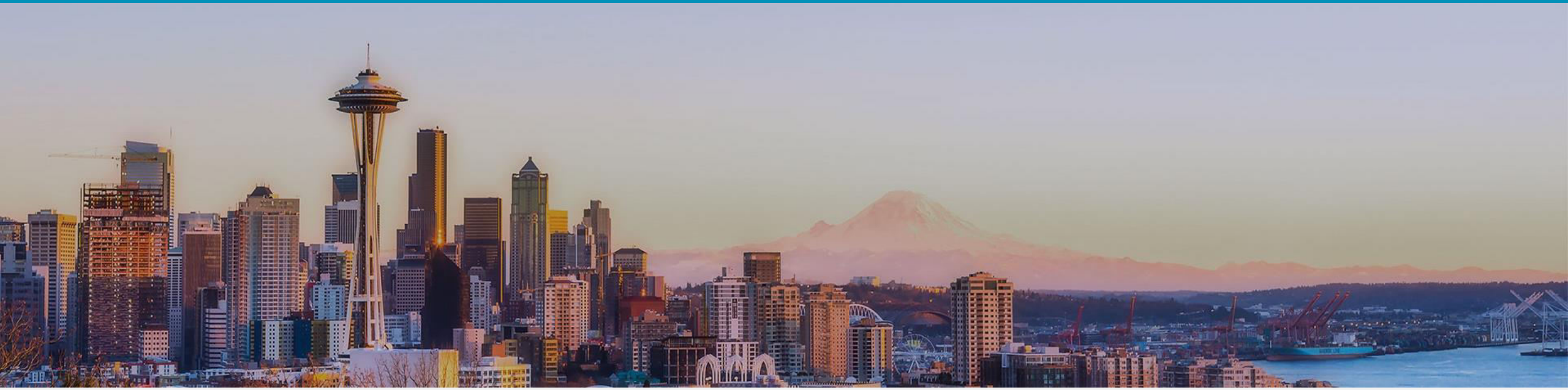
HL7® FHIR® DevDays 2019

HL7 FHIR

How and when to use FHIR Terminology Service APIs

Jim Steel



Redmond, June 10 – 12 | @HL7 @FirelyTeam | #fhirdevdays | www.devdays.com/us

Terminology: Why?

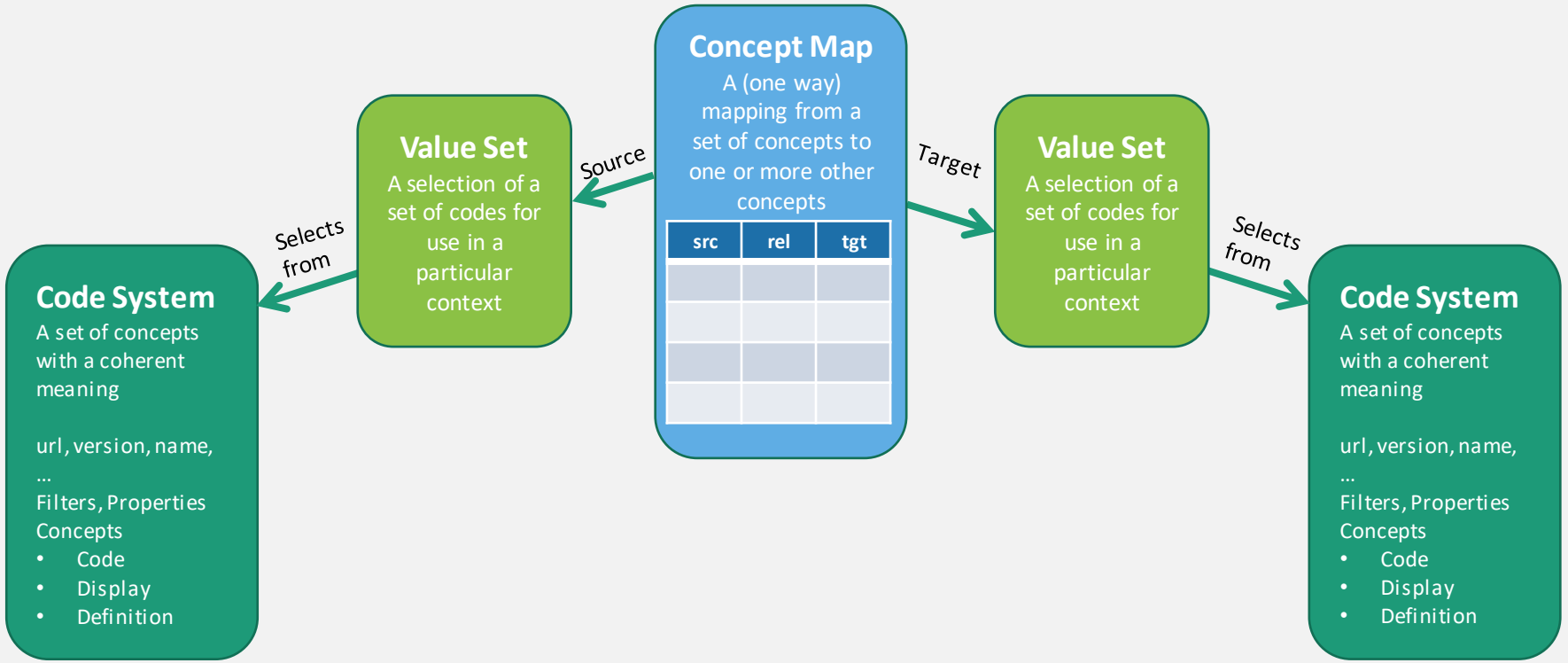
- Terminology is the basis of ***shared meaning*** between different systems
  - StructureDefs are the bones/branches, terminology is the meat/leaves/fruit
  - Terminology (especially ValueSets) is a very important part of defining profiles
- Especially in a large integrated system, having a single reference for terminology is very valuable for interoperability
- Externalized terminology makes it much easier to update terminology when it is updated/changes

How does terminology affect what I'm doing in FHIR?

- Data entry
  - Quickly find the right code to record in a specific context
- Validation
  - Is the code I've been given valid for this context?
- Display / understanding
  - Tell me about this code/set of codes (from this code system)
- Mapping
  - Find me a code in that set related to the code I have from this set
- Analytics
  - Arrange this coded data into categories so I can understand it
- Authoring
  - I need to make my own CodeSystem/ValueSet/ConceptMap

- Server: I want my server to be smart about terminology

FHIR Terminology Resources



FHIR Basics

SCRUD

- Search: GET /Resource?searchParams or POST /Resource/_search
- Create: POST /Resource
- Read: GET /Resource/id (or /Resource/id/_history/n for VRead)
- Update: PUT /Resource/id
- Delete: DELETE /Resource/id

Operations

- GET/POST /$operation or /Resource/$operation or /Resource/id/$operation
  - Any parameters go in a 'Parameters' resource in the request body
- GET only for idempotent (all can POST)
  - Can be cached using HTTP caching
- Server/type/instance-level depending on the operation

What if I am …

- Designing interfaces for data entry?
  - Choose your CodeSystem/s (ideally standardized ones!)
  - Choose or define your ValueSets
  - If your ValueSet is small, a picklist can be populated using $expand
  - If your ValueSet is large, a typeahead widget can use $expand?filter=xxx

Valueset operations: $expand

- Retrieve the expansion of the ValueSet subject to a number of parameters
  - Result is a ValueSet with an 'expansion' element
  - Parameters include: filter, count, offset, includeDesignations, includeDefinition, activeOnly, excludeNested, excludeNotForUI, excludePostCoordinated, displayLanguage, limitedExpansion, profile*
- Type or instance level, idempotent
  - GET/POST /ValueSet/$expand (must have 'url' or 'valueSet')
  - GET/POST /ValueSet/id/$expand

This is the main/best way to search for a code!

Example: $expand

GET /ValueSet/$expand?
url=http://hl7.org/fhir/ValueSet/administrative-gender

Accept: application/fhir+json


```
{
    "resourceType": "ValueSet",
    "language": "en",
    "url": "http://hl7.org/fhir/ValueSet/administrative-gender",
    "version": "4.0.0",
    "name": "AdministrativeGender",
    "expansion": {
        "identifier": "d71ae791-26d5-4cb3-9774-83235a17f99e",
        "timestamp": "2019-06-08T19:11:56+10:00",
        "total": 4,
        "parameter": [
            {
                "name": "version",
                "valueUri": "http://hl7.org/fhir/administrative-gender?version=4.0.0"
            },
            {
                "name": "count",
                "valueInteger": 2147483647
            },
            {
                "name": "offset",
                "valueInteger": 0
            }
        ],
        "contains": [
            {
                "system": "http://hl7.org/fhir/administrative-gender",
                "code": "male",
                "display": "Male"
            },
            {
                "system": "http://hl7.org/fhir/administrative-gender",
                "code": "female",
                "display": "Female"
            },
            {
                "system": "http://hl7.org/fhir/administrative-gender",
                "code": "other",
                "display": "Other"
            },
            {
                "system": "http://hl7.org/fhir/administrative-gender",
                "code": "unknown",
                "display": "Unknown"
            }
        ]
    }
}
```

Example: $expand with filter

- GET /ValueSet/$expand?
  url=http://loinc.org/vs
  &***filter***=urinalysis
  &***count***=10


```
{
    "resourceType": "ValueSet",
    "language": "en",
    "url": "http://loinc.org/2.64/vs",
    "name": "LOINC Value Set for All Codes",
    "status": "active",
    "expansion": {
        "identifier": "8b77a009-2480-4add-887b-d45a3626c584",
        "timestamp": "2019-06-08T19:15:08+10:00",
        "total": 52,
        "parameter": [
            {
                "name": "version",
                "valueUri": "http://loinc.org?version=2.64"
            },
            {
                "name": "count",
                "valueInteger": 3
            },
            {
                "name": "offset",
                "valueInteger": 0
            },
            {
                "name": "filter",
                "valueString": "urinalysis"
            }
        ],
        "contains": [
            {
                "system": "http://loinc.org",
                "code": "LP32744-2",
                "display": "Urinalysis"
            },
            {
                "system": "http://loinc.org",
                "code": "LP14150-4",
                "display": "Urinalysis panel"
            },
            {
                "system": "http://loinc.org",
                "code": "LP74376-2",
                "display": "Urinalysis studies"
            }
        ]
    }
}
```

$expand using context

- GET /ValueSet/$expand?***context***=Medication.status


```
{
    "resourceType": "ValueSet",
    "language": "en",
    "url": "http://hl7.org/fhir/ValueSet/medication-status|4.0.0",
    "version": "4.0.0",
    "name": "Medication Status Codes",
    "expansion": {
        "identifier": "e8e479c8-2aa8-41bc-bc72-5c88de700310",
        "timestamp": "2019-06-08T19:22:05+10:00",
        "total": 3,
        "parameter": [
            {
                "name": "version",
                "valueUri": "http://hl7.org/fhir/CodeSystem/medication-status?version=4.0.0"
            },
            {
                "name": "count",
                "valueInteger": 2147483647
            },
            {
                "name": "offset",
                "valueInteger": 0
            }
        ],
        "contains": [
            {
                "system": "http://hl7.org/fhir/CodeSystem/medication-status",
                "code": "entered-in-error",
                "display": "Entered in Error"
            },
            {
                "system": "http://hl7.org/fhir/CodeSystem/medication-status",
                "code": "active",
                "display": "Active"
            },
            {
                "system": "http://hl7.org/fhir/CodeSystem/medication-status",
                "code": "inactive",
                "display": "Inactive"
            }
        ]
    }
}
```

What if I am …

Analyzing/validating coded data?

- Choose your CodeSystem/ValueSets
- Use **$validate-code** to check whether the codes are valid in the context in which you're using them, and whether the display text is correct (many clinical systems allow users to override the display text for the term)
- Use **$translate** to map 'foreign' coded data into a normalized CodeSystem/ValueSet for analysis
- Use **$subsumes, $closure, or $validate-code** with an inline ValueSet, to categorize data, or **$translate** to map into categories

ValueSet operations: $validate-code

- Validate a code (and display text) against a ValueSet
  - Existing ValueSet (including implicit ValueSets) or POST one
  - Determine whether the code is included in the ValueSet
  - (optionally) Determine whether the provided display text is the correct display text for the code
  - Code as code/system/version, or coding, or CodeableConcept (multiple)
  - This is the main method for validating coded data!

Example: $validate-code

- GET /ValueSet/$validate-code?***url***=http://snomed.info/sct?fhir_vs&***system***=http://snomed.info/sct&***code***=38362002

```
{
    "resourceType": "Parameters",
    "parameter": [
        {
            "name": "result",
            "valueBoolean": true
        },
        {
            "name": "display",
            "valueString": "Dengue fever"
        }
    ]
}
```

Example: $validate-code with display

- GET /ValueSet/$validate-code?***url***=http://snomed.info/sct?fhir_vs&***system***=http://snomed.info/sct&***code***=13644009&***display***=hypercholesterolemia


```
{
    "resourceType": "Parameters",
    "parameter": [
        {
            "name": "result",
            "valueBoolean": false
        },
        {
            "name": "display",
            "valueString": "Hypercholesterolaemia"
        },
        {
            "name": "message",
            "valueString": "The code 13644009 exists in the ValueSet,
                but the display \"hypercholesterolemia\" is incorrect"
        }
    ]
}
```

What if I am …

- Exploring a set of concepts and how they relate to one another?
  - Use $lookup to retrieve their properties and display them in a table
  - Use the ‘child’ and ‘parent’ properties, or $subsumes/$closure, to explore the hierarchy that exists between concepts

CodeSystem operations: $lookup

- Retrieve details about a code (as code/system/version, or Coding)
- Can be used to determine whether a code exists in the CodeSystem
- Can be used to retrieve specific/all properties/designations

Example: $lookup

- GET /CodeSystem/$lookup?***system***=http://csiro.au/cs/au-jurisdictions&***code***=WA


```
{
    "resourceType": "Parameters",
    "parameter": [
        {
            "name": "name",
            "valueString": "Australian jurisdictions"
        },
        {
            "name": "version",
            "valueString": "0.0.1"
        },
        {
            "name": "display",
            "valueString": "Western Australia"
        },
        {
            "name": "designation",
            "part": [
                {
                    "name": "use",
                    "valueCoding": {
                        "system": "http://snomed.info/sct",
                        "code": "900000000000013009"
                    }
                },
                {
                    "name": "value",
                    "valueString": "West Australia"
                }
            ]
        },
        {
            "name": "property",
            "part": [
                {
                    "name": "code",
                    "valueCode": "parent"
                },
                {
                    "name": "value",
                    "valueCode": "AU-state"
                }
            ]
        }
    ]
}
```

CodeSystem operations: $subsumes

Check what (if any) subsumption relationship exists between two codes

- codeA and codeB, as code/system/version, or codingA and codingB
- Result will be ‘equivalent’, ‘subsumes’, ‘subsumed_by’, ‘not_subsumed’
- Depends on the code system’s ‘hierarchyMeaning’

Can also use $closure: see Rob Hausam’s session!

Example: $subsumes

GET /CodeSystem/$subsumes?
system=http://snomed.info/sct
&***codeA***=85562004
&***codeB***=302540006

```
{
    "resourceType": "Parameters",
    "parameter": [
        {
            "name": "outcome",
            "valueCode": "subsumes"
        }
    ]
}
```

What if I am …

- Mapping between codes in one system and codes in another
  - For example, mapping clinical codes to administrative codes for billing or reporting, or mapping legacy codes to replacement codes
  - Find or build a ConceptMap
  - Use $translate

ConceptMap operations: $translate

- Translate a code from one ValueSet to another, according to the server's resources, and/or other knowledge available to the server
- Typically, this means look up a ConceptMap resource and show the corresponding map entries (but not always!)
- Code as code/system/version, or coding, or CodeableConcept
- Specify source ValueSet, and target ValueSet (or targetSystem)
- Can be run in reverse (given target, show source)
- Results come back as Parameters with 'match' elements
- Match elements each have a relationship, e.g. 'equivalent', 'narrower', 'related-to'

Example: $translate

GET ConceptMap/$translate?
url=http://snomed.info/sct?fhir_cm
=900000000000526001
&***target***=http://snomed.info/sct?fhi
r_vs
&***system***=http://snomed.info/sct
&***code***=399144008


```
{
    "resourceType": "Parameters",
    "parameter": [
        {
            "name": "result",
            "valueBoolean": true
        },
        {
            "name": "match",
            "part": [
                {
                    "name": "equivalence",
                    "valueCode": "equivalent"
                },
                {
                    "name": "concept",
                    "valueCoding": {
                        "system": "http://snomed.info/sct",
                        "code": "1046151000168100",
                        "display": "Diabetes mellitus co-occurrent and due to haemochromatosis"
                    }
                },
                {
                    "name": "source",
                    "valueString": "http://snomed.info/sct/32506021000036107/version/20190531?fhir_c
                }
            ]
        }
    ]
}
```

What if I am …

Performing more advanced analytics with coded data

- Prepare more sophisticated, often dynamic, ValueSets using property or constraint filter
- Again use $validate-code, $subsumes or $closure
- Use a FHIR server with terminology support!

What if I am …

Authoring Terminology resources

- Creating a new UI, need to create ValueSets for data entry fields
- Editing profiles, need to create some new ValueSets for bindings
- Representing proprietary or local codes as a CodeSystem resource
- Creating a new ConceptMap for mapping local codes to a standard

CodeSystem resource

- Metadata
  - url, version
  - Name, title, identifier*, status, experimental, date, publisher, useContext, jurisdiction, description, copyright
  - CaseSensitive, compositional, hierarchyMeaning, versionNeeded, content, supplements*, count
- Properties, filters


```
{
    "resourceType": "CodeSystem",
    "id": "au-jurisdictions",
    "url": "http://csiro.au/cs/au-jurisdictions",
    "identifier": [
        {
            "system": "urn:ietf:rfc:3986",
            "value": "http://csiro.au/cs/au-jurisdictions"
        }
    ],
    "version": "0.0.2",
    "name": "Australian jurisdictions",
    "status": "draft",
    "experimental": true,
    "caseSensitive": false,
    "valueSet": "http://csiro.au/vs/au-jurisdictions",
    "hierarchyMeaning": "is-a",
    "versionNeeded": false,
    "content": "complete",
    "property": [
        {
            "code": "capital",
            "uri": "http://csiro.au/cs/jurisdiction-capital",
            "description": "Capital city of the jurisdiction",
            "type": "string"
        },
        {
            "code": "neighbour",
            "uri": "http://csiro.au/cs/jurisdiction-neighbour",
            "description": "Neighbouring jurisdiction",
            "type": "code"
        }
    ],
```

CodeSystem Resource

- Concepts
  - Hierarchy
  - Display
  - Designations
  - Properties


```
"concept": [
    {
        "code": "AU",
        "display": "Australia",
        "definition": "Australian jurisdiction",
        "concept": [
            {
                "code": "AU-state",
                "display": "Australian state",
                "definition": "Australian state jurisdiction",
                "concept": [
                    {
                        "code": "WA",
                        "display": "Western Australia",
                        "definition": "Western Australia jurisdiction",
                        "designation": [
                            {
                                "use": {
                                    "system": "http://snomed.info/sct",
                                    "code": "900000000000013009"
                                },
                                "value": "West Australia"
                            }
                        ],
                        "property": [
                            {
                                "code": "capital",
                                "valueString": "Perth"
                            },
                            {
                                "code": "neighbour",
                                "valueCode": "NT"
                            },
                            {
                                "code": "neighbour",
                                "valueCode": "SA"
                            }
                        ]
                    },
```

ValueSet resource

- Metadata
  - Url, version
  - Lots of the same ones as CodeSystem
- Compose
  - Includes and excludes
  - Whole CodeSystems
  - Lists of codes
  - Filters
  - Combinations

```
"compose": {
    "include": [
        {
            "system": "http://csiro.au/cs/au-jurisdictions",
            "filter": [
                {
                    "property": "concept",
                    "op": "descendent-of",
                    "value": "AU-state"
                }
            ]
        }
    ],
    "exclude": [
        {
            "system": "http://csiro.au/cs/au-jurisdictions",
            "concept": [
                {
                    "code": "TAS"
                }
            ]
        }
    ]
}
```

ConceptMap resource

- url, version
- Metadata (same as CS, VS)
- Groups
  - Shorthands for system/version
  - Mappings between codes
  - Equivalence relationships, e.g. equivalent, narrower, related-to


```
"sourceUri": "http://csiro.au/vs/australian-mainland-states",
"targetUri": "http://snomed.info/sct?fhir_vs",
"group": [
    {
        "source": "http://csiro.au/cs/au-jurisdictions",
        "target": "http://snomed.info/sct",
        "element": [
            {
                "code": "QLD",
                "target": [
                    {
                        "code": "223778006",
                        "equivalence": "equivalent"
                    }
                ]
            },
            {
                "code": "WA",
                "target": [
                    {
                        "code": "223782008",
                        "equivalence": "equivalent"
                    }
                ]
            },
```

What if I am …

Building a FHIR server that uses search or validate

Using a FHIR server that supports terminology-aware search or validation

Servers can support terminology-aware search such as code:below or code:in

- /Condition?code:below=73211009 (all descendants of diabetes)
- /Procedure?code:in=http://acme.com/vs/elective-procedures

Servers and externalized terminology

- Call the operations I've described (also $closure)
  - Easy, probably not performant
- Build virtual tables (or equivalent) in your database
  - Database handles caching
- Manage local caches of ValueSet membership
  - Efficient, doesn't deal with dynamic ValueSets

Things I haven't covered here

- SNOMED details
  - See Rory's talk
- LOINC details
  - See Dan's talk
- $closure
  - See Rob's talk
- CodeSystem supplements
- $validate

Useful tools

https://ontoserver.csiro.au/shrimp

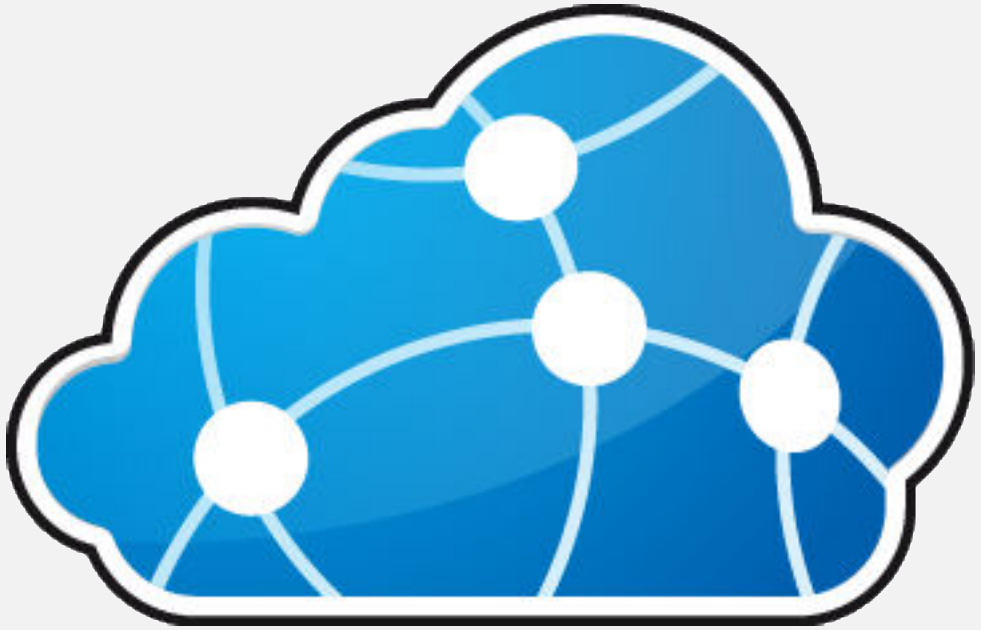
- Terminology browser

https://ontoserver.csiro.au/snapper2

- Terminology authoring

https://ontoserver.csiro.au/

- Terminology server



Thank you!

Ask me questions!

(or send them to me at Jim.Steel@csiro.au)