DevDays HL7® FHIR®

Multiple FHIR Versions

Grahame Grieve

November 20-22, Amsterdam | @HL7 @FirelyTeam | #fhirdevdays | www.devdays.com

Multiple releases of FHIR

- We make multiple releases of FHIR (working on R5 now)
- Over time, we change things – make breaking changes
- It would be better if we got it right first time
- But it's better to fix things when we don't
- Maturity rating reflects our process – change slows down over time
- No changes to normative content (some in R4)
- In the mean time, we support multiple versions…

Full release history

- http://hl7.org/fhir/directory.html
- Or http://hl7.org/fhir/package-list.json

Major Milestones:

Dec 27, 2018	4.0.0	Release 4 (1st Normative Content + Trial Use Developments)
Feb 21, 2017	3.0.0	Release 3 (STU - Standard for Trial Use)
Oct 24, 2015	1.0.0	DSTU2 (Second Draft Standard for Trial Use)
Sept 30, 2014	0.0.82	DSTU1 (First Draft Standard for Trial Use)

[canonical]/package-list.json


```
{
  "package-id" : "hl7.fhir.core",
  "title" : "FHIR Specification",
  "canonical" : "http://hl7.org/fhir",
  "introduction" : "This table provides a list of all the versions of FHIR (Fast
  "footnote" : "Note: Subsequent to Sept 2013, the FHIR version policy was change
  "list" : [ {
    "version" : "current",
    "date" : "n/a",
    "desc" : "Current Development build (about 30min behind version control, may
    "path" : "http://build.fhir.org",
    "status" : "ci-build",
    "current" : true
  },{
    "version" : "4.0.0",
    "date" : "2018-12-27",
    "desc" : "FHIR Release #4: First Normative Content",
    "path" : "http://hl7.org/fhir/R4",
    "status" : "normative+trial-use",
    "sequence" : "R4",
    "current" : true
  },{
```

Versioning is expensive

- Historically, version changes have been very expensive
  - Or, profitable for some, but bad for health
- Much argument about everything to do with versions….

Supporting Multiple Versions

- Converting between versions
- API Strategy
- Persistence Strategy
- Documentation Strategy

Converting between versions

- R3 Diff: http://hl7.org/fhir/diff.html

Changes since R3

Observation	
Observation.partOf	• Type Reference: Added Target Type MedicationUsage • Type Reference: Removed Target Type MedicationStatement
Observation.status	• Change value set from http://hl7.org/fhir/ValueSet/observation-status|4.0.0 to http:/

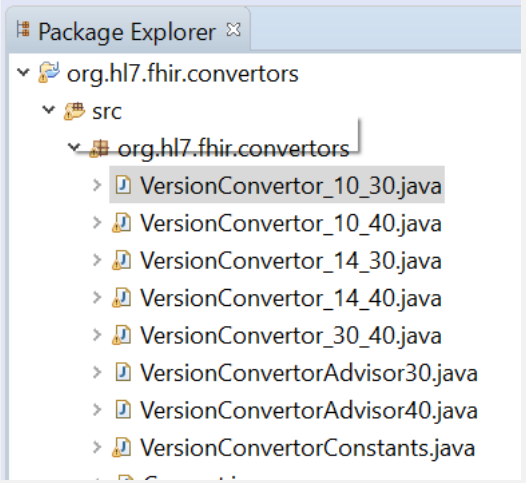
See the Full Difference for further information

Converting between versions

- R3/R4 Transform:

Resource	# Tests	% Execute OK	% RoundTrip Ok	% R4 Valid	R4 Error Count
Account	2	100	0	100	
ActivityDefinition	9	100	77	0	
AdverseEvent	1	100	0	0	
AllergyIntolerance	3	100	100	100	
Appointment	3	100	100	100	
AppointmentResponse	2	100	100	100	
AuditEvent	8	100	100	100	
Basic	3	100	100	100	
Binary	2	100	100	100	
BiologicallyDerivedProduct	No r2:r3 maps available				

Java Converter




```
21904
21905
21906  public static org.hl7.fhir.r4.model.Resource convertResource(org.hl7.fhir.dstu3.model.Resource src, boo
21907    if (src == null)
21908      return null;
21909    if (src instanceof org.hl7.fhir.dstu3.model.Parameters)
21910      return convertParameters((org.hl7.fhir.dstu3.model.Parameters) src);
21911    if (src instanceof org.hl7.fhir.dstu3.model.ActivityDefinition)
21912      return convertActivityDefinition((org.hl7.fhir.dstu3.model.ActivityDefinition) src);
```

- Only supported fully for conformance resources
- Contributions for other resources are welcome

Version independent logic

- Use a façade in front of versions e.g.

```
IPatient = interface (IDomainResource) {
  IHumanName getNameI();
}

R3.Patient = class (DomainResource, IPatient) {
 public HumanName getName() {…}
 public IHumanName getNameI() {…}
}
```

- This is a lot of work, but partially done in some reference implementations – you can contribute

API Versioning Strategy

Resource Conversion isn't everything

- GET [base]/Patient/[id]?(params)
  Accept: [content-type]

- HTTP 200 OK
  Content-Type: [content-type]

  { … body …}

- The entire exchange has a version (can't mix with one exception)

Simplest Approach: multiple end-points

- http://test.fhir.org/r2
- http://test.fhir.org/r3
- http://test.fhir.org/r4

- fhirVersion element in the applicable CapabilityStatement applies
- Pro: Simple
- Con: Logical records get multiple URLs

Single end-point, multiple versions

- http://test.fhir.org/rX
- The fhirVersion parameter on the MIME-type that applies to the resource (but fixes the whole exchange)

```
GET [base]/metadata
Accept: application/fhir+json; fhirVersion=4.0
```

Determining Server Versions

```
GET [base]/$versions
Accept: application/fhir+json
[other headers]
```

```
"resourceType": "Parameters",
"parameter": [{
  "name": "version",
  "valueString": "3.0"
 }, {
  "name": "version",
  "valueString": "4.0"
 }, {
  "name": "default",
  "valueString": "4.0"
 }]
```

Determining Server Versions

```
GET [base]/$versions
Accept: application/json
[other headers]
```

```
{
  "versions": ["3.0", "4.0"],
  "default" : "4.0"
}
```

Single end-point, multiple versions

- Server specifies what versions it supports, with a default
- Client chooses a version using the fhirVersion parameter
- Fixes the whole exchange

- Conversion information for resource names and search parameters: https://github.com/FHIR/interversion/tree/master/package

$convert

- Ask server to convert versions

```
POST /base/$convert
Accept: application/fhir+json; fhirVersion=3.0
Content-Type: application/fhir+json; fhirVersion=4.0
```

Non-API Exchange

- There's almost always a mime type:

```
application/fhir+json; fhirVersion=4.0
```

- If that's not possible:

```
"meta" : {
  "profile" : ["http://hl7.org/fhir/4.0/StructureDefinition/Patient"]
}
```

Persistence Strategy

Persisting Multiple versions

- Store Resources with known version (implicit, or explicit)
- Use the profile marker if you really need to

Persistence and Conversion

In general 3 options:

- Store resources as you get them (and convert on the fly if needed)
- Store resources in your preferred version (and convert if needed)
- Extract information from resources and store in (relational?) database

Or… Do all 3 things at once:

- Store resources as you first received them (for audit trail)
- Store resources in your preferred version (for flexibility)
- Build specific tables for particular indexing (for performance)

Documentation Strategy

- Simple: Different documentation for different versions
  - Multiple repetitions of narrative
  - Implementers have to compare between versions
- Complex: One set of documentation, with different profiles/examples
  - One combined narrative with explicit differentiation
  - Implementers explicitly deal in multiple versions

- Which is better depends on the implementers
- Do the business rules differ? What about documentation versions?

Managing Multiple Versions

- Versions are expensive and painful
- There's some fantasy land where they don't happen
- Questions…