1. <http://www.healthintersections.com.au/>

**Question**:

How to extend severity in AllergyIntolerance resource.  Is the following resource where severity is extended done correctly?

<AllergyIntolerance xmlns="http://hl7.org/fhir">

<id value="uWErIN0G0" />

<recordedDate value="05/10/1980" />

<patient>

<reference value="Patient/DFQA\_ghn000003" />

<display value="Newman, Alice Jones" />

</patient>

<substance>

<coding>

<system value="RxNorm" />

<code value="1659592" />

</coding>

<text value="ampicillin-sulbactam" />

</substance>

<status value="confirmed" />

<event>

<manifestation>

<coding />

<text value="urticaria (hives)" />

</manifestation>

<severity>

<extension url="http://phinvads.cdc.gov/vads/ViewValueSet.action?oid=2.16.840.1.113883.3.88.12.3221.6.8">

<valueCodeableConcept>

<coding>

<system value="" />

<code value="371923003" />

<display value="mild to moderate" />

</coding>

</valueCodeableConcept>

</extension>

</severity>

</event>

</AllergyIntolerance>

**Answer:**

Severity is a code, with cardinality 0..1 and a required binding to mild | moderate | severe. You can replace the code with an extension, as is done here, so that part is valid. But it’s much better to provide both a code out of the defined choices, so that systems that only know those choices can process the information correctly, while those that know the refined list can process the refined list:

Severity是一个代码，其基数为0..1，需要绑定到mild | moderate | severe。 您可以使用扩展名替换代码，如此处所做，以使该部分有效。 但是，从定义的选择中提供一个代码是更好的，所以只知道这些选择的系统可以正确处理信息，而那些知道精简列表的系统可以处理精简列表：

<severity value="mild">

<extension url="http://phinvads.cdc.gov/vads/ViewValueSet.action?oid=2.16.840.1.113883.3.88.12.3221.6.8">

<valueCodeableConcept>

<coding>

<system value="" />

<code value="371923003" />

<display value="mild to moderate" />

</coding>

</valueCodeableConcept>

</extension>

</severity>

Aside: you could map this code to either mild or moderate, yes. Choosing either is better than choosing neither.

另外,可以将code映射到mild或者moderate。只要选择就比不选择好。

There’s several other issues here, though:

* The URL given that identifies the extension actually identifies the value set, not the extension (aka the data element). The correct thing to do would be to define an extension that use the value set (see at the bottom of the post)
* According to the value set definition, this is a SNOMED CT code, and the system should be filled out
* the recorded date format is not a valid XML date
* the system for RxNorm is not correct
* manifestation coding can’t be empty

但是，这里还有一些其他的问题：

标识扩展的URL实际上标识了值集，而不是扩展名（也称为数据元素）。正确的做法是定义一个使用值集的扩展名。

根据值集定义，这是一个SNOMED CT 编码，应该填充system值。

记录的日期格式不是有效的XML日期

RxNorm的system不是正确的。

表现的coding值不能为空

Here it is corrected:

<AllergyIntolerance xmlns="http://hl7.org/fhir">

<id value="uWErIN0G0" />

<recordedDate value="1980-10-05" />

<patient>

<reference value="Patient/DFQA\_ghn000003" />

<display value="Newman, Alice Jones" />

</patient>

<substance>

<coding>

<system value="http://www.nlm.nih.gov/research/umls/rxnorm" />

<code value="1659592" />

</coding>

<text value="ampicillin-sulbactam" />

</substance>

<status value="confirmed" />

<event>

<manifestation>

<text value="urticaria (hives)" />

</manifestation>

<severity>

<extension url="http://example.org/fhir/StructureDefinition/my-extension">

<valueCodeableCoding>

<system value="http://snomed.info/sct" />

<code value="371923003" />

<display value="mild to moderate" />

</valueCodeableCoding>

</extension>

</severity>

</event>

</AllergyIntolerance>

With an associated Extension definition:相关的扩展定义

<?xml version="1.0" encoding="utf-8"?>

<StructureDefinition xmlns="http://hl7.org/fhir">

<meta>

<lastUpdated value="2017-07-12T07:05:27.311+10:00" />

</meta>

<url value="http://example.org/fhir/StructureDefinition/my-extension" />

<name value="RefinedSeverity" />

<status value="draft" />

<date value="2017-07-12T07:02:31.9614187+10:00" />

<publisher value="Health Intersections" />

<description value="My definition for FHIR extension" />

<purpose value="an extension" />

<fhirVersion value="3.0.1" />

<kind value="complex-type" />

<abstract value="false" />

<contextType value="resource"/>

<context value="AllergyIntolerance.event.severity"/>

<type value="Extension" />

<baseDefinition value="http://hl7.org/fhir/StructureDefinition/Extension" />

<derivation value="constraint" />

<differential>

<element id="Extension">

<path value="Extension" />

<definition value="This is a description of the level of the severity of the problem." />

</element>

<element id="Extension.url">

<path value="Extension.url" />

<fixedUri value="http://example.org/fhir/StructureDefinition/my-extension" />

</element>

<element id="Extension.value[x]:valueCoding">

<path value="Extension.valueCoding" />

<sliceName value="Severity" />

<type>

<code value="Coding" />

</type>

<binding>

<strength value="required" />

<description value="This is a description of the level of the severity of the problem." />

<valueSetUri value="http://phinvads.cdc.gov/vads/ViewValueSet.action?oid=2.16.840.1.113883.3.88.12.3221.6.8" />

</binding>

</element>

</differential>

</StructureDefinition>

Categories: [FHIR](http://www.healthintersections.com.au/?cat=35), [Question](http://www.healthintersections.com.au/?cat=24)

## [Report from joint #openEHR / #FHIR meeting](http://www.healthintersections.com.au/?p=2648)

Posted on June 15, 2017 by [Grahame Grieve](http://www.healthintersections.com.au/?author=1)

[4 Comments](http://www.healthintersections.com.au/?p=2648#comments)

Last month, the openEHR and FHIR communities met for a day in Norway. From the openEHR community, Ian McNicoll and Silje Ljosland Bakke were present, and from the FHIR community, Ewout Kramer and myself were present, along with a group of Norwegians who are involved in FHIR and/or openEHR. My thanks to [HL7 Norway](https://www.hl7.no/), [DIPS](https://www.dips.no/) ([en](https://www.dips.com/uk)) and the [Direktoratet for e-helse](http://ehelse.no/) for collaborating to bring the meeting together.

上个月，进行会议讨论。

The overall agenda for the day:

* Welcome, introductions / complaints about the weather (de rigueur, but Norway was lovely, there was even some sun one day)
* Opening presentations from Ian and myself
* Discussion on specific model mapping HL7 FHIR – openEHR
* Discussion on a general form solutions
* Some general questions on HL7 FHIR – openEHR: Mapping, query, Workflow, Pub/Sub, Terminologies, Modeling

会议议程为：

讨论具体的模型映射

讨论一般形式的解决方案

关于HL7 FHIR的一般问题：openEHR、映射、查询、工作流、Pub/Sub、术语、建模。

And then drinks afterwards, where we solved all the worlds problems (except for a few big ones).

I enjoyed the day greatly – there was a real positive spirit, based on a recognition that both communities are vibrant productive communities that have much to offer each other. Technically, the FHIR and openEHR communities have different scopes, with some overlap, and there’s plenty of real world problems that FHIR is not attempting to solve that openEHR has invested in – and vice versa. Given good will, and a focus on pragmatic outcomes, we can work well together and both derive benefit from that collaboration.

在技​​术上，FHIR和openEHR社区有不同的范围，有一些重叠，FHIR没有试图解决openEHR所投入的大量真实世界的问题，反之亦然。

Given that, most of the day focused on immediate challenges faced by Norwegian implementers today. The most obvious question is ‘I have content described by an archetype, and I want to exchange that using FHIR. How do I go about that?’  Right now, there’s no clean simple answer. There’s several different paths, which work variably well. We discussed some of the issues with a focus on mapping between openEHR and HL7 FHIR of observations like body temperature. The outcome of the discussion is that the place to focus is mapping openEHR templates to/from FHIR Profiles. We’ve created a [joint workspace](https://github.com/openEHR/fhir-guide) in the openEHR Github repository so we can work together on that (as of today, it’s empty. It’s work in progress and will gradually start to fill up, and hopefully with useful stuff).

我们讨论了一些重点关注openEHR和HL7 FHIR之间的映射的一些问题，如体温等观察。讨论的结果是，重点放在的地方是将openEHR模板映射到/从FHIR配置文件。

The plan is that we’ll automatically convert openEHR archetypes and templates to FHIR Logical models, and then map to FHIR resources using the FHIR Mapping language, and generate profiles and implementation guides from that. If that works well (not yet known), this would establish a single demonstrated path to interoperability. Note that I didn’t say a simple path – to make it simple will require further content reconciliation between the communities – perhaps this will provide the impetus for more investment in that (we’ve done some, but it’s hard work).

该计划是，我们将自动将openEHR原型和模板转​​换为FHIR逻辑模型，然后使用FHIR Mapping语言映射到FHIR资源，并从中生成配置文件和实施指南。如果这个工作很好（尚不知道），这将建立一个单一的演示路径，以实现互操作性。

Then we discussed the different and similarities of the FHIR and openEHR approaches to forms/questionnaires, and what would be involved to interconvert between the different formats. We agreed to work on this further, including a set of activities on Form Mapping at the [FHIR DevDays meeting](https://www.fhirdevdays.com/) in Amsterdam in November.

我们讨论了FHIR和openEHR方法对表单/调查问卷的不同和相似之处，以及不同格式之间的相互转换。我们同意在11月份在阿姆斯特丹举行的[FHIR DevDays会议](https://www.fhirdevdays.com/)上进一步开展工作，包括在Form Mapping上的一系列活动。

One idea that has been floating around recently is for openEHR to use the FHIR binding syntax and terminology services – we discussed this a little, and hopefully we can work towards a concrete proposal for this in the next few months to see how it flies in the wider openEHR community.

Overall, the meeting created a lot of prospects about what we can do together, but now we actually have to follow up and do them.

总的来说，这次会议为我们可以一起做的事情创造了很多前景，但现在我们实际上要跟进。

p.s. Thanks to Sigurd From from DIPS for keeping notes from the meeting which I used for this report

p.s. Also, thanks to the Norwegians for being great hosts. In addition to [Norwegian national day](https://www.visitnorway.com/about/history-traditions/national-day/), I [enjoyed the fjords](http://nordicventures.com/) greatly. Now I just have to find the right supplier of Fjord Cider here in Australia, and all will be good.

Categories: [FHIR](http://www.healthintersections.com.au/?cat=35), [openEHR](http://www.healthintersections.com.au/?cat=54)

## [Interconversion between #FHIR and HL7 v2](http://www.healthintersections.com.au/?p=2646)

Posted on June 12, 2017 by [Grahame Grieve](http://www.healthintersections.com.au/?author=1)

[2 Comments](http://www.healthintersections.com.au/?p=2646#comments)

**Question**

What advice do you give for introducing FHIR in new software, while continuing to maintain HL7v2 interoperability with client applications that do not speak FHIR?

在一个新的软件中引入FHIR应该给出什么建议，不使用FHIR客户端的应用程序怎样继续进行HL7 V2的互操作性。

For example, are FHIR resources the way to go as an internal representation of an application’s health care data? 例如：FHIR资源是否是应用程序的医疗保健数据的内部表示方式？

If yes, is it practical to convert HL7 messages into FHIR resources (e.g. Patient, Practitioner, ProcedureRequest, ReferralRequest, Appointment…)? What open source software do you recommend for converting HL7 messages into FHIR resources (and vice-versa)?

Or is it better to use FHIR for external information exchange only (with outside FHIR clients)?

如果是，可以将HL7消息转换为FHIR资源（例如：Patient, Practitioner, ProcedureRequest, ReferralRequest, Appointment）吗？推荐一款软件可以将HL7消息转化为FHIR资源。或者是否可以更好地使用FHIR进行外部信息交换？

**Answer**

I’ve worked with several projects rebuilding their products around FHIR resources. Like all development methodologies, this has pros and cons. What you get out of the box is a highly interoperable system, and you get a lot of stuff for free. But when your requirements go beyond what’s in the specification, it starts to get hard – FHIR is an interoperability standard, that focuses on the lowest common denominator: what everyone agrees with. Whether that’s a net benefit depends on how far beyond common agreement your going to go. (This, btw, is a demonstration of [my 3rd law](http://www.healthintersections.com.au/?p=55) of interoperability).

已经与多个项目合作，围绕FHIR资源重建产品。就像所有的开发方法一样，这有利弊。开箱即用是一个高度互操作性的系统，可以免费获取很多东西。但是当超出规范中的请求的时候，这就变得比较困难。FHIR是一个互操作性标准，侧重于最低的共同点（每个人都同意）。是否是净收益取决于你使用的共同协议。

It is practical to convert HL7 messages to FHIR resources and vice versa, yes. We’ve seen plenty of that going on. But there’s no canned solution, because to do the conversion, you have to do two things:

* Figure out all the arcane business logic and information variants and code this into the conversion
* Figure out how to integrate your conversion logic into your application framework

可以将HL7消息转化为FHIR资源，反之亦言。因为要进行转换，必须做到：

找出所有神秘的业务逻辑和信息变体，并将code进行转化。

了解如何将逻辑转化为应用程序框架的整合方案。

The upshot of this is that you have a programming problem, and most people solve this by taking a open source libraries for v2 and FHIR in the language of their choice (most languages have one of those) and writing the business logic and application integration in their development language of choice. Hence, there’s no particular open source library to do the job other than the parsers etc. There are some commercial middleware engines that include FHIR as one of the formats that can be supported.

这样做的结果是涉及到一个编程问题，大多数通过V2和FHIR的开源库解决这个问题，将业务逻辑和应用程序集成在可选的开发语言中。因此，除了解析器之外，没有特定的开源库来解决这个问题。其中一个中间件引擎将FHIR作为可支持的格式。

In the FHIR spec, we’ve defined a mapping language that tries to abstract this – so you can separate the business logic from the application integration, and a platform independent business logic that has libraries for whatever platform. That’s an idea that is gradually starting to gather some interest, but is still a long way from maturity.

在FHIR规范中，我们定义了一种试图抽象出来的映射语言，因此可以将业务逻辑与应用程序集成分离开来，并将平台独立业务逻辑与任何平台的库进行分离。这是一个感兴趣的想法，但是远远不成熟。

With regard to using FHIR for external exchange only… what I usually say about this that is that it makes sense to implement FHIR for new things first, and then to replace old things only when they become a problem. And most new stuff is on the periphery, where the architectural advantages to FHIR are really big. But internally, v2 will increasingly become a major service limitation in time, and will have to be replaced. The open question is how long that timeline is. We don’t know yet.

Categories: [FHIR](http://www.healthintersections.com.au/?cat=35), [Interoperability](http://www.healthintersections.com.au/?cat=49), [Question](http://www.healthintersections.com.au/?cat=24), [v2](http://www.healthintersections.com.au/?cat=8)

## [#FHIR Report from Madrid Working Group Meeting](http://www.healthintersections.com.au/?p=2643)

Posted on May 19, 2017 by [Grahame Grieve](http://www.healthintersections.com.au/?author=1)

[No Comments](http://www.healthintersections.com.au/?p=2643#respond)

see my post on the official FHIR Product director blog: [#FHIR Report from Madrid Working Group Meeting](https://onfhir.hl7.org/2017/05/19/fhir-report-from-madrid-working-group-meeting/)

Categories: [FHIR](http://www.healthintersections.com.au/?cat=35), [Standards](http://www.healthintersections.com.au/?cat=11)

## [FHIR Product Priorities for Release 4](http://www.healthintersections.com.au/?p=2630)

Posted on May 4, 2017 by [Grahame Grieve](http://www.healthintersections.com.au/?author=1)

[No Comments](http://www.healthintersections.com.au/?p=2630#respond)

Now that we’ve published Release 3 of FHIR, it’s time for us to consider our main priorities for the next FHIR release. This is my draft list of product priorities that we’ll be discussing – and trying to execute – at the Madrid meeting next week:

现在已经发布了FHIR第三版本，现在要考虑下一个FHIR版本的优先事项。

* Normative: push to normative for
  + Foundation / API / XML / JSON / Bundle / OperationOutcome
  + Terminology Service (ValueSet / CodeSystem / ExpansionProfile)
  + StructureDefinition / CapabilityStatement
  + Patient / RelatedPerson / Practitioner / Organization / ?Endpoint
* Position a core set of clinical resources (‘health base’?) for normative in R5 (or Observation | AllergyIntolerance | MedicationStatement normative for R4?)
* JSON: ? use manifest for extensions, parameters resource ([see blog post](http://www.healthintersections.com.au/?p=2626)) (note that discussion on this didn’t go very well – probably will be dropped)
* RDF: more ontology bindings + resolve status of JSON-LD
* Data Analytics: support for a bulk data analysis bridge format (Apache Parquet?)
* API: better control over retrieving graphs, and value added query support (tabular format?)
* Patterns: change the W5 framework to a pattern (logical model), tie the patterns to ontology, and use of patterns to drive more consistency (and how to do this without decreasing quality)
* Services: more services. Candidates: conformance, registry, personal health summary?, etc?
* Deployment: get a clear standards path for smart on fhir / cds-hooks (and alignment with UMA/Heart)
* FM: work on alignment between FM resources and the rest of FHIR

规范性：推动规范化。Foundation / API / XML / JSON / Bundle / OperationOutcome；Terminology Service (ValueSet / CodeSystem / ExpansionProfile)；StructureDefinition / CapabilityStatement；Patient / RelatedPerson / Practitioner / Organization / ?Endpoint。

在R5（或规范4中的Observation | AllergyIntolerance | MedicationStatement）中定义一套核心的临床资源（或健康资源）。

JSON：?使用manifest扩展名，Parameters资源。（这个问题没有进行很好的讨论）

RDF：更多的本体绑定+解析JSON-LD的状态

数据分析：支持批量数据分析

API：更好地控制检索图形和增值查询支持

Patterns：将W5框架更改为模式（逻辑模型），将模式与本体结合起来，并使用模式来提高更一致性（以及如何在不降低质量的情况下实现）。

Service：更多的服务。Candidates: conformance, registry, personal health summary等等。

Deployment：在fhir / cds-hooks上获得明确的标准路径（与UMA / Heart对齐）

FM：在FM资源与FHIR的其余部分之间进行协调。

Note that this list is written anticipating that the normal standards development process occur, and the content as a whole is maintained. I’d expect that this would amount to 1000s of tasks. So this list is not a list of ‘what will change in R4’, but an indication of where particular focus will be applied by the FHIR leadership (so don’t be concerned if a particular issue of yours is not on this list, as long as it’s in [gForge](http://gforge.hl7.org/gf/project/fhir/tracker/?action=TrackerItemBrowse&tracker_id=677))

请注意，此列表的预期是正常的标准开发过程发生，整个内容都得到维护。我希望这将达到1000多个任务。所以这个列表不是“R4中会有什么变化”的列表，而是指示FHIR领导层应该特别关注的重点。

Categories: [FHIR](http://www.healthintersections.com.au/?cat=35), [Standards](http://www.healthintersections.com.au/?cat=11)

## [Proposal for #FHIR JSON format change: @manifest](http://www.healthintersections.com.au/?p=2626)

Posted on April 20, 2017 by [Grahame Grieve](http://www.healthintersections.com.au/?author=1)

[6 Comments](http://www.healthintersections.com.au/?p=2626#comments)

There’s a long running discussion in the FHIR community about the way the JSON format handles extensions, and operation invocations (“Parameters”) resource.  Various implementers keep proposing format changes to the JSON format around extensions, but the last time we made an attempt to change this, it was roundly quashed at ballot.

FHIR社区中有关JSON格式处理扩展的方式以及操作调用（“参数”）资源的长时间讨论。各种实施者不断提出对扩展名的JSON格式进行格式更改。

The underlying problem is that there’s 2 different (though overlapping) communities that use the JSON format for FHIR:

* the interoperability community, who value consistency and robustness
* the app writing community who value conciseness much more

基本的问题是，使用FHIR的JSON格式有2个不同（虽然重叠）的社区：

互操作性社区：一致性和鲁棒性。

简洁的应用程序写作社区。

From the perspective of the second community, the current JSON format doesn’t work well for representing either extensions, or the parameters of an operation. With this in mind, and drawing on the practices of the [JSON-LD](http://json-ld.org/) community, I’d like to advance a proposal for a **manifest** approach to extensions and parameters in the FHIR JSON format.

从第二个社区的角度来看，目前的JSON格式对于表示扩展或操作的参数来说并不奏效。考虑到这一点，并借鉴了[JSON-LD](http://json-ld.org/)  社区的实践，我想提出一种使用FHIR JSON格式的扩展和参数的**manifest** 方法的提案。

The way this would work is that we start with the existing format, and add a “@manifest” property, which contains information about how extensions and parameters have been represented in the json format. Applications reading the JSON format can either read the properties directly, based on their implicit knowledge of the manifest, or read the manifest and process accordingly.

这样做的方式是，我们从现有的格式开始，并添加一个“@manifest”属性，其中包含有关如何以json格式表示扩展名和参数的信息。读取JSON格式的应用程序可以直接读取属性，基于它们对清单的隐含知识，或者相应地读取清单和过程。

As an example, consider this example Patient resource:

{

"resourceType": "Patient",

"id": "ex1",

"extension": [

{

"url": "http://example.org/StructureDefinition/trials",

"valueCode": "renal"

}

],

"active": true

}

This uses an extension following as specified in FHIR Release 3. The same resource rendered using a manifest might look like this:

使用FHIR V3中指定的扩展名，使用manifest渲染的相同资源可能如下表示：

{

"resourceType": "Patient",

"id": "ex1",

"@manifest" : {

"trials" : {

"extension" : "http://example.org/StructureDefinition/trials",

"type" : "code",

"list" : false

}

},

"trials": "renal",

"active": true

}

Note: It’s important to note that processing the JSON directly and ignoring the manifest is a convenient but fragile approach; changes in naming or type would be transparent to an application that processed via the manifest, but would likely break an application that processed using the ‘trials’ name directly. That doesn’t mean that applications should not do this; just that it should only be used where the client and server are fairly closely linked and managed.

注意：直接处理JSON并忽略manifest 是一种方便但是比较脆弱的方法。命名或类型的更改对于通过清单处理的应用程序将是透明的，但可能会直接破坏使用“试用”名称处理的应用程序。这并不意味着应用程序不应该这样做; 只是它应该只能用于客户端和服务器相互紧密联系和管理的位置。

Aside: I think of this as ‘interoperability’ vs ‘operability’. At heart, FHIR is a specification for an API between disparate systems with different designs and life cycles (and customers – see ‘[drive-by interoperability](http://www.healthintersections.com.au/?p=141)‘). But lots of people are using it as a client/server format for singly maintained applications (often because there’s no strong technical boundary between the internal and external use) – and it’s in that tightly managed context that the manifest approach brings the most benefit with a manageable risk.

另外：这是“互操作性”和“可操作性”之间的抗衡。核心内容，FHIR是不同设计和生命周期的不同系统之间的API规范。但是很多人将用作单独维护的应用程序的客户端/服务器格式（通常是因为内部和外部使用之间没有强大的技术边界），而且在紧密管理的上下文中，清单方法最有利于一个可控风险。

还可以将其分为两部分。

It’s also possible to take the manifest and move it out of band:

{

"resourceType": "Patient",

"id": "ex1",

"@manifest" : "http://healthintersections.com.au/patient.manifest.json",

"trials": "renal",

"active": true

}

And then, at http://healthintersections.com.au/patient.manifest.json:

{

"@manifest" : {

"trials" : {

"extension" : "http://example.org/StructureDefinition/trials",

"type" : "code",

"list" : false

}

}

}

Of course, if the manifest is not available at the nominated address, applications that use the manifest will not be able to process the instance correctly – if at all. So that’s an obvious risk that needs to be managed.

当然，如果清单在提名地址不可用，则使用清单的应用程序将无法正确处理实例 - 如果有的话。这是一个明显的风险，需要管理。

Readers familiar with JSON-LD will have seen the obvious similarities with JSON-LD’s @context. We’re not actually using ‘@context‘, though, because while what we are doing is structurally similar, we’re using it for a different purpose.

熟悉JSON-LD的读者​​将看到与JSON-LD的@context有明显的相似之处。我们实际上并没有使用' *@context* '，因为虽然我们在做的是结构上相似的，但是我们使用它是为了不同的目的。

You could use the same technique with regard to parameters on an operation. Take, for example, this input to the $expand operation: 可以对操作中的参数使用相同的技术。例如，输入的$ expand操作

{

"ResourceType" : "Parameters",

"parameter" : [

{

"name" : "coding",

"valueCodeableConcept" : {

"coding" : {

"system" : "http://loinc.org",

"code" : "1963-8",

"display" : "test"

}

}

},

{

"name" : "valueSet",

"resource": {

"resourceType" : "ValueSet",

[etc]

}

}

]

}

With an in-line manifest, this might look like this:

{

"ResourceType" : "Parameters",

"@manifest" : {

"code" : {

"parameter" : " http://hl7.org/fhir/OperationDefinition/ValueSet-validate-code#coding",

"type" : "Coding",

"list" : false

}

"vs" : {

"parameter" : " http://hl7.org/fhir/OperationDefinition/ValueSet-validate-code#valueSet",

"type" : "Resource",

"list" : false

}

}

"code" : {

"coding" : {

"system" : "http://loinc.org",

"code" : "1963-8",

"display" : "test"

}

},

"vs" : {

"resourceType" : "ValueSet",

[etc]

}

}

}

Or, we could refer to a manifest defined in the specification itself:

{

"ResourceType" : "Parameters",

"@manifest" : "http://hl7.org/fhir/r4/validation.manifest.json",

"code" : {

"coding" : {

"system" : "http://loinc.org",

"code" : "1963-8",

"display" : "test"

}

},

"vs" : {

"resourceType" : "ValueSet",

[etc]

}

}

}

Several Questions I’ve had from the few people who’ve looked at this idea already:

* Why not do this in XML too? Well, we could. But I don’t think it has value, because people using FHIR in tightly bound client/server type environments (where the @manifest approach is nost beneficical) are almost exclusively using JSON. So the cost/benefit is not there for XML. Also, in XML, schema validation matters more.
* What about JSON schema then? It’s possible to generate a JSON schema for this, if the generation tooling knows what the manifest is going to say. No such tooling exists right now, but it could be written. Or else someone could easily write a convert to convert from the @manifest form to the existing normal form.
* What about the reference implementations? They’d be enhanced to support this transparently on read, and there would be some kind of configuration to allow the user to control the manifest, and then it would write according to the manifest.
* Would this be instead of the existing approach? I propose that it’s an additional approach: the existing extension and parameter format is still valid, and can still be used, but implementations can use the @manifest if they want – and can mix and match. e.g. some extensions represented using @manifest, and others (not known in the manifest) represented the existing way

几个已经看过这个想法的人已经有了几个问题：

* 为什么不这样做在XML呢？那我们可以。但是我不认为它有价值，因为在紧密绑定的客户端/服务器类型环境（其中@manifest方法是最有益的）中使用FHIR的人几乎完全使用JSON。所以XML和XML的成本/收益是不一样的。而且，在XML中，模式验证更重要。
* 那么JSON模式呢？如果生成工具知道清单将要说什么，可以为此生成一个JSON模式。现在没有这样的工具，但可以写成。或者有人可以轻松地编写一个转换器，从@manifest表单转换为现有的正常格式。
* 参考实现怎么样？他们将得到增强，以便透明地支持读取，并且将有一些配置允许用户控制清单，然后根据清单进行写入。
* 这不是现在的做法吗？我建议这是一个额外的方法：现有的扩展和参数格式仍然有效，仍然可以使用，但实现可以使用@manifest，如果他们想要 - 并且可以混合和匹配。例如使用@manifest表示的一些扩展名，以及其他（在清单中未知）的代表现有方式

For follow up / discussion, see [fhir.chat.org](https://chat.fhir.org/#narrow/stream/implementers/subject/JSON.20Manifest.20Proposal), though comments here are also welcome.

Categories: [FHIR](http://www.healthintersections.com.au/?cat=35), [Standards](http://www.healthintersections.com.au/?cat=11)

## [#FHIR Testing is Coming](http://www.healthintersections.com.au/?p=2622)

Posted on April 20, 2017 by [Grahame Grieve](http://www.healthintersections.com.au/?author=1)

[1 Comment](http://www.healthintersections.com.au/?p=2622#comments)

The FHIR Team has been working with the HL7 Education Work Group to introduce FHIR certification testing so that members of the FHIR community can demonstrate their knowledge of the specification. There’s going to be 2 levels of certification test.

**FHIR Proficiency Test 能力测试**

This test ascertains whether a candidate has basic knowledge of the FHIR specification – what areas it covers, what resources, data types, and profiles are, some basic overview of the way RESTful interfaces work. This test is open to anyone, and it works very much like the existing V2 and CDA tests – though it’s a little easier than them.

该测试确定候选人是否具有FHIR规范的基本知识 - 其涵盖的领域，资源，数据类型和配置文件，RESTful接口工作方式的一些基本概述。该测试对任何人都是开放的，它的工作原理非常像现有的V2和CDA测试 - 尽管它比它们容易一些。

Anyone can sit – and pass – this closed book test.

**FHIR Professional Credentials 专业证书**

This is a much harder test – it explores the functionality of the FHIR specification deeply, and to pass it requires considerable experience working with the specification. The idea of this test is that if you pass it, you’ve met our expectations for being an expert and providing advice to other implementers about how to implement the specification properly.

这是一个更艰难的测试：深入探索FHIR规范功能，并通过他将需要相当多的经验来处理规范。这个测试的想法是：如果通过了测试，则满足了专家的期望值，并向其他实施者提供如何正确实现规范的建议。

This is an open book test – you have a copy of the specification when sitting it – and it has pre-requisites including demonstrated practical experience in healthcare IT, and ongoing exposure to the FHIR community. The credentialing process will itself be approved by the appropriate authorities so that if you have met the credentialing criteria (including passing the test), you’ll be allowed to put letters after your name. The Professional Credentials will require ongoing maintenance.

这是一个开放的书面测试，由一份说明书的副本，并且前提条件是：必须包含在医疗信息领域展示实践经验以及持续接触FHIR社区。认证过程将由相关部门批准，以便符合认证标准（还包括测试）。允许在姓名后面加字母，将持续维护专业证书。

**Introducing the tests**

There’s not a lot of detail here – we’re still working to resolve the process and requirements for the tests. So I can’t tell you, for instance, how much the tests will cost. At least, not at this stage. These details will be released as final decisions are made. The education committee plans to announce the proficiency test at the Madrid HL7 meeting in a few weeks, and then have it available by the September meeting. The Professional Credentials will follow.

At this point, I just wanted to give everyone a heads up about what is coming

Note: Some HL7 insiders have already worked with us prototyping the tests – we thank you for your support, and we’re planning to grandfather you in when the time comes.

2.1 Creating Clinical and Analysis models（ <https://fhirblog.com/2017/08/27/creating-clinical-and-analysis-models/#more-160746> ）

AUGUST 27, 2017 [LEAVE A COMMENT](https://fhirblog.com/2017/08/27/creating-clinical-and-analysis-models/#respond)

As you may know, one of the things that excites me about FHIR is the potential that it has to involve the clinician in the design of health IT systems (and I’m using ‘clinician’ in the widest sense – doctor, nurse allied health etc – anyone who delivers care to a patient).

FHIR中激发我的是让临床医生参与健康IT系统的设计（正在使用最广泛意义上的“临床”——医生、护士、护士联合等等-向患者提供护理的任何人）

So far a lot of my thoughts have been theory rather than practice, so it is great to try these ideas out on a real project – the reporting of Adverse Drug Reactions (ADR) by a clinician to a central service – such as a National EHR!

到目前为止，许多想法都是理论，不是实践。所以在一个真正的项目中尝试这些想法是很好的，临床医生向中央服务机构报告不良药物反应（ADR）——例如：国家HER。

The primary Use Case is:

1. Clinician suspects an ADR
2. They complete a form which is submitted in real-time, and becomes part of the clinical record, but is also used for ‘population’ based reporting.
3. The data is assessed and recommendations given around the likelihood of it being a true allergy, and management advice.

最初的案例：

临床医生怀疑ADR；完成了实施提交的表格，称为临床记录的一部分，也用于“基于人群”的报告；对数据进行评估，并提出建议以确定真实过敏的可能性，以及管理意见。

Eventually, we’d like step 3 to become automated – and, ideally, in real-time using some form of automated Decision Support.

最终，我们希望第三步能够自动化，最理想的情况就是实时使用某种形式的自动化决策支持。

The value of this is not only in reporting (which is the primary use at the moment), but because it’s important clinical data about the patient, it should be made available to other clinicians involved in the care of the patient. And because the data is coded and the source known, it will allow the viewing clinician to make their own judgements as to the accuracy of the reaction when making a prescribing decision, and also to feed back into automated Decision support.

这个值不仅仅记录目前的主要用途，它是关于患者重要的临床资料，所以参与到照顾患者的其他临床医生，而且数据已经编码且源数据已知，将允许观察 临床医生在做出处方的时候，对反应的准确性做出自己的判断，并且还可以反馈到自动决策支持中。

In this post, we’ll take a look at how we might define the contents of the form that is submitted in step 1 – what is the information we need to receive from the submitter?

The overall design process is as follows:

Using the Logical Modeler in [clinFHIR](http://clinfhir.com/), the Clinicians or ‘Subject Matter Experts’ (SME) – create a ‘clinical’ model that represents the information they believe is necessary to properly determine if the reported reaction can be assigned to a specific substance, and should appear in the patient’s allergy list. This model is very much from the perspective of the clinician – there’s no attempt at this stage to decompose it into FHIR resources, but in particular it does attempt to identify the elements that should be coded – and as much detail about the codes as possible (These will, of course, become ValueSets)

整体的设计过程如下：临床医生或临床医师专家（SME）中使用逻辑建模来建立一个“临床”模型，来代表他们认为必要的信息，以正确确定报告的反应是否可以分配给特定的物质，应该出现在患者的过敏名单中。这种模式从临床医生的角度来看非常之多 - 在这个阶段没有尝试将其分解为FHIR资源，但特别是它试图确定应该被编码的元素以及尽可能多的代码细节（当然这些将成为ValueSets）

Using this model as requirements, a ‘FHIR Analyst’ then constructs another model that more accurately reflects the underlying FHIR resources and other artifacts (eg Extension Definitions, ValueSet, NamingSystem) that can, in turn, lead to the definition of the various profiling artifacts either as a manual or an automated process.

使用此模型作为要求，“FHIR Analyst”然后构建另一个更准确地反映基础FHIR资源和其他工件（例如扩展定义，ValueSet，NamingSystem）的模型，这又可导致各种分析工件的定义作为手动或自动化过程。

I suspect that these models will, to some extent, exist at the some time, as the process of defining/building the profiling artifacts may lead to questions back to the SME, and the models will iterate as the design progresses. However, my hope is both models will be ‘understandable’ to the clinician, even though they may not have created them.

我怀疑这些模型在某种程度上将会在某种程度上存在，因为定义/构建分析工件的过程可能会导致中小企业的问题，模型将随着设计的进行而迭代。然而，我希望两个模型对于临床医师来说都是“可以理解的”，尽管它们可能没有创建它们。

So let’s take a look at the example of the Adverse Drug Reaction (ADR). We’ll start with a model that was created by a clinician – [John Fountain](https://www.linkedin.com/in/john-fountain-31a267a/) (Do note that this is incomplete – it’s very much a work in progress).

We’ll take a look at each section in turn, starting with the Patient Details

|  |  |
| --- | --- |
| demo-1.png | demo-2.png |

The column to the left is the ‘Clinician’ model, and to the right is the ‘Analyst’ model. This doesn’t show the details of the models – including the datatypes –  but we’ll call out the important parts in the discussion. 左侧的列是“临床医生”模型，右边是“分析师”模型。这不显示模型的细节 - 包括数据类型 - 但我们将在讨论中提出重要的部分。

* The first thing to note is that in the Analyst model, there are 2 groupings – demographics and observations. The demographics are going to appear as a profile on [Patient](http://hl7.org/fhir/patient.html) resource, while the observations will be [Observation](http://hl7.org/fhir/observation.html) resources – and will have the date/time collected as well.
* Next note that different components of name in the clinical model have become a single entry in the Analyst model. It has a datatype of [HumanName](http://hl7.org/fhir/datatypes.html#humanname), so the other elements aren’t needed.
* We’ve kept the NHI – in the Patient resource it will be an [Identifier](http://hl7.org/fhir/datatypes.html#Identifier) with a fixed ‘system’ to represent the National Health Identifier.
* The ethnicity is going to become an extension – likely a [CodeableConcept](http://hl7.org/fhir/datatypes.html#codeableconcept) referring to a national ValueSet.

首先注意的是，在Analysis模型中，有两个分组：demographics（在Patient资源上显示为个人资料）和Observations（将为Observation资源，并且还收集日期时间）。

接下来注意：临床模型中名称的不同组成部分已成为Analysis模型中的单个条目。含有HumanName的数据类型，因此不需要其他元素。

我们保持NHI，在Patient资源中保留固定的“system”的Identifier来代表国家健康标识符。

种族将成为一个扩展 - 可能指的是国家ValueSet的[CodeableConcept](http://hl7.org/fhir/datatypes.html#codeableconcept)。

The next node we’ll take a look at is the substance node:

|  |  |
| --- | --- |
| substance-1.png | substance-2.png |

Quite a difference here! (But one that is completely expected). Remember that the clinician is describing all the information that they want to be collected so we’d expect it to be a longer list. It’s the Analyst that can then pick the FHIR datatypes that will best record this data (but – once picked – the Clinician can easily verify that all the data they specified is present).

这是相当不一样的，临床模型描述的是想要收集的所有信息， 期望是一个更长的列表。分析模型可以选择最佳记录数据的FHIR数据模型（但是，一旦选择临床医生可以轻松验证指定的所有数据是否存在）

* For a start we’ve collapsed SubstanceCode and MedicineName into a single element – substance. This is because we really only want one place where a consuming application needs to look for the ‘thing’ that the clinician thinks the reaction is to. And there are many similarities between a medication and some other substance (sea food or peanut perhaps) in terms of the data we wish to capture. We’ve included a ‘substance.type’ element – though this might not be necessary as the code element might provide sufficient discrimination (it’s a [CodeableConcept](http://hl7.org/fhir/datatypes.html#codeableconcept), so gives us the name of the substance, code in the terminology ([SNOMED](http://www.snomed.org/snomed-ct) in this case) and a text field).

刚开始，将SubstanceCode 和 MedicineName折叠成一个元素——Substance。这是因为需要一个消费应用程序寻找临床医生认为反应物的地方。在期望获取数据方面，药物和物质之间有许多相似之处，其中包含了substanceType元素，这可能不是必须的，因为substanceCode可能会提供足够的规则。

* The period has a datatype of [Period](http://hl7.org/fhir/datatypes.html#period) – so we get the start and end dates (we might consider re-naming the element to ‘periodfOfExposure’)

Period数据类型为Period，我们需要开始日期和结束日期（考虑将元素重新命名）

* The dose element is where a lot of the elements from the Clinician model has gone to (like dose value/unit, frequence & route). This is actually an ‘internal’ Logical model in the FHIR spec called Dosage as it is so widely used in the pharmacy domain – [here are the details](http://hl7.org/fhir/dosage.html#Dosage). (Of course, in the Logical Modeler there’s an icon that will load this page – or the details of any datatype for that matter). In the case of a ‘substance’ it is likely that many of these fields will be unused – but that doesn’t matter. There’s only an issue if there’s a field for a substance that we can’t accommodate where we’d need to think again.

dose元素是临床医生模型中许多元素可取的值（例如：剂量值、剂量单位、频率、路线）。这实际上是FHIR规范中称为Doseage的“内部”逻辑模型，因为在药学领域广泛使用。如果在substance中未被使用，则需要思考另一个地方。

Now for medications:

|  |  |
| --- | --- |
| meds-1.png | med2-2.png |

Again, we’ve been able to represent the required fields more simply – the medications element is simply a repeating series of [MedicationStatement](http://hl7.org/fhir/medicationstatement.html) resources. We may want to consider having a ‘top level’ element called ‘currentData’ with medications and previously identified allergies below that (using [AllergyIntolerance](http://hl7.org/fhir/allergyintolerance.html) for the existing allergies). Maybe even the current [problems](http://hl7.org/fhir/condition.html). Lets see what our clinicians think

再次，可以简单表示所需的领域:medication元素仅仅是对MedicationStatement资源的重复，我们考虑需要一个称为“currentData”的高级别元素，其中含有AllergyIntolerant资源识别出现有的过敏，也可以使用Problem资源。

The last element we’ll explore here is the reaction.

|  |  |
| --- | --- |
| reaction-1.png | reaction-2.png |

Not so different really – just having a single [dateTime](http://hl7.org/fhir/datatypes.html#dateTime) datatype for the onset – though again a period might prove useful. I expect there’s more data to go in here in the next round! Interestingly, there’s talk of a separate [‘AdverseReaction’](http://wiki.hl7.org/index.php?title=AdverseReaction_FHIR_Resource_Proposal) resource being defined (though it hasn’t made it into the spec yet) that may be of interest. We’ll probably need to move before that becomes defined – but it might be an opportunity to feed our requirements back into the community.

定义一个单独的“adverseReaction”资源，可能引起注意，我们可能需要在此之前进行定义 - 但这可能是将我们的要求提供给社区的机会。

So there‘s a first cut at going from a ‘Clinician oriented’ model to a ‘FHIR oriented’ model. Of course this is only the first iteration – there will be plenty more as we challenge both of these models against as many Use Cases as we can find and as many clinicians as we can find before we land on a final version – at least one that all can agree on, even if we don’t get complete agreement! During this process we’ll also work to define the ValueSets and locate/build the Extension Definitions and other artifacts that we’re going to need.

这是从“临床医生为导向”到“FHIR导向”的第一个切入点。当然，这只是第一次迭代，我们挑战这两个模型，可以找到许多案例，我们可以找到许多临床医生，可以在最终版本之前找到许多临床医生，这将是更多的——至少所有中的一些人都可以同意，即使我们没有完全达成协议！在此过程中，我们还将定义ValueSet并定位/构建扩展定义和我们将需要的其他工件。

And what’s nice is that we can open the logical model in the Scenario Builder to enter some real data – it isn’t real FHIR of course, but it let us prove that there is a place for any of the data items we need to capture.

我们可以在场景生成器中打开逻辑模型来输入一些真正的数据 - 当然不是真正的FHIR，但是我们可以证明，我们需要捕获任何数据项。

The next stage (once we’ve agreed on the model) is to convert this into real FHIR profiles and one interesting thing to think on is what these profiles are going to look like – and how many there will be. Take the patient for example, do we create a single patient profile for New Zealand that all of our profiles use (refer to), or do we create one specifically for this project? If we do create one for this project, then we can absolutely describe what are the elements we’re interested in – but if each project has it’s own profile, then we’re going to have a lot of them! (We can have profiles that alter other profiles, so there are quite a few options open to us).

下一个阶段（一旦我们同意了模型），将其转化为真正的FHIR资料，一个有趣的事情就是这些资料将会是什么样的 - 而且有多少。以患者为例，我们是否为新西兰创建了所有我们的个人资料使用（参考）的单个患者资料，还是为此项目专门创建？如果我们为这个项目创建一个，那么我们可以绝对地描述我们感兴趣的元素，但是如果每个项目都有自己的个人资料，那么我们将会有很多这些！（我们可以拥有更改其他配置文件的配置文件，因此有很多选项可供我们使用）

In this case it seems more sensible to create a single Patient profile that everyone uses, and accept the fact that there will be elements that we don’t need – but there are some fish hooks with this approach that we’ll need to think through. For example what do we do if one project needs to make Patient.ethnicity a required field? Time will tell how this gets managed…

在这种情况下，创建每个人使用的单个患者资料似乎更明智，并接受一些事实，即我们不需要的元素 - 但是有一些鱼钩与这种方法，我们需要思考。例如，如果一个项目需要使Patient.ethnicity成为必需领域，我们该怎么办？时间会告诉你如何管理...

We’ll also need to think about the ‘paradigm’ of exchange, and whether we want to define a [Questionnaire](http://hl7.org/fhir/questionnaire.html) resource to help capture the information we want…

我们还需要考虑交换的“范式”，以及我们是否要定义[Questionnaire](http://hl7.org/fhir/questionnaire.html)资源，以帮助捕获我们想要的信息...

Incidentally – this is exactly the exercise (with a bit of training as well) that we be doing both at [HINZ](http://www.hinz.org.nz/page/Workshops) and [Devdays](https://www.fhirdevdays.com/) in November. Come along if you’re interested (and bring your laptop!)

2.2 Using the clinFHIR query tool （ <https://fhirblog.com/2017/08/23/using-the-clinfhir-query-tool/#more-160658> ）

AUGUST 23, 2017 [LEAVE A COMMENT](https://fhirblog.com/2017/08/23/using-the-clinfhir-query-tool/#respond)

I’ve been in China recently (attending the CHIMA conference and presenting on FHIR in a number of events), and while there I had a bit of spare time, so decided to give the clinFHIR query tool a bit of attention (I also needed it for the workshop I gave).

The query tool has been there for quite a while, but it’s been rather basic and a bit klunky, so a facelift was overdue! 查询工具已经存在很长一段时间，但是还是一个整理过程。

At a high level, there are a number of main functions you can use it for.

* Querying a FHIR Server. This is useful when learning FHIR as you can send queries to a server (and there’s a simple query builder built in) and then review the response in a number of different ways (plain JSON, XML, a tree, FHIRPath query).
* Saving/Updating a resource directly against a FHIR server
* Validating a resource. This is useful if you are building a resource by hand and want to be sure it is valid. You can also use it to validate a resource instance against a specific profile.

高层次上，可以使用多种主要功能：

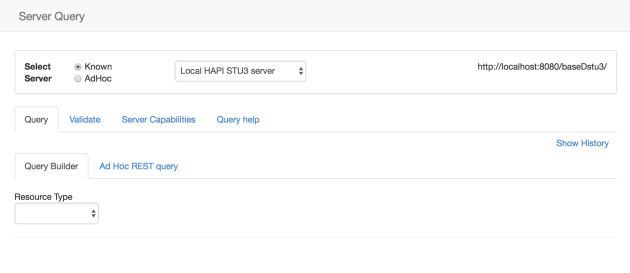
查询FHIR服务器，在学习FHIR的过程中，可以向服务器发送查询（内置个简单的查询构建器），然后以不同的方式（纯JSON、XML或者FHIR Path查询）来查看响应，这非常有用。

直接针对FHIR服务器保存或更新资源

验证资源。对于手工构建资源并确保有效性，将是非常有用的。可以使用它针对特定的profile验证资源实例。

To run the query tool, start clinFHIR and select the tool from the main menu. Here is a screenshot of the main UI.

运行查询工具，启动丛林FHIR，从主菜单中选择该工具，下面是主界面的截图



At the top of the page is where you select the server to query. You can select any of the configured servers (including any that you have added through the ‘Add Server’ dialog on the front page), and you can also directly enter the URL to any FHIR compliant server.

页面顶部是您选择要查询的服务器的位置。您可以选择任何已配置的服务器（包括您在首页上通过“添加服务器”对话框添加的任何服务器），也可以直接输入任何符合FHIR的服务器的URL。

The server must be able to return a ‘[CapabilityStatement](http://hl7.org/fhir/capabilitystatement.html)’ resource (‘Conformance’ resource in STU2). The tool will then use that resource to populate the list of supported resources, and will also generate a display (the ‘Server Capabilities’ tab).

服务器必须能够返回“ [CapabilityStatement](http://hl7.org/fhir/capabilitystatement.html) ”资源（STU2中的“conformance”资源）。然后，该工具将使用该资源填充支持的资源列表，并且还将生成显示（“Server Capabilities”选项卡）。

There are 4 main tabs.

* The **Query** tab. Here is where you can make queries against the selected server
* The **Validate** tab. Used for validating resources built separately. It can also be used to add a resource directly.
* The **Server Capabilities** tab lists what the server supports – according to the CapabilityStatement downloaded when the server was selected
* The **Query Help** tab has some useful summaries and links for querying a FHIR server.

有4个主标签。在查询选项卡，可以在这里对所选服务器进行查询；该验证标签，用于验证分开构建的资源。它也可以直接添加资源；该服务器功能选项卡中列出的信息是-根据下载的CapabilityStatement选择服务器的信息；“ 查询帮助”选项卡具有一些有用的摘要和链接，用于查询FHIR服务器。

We’ll examine each of these.检查下面的内容：

（1）The **Query tab** offers 2 options for generating the query – a simple builder and an ‘ad hoc’ option where you enter the query directly.

Query选项卡提供了用于生成查询的2个选项：一个简单的构建器builder和“ad hoc”选项（可以直接输入查询）

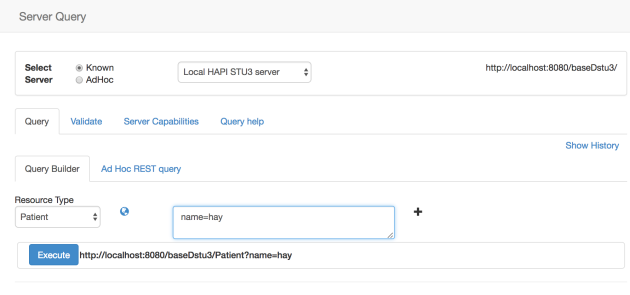
Use the builder as follows.使用构建器如下。

Select the resource you want to query from the dropdown 在下拉菜单中选择要查询的资源

You can either enter the search parameters directly into the text box to the right, or click the plus (+) symbol to get a dialog box that has the search parameters supported by the server for that resource. In either case you will see the actual query displayed below the parameter box. Note that you don’t need to enter the full url – the tool will do that for you.

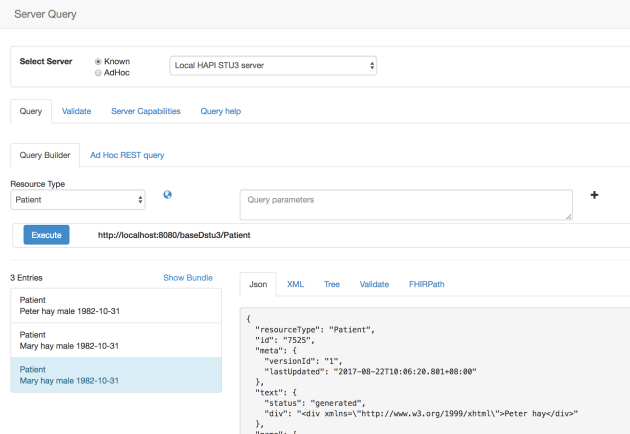
可以直接在右侧的文本框中输入搜索参数，也可以单击加号（+）来获取一个对话框，该对话框具有该资源的服务器支持的搜索参数。在这两种情况下，您将看到显示在参数框下面的实际查询。请注意，您不需要输入完整的网址 - 该工具将为您做到这一点。

Here’s an example:示例如下



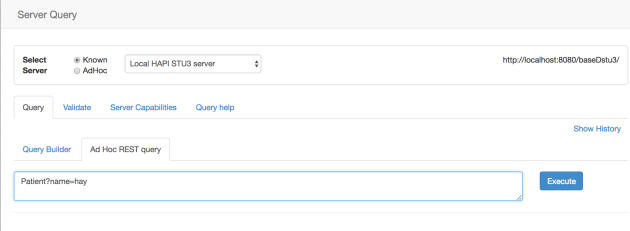
Once you have completed the query, click the ‘Execute’ button to run the query against the selected server. 完成查询之后，单击“Execute”按钮对所选的服务器运行查询。

Here is an example of the result from running a query. The resources from the result bundle are displayed to the left, with various display options to the right. In this example, the Json tab has been selected.一下就是查询的结果示例。结果bundle中的资源显示在左侧，右侧有详细的显示选项。还可以选择XML或JSON的表示形式。



The Ad hoc REST query allows you to enter the query directly. You don’t need to enter the server root, but you do need to enter the resource type. Once you’ve executed the query, the response viewer is the same. Here’s a screen shot:

Ad hoc REST查询允许直接输入查询，此时不需要输入服务器的根目录，但是需要输入资源类型。执行查询之后，响应查看器的结果值是一样的。



Some general notes about querying and displaying the responses. 关于查询和显示响应的一些一般注释。

If you want to return a single resource by its id on the server, then you need to use a special syntax – [querying by \_id](http://hl7.org/fhir/search.html#id).  This will return a bundle with one resource rather than the resource itself. Actually, this is quite handy as it allows you to use other search parameters, such as [\_include](http://hl7.org/fhir/search.html#include). For example the following query returns a bundle containing a single observation with an id of 160250 and the Patient that is the subject of the observation (the server base url is added automatically).

如果你想返回服务器id的单个资源，则需要使用特殊的语法——通过\_id查询。这将会返回一个包含一个资源而不是资源本身的一个bundle。实际上，这是非常方便的，允许使用查询参数，例如\_include。例如，以下返回一个包含id=160250的单个Observation的bundle，作为Observation对象的Patient（自动添加服务器的基本URL）。

Observation?\_id=160250&\_include=Observation:subject

（2）The Validation sub-tab allows you to validate the selected resource against a given profile. You can choose a different server to perform the validation than the one you retrieved the resource from – useful when the server does not support validation.

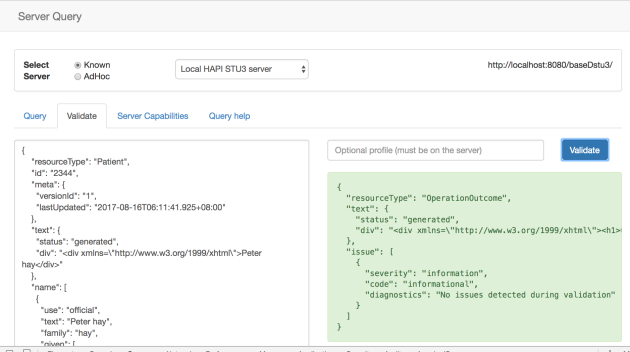
验证选项卡允许根据给定的profile来验证所选的资源。当服务器不支持验证的时候，可以选择不同的服务器来执行验证而不是从检索到资源的服务器中执行验证。

There’s a link labelled ‘Show Bundle’ immediately above the list of resources in the bundle. Clicking that link will show the raw bundle received back from the server, plus the FHIRPath query facility that allows you to make queries directly against that bundle.

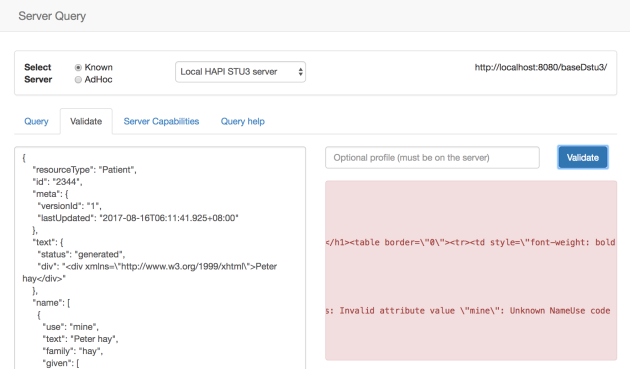
在bundle中资源列表上方有一个标记为“show Bundle”的链接，单击该链接将显示从服务器返回的原始bundle，以及FHIR Path查询工具，允许直接对bundle进行查询。

The **Validate tab** (the main one – not the one that appear when you select a resource returned in a query described above) allows you to validate that a resource instance you have created externally is valid – either against the core spec or against a particular profile. Selecting the tab will show a box where you can paste the resource (currently it must be a single resource in Json format) and then validate it against a given profile. Here’s an example of a valid resource:

Validate tab（主要选项卡，当在上述查询中选择资源时候不会显示该选项）允许验证外部创建的资源实例是有效的：根据核心规范或者特定的profile，选择选项卡将显示一个可以可以粘贴资源的文本框，然后会根据给定的profile进行验证，以下是有效资源的示例。

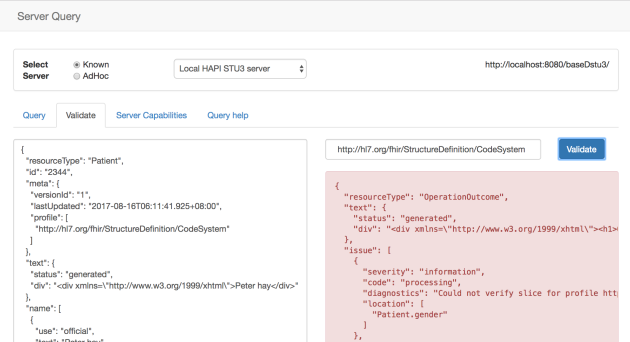


And here’s an example where the resource is not valid (name.use has not got a valid value). Note that you may need to scroll the error text sideways to see the entire response.注意这是一个资源无效的例子（name.use没有一个有效值）。可能需要滚动侧面的错误文本以查看整个资源响应。



If you don’t specify a profile to validate against, then the tool will validate against the core definition for that resource type. If you do enter a profile (and the server you have selected has a copy of that profile) then it will use the profile you enter. Here’s an example of seeing is a Patient resource instance is conformant to the CodeSystem profile. Unsurprisingly, it isn’t. Note that the profile was added to the *meta.profile* element on the resource instance by the tool.

如果没有指定要验证的profile，则该工具将会根据资源类型的核心定义进行验证。如果输入profile（并选择服务器中应该具有的配置文件的副本）则将使用输入的profile。下面是一个实例，一个Patient资源实例符合CodeSystem profile。不出所料。profile已经通过该工具添加资源实例上的meta.profile元素中。



Once an instance has been validated, a red ‘Send to Server’ button will appear at the bottom right. Clicking that will save the instance to the server – any errors will be displayed. If the instance has an id property, then an update will occur – otherwise a new resource will be created – and can be queried for on the query tab.一旦验证完实例，将在右下方出现 “Send to Server”的红色按钮，单击它将实例保存到服务器中：将会显示任何错误信息。如果实例有一个id属性，则会发生更新，否则创建一个新的资源，并且可以在query tab中进行查询。

（3）The **Server capabilities tab** displays the CapabilityStatement resource that was received from the server when selected. It has a number of sub-tabs:

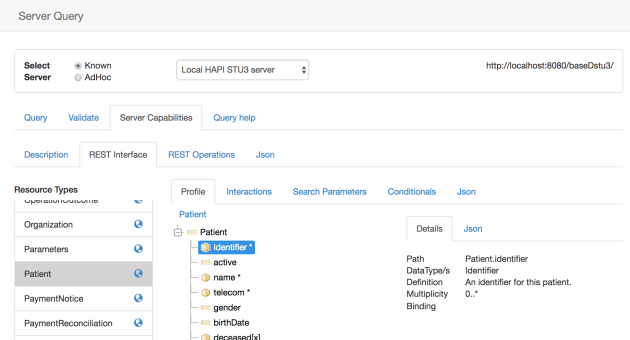
* The Description tab shows – well – the description of the server
* The REST interface tab lists the resource types that are supported over a REST interface. There’s a ‘globe’ icon that will display the details of the type in the spec, and selecting a type will display a ‘tree view’ of the type.
* The operations tab shows the supported operations (without any documentation unfortunately – I’ll look into that)
* And the Json tab shows the raw json.

“Server capabilities tab”显示定时从服务器接收的CapablityStatement资源。有一些子标签：

Description tab显示：服务器的描述

REST interface tab列出了：通过REST 交互支持的资源类型。“global”icon将会显示规范中资源类型的详细信息，在左侧选择资源类型，将会在右侧显示类型的“tree view”树状图。

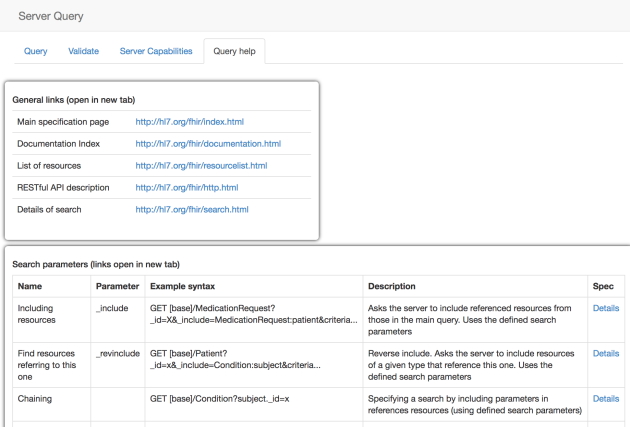
Here’s the REST interface tab:



Note that if the server specifies a base system profile for a resource type, then a tree view is displayed.注意：如果服务器指定资源类型的基本系统profile，则将显示一个树状图。

（4）And finally **the Query Help tab** has a number of useful links.

**“Query Help tab”**有许多有用的链接。



2.3 GraphQL <https://fhirblog.com/2017/08/17/graphql/#more-160637>

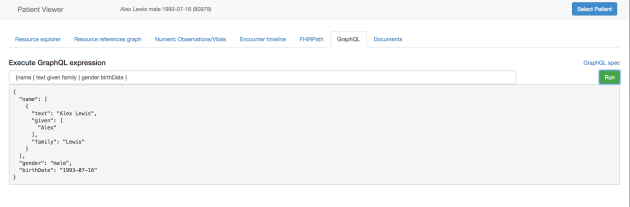
AUGUST 17, 2017 [LEAVE A COMMENT](https://fhirblog.com/2017/08/17/graphql/#respond)

Following my post yesterday, someone who shall remain nameless (you know who you are [Brian](https://www.linkedin.com/in/brianpos/)) suggested that it would also be good to be able to make [GraphQL](http://graphql.org/) queries from [clinFHIR](http://clinfhir.com/). I know even less about GraphQL than I did about FHIRPath, but as Grahame has an implementation on his server, it was a reasonably straightforward matter to put a simple UI in so you can experiment with that against a Patient resource. (GraphQL can do a lot more than that, but this is a start).

The specification for the FHIR version of GraphQL is available [here](http://build.fhir.org/graphql.html) (be warned, it’s very early in the process). FHIR版本的GraphQL的规范在[here](http://build.fhir.org/graphql.html) 可用。

If you select Grahames R3 server as the data server, then select a patient in the Patient Viewer of clinFHIR, you’ll see a new tab as shown in the screenshot below.

如果选择Grahames R3服务器作为数据服务器，则在丛林FHIR中的Patient Viewer中选择一个Patient，则将会显示如下：



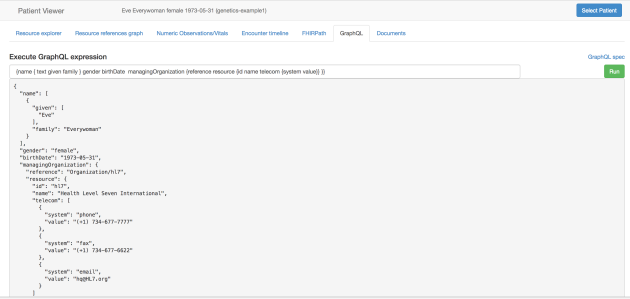
You can enter a GraphQL query that makes sense against the current patient, and then run it to see the results (or an error) displayed below the query.

如果可以输入对当前患者有意义的GraphQL查询，则运行它来查询下方显示的结果（或错误）。

One of the neatest things is that way you can easily resolve references across resources. You can do this using plain REST and FHIRPath as well, but GraphQL seems to be a good choice when you are after only a subset of the data – The REST option in general returns whole resources, and FHIRPath is (I think) more for describing paths and references rather than wholesale data extraction.

最简单的一件事情就是：可以轻松的解决跨资源引用的问题。可以使用普通的REST和FHIR Path来执行此操作，但是当数据仅仅是一个子集的时候，GraphQL是一个不错的选择——REST选项常常返回全部资源，而FHIR Path更多用于描述路径和引用而不是对真个数据的提取。

Oh, and here’s a more interesting example where we pull in details from a referenced resource (the managingOrganization)下面是一个有趣的例子，可以从引用Organization资源中提取细节。



One thing to be quite clear about is that GraphQL does not return resources (even though it may look like it). It allows you to pull out the bits you a want and assemble a completely custom response.

注意：GraphQL不返回资源（即使看起来很像）。允许所需的bits，并组合完全自定义的响应。

I have to say that implementing GraphQL does look like quite an exciting addition to the FHIR family – especially in collecting information across resources, and it will be interesting where it ends up.  I’ll try to enhance the support in clinFHIR (as a client of course) and if you are implementing it on a server using the FHIR interface (with [CORS](https://en.wikipedia.org/wiki/Cross-origin_resource_sharing) enabled) , than let me know and I’ll add your server to the list of supported servers.

不得不说，实现GraphQL对FHIR系列来说是一个令人激动的补充，特别是在资源之间收集信息的时候，将会非常有趣。尝试在clinFHIR中支持此项工作，如果在服务中使用FHIR交互实现，则会将使用的服务器添加到支持的服务器列表中。

2.4 FHIR Path

AUGUST 16, 2017 [2 COMMENTS](https://fhirblog.com/2017/08/16/fhirpath/#comments)

I’ve known about [FHIRPath](http://hl7.org/fhirpath/) for some time, though I must admit I haven’t paid a lot of attention to it.

Put briefly, FHIRPath is a specification that describes how to identify (and potentially extract) data from a resource using a path based syntax. From the spec:

简单的来说，FHIR Path是一个规范，描述如何使用基于path语法来识别数据。规范如下：特别重要的是能够轻松准确地表达基本的逻辑条件，例如：在所需的约束中查找内容（例如：Patient资源必须有name属性）、决策支持（例如：患者患有糖尿病并且尚未进行全面的足部检查）、队列定义（例如：60-75岁的所有男性患者）、方案描述（例如：测试的样本是对sodium的存在测试为阳性）以及许多其他的环境。

*Of particular importance is the ability to easily and precisely express conditions of basic logic, such as those found in requirements constraints (e.g. Patients must have a name), decision support (e.g. if the patient has diabetes and has not had a recent comprehensive foot exam), cohort definitions (e.g. All male patients aged 60-75), protocol descriptions (e.g. if the specimen has tested positive for the presence of sodium), and numerous other environments.*

with these features:

* ***Graph-traversal:****FHIRPath is a graph-traversal language; authors can clearly and concisely express graph traversal on hierarchical information models (e.g. HL7 V3, FHIR, vMR, CIMI, and QDM).*
* ***Fluent****: FHIRPath has a syntax based on the Fluent Interface pattern*
* ***Collection-centric****: FHIRPath deals with all values as collections, allowing it to easily deal with information models with repeating elements.*
* ***Platform-independent****: FHIRPath is a conceptual and logical specification that can be implemented in any platform.*
* ***Model-independent****: FHIRPath deals with data as an abstract model, allowing it to be used with any information model.*

具有以下特点：

Graph-traversal：FHIR Path是一个图遍历语言，作者可以在层次信息模型上清晰简单地表示图遍历。

Fluent：FHIR Path具有基于Fluent界面参数的语法。

Collection-centric：FHIR Path处理的所有值作为集合，允许他轻松处理具有重复元素的信息模型。

Platform-independent：FHIR Path是一个概念和逻辑规范，可以在任何平台上实现。

Model-independent：FHIR Path将数据作为抽象模型来处理，允许将其与任何信息模型一起使用。

The reason I’ve taken a closer look is that I’m involved in a project where we want to be able to create a testing environment for data extracted from one type of message (HL7 V2), persisted in a store and then exposed as FHIR resources. I’m aware of course of the [mapping language](http://hl7.org/fhir/mapping-language.html) that could potentially perform this conversion directly, but this is more of a ‘black box’ test for the outcome of such a conversion regardless of how it was done, rather than a way to actually do it.

希望从一种消息类型中，为提取的数据创建一个测试环境，持续存储在一个存储器中，并显示为FHIR 资源。已经意识到映射语言的过程可能直接执行这种转换，但是对于这种转换的结果来说，更像是一个“黑盒子”测试，不管他是如何完成的，而不是一个实际执行的方式。

The approach we’re taking is to create a Logical model that expresses the information we are converting in an easy to visualize way, then for each element in the model use the map to describe where it came from, and where it appears. Eventually this should lead to automated end to end testing, but it is harder than you might think! However, if you can use this to create a view where a person can easily visualize where the data comes from and check that it is correct, then that is a step forward.

采取的方法是创建一个逻辑模型，易于可视化的方式表示正在转换的信息，然后在模型中的每个元素都是以map映射来描述他的来源以及出现的位置。最终导致自动化的端点测试，但是这比想象的难。但是如果使用它创建一个视图，即人们可以轻松地知道数据来自哪里，并检查是否正确，则这是一个进步。

Take encounters for example. 以就诊为例

Suppose we want to check the clinicians involved in the encounter – attending, referring and consultant. In our model we’ll create them as 3 different elements (because that is easy to understand, and then map them as follows:假设，检查就诊过程中的临床医生：参与者、转诊人和咨询人。则在模型中创建3个不同的元素（因为这很容易理解，其映射如下）

|  |  |  |
| --- | --- | --- |
|  | HL7 v2 | FHIR |
| Attender | PV1-7 | Encounter.participant where the type is ‘ATND’ |
| Referrer | PV1-8 | Encounter.participant where the type is ‘REF |
| Consultant | PV1-9 | Encounter.participant where the type is ‘CON |

So the question was how to express the FHIR mapping? And FHIRPath seems to fit the bill nicely. (Note that in theory I could also use if for specifying the v2 maps – but I’m not aware of an implementation for this).

如何表达FHIR映射？而FHIRPath似乎很适合这个账单。（请注意，理论上如果指定v2映射也可以使用- 但我不知道怎样实现）。

I don’t intend to repeat what is in [the spec](http://hl7.org/fhirpath/) – smarter folk than I have described that and you can read it for yourself – and test it online using neat tool  [here](http://niquola.github.io/fhirpath-demo/#/) which uses the Javascript implementation of FHIRPath (and, not surprisingly the same one that I decided to use – thanks [Nicola](https://www.linkedin.com/in/nicola-ryzhikov-586a6913/)!)

So the paths for our clinicians becomes:

* Attender: **Encounter.participant.where(type.coding.code=’ATND’)**
* Referrer: **Encounter.participant.where(type.coding.code=’REF’)**
* Consultant: **Encounter.participant.where(type.coding.code=’CON’)**

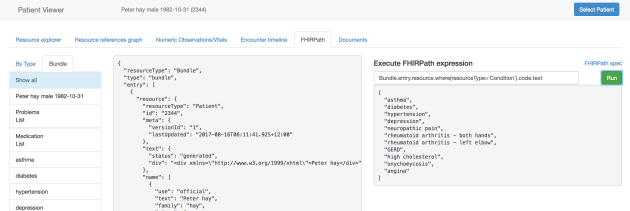
Because I think it’s a very useful way of diving into (and understanding) resources (and this includes Bundles of course) I decided to implement it in various places in [clinFHIR](http://clinfhir.com/). This wasn’t without its technical challenges, as I couldn’t get the library to run in the browser, which means that I needed to create a server (nodejs) based module, and pass the resource and path to it via HTTP to be evaluated server side. This has performance implications that I’d quite like to get rid of, but it does work. The other issue is that the library is not up to date with the current spec, so not all of it will work, but I’m hopeful that there will be an update soon…

因为，认为这是潜入资源的一个非常有用的方式（当然包括bundle），决定在clinFHIR的各个地方进行实施，这具有技术挑战。无法使用浏览器运行库，意味着需要创建一个基于服务器的模块，并通过HTTP将资源和路径传递给评估的服务器端。非常想要摆脱的性能影响，但它确实有效。另一个问题是库不符合现行规范，所以并不是全部都可以正常运作，但我希望很快会有更新...

So here are the places where you can use it in clinFHIR:

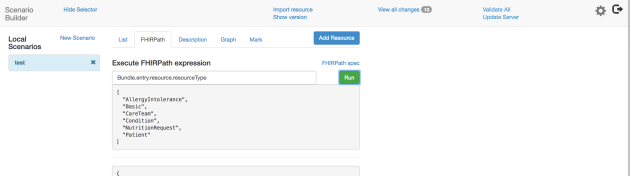
In the **Patient Viewer**, there’s a new tab after you’ve selected the patient that allows you to run a FHIRPath expression on either a single resource or the entire bundle of data for that patient (Select the Bundle tab to the upper right, then click ‘Show all’). Here’s an example that lists the names of all Conditions in the bundle:

在Patient Viewer中，选择了允许在单个资源或该患者整个数据bundle上运行FHIR Path表达式的患者之后，会出现一个新的选项卡（选择右上角的“bundle”，单击“show all”）。以下是一个示例，列出了bundle中所有conditions。



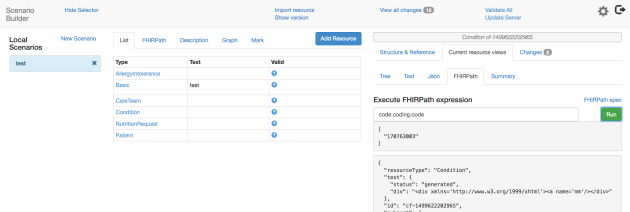
It’s in 2 places in the **Scenario Builder**. In the middle pane you can enter expressions against the bundle that holds all the resources in the scenario. Here’s an example:

在“Scenario Builder”中的第二个位置（FHIR path选项卡中），在中间面板中可以输入包含场景中的所有资源的bundle的表达式。下面是一个示例：



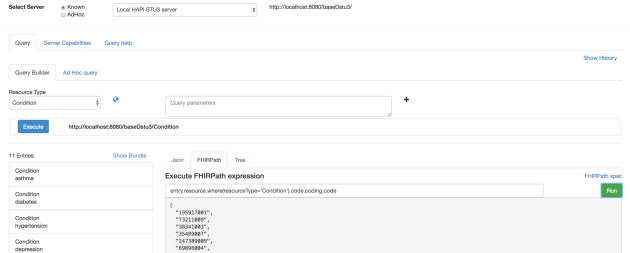
And if you view a single resource, then there’s a tab that runs expressions against that resource only – like this:

如果查看一个资源，则还有仅仅针对该资源表达式的一个选项卡（List 选项卡），如下所示：



And finally in the Query tool, you can examine the bundle returned by a query:

在下面的查询工具中，可以检查查询所返回的bundle



2.5 Implementation Guide viewer

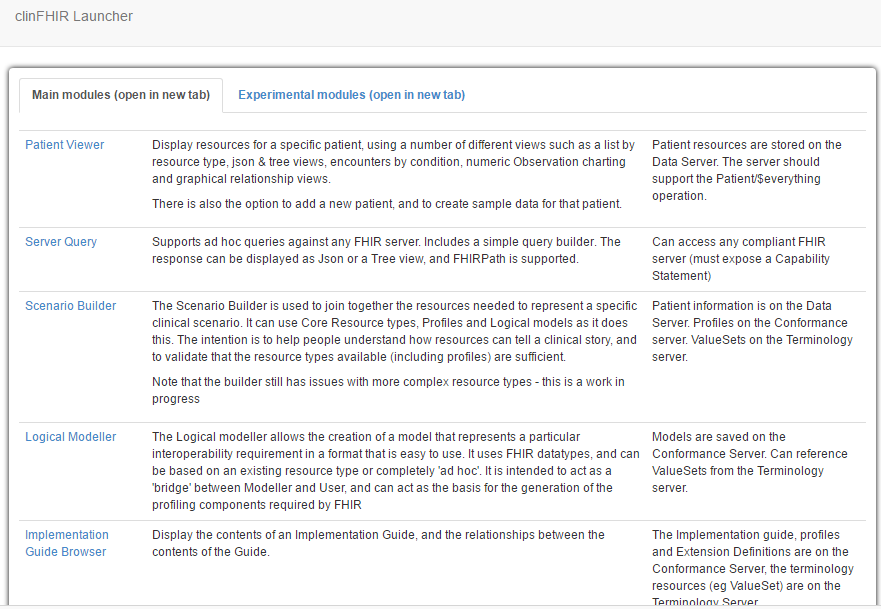
<https://fhirblog.com/2017/07/25/implementation-guide-viewer/#more-160560>

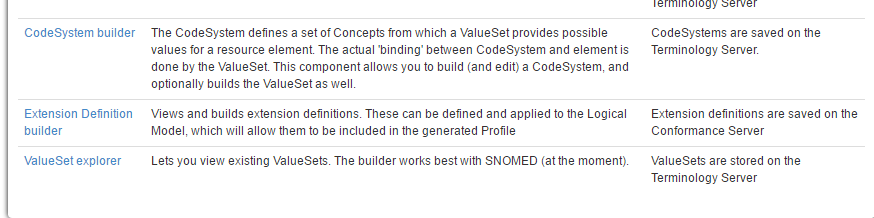
Just a short post to describe some updates to the Implementation Guide viewer (which I’ve renamed from ‘profile viewer’ as it was described [in this post](https://fhirblog.com/2017/06/26/clinfhir-profile-viewer/).) Thanks to some comments in the FHIR chat from my friend [John Moehrke](https://www.linkedin.com/in/john-moehrke-6841414/) I’ve done some work on the ‘Graph visualizer’ component of the viewer.

短时间内，描述实施向导viewer的更新。

The idea is to make it easier to understand the contents of an Implementation Guide, and the relationships between them. Currently limited to Profiles, Extensions & ValueSets – but no reason why it couldn’t be extended (or won’t be  ).

实施向导目前仅仅限于“profile”、Extensions、ValueSets，但是无法扩展。



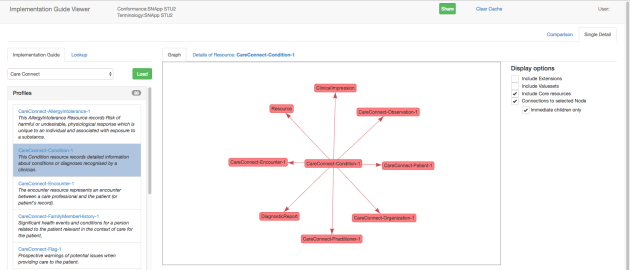


To take a look, start [clinFHIR](http://clinfhir.com/) and select the Implementation Guide view (as described in the [previous post](https://fhirblog.com/2017/06/26/clinfhir-profile-viewer/) – remembering the name change).

启动clinFHIR并选择“实施指南”视图（就像在上面介绍的一样）

If you load the [CareConnect](https://nhsconnect.github.io/CareConnectAPI/)Implementation Guide (which is the one I’ve been working with), you’ll see a (small) graph to the right hand side. It is certainly possible to work with this graph (though it is a bit cluttered), but it really becomes useful when you limit the view. To do so, expand the ‘Profiles’ accordion to the left, and select one of the profiles – say Condition.

如果加载了[CareConnect](https://nhsconnect.github.io/CareConnectAPI/)实施指南（这是我一直在使用的指南），那么右侧将会看到一个（小）图。当然可以使用这个图（虽然它有点混乱），但是当你限制视图时它真的很有用。为此，请将左侧的“Profiles”展开，然后选择其中一个profiles - 说“Condition”。



The graph in the middle shows the Condition profile as a central node, all of the resources that it references (ie the direction is from the Condition profile to the other resource), and the connections between them.

中间的图表显示了Condition profile作为中央节点，以及引用的所有资源（从Condition profile到另一个资源）以及它们之间的链接。

Selecting a node will show a few basic details about that node in the lower right (url & description). Depending on the node type (more on this in a minute) there will be other options:

* **Profile**: There will be a link labelled ‘Select Node’. Clicking that link makes the selected node the ‘center’ node and re-draws the graph. It also makes that profile the ‘selected’ profile so the details tab will now show details of that resource rather than the one originally selected. Selecting that tab shows the views described in the earlier post. (Somewhat enhanced since then).
* **Extension**. A link that when clicked shows a description of the extension.
* **ValueSet**. A link that when clicked displays the ValueSet browser. The ValueSet is able to be expanded if the Terminology server has the definition of that ValueSet, and is capable of performing the expansion.

选择一个节点将显示一些关于右下角的节点（url&description）的基本细节，根据节点类型 ，将有其他选项：

Profile：标记为“Select Node”的链接。单击该链接将使所选节点成为“center”节点并重新绘制图形。使该profile成为“selected”的profile。因此detail tab将显示该资源的详细信息，而不是最初选择的资源。选择该tab将显示前面帖子中描述的视图。

Extension：单击显示扩展名描述的link

ValueSet：点击显示ValueSet浏览器的link。如果术语服务器具有该ValueSet定义，则值集能够扩展，并能够执行扩展。

Selecting one of the arrows (representing the references between nodes) will show the paths from the source to the target resource in to the lower right (Only a single arrow is shown, regardless of the number of references between any two nodes, otherwise the graph is overly cluttered).

选择一个箭头（表示节点之间的引用）将显示从源到目标资源到右下方的路径（仅显示单个箭头，而不管任何两个节点之间的引用数量，否则图形过于凌乱）。

The actual contents of the graph can be changed using the check boxes to the upper right. Options are:

* **Include Extensions**. If checked then extensions in the profile are also shown (with a purple background)
* **Include ValueSets**. If checked then ValueSets in the profile are also shown (with a yellow background)
* **Include Core Resources** (selected by default). This will include resource types and ValueSets from the FHIR spec that are referenced by the profile, but not included in the Implementation Guide.
* **Connections to selected node** (selected by default). If this is not selected, then the entire contents of the Implementation Guide are shown. The graph can be zoomed (mouse wheel or trackpad), but it’s a wee bit cluttered. Selecting it allows a single resource to act as the centre.
* **Immediate Children only** (selected by default). Unchecking this will expand the shown nodes beyond the immediate children of the centre node. So it will show ‘grandchildren’  and ‘great grandchildren’ as well.

可以使用右上方的复选框更改图形的实际内容，选项是：

include Extension：如果选中，则将显示profile中的扩展。

include ValueSet：如果选中，将显示profile中的值集（带有黄色背景）

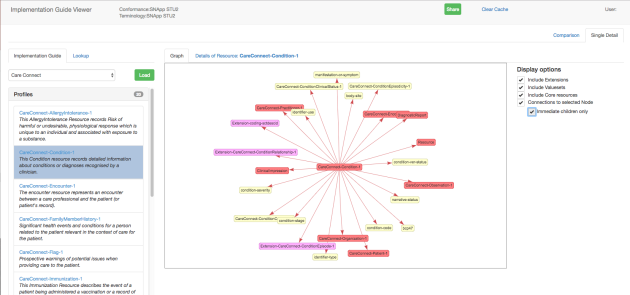
include Core Resource：如果选中，将包括通过profile引用的FHIR规范中的资源类型和值集，但是不包括实施向导。

connections to selected node：如果未选中，则显示“实施指南”的全部内容。该图可以缩放（鼠标滚轮或触控板），但是它有点混乱。选择它可以使单个资源充当中心。

Immediate Children only：取消选中此项将会显示的节点扩展到超过中心节点的直接子节点。

Here’s a shot of the Condition profile, showing all related profiles and ValueSets

下面是一个Condition profile的写照，显示所有相关的profiles和ValueSet。

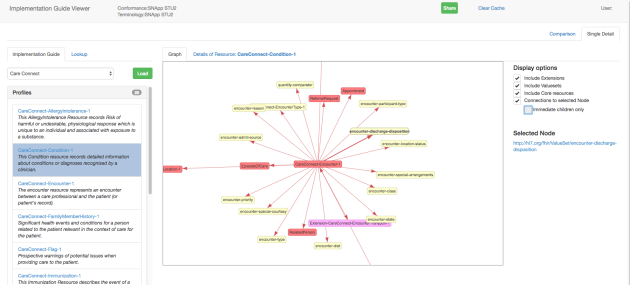


Here it is showing the ‘extended family’:



You’ll definitely need to zoom to see detail – here’s an example:

需要放大以查看详细信息 - 以下是一个示例



There are still a number of enhancements planned – for example the ability to actually edit the Implementation Guide but even in its current form it’s useful to help see what is in an Implementation Guide. I’m also not going to guarantee that it is correct! (More testing would be a real help  )

计划还有一些功能，例如：实际编辑实施指南的能力。即使在目前的形式下，帮助您了解“实施指南”中的内容是有用的。我也不会保证是正确的！

If you do want to try it out on another Implementation Guide, then loading one on a FHIR server (making sure all the referenced resources are also on the same server – ValueSets on the Terminology Server) and setting that server as the Conformance server ought to work. I’m pretty sure that there will be plenty of issues with Implementation Guide’s that I haven’t created so let me know of any issues. Also so far I’ve only used the STU2 version of the IG.

如果您想在另一个实施指南中尝试使用，请在FHIR服务器上加载一个（确保所有引用的资源也位于同一服务器上 - 术语服务器上的ValueSets），并将该服务器设置为一致性服务器应该工作。我很确定实施指南中有很多问题，我没有创建，所以让我知道任何问题。到目前为止，我只使用了IGU的STU2版本。

Note that there’s also a potential problem with storage – all the resources are saved in the browser cache to improve performance, so a large Implementation Guide will likely overflow that cache (you’ll get a message and the chance to clear the cache, but it needs to be properly fixed). 请注意，存储还存在潜在的问题 - 所有资源都保存在浏览器缓存中以提高性能，因此大型实施指南可能会溢出该缓存

## 2.6 ‘Extending’ a ValueSet in a profile

<https://fhirblog.com/2017/07/20/extending-a-valueset-in-a-profile/#more-160524>

JULY 20, 2017 [LEAVE A COMMENT](https://fhirblog.com/2017/07/20/extending-a-valueset-in-a-profile/#respond)

In the [last post](https://fhirblog.com/2017/07/20/default-namespaces-in-hl7-v2/) we talked about [ValueSets and CodeSystems](http://hl7.org/fhir/terminology-module.html) – and in particular how the ValueSet can be thought of as the set of possible values from one or more CodeSystems for a particular element in a given context.

讨论了[ValueSets和CodeSystems](http://hl7.org/fhir/terminology-module.html)，特别是如何将ValueSet视为给定上下文中特定元素的一个或多个CodeSystems的一组可能值。

As you know, the spec provides ValueSets and bindings for all coded elements, and a common need when profiling is likely to be to ‘extend’ the set of possible values – take the contents already in the ValueSet and add others. What’s the best pattern to do that?

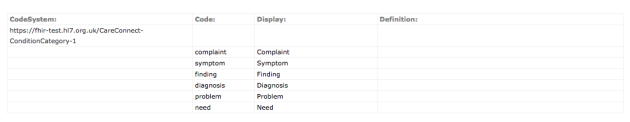
该规范为所有的编码元素提供了ValueSet和binding，并且在profiling可能扩展“可能的值集”，通常需要在ValueSet中的内容中添加其他的元素。

This thought was triggered when reviewing the new [CareConnect](https://nhsconnect.github.io/CareConnectAPI/) profiles from the UK. This is an initiative being driven by the [interOpen](http://www.interopen.org/) project (disclaimer: I’m a vendor representative to the board) and they have published an initial set of profiles and API’s (technically an Implementation Guide) which will going for ballot shortly by [HL7 UK](http://www.hl7.org.uk/).

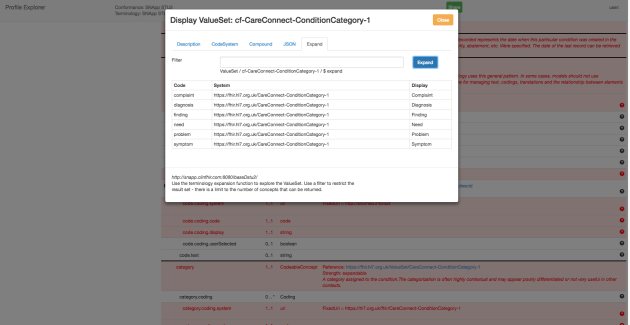
当从UK中检索新的CareConnect profile的时候，引发了这种想法。这是由[InterOpen](http://www.interopen.org/)项目（免责声明：我是董事会的供应商代表）推动的一个举措，他们已经发布了一组初始的配置文件和API（技术上是一个实施指南）。

Take a look at [the profile](https://fhir-test.hl7.org.uk/StructureDefinition/CareConnect-Condition-1) for the Condition resource. What you can see is that the Condition.category field is bound to a [new ValueSet](https://fhir-test.hl7.org.uk/ValueSet/CareConnect-ConditionCategory-1) which takes the current values from the spec and adds a few others: (They also tightening the binding from ‘example’ to ‘extensible’, which makes sense).

看Condition资源的profile，可以看到Condition.category字段绑定到一个新的ValueSet中，从规范中获取当前值，并添加一些内容（也将绑定从“example”转为“可扩展”）



(Here’s a screenshot of what it looks like when viewed in [clinFHIR](https://fhirblog.com/2017/06/26/clinfhir-profile-viewer/))

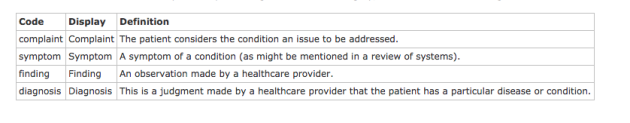


The interesting thing is that all of the actual values in this ValueSet point to concepts from a new Code System with the Url <https://fhir-test.hl7.org.uk/CareConnect-ConditionCategory-1>.

该值集中的所有实际值都指向url为<https://fhir-test.hl7.org.uk/CareConnect-ConditionCategory-1>的新的编码系统。

Now, there’s nothing wrong with this approach – but it could be improved.

Here’s the contents of the [ValueSet](http://hl7.org/fhir/DSTU2/valueset-condition-category.html) from the spec:



(Note that it’s STU-2). You can clearly see that the values from the codesystem in this set have been duplicated in the CareConnect one – and so are now in a new system.

Why is this undesirable? Well it’s just that it takes a little away from semantic interoperability. We now have 2 codes for the ‘symptom’ concept. One from <http://hl7.org/fhir/ValueSet/condition-category> and one from <https://fhir.hl7.org.uk/CareConnect-ConditionCategory-1> – but they both mean the same thing. More work for the consumer.

我们现在有两个“症状”概念的代码。一个来自<http://hl7.org/fhir/ValueSet/condition-category>，一个来自<https://fhir.hl7.org.uk/CareConnect-ConditionCategory-1> - 但它们都意味着相同的事情。更多的工作为消费者。

A better approach would be to define the contents of the CareConnect ValueSet as follows:

* **complaint**(<http://hl7.org/fhir/ValueSet/condition-category>) 主诉
* **symptom**        (<http://hl7.org/fhir/ValueSet/condition-category>) 症状
* **finding**            (<http://hl7.org/fhir/ValueSet/condition-category>) 查找
* **diagnosis**        (<http://hl7.org/fhir/ValueSet/condition-category)> 诊断
* **need**                 (<https://fhir.hl7.org.uk/CareConnect-ConditionCategory-1>) 需要
* **problem**          (<https://fhir.hl7.org.uk/CareConnect-ConditionCategory-1>) 问题

In this way, the 4 concepts that are in both ValueSets come from the same CodeSystem making it just that little bit easier for the consumers. (You could also argue that all of them should come from SNOMED – but I’m not going to go there!) 以这种方式，ValueSet中的4个概念来自相同的CodeSystem，使消费者更容易一点理解这一点。

BTW – I’m not intending to criticize CareConnect – far from it! It’s great to see the sector coming together to do this work – with National buy-in – and to make the work publicly available for comment. These are issues that will arise very commonly by everyone involved in profiling FHIR and it’s great to be able to share the learnings and issues that arise with the FHIR community when doing so.

Indeed, one of the precepts for [interOpen](http://www.interopen.org/about) is to be – well – open, and this is a good example of that.

## 2.7‘Default’ namespaces in HL7 v2

<https://fhirblog.com/2017/07/20/default-namespaces-in-hl7-v2/#more-160495>

JULY 20, 2017 [1 COMMENT](https://fhirblog.com/2017/07/20/default-namespaces-in-hl7-v2/#comments)

So I had an interesting question this week about coded values when importing data from HL7 v2 messages into a data store, and then subsequently exporting them in FHIR interfaces.

有趣的问题是：将数据从HL7 v2消息导入到数据存储中，然后在FHIR接口中导出数据。

The specific question was what should the ‘default’ namespace be when the namespace is not specified in the v2 message?

具体问题是在V2消息中没有指定命名空间的时候，“默认”命名空间应该是什么？

The message and field in question is the AL1-2 “Allergen type Code” that you find in ADT messages. It’s a CWE (Coded With Extensions) datatype, and there’s a user defined table of suggested values – number [0127](http://hl7.org/fhir/v2/0127/index.html).

在ADT消息中查找的问题的消息和字段是AL1-2“Allergen type Code”。它是一个CWE（编码扩展）数据类型，并且有一个用户定义的建议值——number0127。

I must admit that my first reaction was ‘”It’s a user defined table with example values so just use the namespace (or system) for that table and add the value to it”. But on reflection (to be honest, after discussion with a colleague more detail orientated than I am – you know who are [Rod](https://www.linkedin.com/in/rod-mcdonald-16125851/)!) that’s wrong.

我的第一个反应是：这是一个包含示例值的用户自定义的表，因此只需使用该表的命名空间，并将其添加到表中。但是仔细想想，这是错误的。

Let’s start with how we need to correctly represent the data in the repository. Being a coded element, we need to have (at a minimum) the Code and the Code System  from which it is defined. For example if this were a diagnosis code, we might use SNOMED to encode it – in which case the coding system would be ‘SNOMED’ (or [the url](http://hl7.org/fhir/snomedct.html) that represents SNOMED) and the code for the specific concept in SNOMED.

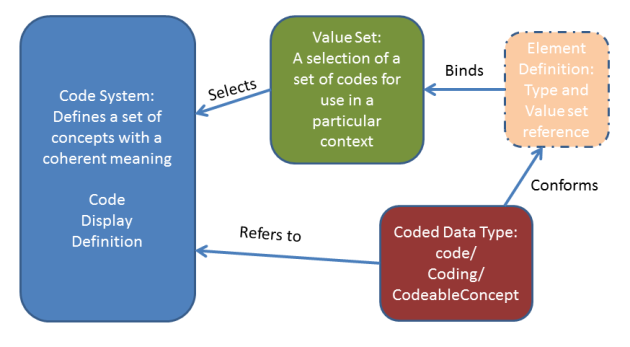
开始的时候，需要正确表示存储库中的数据。对于一个编码元素，需要具有定义的code和code system。例如：这是一个diagnostic code，我们将会使用SNOMED进行编码，在这种情况下，编码系统将是“SNOMED”（或表示SNOMED [的URL](http://hl7.org/fhir/snomedct.html)）和SNOMED中特定概念的代码。

So what is the coding system here? Well, it’s a user defined table, but there are a number of example values in the v2 spec, so we can think of that as the codesystem. That’s all well and good, but what happens when we get a value that’s not in the spec? What codesystem do we use then? Can we just add the code to the codesystem/table?

这里的编码系统是什么？ 这是一个自定义的表，但是在V2规范中有许多示例值，我们可以将其视为编码系统。但是，当获取不符合规范的值的时候，会发生什么？我们应该使用什么代码系统，还可以将code添加到code system中吗？

The short answer is no, we can’t. We’re not the custodian of the values in the codesystem – HL7 is. To help understand how we solve this, take a look at the following picture (which we’ve seen before) from the [FHIR Spec](http://hl7.org/fhir/terminology-module.html) (the overall concepts are the same in v2 & FHIR)

简短的答案是否定的。我们不是codesystem的取值的保管者（HL7是保管者）为了帮助如何解决这个问题，参看下面FHIR规范的内容。



Note that the data in the ‘instance’ (lower right – which will be the v2 message and its representation in the database) refers directly to the CodeSystem, while the ‘element definition’ (upper right – which is the equivalent of the v2 specification) refers to a ValueSet (that in turn refers to a CodeSystem – actually one or more codesystems).

注意：“instance”中的数据（右下角是v2消息及其在数据库中的表示）直接指向CodeSystem，而“element Definition”（右上角相当于V2规范）指向ValueSet（相继指向CodeSystem——实际上是一个或多个编码系统）。

An important point here:

*The actual definition of a code – and the details that need to be in the repository – come from the CodeSystem. The ValueSet is just a selection of codes from one or more CodeSystems in a given context*

重点：code的实际定义——存储库中需要的详细信息来自CodeSystem——来自CodeSystem。ValueSet仅仅是在一个给定的上下文中，来自多个CodeSystem的codes选择。

So applying this to the v2 spec, what is happening is that the contents of a given field is *bound* to a ValueSet – which happens to include all the values from a CodeSystem (being the table in the spec). And in this case, it’s a user defined table with ‘suggested’ values – and the datatype is CWE ([Coded With Extensions](http://wiki.hl7.org/index.php?title=CWE)), so the binding strength in FHIR terms will be [extensible](http://hl7.org/fhir/terminologies.html#extensible) (we really should use these values if the meanings match up, but we can add others).

将其用于V2规范中，给定的字段内容绑定到一个值集中，恰好包含了codesystem的所有值。在这种情况下，是一个具有“建议的”值的用户自定义表，数据类型是CWE（含有扩展码）所以FHIR术语中的绑定强度将是[可扩展的](http://hl7.org/fhir/terminologies.html#extensible)（如果意义匹配，我们真的应该使用这些值，但我们可以添加其他）。

Here’s a screenshot from the v2 spec:



In the top table seq 2 refers to the ‘ValueSet’ 0127, which in turn refers to the contents of the ‘CodeSystem’ 0127 (the lower table).

在顶部的表格中，Seq 指的是‘ValueSet’ 0127，反过来指的是“CodeSystem”0127（下表）的内容。

This distinction between ValueSet and CodeSystem is really important – and really confusing! (especially when they seem the same…)

ValueSet和CodeSystem之间的区别很重要，但是也很混乱。（特别是当他们看起来一样）

Think of the ValueSet as being a ‘design time’ thing – “here are the codes’ you can/should use”, while the CodeSystem is the definition of an actual code in a real message. The ValueSet is really just a pointer to the actual codes defined by the CodeSystem (even if they seem to be the same thing) – it’s the concepts from the CodeSystem that will be in the message/database – not the ValueSet.

考虑到ValueSet是一个“设计时间”的事情，这里是可以或者应该使用的代码，而codesystem是真实消息中实际代码的定义。ValueSet只是一个指向由codesystem定义的实际编码的指针，这是codesystem中将在消息或数据库中的概念而不是ValueSet。

So with all that background, how do we answer the original question? What do we do if we get a value in the AL1-2 field that doesn’t have a namespace? Well, the right thing to do is to see if the value exists in the HL7 Table (CodeSystem). If it does, then it’s likely safe to assume that the CodeSystem is the HL7 one – so we can use that. But if it doesn’t, then we need to associate it with another CodeSystem when we store it. How we do that is up to the implementation. We could:

* Reject the message and make the sender choose a value from the defined set (a bit harsh)
* Use codes from an existing CodeSystem like SNOMED – always preferable if possible
* Create a new [CodeSystem](http://hl7.org/fhir/codesystem.html) for the different codes. This could be agreed in advance with the message sender – or we could add it automatically and add descriptions later. Note that the CodeSystem would likely be specific to the context (“Allergy types not in the spec that we have agreed we’ll use”) – though it doesn’t have to.

在没有命名空间的AL1-2字段中获取值，我们该怎么办？那么正确的做法是看看这个值是否存在于HL7表（CodeSystem）中。如果是这样，那么可以安全地假设CodeSystem是HL7，所以我们可以使用它。但是如果没有，那么当我们存储它时，我们需要将它与另一个CodeSystem相关联。我们如何做到这一点得到实施。我们可以：

拒绝该消息，并使发件人从定义的集合中选择一个值。

使用现有codesystem的代码，例如：SNOMED

为不同的codes创建一个新的codeSystem，可以与消息发送者提前约定或者我们可以自动添加它，并在以后添加说明。注意：codesystem用于具体的上下文中（过敏类型不符合我们已经同意使用的规范）。

But, at the end of the day, the data that we store needs have both a code, and a reference to the CodeSystem where that code is defined.

我们存储的数据需要有一个代码和一个代码定义的CodeSystem引用

So this covers getting the data into the data store in a form where it can be manipulated. There are other issues around exposing them in a FHIR API – we can think about that later.

## 2.8 [EU Data Protection](https://fhirblog.com/2017/06/28/eu-data-protection/) 数据保护

<https://fhirblog.com/>

JUNE 28, 2017 [LEAVE A COMMENT](https://fhirblog.com/2017/06/28/eu-data-protection/#respond)

[Great post](http://www.ringholm.com/column/GDPR_impact_on%20healthcare_data_interoperability.htm) from Rene Spronk on the European Union GDPR (General Data Protection Regulation) standards in the context of healthcare interoperability…

## 2.9 clinFHIR profile viewer

<https://fhirblog.com/2017/06/26/clinfhir-profile-viewer/#more-160438>

JUNE 26, 2017 [2 COMMENTS](https://fhirblog.com/2017/06/26/clinfhir-profile-viewer/#comments)

Over the years I’ve made a number of attempts to build a profile viewer – to a mixed amount of success. The issue is becoming more urgent though, as profiles (as part of Implementation Guides) start to become published, and vendors such as Orion Health need to think about how we are to support them.

The issue is even more important for vendors in the international space, as our solutions are going to have to support different profiles in different countries, and we cannot assume that the profiles will be in alignment even for the same concept.

我们的解决方案将不得不支持不同国家的不同配置文件，我们也不能假定配置文件将在同一个概念上保持一致。

We’re not going to solve that issue right now (though it does highlight that the developers of profiles need to be aware of it and ideally working to avoid it as much as possible), but the ability to view profiles from different jurisdictions and analyse them in a common way is going to be important.

现在不会解决这个问题（尽管突出说明profiles的开发人员需要认识到这一点，并且尽量避免这一点），但是可以从不同的司法管辖区查看profile并以一种共同的方式分析他们。

So this post documents the latest attempt in clinFHIR to develop a Profile Viewer that will work with any compliant profile.

这篇文章记录了clinFHIR的最新尝试，以开发一个可以与人格兼容的profile配合使用的Profile Viewer。

We tend to use the term ‘Profile’ a bit loosely, so let’s be a bit more specific about what we need the app to do. We want to:

* Be able to view the restrictions, additions and changes on a single resource type
* View the extension definitions that are included
* Understand the ValueSets used – whether a new one on a ‘core’ element, or as part of a coded extension
* Review other ‘conformance’ resources such as ConceptMap and NamingSystem that might also be needed
* Show the differences between profiles that are adapting the same core resource type

倾向于使用“Profile”，更具体地说明所需的应用程序：

能够查看单个资源类型的约束、添加和更改；

查看包含的扩展定义

了解使用的ValueSet，是否需要在“core”元素中建立一个或作为编码扩展的一部分

查看“conformance”类的其他资源，例如：所需的Conceptmap、NamingSystem资源。

显示适用于相同core资源类型的profile之间的差异。

Given that it is common that these profiles are published as part of an Implementation Guide, it makes sense that we use the [ImplementationGuide](http://hl7.org/fhir/implementationguide.html) resources to group all the resources in a single UI rather than as discrete resources.

鉴于这些profile作为“实施指南”的一部分，因此使用ImplementationGuide资源将所有资源分组在单个UI中，而不是离散资源。

To build the app, I started with the ‘[CareConnect](https://nhsconnect.github.io/CareConnectAPI/)’ conformance artifacts that are being built by the [interOpen](http://www.interopen.org/) project in the UK. They are being created using the [Forge](https://fhir.furore.com/forge/) profiling tool, and stored in a [GitHub repository.](https://github.com/nhsconnect/CareConnectAPI/tree/develop) So I cloned the repo onto my computer, then created a small [nodejs](https://nodejs.org/en/) application that iterated through the files, uploaded them to a FHIR server (an instance of HAPI 2) and created an ImplementationGuide resource as it went. This means that the Profile Viewer can download the ImplementationGuide resource, and use that to display all the different parts that comprise the careConnect ‘profile’ (which we should really call an Implementation Guide I guess).

为了构建app，开始使用由英国的[InterOpen](http://www.interopen.org/)项目构建的“ [CareConnect](https://nhsconnect.github.io/CareConnectAPI/) ”一致性工件。它们正在使用[Forge](https://fhir.furore.com/forge/)分析工具创建，并存储在[GitHub存储库中。](https://github.com/nhsconnect/CareConnectAPI/tree/develop)将repo克隆到我的电脑上，然后创建了一个小型的[nodejs](https://nodejs.org/en/)应用程序，来遍历文件，并将它们上传到FHIR服务器（HAPI 2的一个实例），并创建了一个ImplementationGuide资源。这意味着Profile Viewer可以下载ImplementationGuide资源，并使用它来显示构成careConnect'profile'的所有不同的部分（应该称之为实现指南）。

To see this in operation, start [clinFHIR](http://clinfhir.com/) and set the servers to ‘snapp 2’ (which is where I uploaded the resources to). On the left pane, you’ll see that there are now 2 tabs. Select the ‘experimental’ tab, and then the profile Browser from there.

查看此操作，启动clinFHIR并将服务器设置为“snapp2”（这是上传资源的地方）左侧窗格中，看到2个选项卡，并选择“experimental”选项卡，并选择profile 浏览器

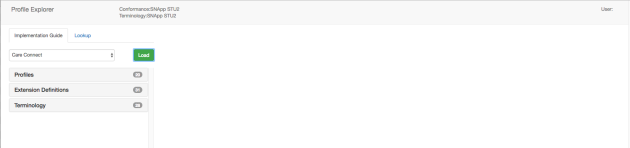
As you can see, when it loads it reads all the ImplementationGuide resources it finds on the conformance server and displays them in the dropdown. Select ‘careConnect’ and then load.

如上所述，加载它的时候，读取在conformance服务器中找到的所有ImplementationGuide资源，并将其显示在下拉列表中。选择“careConnect”并进行加载。

The app examines the ImplementationGuide and generates a display with the resources in the guide in an accordion to the left (divided into profiles, extension definitions and terminology resources).

该app检查ImplementationGuide，并在向导的左边显示资源（分为Profile、extension和术语资源）

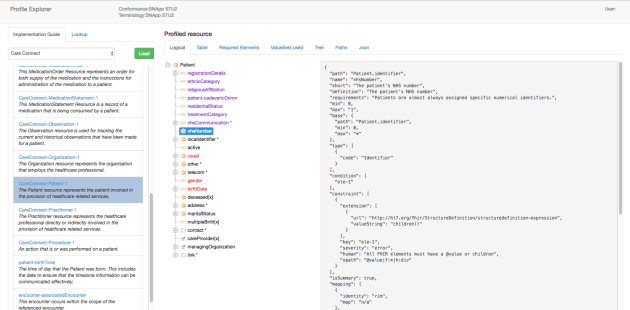
Here’s a screen shot of the resultant page:



The app groups all the resources into 3 main groups:

* Profiles on individual resource types 单个资源类型的profiles
* Extension Definitions used by those profiles 这些profile使用的扩展定义
* Terminology artifacts (like ValueSets) 属于工件（例如ValueSet）

Select a profile – say the Patient profile, and details of that profile are then displayed in the right pane. 选择一个profile（Patient profile），其详细信息显示在右侧窗格中。



There are a number of different views that are available. 右侧窗格中有不同的视图（选项卡）

The **Logical** tab (shown above) shows a simple tree view of the profile. Extensions are retrieved and a human readable name displayed (from the ‘name’ element in the profile). It is intended to give a high level overview of the profile – suitable for a clinician or business analyst who just wants to know what information is being represented, rather than the details of the profile. Extensions are represented in purple, required elements are red, and an asterisk (\*) indicates that the element can repeat.

Logical选项卡：显示profile的简单的树状视图，可以检索扩展并显示人可读的名字（profile中的“name”字段）。目的是高度概述profile：适用于临床医生或者业务分析师需要知道的正在表示的信息，而不是profile的详细信息。扩展以紫色表示，所需的元素为红色，星号表示元素可以重复。

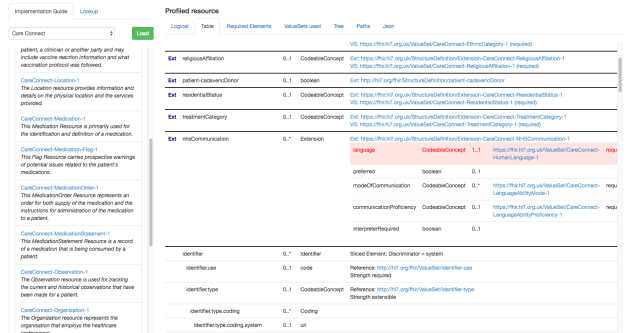
Note that the tree view will not necessarily correspond directly to the core resource paths. For example, in the Patient profile there is both an nhsNumber and a localIdentifier. These are both at the Patient.identifier path in the profile.

注意：树状视图不一定对应于核心资源paths。例如：Patient profile中，存在nhsNumber 和 localIdentifier。这些都是在profile中的Patient.identifier路径上。

The **Table** tab is intended for more in-depth review.  Each row in the table represents a ‘row’ from the profile (strictly speaking an ‘ElementDefinition element from the profile StructureDefinition.snapshot.element) – with some un-needed ones removed to reduce the ‘clutter’.

Table选项卡：目的是更深入的审查。表格中的每一行表示一个profile（严格上来说，来自profile的StructureDefinition.snapshot.element的ElementDefinition字段）——删除了一些不必要的元素以减少“杂乱”。

Here’s an example:



There are some thicker horizontal lines that group together the ‘expanded’ datatype children for elements with complex datatypes (if they are expanded in the profile).  This is because the children of complex datatypes can be profiled as well as the elements – for example you can specify that an identifier must have a system – and the actual value of the system as well – as we will see shortly.

较粗的水平线将“扩展”数据类型的子元素与复杂数据类型（如果在配置文件中扩展）组合在一起。这是因为与元素一样也可以对复杂数据类型的子级进行概要分析- 例如，可以指定必须具有的系统标识符- 以及系统的实际值 - 我们将尽快看到。

There are 4 columns.The first column is the path in the profile. If it’s an extension then the path is replaced by the name of the extension (from the extension definition) and the word ‘Ext’ is displayed in front of it.

有四列内容。第一列是profile的path，如果是扩展，则以扩展的名字取代path，并且“Ext”显示在前面。

Next is the multiplicity of the element. If required (min = 1) then the whole row is colored red.

接下来是元素的多重性，如果需要（min=1），则整行为红色。

Then the datatypes are shown. (There are multiple possible types in some cases of course). If the datatype is a reference (either to a profiled resource in the ImplementationGuide or a ‘core’ resource type) then the datatype is a hyperlink that, if clicked, will load that profile and display it. (btw – at the moment the accordion display to the left is not updated to indicate a new profile is being displayed – I’m working on that  )

然后显示数据类型。（当然在有些情况下，可能有多种可能的类型）。如果数据类型是一个引用（不论是ImplementationGuide中的profiled资源还是核心资源类型），这个数据类型是一个超链接，点击之后，将加载profile并显示它。（目前左侧窗格并没有显示更新，表示正在显示一个新的profile）

The fourth column serves a number of purposes.

第四列有很多用途。

If the row is an extension, then: the url for the extension is displayed (as a hyper link)

如果这一行是扩展，则第四列显示扩展的url。

If the extension is to a coded datatype, then the url of the ValueSet is displayed. Clicking on the hyperlink will show the ValueSet browser in a modal dialog so you can see the concepts within it.

如果扩展是一个编码数据类型，显示值集的url。单击超链接在模态框中显示ValueSet浏览器，以便在其中查看概念。

If the extension is a complex extension (ie has its own children) then a  internal table is shown with a row for each child indicating the name, datatype, multiplicity, ValueSet and binding strength (with a hyperlink for any ValueSets).

如果扩展是一个复杂的扩展（即有自己的子节点），则会显示一个内部表，其中每个子节点都有一行，显示名称、数据类型、多重性、ValueSet和绑定强度。

If the row is an existing element from the base, then the valueset url and binding strength is displayed.

 如果该行是基本的现有元素，则显示值集的url和绑定强度。

One aspect that I‘m not sure is rendered nicely yet are ‘sliced’ elements.

[Slicing](http://hl7.org/fhir/profiling.html#slicing) is a technique whereby an element that is multiple in the spec is ‘sliced’ into sub-elements with specific properties. The [example from the spec](http://hl7.org/fhir/profiling.html#slicing) is of Observation.component where the profile uses slicing to specify the Observation.component.code values that are expected in the profile (in this case a Blood Pressure with a systolic and a diastolic value).

Slicing是一种技术，在规范中可以将多重性的元素切片为具有特定属性的子元素。该规范中实例是：Observation.component，其中profile可以使用slicing来指定在profile中预期的Observation.component.code的取值。（此例子是收缩压和舒张压）.

When you define a slice on an element in the profile, you indicate the element property that is used to distinguish between slices – in this case it would be Observation.component.code – and this is a ‘row’ in the profile. In the profile viewer, the word: ‘Sliced Element:’ appears – followed by the distinguishing element (termed the ‘discriminator’). After this are the details of the sliced elements (which can get quite detailed).

当在profile中元素上定义一个slice的时候，应该指出区别slices的元素属性——即Observation.component.code来区别。在profile viewer中，出现：'Sliced Element：'，后跟区分元素（称为“鉴别器”）。之后是切片元素的细节（可以得到很详细的）。

To see this in operation, take a look at the CareConnect-Patient-1 profile.

要查看此操作，请查看CareConnect-Patient-1 profile.

You’ll see that there are quite a few rows starting with ‘identifier’ and that they are in 3 ‘groups’ (as defined by the thicker horizontal lines).

可以看到多行以“identifier”开头。

The first group represents the discriminator element – it’s the ‘system’ element. In other words, the individual ‘slices’ of Identifier will each have a different value for ‘system’. This group also includes all the child elements for an Identifier datatype. This contains the ‘expected’ values for these fields – for example that Identifier.type and identifier.use elements are coded, and giving the ValueSets and bindings.

第一组表示鉴别器元素：system元素。换句话说，Identifier的单个slices将会有不同的system值。这个分组常常包括Identifier数据类型的子元素。包含这些字段的“expected”值，例如编码的Identifier.type和identifier.use元素。并给出了ValueSet和binding。

The second group is a slice that represents the NHS number, and specifies that the system is required with a fixed value. There is also an extension on this slice to indicate the verification status which is drawn from a specific ValueSet. There can only one of these types of identifier in an instance, which is optional.

第二个分组表示NHS编号的slices，并指定需要具有固定值的system。此切片上有一个扩展名，用于指示从特定ValueSet中提取验证状态。在一个实例中，只能有一种类型的标识符，这是可选的。

The third group is multiple and the system is fixed as well. (I’m not quite clear why this system is fixed – it seems to specify a ‘local’ identifier, but the value of which still needs to be unique within that system).

第三个分组是具有多重性，固定的system（我不清楚为什么这个系统是固定的 - 它似乎指定一个“本地”标识符，但其值在该系统中仍然是唯一的）

Looking at the table, you can also see that the *name* element is also sliced. Slicing can become really complicated – and creating a generic display for sliced elements is not going not be straightforward

查看表格，可以看到分片的name元素。slicing变得非常复杂，为切片创建通用的显示并不是很简单。

The other tabs are really more intended for techies – and for helping to develop the App. ‘Required Elements’ and ‘ValueSets used’ is self explanatory, while the ‘Paths’ tab is a 1:1 rendition of the StructureDefinition.snapshot.element array. Json is, of course, the raw Json for the profile.

其他的选项卡适用于技术人员，并帮助开发应用程序。“required Element”和“ValueSet”用于自解释。而“Path”选项卡是StructDefinition.snapshot.element数组的1：1。JSON选项卡是原始的Json的profile。

The other parts of the app are self-explanatory, so I won’t spend time on them here.

It’s early days for the app, so expect it to change over the coming months. In particular I’m keen to have some form of ‘differencing’ function – to be able to compare 2 profiles to show where they differ.

At the moment, if you want to view the Argonaut ImplementationGuide – select the SNAPP R3 server (it’s is a STU-2 ImplementationGuide, but seems to be represented in R3) – I expect that will be corrected in time.

## 2.10 Changing a ValueSet in a profile

https://fhirblog.com/2017/06/21/changing-a-valueset-in-a-profile/#more-160379

JUNE 21, 2017 [LEAVE A COMMENT](https://fhirblog.com/2017/06/21/changing-a-valueset-in-a-profile/#respond)

I learned something today.

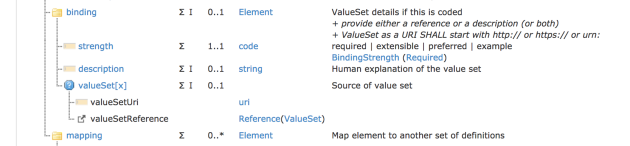
Actually, most days I do learn something (and occasionally remember it later on) but this one is worth recording here.

One of the actions that is commonly done when profiling a resource is to change the binding of a coded element to some other ValueSet than the one in the spec. For example, in the UK [CareConnect](https://nhsconnect.github.io/CareConnectAPI/) [profiles](https://github.com/nhsconnect/CareConnectAPI/tree/develop), the Patient.gender element is bound to a different ValueSet.

在profile资源的时候，通常会执行的操作是将编码元素的binding更改为其他的值集，而不是该规范中的值集。例如：在UK [CareConnect](https://nhsconnect.github.io/CareConnectAPI/) profiles，中Patient.gender绑定了一个不同的值集。

In the StructureDefinition resource that defines the profile, this done by changing the binding – here’s the entry in the [StructureDefinition](http://hl7.org/fhir/structuredefinition.html) (actually a child [ElementDefinition](http://hl7.org/fhir/elementdefinition.html#ElementDefinition)) where this can be done:

在定义profile的StructureDefinition资源中，更改绑定强度。通常在StructureDefinition的entry（实际上是一个ElementDefinition）中完成强度的更改。



Note that there are 2 ways to do this – either the valueSetUri element or a valueSetReference. In the past, I’ve assumed that the valueSetReference (being a standard resource reference to a FHIR resource) required that the ValueSet be present at the URI that is the value of the element, whereas the valueSetUri refers to the canonical URL for the ValueSet (the ValueSet.url element).

注意：两种方法可以执行此操作：ValueSetURI元素或者ValueReference。当为ValueSetReference的时候（引用标准中的FHIR资源）要求以元素uri值表示ValueSet。而ValueSetURI指的是ValueSet的规范URL（valueset.url元素）

So I’ve always used the valueSetUri, as I believed that it supported a registry of ValueSets (as well as other conformance resources) – you can query the registry with a query like:

一直使用valueSetUri，因为我相信它支持一个ValueSets注册表（以及其他一致性资源） - 您可以使用以下查询来查询注册表：[host]/ValueSet?url={value)

But it turns out that that’s not quite right.

* **valueSetUri** is intended to be used when the profile is not pointing at an explicit ValueSet resource – perhaps a document like an RFC that describes the options.
* **valueSetReference** is intended to be used when the profile IS pointing to a ValueSet (or to an [implicit ValueSet](http://hl7.org/fhir/loinc.html#implicit)). It SHOULD resolve, but if it doesn’t then it’s perfectly permissible to look it up in a registry. This isn’t the way that ‘ordinary’ resource references work, but does for resources that have a canonical URL (like [CodeSystem](http://hl7.org/fhir/codesystem.html), [ConceptMap](http://hl7.org/fhir/conceptmap.html) and [CodeSystem](http://hl7.org/fhir/codesystem.html)), and is intended to specifically support saving these resources in a separate registry (like [Simplifier](https://simplifier.net/)) if they can’t be directly resolved.

但是事实证明，这并不完全正确。

ValueSetURI：当Profile没有显式指向ValueSet的时候，使用ValueSetURI，可能是描述选项的RFC文档。

ValueSetReference：当Profile IS执行ValueSet的时候使用ValueSetReference。应该在注册表中查找他，这不是“ordinary”资源引用的方式，而是具有规范URL（如： [CodeSystem](http://hl7.org/fhir/codesystem.html), [ConceptMap](http://hl7.org/fhir/conceptmap.html) and [CodeSystem](http://hl7.org/fhir/codesystem.html)）的资源，并且当不能直接解决的时候，旨在专门支持将这些资源保存在单独的注册表中（simplifier）。

So there you are. If you are using a ValueSet to define the options for a coded element (which is always a good idea) you should always use valueSetReference.

如果您使用ValueSet定义编码元素的选项（这总是一个好主意），则应始终使用valueSetReference。

Now I need to go fix clinFHIR. Bother.

**Addendum**:

As I thought more about this, I realized that this behaviour is not that different to an ‘ordinary’ reference. The reference is saying “The resource can be found at this location” – in fact it’s saying “the **most recent** version of this resource can be found here”. Because the location is globally unique, there’s nothing stopping there a copy being somewhere else – indeed, that’s exactly what we do in a Bundle – especially a Document bundle’. And you can even be [explicit about the version](http://hl7.org/fhir/resource.html#versions) if you need to (as a copy may not be the most recent version of course).

## 2.11 Viewing resource instances in clinFHIR

<https://fhirblog.com/2017/06/15/viewing-resource-instances-in-clinfhir/#more-160350>

JUNE 15, 2017 [LEAVE A COMMENT](https://fhirblog.com/2017/06/15/viewing-resource-instances-in-clinfhir/#respond)

One thing you need to do quite frequently in the FHIR world is to look at resource instances (whether in XML or JSON), and this can be quite complex.

在FHIR中经常需要做的一件事就是，查看资源实例（XML或JSON），这可能是非常复杂的。

A little trick I use quite frequently is to use the [clinFHIR](http://clinfhir.com/) scenario builder to create a hierarchical “tree view” of the resource, which I find easier to review than the raw format.

经常使用的一个小技巧就是使用clinFHIR场景构建器来创建资源的分层“树状图”，这笔原始的格式更容易查看。

To do this, follow these steps.

Load clinFHIR and select the Scenario Builder (It doesn’t really matter which server you choose as we’re not going to be saving the resource to a server, but it’s probably a good idea to choose one that matches the FHIR version of your resource, especially if you are importing an XML bundle). This is because scenarios are saved on the local machine.

加载clinFHIR并选择Scenario Builder（选择的服务器并不是很重要，因为不会将资源保存在服务器中，但是最好服务器的FHIR版本要与资源匹配，特别是正在导入XML包）。这是因为在本机中保存了场景。

Next create a new scenario.

On the nav bar at the top of the screen, select the option ‘Import resource’. This will open a dialog into which you paste your resource. You have a couple of options here.

* You can past in a single JSON resource (preferred approach)
* You can paste in a bundle containing the resource/s in either XML or JSON format. If you do paste in XML, then clinFHIR will use the ‘Bundle’ endpoint of the current conformance server to save the bundle as XML, and then read it back as JSON – which is why the server FHIR version should match the resource.

接下来创建一个新场景：

在顶部导航栏中，选择“Inport Resource”选项。这将打开一个对话框。可以对话框中粘贴资源，有几个选项：

可以使用单个JSON资源

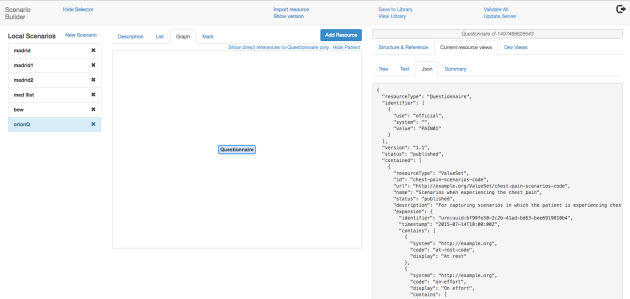
可以是以XML或JSON格式的资源包。如果粘贴XML，则clinFHIR将会使用当前conformance服务器的“Bundle”端点作为XML，然后将其作为JSON进行读取。这就是为什么服务器的FHIR版本要与资源匹配。

Then click the ‘Import’ button which will appear to the upper right. All going well, the resource/resources will be imported into the scenario. You will get issues if the resource/bundle is mal-formed, but otherwise it seems to work OK. It should also be able to maintain resource reference. Do note that it doesn’t validate the resource instance…

点击右上方的“Import”按钮，将资源导入到场景中，如果出现格式错误，则会收到问题，否则似乎可以正常工作，也因该保持资源引用，注意：不会验证资源实例。

Now that the resource has been imported, you can view them in the right pane. Select the resource in the middle pane (eg the graph view) and then select the ‘Current resource views’ tab in the right pane.

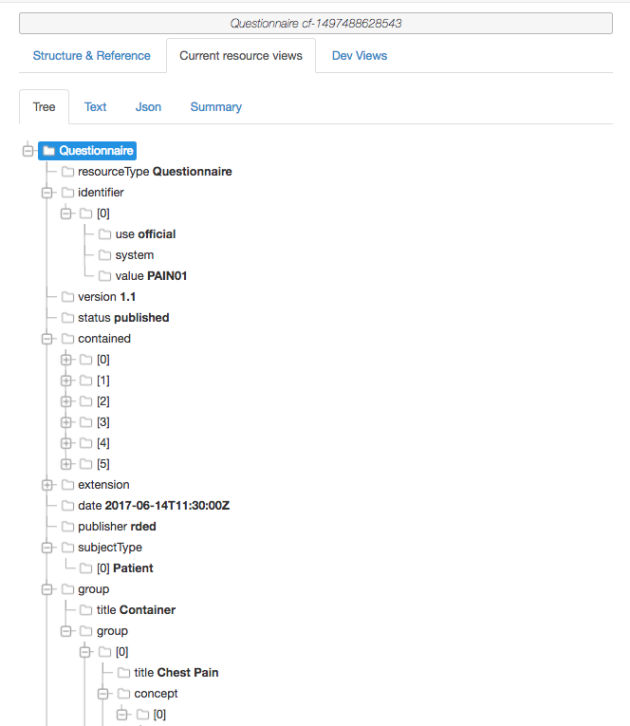
现在已经导入资源，则可以在右侧窗格中查看。在中间窗格中选择资源（例如Graph选项卡），然后右侧窗格中选择“Current Resource view”选项卡



There are a number of view options here

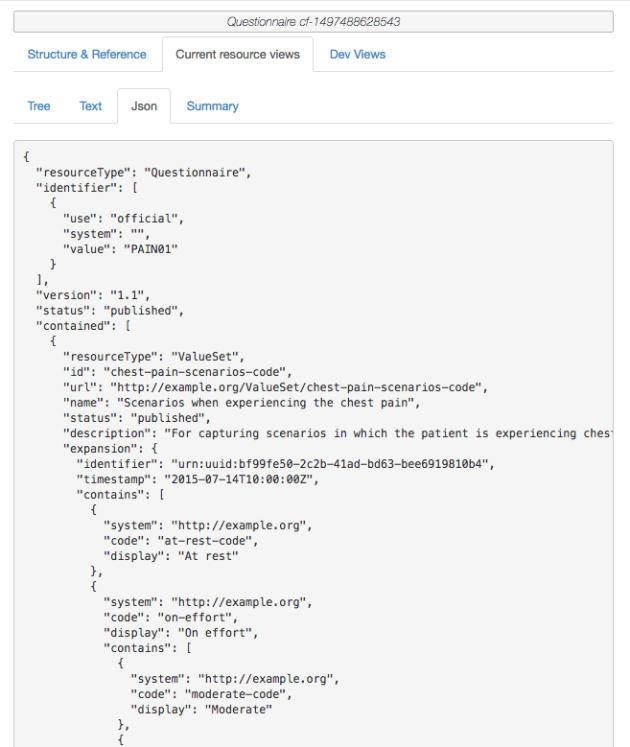
The **Tree view** is the hierarchical view I referred to above: here’s an example of a Questionnaire:

下面显示的内容是 右侧窗格中Tree选项卡的内容



The **Text view** shows the contents of the text element (if any)

The **Json view** shows the raw JSON



The **Summary view** shows incoming & outgoing references.

Summary view选项卡显示了传入或传出的引用。

Note that, in theory, you are able to modify the resource using the ‘Structure & Reference’ tab -including saving to the FHIR server – as you can any other resource. I have come across some issues with more complex resources though (ironically the Questionnaire resource) – so your mileage may vary. Do let me know if you have issues & I see what I can do to resolve them.

注意：从理论上讲，使用“Structure & Reference”选项卡修饰资源——包括将资源保存到FHIR服务器

You can import as many resources into a scenario as you wish – again, it’s best to make them all the same FHIR version, though clinFHIR won’t enforce that.

## 2.11 [Connectathon report](https://fhirblog.com/2017/05/26/connectathon-report/)

https://fhirblog.com/2017/04/29/creating-an-extension-definition-part-3-coded-items/#more-160282

MAY 26, 2017 [LEAVE A COMMENT](https://fhirblog.com/2017/05/26/connectathon-report/#respond)

A report on the recent [FHIR connectathon](http://wiki.hl7.org/index.php?title=FHIR_Connectathon_15) hosted at the [HL7 Site.](http://blog.hl7.org/fhirconnectathon15recap)..

2.12Creating an Extension Definition – part 3: Coded items

<https://fhirblog.com/2017/04/29/creating-an-extension-definition-part-3-coded-items/#more-160282>

APRIL 29, 2017 [4 COMMENTS](https://fhirblog.com/2017/04/29/creating-an-extension-definition-part-3-coded-items/#comments)

A common data type that is likely to be added in an extension are coded types – those where the value in an instance comes from a pre-defined set of possible values. There are a couple of extra things that you need to do for these.

可能在扩展中添加一个常见的数据类型是coded类型：实例中的值来自预定义的可能值集的类型。有些额外的事情，需要做这些。

First, a quick review of the basics.

All coded elements in FHIR (whether in the core spec or a profile) are [bound](http://hl7.org/fhir/terminologies.html#strength) to a ValueSet that describes the set of possible values for the element. The ‘strength’ of the binding determines whether an element in an instance of the resource can have a value that is not in the set. This can vary from a ‘*required*’ strength where the value MUST be in the ValueSet, to ‘*example*’ where the ValueSet is, well, just an example. In general terms, you use the ‘required’ strength sparingly, and only where it is really important to do so (‘status’ elements are the most common examples in the spec).

FHIR中所有的coded元素（无论是核心规范中还是profile中）都绑定到一个ValueSet中，该值集描述了元素的可能值。绑定的强度决定了资源实例中的元素是否可以具有不在值集中的值。可以将元素绑定强度required转为example。必须谨慎使用required强度，只有必须使用的时候，才使用（资源中的status元素是最常见的示例）。

The [ValueSet](http://hl7.org/fhir/valueset.html) gets its contents from one or more [CodeSystems](http://hl7.org/fhir/codesystem.html), where the CodeSystem contains the actual definition of each concept, and the ValueSet is just picking the ones you want for your Use Case. For example you might want to create a ValueSet that has the most common SNOMED diagnoses made in the Emergency Department to use as a simplified ‘pick list’ in an application. The ValueSet would list the SNOMED terms to be included, and you’d likely bind it to the extension with a ‘[extensible’](http://hl7.org/fhir/terminologies.html#extensible) strength to get consistency of use, but allow uncommon diagnoses to be selected.

可以从多个codesystem中获取ValueSet内容，codesystem包含每个概念的实际定义，ValueSet仅仅包含使用的用例。例如;创建一个在急诊中最常见的SNOMED诊断的ValueSet，以在应用程序中用作简化的“pick list”。ValueSet将会列出包含的SNOMED术语，将其绑定到“extension”强度的扩展，以获得一致性使用，但是允许选择不常见的诊断。

‘[Terminologies](http://hl7.org/fhir/terminology-module.html)’ is another word used for CodeSystems (though is often used in a broader context, and can get \*really\* complicated). And to any terminology experts reading this, I know that this description is a simplification – it is a complex domain!

“Terminology”是codesystem中使用的另一个词语（尽管通常在更广泛的上下文中使用，但是可能会更复杂）。对于任何读这些术语的专家，这些描述是一个简化。

It is important to note that in STU2, the ValueSet was able to also define concepts – but in STU3 (Release 3) this is now done by CodeSystem resource

在STU2中，ValueSet还能够定义概念 - 但是在STU3（版本3）中，现在由CodeSystem资源定义概念。

Next, a quick review of coded datatypes in FHIR – there are 3 ½ of them.

The most common type that you’ll use is the [CodeableConcept](http://hl7.org/fhir/datatypes.html#codeableconcept). This is far and away the most flexible one, as it allows you to use codes from different terminologies to represent the same concept by using multiple copies of the Coding datatype (useful when you need to represent the concept in different code systems like a ‘custom’ code and it’s representation in, say, SNOMED) and also has a ‘text’ element that can be used to represent the description of the concept – or when the coded form is not available (distressingly common in the real world!)

最常用的是CodeableConcept类型。这是最灵活的一个。通过使用Coding数据类型的多个副本或者编码形式不可用的时候，允许使用不同术语编码来表示相同的概念，（当您需要在不同的代码系统中表示概念（如“custom” '代码，它表示在SNOMED中），并且还有一个'text'元素，可用于表示概念的描述

[Coding](http://hl7.org/fhir/datatypes.html#Coding) represent a single concept in a single CodeSystem. It has 3 main properties:

* *System* is the uri the identifies the CodeSystem
* *Code* identifies the concept in the code system
* *Display* is the description of the concept in the CodeSystem (This is NOT the same as CodeableConcept.text)

Coding表示在 codesystem中的一个概念，有三个主要的属性：

system是标识codesystem的一个uri

code标识了codesystem的一个概念

display是codesystem的概念描述（与CodeableConcept.text的意思不一样）

Generally you don’t use Coding, as the CodeableConcept is significantly more flexible – unless you are sure that you’ll never need text, or need to represent the concept in multiple CodeSystems.

一般来说，不使用Coding，因为CodeableConcept更加灵活，除非你确定不需要text或者需要在多个codesystem中表示概念。

Next up is the [code](http://hl7.org/fhir/datatypes.html#code) datatype – this is mostly used when there’s a small set of codes and the code is descriptive (there’s no display element). Often used for ‘structural’ elements like Observation.status or where there’s a fixed set of codes that cannot change like AllergyIntolerance.type. The binding strength of a code should always be ‘required’. You can think of code as an [enumeration](https://en.wikipedia.org/wiki/Enumeration) in computer terms – and the libraries generally treat them in that way.

接下来是code数据类型，只有一小段code并且code是具有可描述性的时候，可使用code数据类型。就像Observation.status的“structural”元素或者就像AllergyIntolerance.type没有改变的固定值的编码。code的绑定的强度总是“required”。可以将code作为计算机术语的枚举。

And the ½ coded datatype is [Quantity](http://hl7.org/fhir/datatypes.html#quantity), as it has a coded *Quantity.unit* / *Quantity.system* (generally [UCUM](http://unitsofmeasure.org/trac))

而½编码数据类型为[Quantity](http://hl7.org/fhir/datatypes.html#quantity)，因为它具有编码的*Quantity.unit* / *Quantity.system*

So when creating an extension that has a coded element, you will need to follow these steps.

1. Decide on the actual datatype you need. A safe option will be CodeableConcept.  
2. Determine the ValueSet to use. If you’re lucky there will be one available for you, but if not:

a) Find the CodeSystem/s that have the concepts you need. You may need to build one if they are custom codes, though using a standard is always better. A good starting point for the search is the [terminology module](http://hl7.org/fhir/terminology-module.html) in the spec as there are a number of lists there (including ValueSets).

If you do have to build a CodeSystem, then remember that in STU2 it is part of the ValueSet resource.

b) Build the ValueSet – referencing the CodeSystem/s  
For both CodeSystem and ValueSet you’ll need a url – the [previous post on url’](https://fhirblog.com/2017/04/12/creating-an-extension-definition-part-2-the-url/)s, though focussed on StructureDefinition is still applicable.

3. Determine the binding strength

4. Build the Extension Definition, selecting the coded datatype and referencing the ValueSet with the appropriate strength.

所以当创建一个coded元素的扩展的时候，可以遵循下面的步骤：

1、确定所需的实际数据类型。最安全的数据类型就是CodeableConcept。

2、决定使用的ValueSet。可能具有所需的值集。当没有的时候：

查找需要此概念的codesystem。尽管使用标准中的code更合适，但是如果是自定义的code，则需要建立一个codesystem。搜索的一个好的起点就是规范中的术语模块，因为有很多列表（0包括ValueSet）；

如果不得不建立codesystem，则记住在在STU2中它是ValueSet资源的一部分。

构建ValueSet——引用codesystem。对于codesystem和ValueSet，都需要一个url，但是这侧重于StructureDefinition。

3、确定绑定强度

4 、构建扩展定义，选择coded数据类型，引用合适强度的ValueSet。

[ClinFHIR](http://clinfhir.com/) does offer some tooling to assist with this – though the functionality is quite basic at the moment (and does need to be exercised to find the bugs!)  [ClinFHIR](http://clinfhir.com/)确实提供了一些工具来帮助这一点 - 虽然功能现在是非常基本的

There are other terminology tools being developed (like [Ontoserver](http://ontoserver.csiro.au/snapper2-dev/index.html#/) in Australia, and [Apelon](http://www.apelon.com/solutions/terminology-tooling/dts) in the US) that are far more sophisticated, but I’m not sure if they are openly available. [Forge](https://fhir.furore.com/forge/), is the official tool for creating profiles.

Enjoy your extending!

Other posts in this series:

* Part 1 – [Creating an extension](https://fhirblog.com/2017/04/12/so-you-want-to-create-a-fhir-extension/)
* Part 2 – [Choosing the URL](https://fhirblog.com/2017/04/12/creating-an-extension-definition-part-2-the-url/)

## 2.12 Using Scenario Builder for developing Resources

https://fhirblog.com/2017/04/22/using-scenario-builder-for-developing-resources/#more-160209

APRIL 22, 2017 [LEAVE A COMMENT](https://fhirblog.com/2017/04/22/using-scenario-builder-for-developing-resources/#respond)

I had an email this morning from the organizers of the ‘[Clinicians On FHIR](http://wiki.hl7.org/index.php?title=Clinicians_on_FHIR)’ event at the upcoming [Working Group Meeting](http://www.hl7.org/events/working_group_meeting/2017/05/) asking if it was possible to use the Scenario Builder to examine a resource type that had been altered after publication (i.e. one that was being worked on for the next release of FHIR – R4). As it turns out, this is quite straightforward to do – albeit with some limitations.

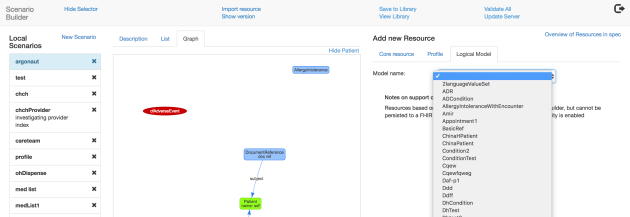
是否可以使用“场景构建器”来检查出版后修改的资源类型。

The way to approach this is to treat the altered resource type (technically a [StructureDefinition](http://hl7.org/fhir/structuredefinition.html) resource of course) as if it were a logical model. We’ve [talked about logical models a bit](https://fhirblog.com/?s=logical), but briefly they are models (or types) that use the FHIR infrastructure (conformance resources, datatypes etc) but whose contents don’t match any of the core resource types. This is not the same as profiling a resource, as you don’t use extensions to add additional elements – you just create the element. This results in a ‘non-conformant’ type, which means that you can’t easily share it and you can’t save it on a validating FHIR server, but it’s perfect for ‘trying things out’.

要解决这个问题的方法是将改变的资源类型（技术是StructureDefinition资源），就像是一个逻辑模型一样。（其中逻辑模型，简单的来说是使用FHIR基础架构——conformance资源、数据类型等等，但是内容与其核心资源类型不匹配）这与profiling资源不一样，就像不能使用扩展来添加其他元素一样，只是创建元素。将导致“不一致”的类型，意味着不能共享，并且无法将其保存在验证的FHIR服务器中，但是可以进行完美的尝试。

To use a logical model in the Scenario Builder, you click the ‘Add resource’ button, and then select the ‘Logical Models’ tab to the right. This exposes a dropdown that lists all the logical models in the current conformance server, and you just select the one you want. Here’s an image:

要在场景生成器中使用逻辑模型，点击“Add resource”，则右侧的“Logical Models”选项卡会显示一个下拉列表，列出当前conformance服务器中所有的逻辑模型，进而可进行自选。



Then you can use it in the scenario like any other resource, including linking to others.

然后可以在场中中的任何资源一样使用它，包括链接到其他资源。

There are a couple of ways of creating the logical model. 有几种创建逻辑模型的方法。

The first is to use the Logical Modeller component in [clinFHIR](http://clinfhir.com/) and just build whatever you want. You can create one based on an existing core type and add/remove/alter elements as needed (often good for quick ‘what if’ scenarios), or just build one from scratch (often useful for capturing requirements). Once built and saved, it will show up in the Scenario Builder as described above. (One trap is that unlike the Scenario Builder, logical models need to be explicitly saved to the server)

第一种方法是：在clinFHIR中使用“Logical component”，只需要构建所需的东西。可以基于已有的核心类型创建元素，并根据需要添加、删除、更改元素。或者从scratch中建立一个。一旦构建并保存，将显示在“Scenario Builder”。（与场景生成器不同，逻辑模型需要显示地保存在服务器中）

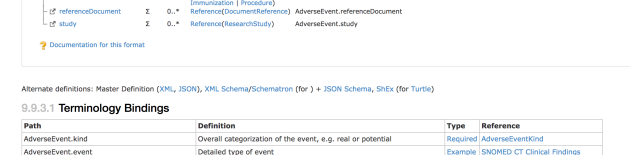
The second way (and the one we’ll use here) is to take the StructureDefinition for a ‘candidate’ resource type being altered (in this case it was changes to the [AdverseEvent](http://build.fhir.org/adverseevent.html) resource), change that to be a Logical Model and save it as such on the conformance server. Then – voila! – you can build instances in the Scenario Builder (but not save them to the data server of course). You will need to be comfortable downloading files and manipulating them directly in an XML (or Json) editor to do this.

第二种方法：将StructureDefinition替换为可变的“候选”资源类型。（在这种情况下，是对AdverseEvent资源的更改），将其更改为逻辑模型并保存在conformance服务器中。可以在场景生成器中构建实例（但是当然不会将它们保存到数据服务器）。需要舒适地下载文件并直接在XML（或JSON）编辑器中进行操作。

Lets take a look at the steps required.需要几个步骤

First, grab the StructureDefinition for the resource. You can find this by going to the [page above](http://build.fhir.org/adverseevent.html) and just below the resource content structure are links to the StructureDefinitions in XML and Json formats (Alternative Definitions).

首先获取StructureDefinition资源（此实例是更改adverseEvent资源类型）



Grab the XML format and open in your favourite XML editor.

获取XML格式并在喜欢的XML编辑器中打开XML内容。

Next, you need to create a new ‘*type*’ element value, that is different to the current one. In this case the type value is ‘AdverseEvent’. I changed it to ‘CfAdverseEvent’ by doing a global replace (AdverseEvent -> CfAdverseEvent) in the file. Note that it must start with a capital letter.

接下来需要创建一个新的“type”元素值（StructureDefinition.type取值），将其更改为CfAdverseEvent。注意：必须以大写字母开头。

Next, I set the *id* to one that would be unique on the conformance server (c*fAdverseEvent* worked for me).

然后在conformance服务器中设置唯一的id。然后将StructureDefinition.kind字段值更改为logical。

Then I changed the value of the ‘*kind*’ element to ‘*logical*’

Finally, I inserted this identifier (as this is how the ScenarioBuilder will find the right files):

<identifier>

        <system value="http:clinfhir.com"/>

        <value value="author"/>

</identifier>

 最后插入identifier（便于查找）

After making these changes I saved the resource on to the R3 HAPI server (I PUTted it to <http://fhirtest.uhn.ca/baseDstu3/StructureDefinition/cfAdverseEvent)>. The server did complain quite a bit as it doesn’t support logical models yet, but it did save it. (Thank you James!).

更改之后，将资源保存在R3 HAPI服务器中（将其输入到http://fhirtest.uhn.ca/baseDstu3/StructureDefinition/cfAdverseEvent）服务器不支持逻辑模型，但是确实保存了。

And that was it! I started up the Scenario Builder and was able to load the Logical model and build a scenario with it. 启动了场景生成器，并且能够加载逻辑模型并使用它来构建场景

So I hope that proves useful – just remember that you can’t save instances based on the logical model to the server (Actually you probably can – but it will drop any elements it doesn’t understand). And there are likely other changes to make it a fully compliant logical model, but this is enough to get started.

## 2.13 A sense of history with clinFHIR

https://fhirblog.com/2017/04/16/a-sense-of-history-with-clinfhir/#more-160166

APRIL 16, 2017 [2 COMMENTS](https://fhirblog.com/2017/04/16/a-sense-of-history-with-clinfhir/#comments)

I’m definitely going to finish off the series on building extensions (we have to cover coded extensions) but I must admit I’ve been a bit sidetracked in the last couple of days. You see we had a call with the ‘Clinicians on FHIR’ team talking about plans for the upcoming event at the [WGM in Spain](http://www.hl7.org/events/working_group_meeting/2017/05/), and during the course of the discussion, Emma asked if clinFHIR could show versions of resources as she wanted to describe medication reconciliation.

At the time I said we could certainly do something like that (though I’ve not yet got that working in the Scenario Builder) but after the call I was thinking about it, and it occurred to me that what we really need is to be able to version scenarios – so we can show how they are built up, and potentially to model a workflow like fulfillment of an order or reconciliation.

So, may I present…

If you load the latest version of the Scenario Builder and load (or create) a scenario, you’ll see a link in the navbar – ‘show version’ (under the ‘Import’ option). Clicking on that will display a version bar across the middle pane (once a scenario has versions the bar is shown automatically).

如果加载最新版本的场景生成器并加载场景，则需要在导航栏中：导航栏 - “显示版本”（“导入”选项）下看到一个链接。单击该选项将在中间窗格中显示一个版本栏（一旦场景有版本，该栏将自动显示）。

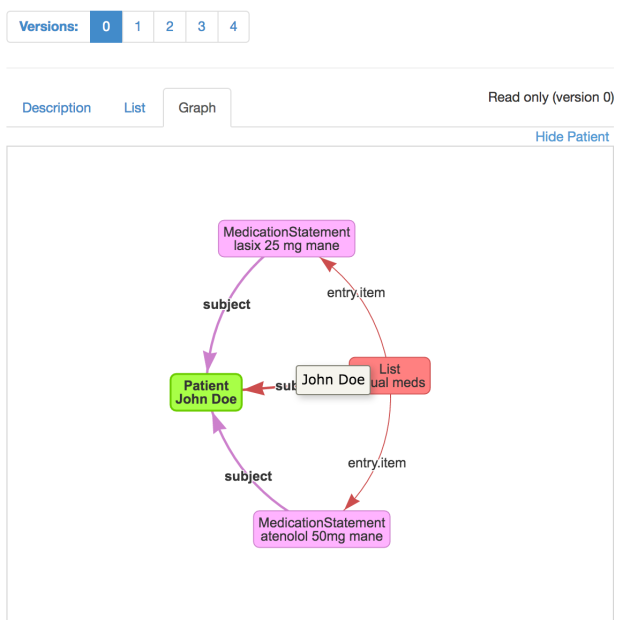
Each version (there will be one at the start) is in a little box, and clicking on the box shows that version (it works best when the graph tab is displayed). If you have the most recent version displayed, a plus (+) sigh is shown to the far right of the version bar and clicking that will ‘freeze’ the current scenario as a version. You can carry on editing the scenario and clicking the + button to create a series of ‘frames’ that represent the story you are trying to tell.

每个版本都在一个box中，点击之后显示当前版本。显示最新版本，则版本栏的最右侧会显示加号（+）叹号，并单击将冻结当前场景作为版本。您可以继续编辑场景，并点击+按钮创建一系列“框架”。

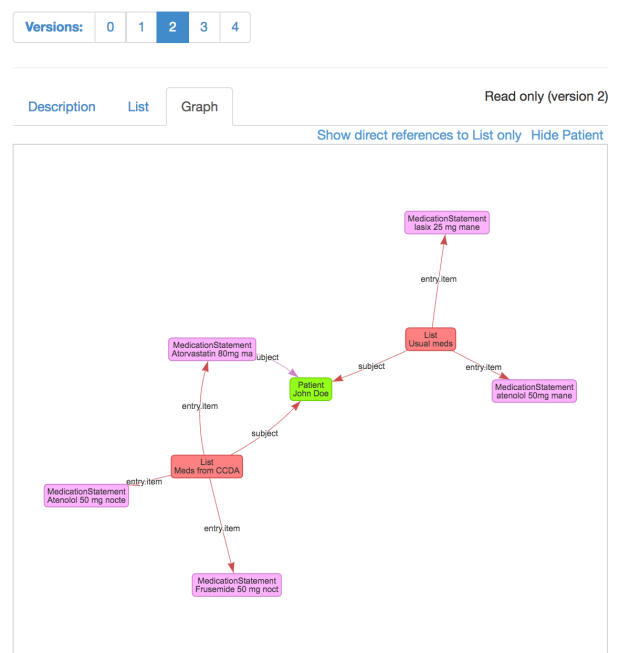
Here’s a simple example.

We start with a Patient, a List and 2 MedicationStatement resources. Note that all resources have a reference to Patient. A simple medication list.

患者、列表、药物清单资源。注意：所有的资源都会引用Patient资源。



Now let’s pretend that we have another list – maybe from a CCDA – that also has a list for the patient, but the list of meds is different – there’s atorvastatin in there as well (How do they make up those names!).



So what we want is to update the patients Usual Medications list with atorvastatin.

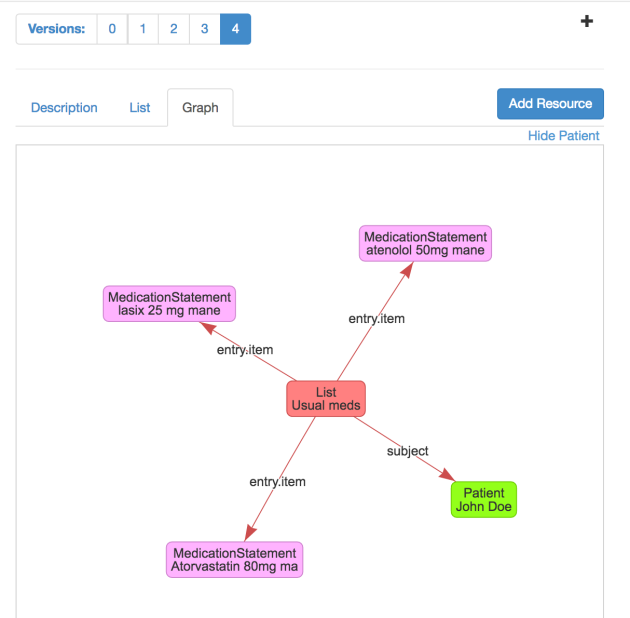
(btw – note that in the interest of clarity I’ve removed the references from most of the MedicationStatement resources to Patient. They will be present in a real scenario of course. To do that, select the MedicationStatement resource instance, and in the tree view of that resource to the right (Current Resource Views > Tree view), select the reference to Patient, and click the delete button that appears. Note that this will remove the whole node – if you try it on List.entry you’ll delete them all)

First, we update the ‘Usual meds’ List by adding a reference to the atorvastatin resource (which is now referenced by both Lists):



And now we can remove the rest of the CCDA set of resources, leaving only the usual list:

我们可以删除CCDA的其余资源集，只剩下通常的列表



Now, I’m not suggesting that this is the best way to do reconciliation, and I’ve ignored the fact that the CCDA version of the meds is taken at night – I just wanted a realistic example. The whole point of this new functionality is to be able to experiment with these things.

A couple of things to note:

The versioning is of the scenario on your local machine (including the resources and references in it) – but they are not saved to the data server, unless you do this manually between versions – ie first update the server and then create a new version. This does need work to tidy up – should be done by the time of the WGM

版本控制是本机的场景，不保存数据服务器，除非在版本之间手动执行此操作。首先更新服务器，然后创建新版本。

I also got confused trying to select the resource I wanted in the graph when there are many of the same type, so I added the first 20 characters of the text to the display. I’ll make that an option shortly, as I suspect it will clutter the larger scenarios…

I also noted that the way in which you create references to multiple resources from a repeating backbone element (like List.entry.item -> MedicationStatement) is – to be kind – rather klunky. What you have to do is:

1. Create the first reference (ie List.entry.item -> MedicationStatement). This updates nicely in the graph display
2. Then click on another resource (any one) and back on the List resource. If you select the List.entry.item element, you’ll see the current contents of that branch just above the datatype, and an ‘Add’ link to the right of it. You have to click on that link to create a new List.entry.item branch, then you can create the new reference – otherwise you just wind up replacing the existing one.

从重复的骨干元素中创建多个资源的引用方式是非常实际的，需要进行：

创建第一个引用（即List.entry.item - > MedicationStatement）

然后单击另一个资源返回List资源。如果选择了List.entry.item元素，那么您将在该数据类型的上方看到该分支的当前内容，并在右侧显示一个“添加”链接。您必须单击该链接才能创建一个新的List.entry.item分支，然后可以创建新的引用 - 否则只需更换现有的引用。

Once you’ve created 2 or more branches, you can create as many more as you like by clicking the ‘Add’ link. You can also select (and edit) the other branches by clicking on the appropriate box in the set of branches.

一旦创建了多个分支之后，可以通过点击“添加”连接创建更多的分支，还可以单击分支集合中相应的框来选择其他分支。

But it’s not very user friendly, so I need to work on that. It’s all a bit complicated under the hood, but I’ll see what I can do.

But nevertheless, you can now a ‘story board’ of ‘frames’ that you can use to illustrate a process, or how to build a complex scenario.

I have to say that I found it a really instructing episode – not just developing the new functionality, but also understanding the usability issues.

这是一个非常有意思的故事 - 不仅仅是开发新的功能，而且还要了解可用性问题

To remind myself:

* Optionally synchronize resource versions in the scenario to the data server
* Allow individual branches (eg List.item) to be deleted
* Make the adding of new branches more user friendly
* Add an option to hide/show the text in the graph.
* A text box to describe each frame would be nice
* The nav bar has become a bit disorganized
* And maybe being able to edit / delete frames…

可选地将场景中的资源版本同步到数据服务器；允许删除单独的分支（例如List.item）；使新分支机构的添加更加用户友好；添加一个选项以隐藏/显示图形中的文本。描述每一帧的文本框将是不错的；导航栏已经变得有点混乱了；也许可以编辑/删除帧...

There’s some work to do! Well, it’s never boring…

## 2.14 Creating an Extension Definition – part 2: The URL.

https://fhirblog.com/2017/04/12/creating-an-extension-definition-part-2-the-url/#more-160105

APRIL 12, 2017 [2 COMMENTS](https://fhirblog.com/2017/04/12/creating-an-extension-definition-part-2-the-url/#comments)

In the [previous post](https://fhirblog.com/2017/04/12/so-you-want-to-create-a-fhir-extension/), we talked about the steps you might follow when creating an Extension Definition as part of profiling a resource. There were 2 steps that we glossed over because of space – the url, and extra work required for coded datatypes and identifiers. Let’s take a look at the url. 对于扩展，着重说一下url元素

The url property (for example [StructureDefinition.url](http://hl7.org/fhir/structuredefinition-definitions.html#StructureDefinition.url) or [ValueSet.url](http://hl7.org/fhir/valueset-definitions.html#ValueSet.url)) are intended to be globally unique identifiers for the resource (‘identifier’ is an overloaded word – we’re not talking about the [Identifier](http://hl7.org/fhir/datatypes.html#identifier) datatype). When another resource wants to refer to it, then it can use the url to do so. (Note that we are not talking about [FHIR references](http://hl7.org/fhir/references.html) between resources – just that the resource wants to identify it)

url属性（例如：StructureDefinition.url或者ValueSet）是资源的全局唯一标识符（identifier是一个重载字段（并不是说identifier的数据类型））。当其他资源引用它的时候，也可以使用url来实现（注意：这不是资源之间的FHIR引用，仅仅是通过资源来识别它）。

When thinking about extensions, there are 2 main circumstances where we use the url:

* Where a resource instance has an extension value, there is a [url property](http://hl7.org/fhir/extensibility.html#extension) on that extension instance that identifies the Extension Definition (StructureDefinition resource).
* Within the Extension Definition, if the datatype is a coded one, then the url is used to refer to the ValueSet containing the possible values for that item. (Actually, it is also possible to have a direct FHIR reference to the ValueSet as well, though that’s less flexible).

考虑扩展，则url 有两个主要的使用环境：

在资源实例中具有扩展值的地方，该扩展实例上有一个url属性，用于标识扩展的定义（StructureDefinition资源）。

在扩展定义中，如果数据类型是code，则url用于引用包含该项可能值的ValueSet。实际上，可以对ValueSet进行直接引用

We’ll consider the first use here – how the instance can identify the extension definition. Using it to refer to the ValueSet (and other resources) will be discussed in the next post (though it works in pretty much the same way)

首先考虑使用：实例如何识别扩展定义。使用url引用ValueSet将在下一文章中讨论。

The first thing to point out is that the URL doesn’t have to be the actual location of the extension definition. In the words of [the spec](http://hl7.org/fhir/extensibility.html#extension):

*Except for child extensions defined within complex extensions, the URL SHALL be an absolute URL. The StructureDefinition that defines the extension SHOULD be available at the identified location so that resource processors can access the definition from the URL*

需要指出的是：url不具有可扩展定义的实际位置。除了在复杂扩展中定义的子扩展外，url应该为绝对URL。定义扩展的StructureDefinition应该在可标识的位置可用，以便资源处理器可用从url访问扩展定义。

So you don’t HAVE to store the extension definition at the url, but it’s best practice to do so.

Lets take a step back and have a think about extensions and interoperability in general.

Extensions were developed to allow the resources to remain small and tightly focussed, yet support the additional elements that most real use cases will require to be exchanged. As the author of a resource, if we are populating the ‘core’ elements in the resource (eg Procedure.code) then we can be quite sure that any FHIR compliant client will know what it means (They might not actually process it – but that’s a different story).

考虑一下互操作性。开发扩展以允许资源保持精简且具有侧重点，但是支持需要交换大多数真实用例的其他元素。作为资源的作者，如果在资源中填充“core”元素（例如：Procedure.code），则可以确定任何符合FHIR客户端都知道此元素的含义（可能不会真正的处理它）

But this doesn’t apply to our extension. To understand it, the client is going to need to retrieve the extension definition, decide if they are interested in it and update the application accordingly. There’s a certain amount of work involved in doing this and so the more widely used an extension is used, the more chance that the client will do that work – in fact the more chance that they already understand it. It’s in our interest to promote re-use.

客户端需要检索扩展定义，来确定是否有兴趣并相应地更新应用程序。

You can think of different ‘levels’ of extensions 可以认为扩展分为不同的级别：

At the top there are those extensions that are defined in the spec. These are elements that are not so commonly used as to justify being in the core spec, but sufficiently common to try to get implementers to use them rather than developing their own – think [religion](http://hl7.org/fhir/extension-patient-religion.html), [time of birth](http://hl7.org/fhir/extension-patient-birthtime.html) & [nationality](http://hl7.org/fhir/extension-patient-nationality.html). Because they are so easy to find – they are listed at the top of each resource in the spec – the likelihood is very high that they will be used.

顶部：在规范中定义的这些扩展。这是不常用的元素，以证明核心规范的正确性，但是一般让实施者直接使用它们。

The next level is National extensions – those appropriate for a country. For example the [New Zealand](http://www.planetware.com/tourist-attractions/new-zealand-nz.htm) Maori tribal affiliations of [Iwi](https://en.wikipedia.org/wiki/Iwi) and [Hapu](https://en.wikipedia.org/wiki/Hap%C5%AB). Of great interest in [New Zealand](http://www.newzealand.com/int/destinations/) – less so in the UK. Generally, these extensions will be managed by the HL7 Affiliate for that country

下一个层级是国家扩展：适合一个国家的扩展。例如：新西兰。

And then there are local extensions – those that are specific to a single vendors implementation for example.

本地扩展：例如，单个供应商实现特定的扩展。

(This is not an ‘official’ list – there may well be others – but hopefully you get the point)

Each of these levels needs a registry – a place to store the extension definitions (and other artefacts), and the url to that registry is a logical ‘root’ for the url. For example:

* <http://hl7.org/fhir> is the root for the ones in the spec (Managed by HL7 International. So we have [http://hl7.org/fhir/StructureDefinition/birthPlace](http://hl7.org/fhir/StructureDefinition/birthPlacehttp:/hl7.org/fhir/extension-birthplace.html) for the patients birthplace
* [http://fhir.hl7.org.nz](http://fhir.hl7.org.nz/) could be the [New Zealand](http://www.hl7.org.nz/) one
* <http://orionhealth.com/fhir> would be the one for [Orion Health](http://orionhealth.com/) ones.

扩展的这几个层级都需要一个注册表（即存储扩展定义的一个地方），该注册表的url是url逻辑根路径。例如：

[http://hl7.org/fhir/StructureDefinition/birthPlace](http://hl7.org/fhir/StructureDefinition/birthPlacehttp:/hl7.org/fhir/extension-birthplace.html) 为患者的出生地。（实施者直接使用的扩展）

[http://fhir.hl7.org.nz](http://fhir.hl7.org.nz/) 新西兰（国家扩展）

<http://orionhealth.com/fhir> （本地扩展）

The point of all this is that the url you use will depend on which level your extension definition is pitched at – the ‘higher’ the better. Right now the governance processes are patchily defined, but if you do think that you have an extension that belongs at the HL7 level, then contact the Work Group that looks after the resources you are interested in (there’s a link at the top of each resource page)

使用的url决定了定义的扩展的层级，层级越高越好。如果认为扩展名术语HL7层级，则联系工作组人员。

If you think you’re extension has national interest, then a good place to start will be to contact the local affiliate for advice. There’s a list of them – along with contact details – on the[HL7 web site](http://www.hl7.org/Special/committees/international/leadership.cfm?ref=nav).

如果认为扩展具有国家利益，则刚开始就可以联系当地的分支机构。

Otherwise, you’ll need to make your own arrangements. One option you could consider is to stand up an instance of the HAPI web server (We talked about that [last year](https://fhirblog.com/2016/10/19/setting-up-your-own-fhir-server-for-profiling/)) as it has all the capabilities needed to store and expose these artifacts. You will need to think about all of the ‘-ities’ – availability, reliability, scalability, security. It is unlikely that you will get a high volume of traffic, but because it’s such an integral part of your resources being useable you do want to make sure that it is available most of the time. A good idea is to lodge a copy of your artefact (extension definition, valueset) in [simplifier](https://fhir.furore.com/simplifier-net/) – it’s just a matter of getting an account there and uploading it.

And remember that regardless of where the Extension Definition is hosted, once it has been used in production it needs to remain there – unchanged – pretty much forever as it is describing clinical data. Which brings us to the topic of versioning – what do you do if you want to change an extension definition?

Well, in short, you can’t.

记住，无论扩展定义在哪里被托管，一旦它被用于生产，它需要保持在那里 - 不变 - 几乎永远是描述临床数据。这使我们了解版本控制的主题 - 如果要更改扩展定义，您该怎么做？ 简而言之，不能这么做。

I guess you could add to descriptive elements – improved description, adding keywords but because someone may have taken and stored one of your resources with an extension in it, you cannot change its meaning.

可以添加描述性元素 - 改进的描述，添加关键字，但是由于有人可能已经将其中一个资源与其中的扩展名一起存储，所以您无法更改其含义。

What you can do is to change the status to ‘retired’ so that a client knows that it is no longer in use, and create a new one. But for the reasons mentioned above, even if you’re not using it any more, it does need to remain.

可以做的就是将status更改为“retired”，以便客户端知道不可用，并创建一个新的。即使不在使用它，也需要进行保留。

Creating extensions is a serious business! 创建扩展是一个严肃的事情

Next up (and last in this little series) will be coded elements.

Other posts in this series:

* Part 1 – [Creating an extension](https://fhirblog.com/2017/04/12/so-you-want-to-create-a-fhir-extension/)
* Part 3 – [Coded items](https://fhirblog.com/2017/04/29/creating-an-extension-definition-part-3-coded-items/)

## 2.15 So you want to create a FHIR extension?

https://fhirblog.com/2017/04/12/so-you-want-to-create-a-fhir-extension/#more-160070

APRIL 12, 2017 [2 COMMENTS](https://fhirblog.com/2017/04/12/so-you-want-to-create-a-fhir-extension/#comments)

Most people familiar with FHIR will be aware of [profiling resources](http://hl7.org/fhir/profiling.html) – and in particular adding new elements to resources for a specific use case – adding an [extension](http://hl7.org/fhir/extensibility.html). It seems straightforward, but there are a number of things to think about when you do this so – or so we’ve found at [Orion Health](https://orionhealth.com/nz/)!

大多数熟悉FHIR的人都会意识到profiling资源：尤其是为特定的用例添加新元素以增加扩展。这似乎很简单，但是当这样做的时候，还需要考虑很多事情。

In this post we’ll talk about some of the factors involved – less about the technology and how they work in FHIR and more about some of the design time issues.

在这一部分将讨论，涉及到的一些因素：关于技术以及在FHIR中的工作原理以及有关设计时间问题的信息。

Before we start, let’s define a few of the terms we’ll be using.

The **Extension Definition** is the resource that will hold the definition of the extension. It’s actually a [StructureDefinition](http://hl7.org/fhir/structuredefinition.html) resource.

The **Profile** is how we ‘attach’ the extension definition to the resource. It’s also a StructureDefinition resource, but we won’t discuss profiling further here.

The **ValueSet** is a resource that describes a set of concepts that are the possible values for a coded element. The [ValueSet](http://hl7.org/fhir/valueset.html) actually refers to one or more CodeSystems that describe the actual concepts. A ValueSet is **bound** to the element, which determines whether values must be in the ValueSet

The **CodeSystem** contains the list of actual concepts – each of which has a code. The [CodeSystem](http://hl7.org/fhir/codesystem.html) might simply be a description of an external terminology (like SNOMED or LOINC), or it might actually contain the definition of those codes.

A **system** is a ‘child element’ found in Identifiers and coded elements that links the element to the set of permissible values (and the **code** value will be unique within that system). It’s like a namespace – in the case of an Identifier it is defined in a NamingSystem resource, for a coded element it’s the Url of the ValueSet. Technically, it’s a URI.

A **NamingSystem** resource is used to [describe a code system](http://hl7.org/fhir/namingsystem.html) or identifier system. It’s not the actual code/identifier system itself, but rather “*Defines a specific code system or identifier system, so that it can be noted in a registry for other systems to find and understand an identifier*”.

在开始之前，需要定义一些将要使用的术语：

ExtensionDefinition是一个含有扩展定义的资源，实际上是StructureDefinition资源；Profile是将扩展定义附加到资源中。常常是一个StructureDefinition资源；

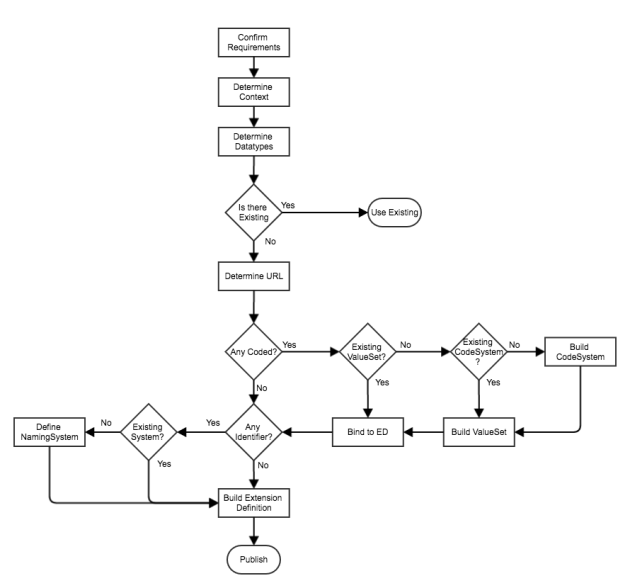
ValueSet是描述了编码元素可能取值的一组概念。ValueSet实际上值的是描述实际概念的多个codesystem。值集绑定到元素中，以确定该值是否存在与ValueSet中。

codeSystem包含实际概念的列表，每个codesystem都有一个code。Codesystem可能仅仅是一个外部术语的描述（例如：SNOMED或者LOINC）或者实际包含这些code的定义。

system是在标识符Identifier和将元素链接到一组允许值的编码元素中的子元素（code值在system中是唯一的）。就像一个命名空间——即对于Identifier场景，指的是在NamingSystem资源中的定义，以及在编码元素场景中，指的是ValueSet中的Url。

Here’s a flow chart that gives an overview of the process of creating an Extension Definition. And a reminder that this is not an ‘official’ FHIR recommendation – I take full responsibility for errors or omissions!

流程图展示了创建扩展定义的过程。单这不是一个官方的FHIR建议



So we start by **Confirming Requirements**– what is it that we are wanting to represent? It pays to spend a bit of time to get this right – and to consult as widely as possible. At Orion, we use an internal wiki to describe what we want it to do and to record comments and feedback, but we’ll also go out to the community (via the [FHIR chat](https://chat.fhir.org/)) to get other input. It often takes several weeks (or even longer) to get this right.

首先“确认请求”——我们想要代表什么？这将有助于确定：需要的扩展是否具有单个值或者多个值；如果接收人懂得这个扩展，即改变资源的含义，如不了解使其临床不安全，则是一个修饰符扩展；如果扩展名有一组特定的值。

This will help to determine:

* whether the extension we need will have a single value (simple extension) or more than one value (complex extension).
* if this extension MUST be understood by the recipient – i.e. it changes the meaning of the resource making it clinically unsafe not to understand it, in which case it is a [modifierExtension](http://hl7.org/fhir/extensibility.html#modifierExtension). eg the patient does NOT have this condition.
* If there is a specific set of values for the extension (more on that later)

And as part of the determining these requirements it is helpful to collect documentation for things like title, description, purpose, author, key words – things that we can record in the main part of the StructureDefinition that describe what it is intending to represent to a recipient, and to aid with discovery.

作为这些要求的一部分，有助于收集文件，例如：title、purpose、author、key word。可以在StructureDefinition中主要部分中记录这些内容，用来描述接收者的目的。

For the sake of this discussion, let’s assume that we want to be able to represent the age of the patient when some event occurred – say the age of the patient when a procedure was performed (“I had an appendicectomy as a child – don’t remember exactly when”). It is a simple extension (it will have only a single value), and can be ignored if the user doesn’t understand it.

可以假设当发生某件事情的时候，表示患者的年龄——例如，手术时候，患者的年龄。这是一个简单的扩展（仅有一个值），用户如果不了解他，可以进行忽略。

Next, what is the **context** of our extension – which resources/s will it apply to, and whereabouts in those resources will it apply. Pretty much any part of a resource can be extended:

* We can add it to the resource root as a new element
* It can be added to a ‘backbone’ element (one that has child elements like Condition.stage)
* It can be added as an ‘extra’ to an existing element

接下来，扩展的context：即将应用到哪些资源中，以及应用到资源的哪些场景中。可以扩展资源的任何部分：

可以将其作为新元素添加到资源根目录中

可以添加“backbone”元素中（具有子元素，例如Condition.stage）

可以作为“extra”添加到现有元素中

In our example, we are wanting to represent the patients age when a procedure was performed.  There is already a *performed* element, but it only has date and period. So we’ll add our extension to that element, with the context therefore being Procedure.performed Remember that multiple resources can use the same extension.

在上述例子中，想要表示手术进行时的患者的年龄。其中Procedure具有Procedure.performed[]元素，但是仅仅含有performeddate和performedperiod。故我们可以添加扩展到performed元素中，则该扩展的上下文就是Procedure.performed，多个资源可以使用相同的扩展名。

Next, we need to decide **what datatype (or datatypes)** are needed. From our requirements, we know that this information can come in a variety of different forms. It might be ‘aged 5’ or it might be ‘as a child’ so it sounds like both ‘string’ and ‘age’ are going to be needed.

接下来决定需要的数据类型。根据要求，我们知道这些信息可以由多种不同的形式。可能是“5岁”或者“可能是个孩子”，故需要“string”或“age”数据类型。

(You could argue that if we have the age then we could calculate the date, but we might not want to do that as it would sound like we actually knew what the date was rather than guessing based on the age.)

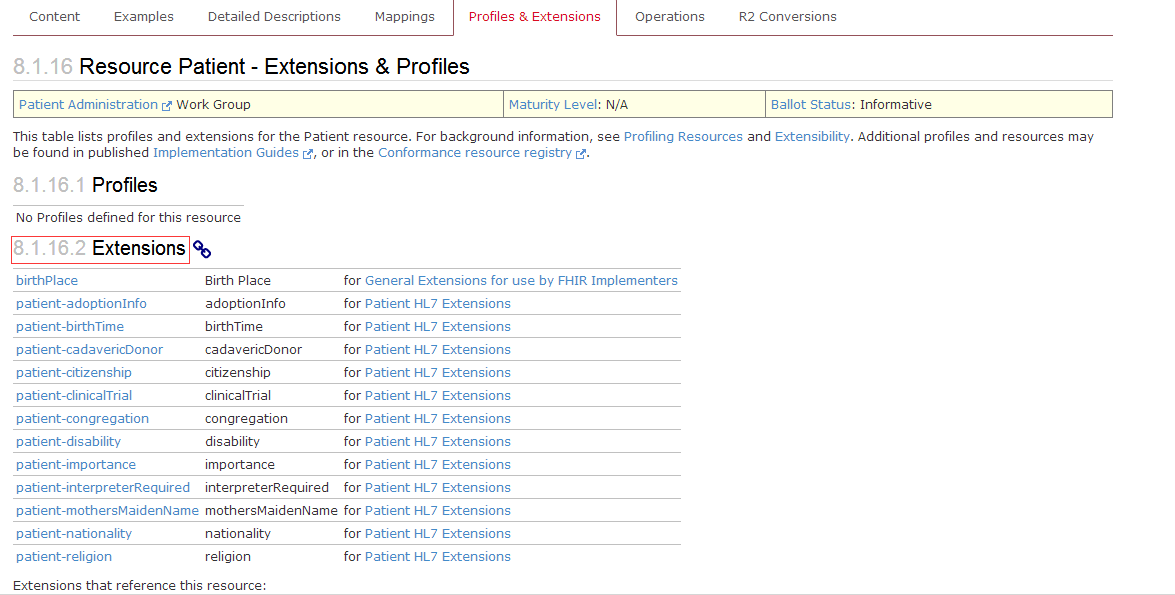
We’ll also let the committee responsible for Procedure know that we’ve needed to do this with a suggestion that ‘Age’ be added as an option for performed – if they agree, then our extension won’t be needed in future releases.

可以将“age”作为performed的数据类型

Note that our extension can have any of the FHIR [datatypes](http://hl7.org/fhir/datatypes.html) – including a reference to another resource. 注意：扩展可以使用具有任何的FHIR数据类型，包括对另一个资源的引用。

Once we know what we want to represent, the next thing is to see if there is **already an extension definition** that does what we want so we can re-use it. Good places to look are in the spec itself (each resource has a list of the official extensions – see the ones for [patient](http://hl7.org/fhir/patient-profiles.html)), in the [Implementation Guides](http://www.fhir.org/guides/registry) that have been published (e.g. [US Core](http://hl7.org/fhir/us/core/), previously known as DAF) and in [Simplifier](https://simplifier.net/) – a registry built by the good folk at [furore](https://fhir.furore.com/).

一旦知道需要的内容，接下来需要看看，我们需要的内容是否是已经存在的扩展定义。在规范中，每个资源都有官方的扩展，例如Patient



In the longer term the simplifier registry will hopefully have a copy of most of the extension definitions that people have built so we’ll be able to just go straight there. There are a few enhancements (mostly around searching capabilities through the API) that need to be completed, but once that’s done then re-use should be simpler…

长远看，simplifier registry将希望具有人们建立的大部分的扩展定义的副本。

Assuming that we can’t find one to reuse, then we’re going to need to build our own so we’re going to need a **unique URL** for it. This is the url that a resource instance that has this extension will use to point to the definition as required by the [specification](http://hl7.org/fhir/extensibility.html).

假设找不着可以重用的扩展，则需要自己构建，则就需要一个唯一的URL。这是具有此扩展名的资源实例，将用于指定规范要求的定义。

The value of the Url will depend on a number of factors, so we will come back to that in a subsequent post. For now, assume we have an appropriate URL in a domain we control, or the domain owner has given one to us.

url的值取决于一些因素，所以随后文章中说明。现在，假设控制区域中有一个合适的URL。

So now we know that we have to create an extension definition, where the extension will go in which resource/s, what datatype/s it can take and what the Url is. We’re almost there!

Except for a couple of things…

* If any of the data types are coded ([code](http://hl7.org/fhir/datatypes.html#code), [Coding](http://hl7.org/fhir/datatypes.html#coding), [CodeableConcept](http://hl7.org/fhir/datatypes.html#codeableconcept) or [Quantity](http://hl7.org/fhir/datatypes.html#quantity)) then we’ll need to create a ValueSet to hold the list of possible values, and figure out the [binding strength](http://hl7.org/fhir/terminologies.html#strength) (if a specific value MUST or SHOULD be in the ValueSet)
* If any of the data types are an [Identifier](http://hl7.org/fhir/datatypes.html#identifier), we’ll need to figure out the system from which the identifier should be drawn.

如果编码类的数据类型（code、coding、CodeableConcept、quantity），则需要创建ValueSet来保存可能的值，并找出绑定强度。

如果是Identifier数据类型，需要指出来自哪个system的identifier。

Both of these are a bit complicated, so for now we’ll just note that we have more work to do for these dataTypes. The next post will explore these in more detail.

With that proviso, we’re good to go. We have the information needed to build our extension definition and we know where to publish it to – the Url that we determined earlier.

(Actually, we don’t have to publish it there, it’s perfectly legitimate to just store it in a registry like simplifier, but it’s good practice to do so – then it is ‘resolvable’ – entering it into a browser will directly retrieve it. Actually, we should do both.)

The last thing we’ll think about in this post is how to actually create the StructureDefinition resource that is the extension definition. There are (at least) 3 options.

* We can create it manually in an XML or Json editor. Perfectly feasible, but hard to get right – under the hood it’s kinda complicated. (If you do go down this route, use Grahame’s server to help validate the resource. It’s not 100% perfect, but catches most of the syntactic errors. And have a look at some of the existing definitions).
* We can use the [Forge](https://fhir.furore.com/forge/) tool from furore. This is the ‘official’ profiling tool for FHIR and by far the most comprehensive.
* [clinFHIR](http://clinfhir.com/) can also create simple extension definitions. You have to create them directly against a FHIR server, and the functionality is not as comprehensive as Forge, but it is simpler to use (though, as the author, I am biased!)

 需要最后考虑的问题是如何实际创建StructureDefinition资源，至少有3个选项：

可以在XML或JSON编辑器中手动创建。如果这么做，使用Grahame的服务器来帮助验证资源，这不是100％完美的，而是捕捉到大部分的语法错误，并且看一些现有的定义

可以使用Forge工具。这是FHIR的官方Profiling工具，且迄今为止是最全面的。

clinFHIR还可以创建一个简单的扩展定义。必须直接针对FHIR服务器创建他们，且功能不如Forge那么全面，但是使用起来更加简单。

So that covers the overall flow of creating an Extension Definition (at least one way of doing it). There were 2 topics we put aside:

* Determining the Url of the Extension Definition
* Extra work needed for coded types and identifiers.

因为涵盖了

We’ll come back to those in the next post.

Other posts in this series:

* Part 2 – [The Url](https://fhirblog.com/2017/04/12/creating-an-extension-definition-part-2-the-url/)
* Part 3 – [Coded items](https://fhirblog.com/2017/04/29/creating-an-extension-definition-part-3-coded-items/)

从此

（2）URL

Creating an Extension Definition – part 2: The URL.

APRIL 12, 2017 [2 COMMENTS](https://fhirblog.com/2017/04/12/creating-an-extension-definition-part-2-the-url/#comments)

In the [previous post](https://fhirblog.com/2017/04/12/so-you-want-to-create-a-fhir-extension/), we talked about the steps you might follow when creating an Extension Definition as part of profiling a resource. There were 2 steps that we glossed over because of space – the url, and extra work required for coded datatypes and identifiers. Let’s take a look at the url.

前面介绍了创建扩展定义作为资源分析的一部分，其中两个步骤没有详细介绍：URL和编码元素的外部工作请求。首先URL

The url property (for example [StructureDefinition.url](http://hl7.org/fhir/structuredefinition-definitions.html#StructureDefinition.url) or [ValueSet.url](http://hl7.org/fhir/valueset-definitions.html#ValueSet.url)) are intended to be globally unique identifiers for the resource (‘identifier’ is an overloaded word – we’re not talking about the [Identifier](http://hl7.org/fhir/datatypes.html#identifier) datatype). When another resource wants to refer to it, then it can use the url to do so. (Note that we are not talking about [FHIR references](http://hl7.org/fhir/references.html) between resources – just that the resource wants to identify it)

url旨在资源的一个全局唯一标识符（例如：StructureDefinition.url或ValueSet.url），其中Identifier是一个重载的字段，此时不是指的这个字段。当另一个资源引用它的时候，使用url来实现（不是资源之间的引用，是资源识别它）

When thinking about extensions, there are 2 main circumstances where we use the url:

* Where a resource instance has an extension value, there is a [url property](http://hl7.org/fhir/extensibility.html#extension) on that extension instance that identifies the Extension Definition (StructureDefinition resource).
* Within the Extension Definition, if the datatype is a coded one, then the url is used to refer to the ValueSet containing the possible values for that item. (Actually, it is also possible to have a direct FHIR reference to the ValueSet as well, though that’s less flexible).

考虑扩展的时候，使用url有两种主要的情况：

在资源实例具有扩展值的地方，该扩展实例上有一个URL属性，用于标识扩展定义（StructureDefinition资源）。

在扩展定义中，如果数据类型是编码类型，则url用于引用包含该项目的可能值的valueSet。（实际上，可以对ValueSet进行直接的FHIR引用）

We’ll consider the first use here – how the instance can identify the extension definition. Using it to refer to the ValueSet (and other resources) will be discussed in the next post (though it works in pretty much the same way)

The first thing to point out is that the URL doesn’t have to be the actual location of the extension definition. In the words of [the spec](http://hl7.org/fhir/extensibility.html#extension):

*Except for child extensions defined within complex extensions, the URL SHALL be an absolute URL. The StructureDefinition that defines the extension SHOULD be available at the identified location so that resource processors can access the definition from the URL*

需要指出的是url不具有可扩展定义的实际位置，用规范的话来说就是：除了在复杂扩展名中定义的子扩展名外，URL应为绝对的url，定义扩展的StructureDefinition应该在标识的位置中可用，以便资源处理器可以从url中进行访问扩展定义。

So you don’t HAVE to store the extension definition at the url, but it’s best practice to do so.

所以并没有将扩展定义存储在URL中。

Lets take a step back and have a think about extensions and interoperability in general.

Extensions were developed to allow the resources to remain small and tightly focussed, yet support the additional elements that most real use cases will require to be exchanged. As the author of a resource, if we are populating the ‘core’ elements in the resource (eg Procedure.code) then we can be quite sure that any FHIR compliant client will know what it means (They might not actually process it – but that’s a different story).

扩展是为了使资源保持小而紧密的关系，但是支持需要交换大多数真实用例的附加元素。作为资源的作者，在资源中填充“核心”元素（例如：Procedure.code）则可以确定任何符合FHIR客户端都知道其含义（可能不会真正的处理它）

But this doesn’t apply to our extension. To understand it, the client is going to need to retrieve the extension definition, decide if they are interested in it and update the application accordingly. There’s a certain amount of work involved in doing this and so the more widely used an extension is used, the more chance that the client will do that work – in fact the more chance that they already understand it. It’s in our interest to promote re-use.

但是，这个核心元素的填充不适用于扩展。客户端将检索扩展定义，决定是否对其有兴趣并相应地更新应用程序。这样做有一定的工作，所以使用更广泛的扩展使用，客户端可以由更多的机会。事实上，了解的机会越多，会更加促进符合我们的利益。

You can think of different ‘levels’ of extensions

At the top there are those extensions that are defined in the spec. These are elements that are not so commonly used as to justify being in the core spec, but sufficiently common to try to get implementers to use them rather than developing their own – think [religion](http://hl7.org/fhir/extension-patient-religion.html), [time of birth](http://hl7.org/fhir/extension-patient-birthtime.html) & [nationality](http://hl7.org/fhir/extension-patient-nationality.html). Because they are so easy to find – they are listed at the top of each resource in the spec – the likelihood is very high that they will be used.

The next level is National extensions – those appropriate for a country. For example the [New Zealand](http://www.planetware.com/tourist-attractions/new-zealand-nz.htm) Maori tribal affiliations of [Iwi](https://en.wikipedia.org/wiki/Iwi) and [Hapu](https://en.wikipedia.org/wiki/Hap%C5%AB). Of great interest in [New Zealand](http://www.newzealand.com/int/destinations/) – less so in the UK. Generally, these extensions will be managed by the HL7 Affiliate for that country

And then there are local extensions – those that are specific to a single vendors implementation for example.

可以想到扩展的不同级别:

顶层，规范中定义的扩展。这些是不常用的元素，以证明核心规范的正当性。试图让实施者使用它们而不是发展自己的扩展。例如：宗教、出生时间、国际。因为列在规范中每个资源的顶部，所以很容易找到，且使用的可能性更高。

下一层是国家性的扩展。适用于一个国家的扩展

本地扩展。例如：单个供应商实现特定的扩展。

(This is not an ‘official’ list – there may well be others – but hopefully you get the point)

Each of these levels needs a registry – a place to store the extension definitions (and other artefacts), and the url to that registry is a logical ‘root’ for the url. For example:

* <http://hl7.org/fhir> is the root for the ones in the spec (Managed by HL7 International. So we have [http://hl7.org/fhir/StructureDefinition/birthPlace](http://hl7.org/fhir/StructureDefinition/birthPlacehttp:/hl7.org/fhir/extension-birthplace.html) for the patients birthplace
* [http://fhir.hl7.org.nz](http://fhir.hl7.org.nz/) could be the [New Zealand](http://www.hl7.org.nz/) one
* <http://orionhealth.com/fhir> would be the one for [Orion Health](http://orionhealth.com/) ones.

这些级别中每个都需要一个注册表，一个存储扩展名的地方，并且注册表的url是URL的逻辑根目录。

The point of all this is that the url you use will depend on which level your extension definition is pitched at – the ‘higher’ the better. Right now the governance processes are patchily defined, but if you do think that you have an extension that belongs at the HL7 level, then contact the Work Group that looks after the resources you are interested in (there’s a link at the top of each resource page)

使用的网址取决于扩展名定义所在的级别。

If you think you’re extension has national interest, then a good place to start will be to contact the local affiliate for advice. There’s a list of them – along with contact details – on the[HL7 web site](http://www.hl7.org/Special/committees/international/leadership.cfm?ref=nav).

Otherwise, you’ll need to make your own arrangements. One option you could consider is to stand up an instance of the HAPI web server (We talked about that [last year](https://fhirblog.com/2016/10/19/setting-up-your-own-fhir-server-for-profiling/)) as it has all the capabilities needed to store and expose these artifacts. You will need to think about all of the ‘-ities’ – availability, reliability, scalability, security. It is unlikely that you will get a high volume of traffic, but because it’s such an integral part of your resources being useable you do want to make sure that it is available most of the time. A good idea is to lodge a copy of your artefact (extension definition, valueset) in [simplifier](https://fhir.furore.com/simplifier-net/) – it’s just a matter of getting an account there and uploading it.

And remember that regardless of where the Extension Definition is hosted, once it has been used in production it needs to remain there – unchanged – pretty much forever as it is describing clinical data. Which brings us to the topic of versioning – what do you do if you want to change an extension definition?

无论扩展定义何处被托管，一旦它被用于生产，它需要保持在那里 - 不变 - 几乎永远是描述临床数据。不能更改扩展定义。

Well, in short, you can’t.

I guess you could add to descriptive elements – improved description, adding keywords but because someone may have taken and stored one of your resources with an extension in it, you cannot change its meaning.

可以添加描述性元素

What you can do is to change the status to ‘retired’ so that a client knows that it is no longer in use, and create a new one. But for the reasons mentioned above, even if you’re not using it any more, it does need to remain.

可以更改其状态取值

Creating extensions is a serious business!

Next up (and last in this little series) will be coded elements.

Other posts in this series:

* Part 1 – [Creating an extension](https://fhirblog.com/2017/04/12/so-you-want-to-create-a-fhir-extension/)
* Part 3 – [Coded items](https://fhirblog.com/2017/04/29/creating-an-extension-definition-part-3-coded-items/)

（3）coded items

<https://fhirblog.com/2017/04/29/creating-an-extension-definition-part-3-coded-items/>

Creating an Extension Definition – part 3: Coded items

APRIL 29, 2017 [4 COMMENTS](https://fhirblog.com/2017/04/29/creating-an-extension-definition-part-3-coded-items/#comments)

A common data type that is likely to be added in an extension are coded types – those where the value in an instance comes from a pre-defined set of possible values. There are a couple of extra things that you need to do for these.

在扩展中添加常见数据类型是编码类型。实例中的值来自预定义的可能值的类型。

First, a quick review of the basics.

All coded elements in FHIR (whether in the core spec or a profile) are [bound](http://hl7.org/fhir/terminologies.html#strength) to a ValueSet that describes the set of possible values for the element. The ‘strength’ of the binding determines whether an element in an instance of the resource can have a value that is not in the set. This can vary from a ‘*required*’ strength where the value MUST be in the ValueSet, to ‘*example*’ where the ValueSet is, well, just an example. In general terms, you use the ‘required’ strength sparingly, and only where it is really important to do so (‘status’ elements are the most common examples in the spec).

FHIR中所有编码元素（无论是核心规范还是profile）都绑定到一个ValueSet中，该ValueSet描述了元素的可能值，绑定的强度确定资源实例中是否可以具有不在集合中的值。可以将绑定强度由“Required”转化为“example”，这仅仅是一个例子。一般来说，谨慎使用“required”。

The [ValueSet](http://hl7.org/fhir/valueset.html) gets its contents from one or more [CodeSystems](http://hl7.org/fhir/codesystem.html), where the CodeSystem contains the actual definition of each concept, and the ValueSet is just picking the ones you want for your Use Case. For example you might want to create a ValueSet that has the most common SNOMED diagnoses made in the Emergency Department to use as a simplified ‘pick list’ in an application. The ValueSet would list the SNOMED terms to be included, and you’d likely bind it to the extension with a ‘[extensible’](http://hl7.org/fhir/terminologies.html#extensible) strength to get consistency of use, but allow uncommon diagnoses to be selected.

ValueSet可以从多个codeSystem中获取内容，其中CodeSystem包含每个概念的实际定义和用例中的valueSet。例如：可能需要创建一个valueSet，并进行常见的SNOMED诊断，以在应用程序中用作简化的“选择列表”，ValueSet将列出包含的SNOMED术语，可能会将其绑定到具有“可扩展”强度的扩展，以获得一致性，但是允许选择不常见的简化

‘[Terminologies](http://hl7.org/fhir/terminology-module.html)’ is another word used for CodeSystems (though is often used in a broader context, and can get \*really\* complicated). And to any terminology experts reading this, I know that this description is a simplification – it is a complex domain!

术语是codesystem的另一个单词。

It is important to note that in STU2, the ValueSet was able to also define concepts – but in STU3 (Release 3) this is now done by CodeSystem resource

Next, a quick review of coded datatypes in FHIR – there are 3 ½ of them.

The most common type that you’ll use is the [CodeableConcept](http://hl7.org/fhir/datatypes.html#codeableconcept). This is far and away the most flexible one, as it allows you to use codes from different terminologies to represent the same concept by using multiple copies of the Coding datatype (useful when you need to represent the concept in different code systems like a ‘custom’ code and it’s representation in, say, SNOMED) and also has a ‘text’ element that can be used to represent the description of the concept – or when the coded form is not available (distressingly common in the real world!)

最常用的类型是CodeableConcept。这是最灵活的一个，因为允许使用不同术语的代码通过使用coding数据类型的多个副本来表示相同的概念。还有text元素表示概念的描述。

[Coding](http://hl7.org/fhir/datatypes.html#Coding) represent a single concept in a single CodeSystem. It has 3 main properties:

* *System* is the uri the identifies the CodeSystem
* *Code* identifies the concept in the code system
* *Display* is the description of the concept in the CodeSystem (This is NOT the same as CodeableConcept.text)

Generally you don’t use Coding, as the CodeableConcept is significantly more flexible – unless you are sure that you’ll never need text, or need to represent the concept in multiple CodeSystems.

一般来说，不使用Coding，因为CodeableConcept显着更灵活 。

Next up is the [code](http://hl7.org/fhir/datatypes.html#code) datatype – this is mostly used when there’s a small set of codes and the code is descriptive (there’s no display element). Often used for ‘structural’ elements like Observation.status or where there’s a fixed set of codes that cannot change like AllergyIntolerance.type. The binding strength of a code should always be ‘required’. You can think of code as an [enumeration](https://en.wikipedia.org/wiki/Enumeration) in computer terms – and the libraries generally treat them in that way.

And the ½ coded datatype is [Quantity](http://hl7.org/fhir/datatypes.html#quantity), as it has a coded *Quantity.unit* / *Quantity.system* (generally [UCUM](http://unitsofmeasure.org/trac))

So when creating an extension that has a coded element, you will need to follow these steps.

1. Decide on the actual datatype you need. A safe option will be CodeableConcept.  
2. Determine the ValueSet to use. If you’re lucky there will be one available for you, but if not:

a) Find the CodeSystem/s that have the concepts you need. You may need to build one if they are custom codes, though using a standard is always better. A good starting point for the search is the [terminology module](http://hl7.org/fhir/terminology-module.html) in the spec as there are a number of lists there (including ValueSets).

If you do have to build a CodeSystem, then remember that in STU2 it is part of the ValueSet resource.

b) Build the ValueSet – referencing the CodeSystem/s  
For both CodeSystem and ValueSet you’ll need a url – the [previous post on url’](https://fhirblog.com/2017/04/12/creating-an-extension-definition-part-2-the-url/)s, though focussed on StructureDefinition is still applicable.

3. Determine the binding strength

4. Build the Extension Definition, selecting the coded datatype and referencing the ValueSet with the appropriate strength.

因此，当创建一个具有编码元素的扩展名时，您需要执行以下步骤。

1.确定您需要的实际数据类型。一个安全的选项将是CodeableConcept。2.确定要使用的ValueSet。如果你幸运的话会有一个可供你使用，但如果没有，

a）找到具有所需概念的CodeSystem / s。您可能需要构建一个，如果他们是自定义代码，虽然使用标准总是更好。搜索的一个好的起点是规范中的[术语模块](http://hl7.org/fhir/terminology-module.html)，因为有许多列表（包括ValueSets）。

如果您必须构建一个CodeSystem，那么请记住在STU2中它是ValueSet资源的一部分。

b）构建ValueSet - 引用CodeSystem / s   
对于CodeSystem和ValueSet，您需要一个url - url的[上一个帖子](https://fhirblog.com/2017/04/12/creating-an-extension-definition-part-2-the-url/)，尽管集中于StructureDefinition仍然适用。

3.确定结合强度

4.构建扩展名定义，选择编码数据类型并以适当的强度引用ValueSet。

[ClinFHIR](http://clinfhir.com/) does offer some tooling to assist with this – though the functionality is quite basic at the moment (and does need to be exercised to find the bugs!)

There are other terminology tools being developed (like [Ontoserver](http://ontoserver.csiro.au/snapper2-dev/index.html#/) in Australia, and [Apelon](http://www.apelon.com/solutions/terminology-tooling/dts) in the US) that are far more sophisticated, but I’m not sure if they are openly available. [Forge](https://fhir.furore.com/forge/), is the official tool for creating profiles.

Enjoy your extending!

Other posts in this series:

* Part 1 – [Creating an extension](https://fhirblog.com/2017/04/12/so-you-want-to-create-a-fhir-extension/)
* Part 2 – [Choosing the URL](https://fhirblog.com/2017/04/12/creating-an-extension-definition-part-2-the-url/)

## 2.16 FHIR Connectathon, May 2017, Madrid Spain

https://fhirblog.com/2017/03/30/fhir-connectathon-may-2017-madrid-spain/#more-160067

MARCH 30, 2017 [LEAVE A COMMENT](https://fhirblog.com/2017/03/30/fhir-connectathon-may-2017-madrid-spain/#respond)

Notice about the next connectathon:

It’s again my pleasure to remind you of the 15th FHIR Connectathon that is coming up on the Saturday/Sunday 6/7 May just before the next HL7 Working Group Meeting in Madrid, Spain. Here’s the link to the WGM: <http://www.hl7.org/events/working_group_meeting/2017/05/>

Here’s the wiki page that describes the connectathon in more detail: <http://wiki.hl7.org/index.php?title=FHIR_Connectathon_15>

We have 9 tracks this time around, and the event is based on [FHIR release 3](http://hl7.org/fhir/index.html):

基于fhir V3的几个测试案例

* Patient Track (particularly suitable for FHIR newcomers) <http://wiki.hl7.org/index.php?title=201705_Patient_Track>
* Workflow Track
* CDS Hooks
* FHIR Genomics
* Devices on FHIR
* Clinical Reasoning
* Terminology
* Financial Track
* Structured Data Capture
* Clinical Research
* Data Analytics

so there’s plenty to choose from.

Each of the tracks has a detailed description of what it is trying to achieve – the roles and the scenarios being tested. In addition, there is a track lead that you can talk to for more information about the track- generally there a place for people new to the track, and those who have done one before. The connectathon wiki page above has links to these descriptions – definitely worth a look. The last item – Data Analytics is a bit different, less focussed on developers and more in informatics.

For those who haven’t been to a connectathon before, they serve a number of purposes. First and foremost they are way to ensure that FHIR meets the needs of implementers by testing it using real-world scenarios. But they are a great way of meeting the folk most active in the community and having the opportunity to talk with them in person. The Patient track in particular is a good choice for someone new to the event as it is quite straight forward, but also good for people interested in testing, as we’re using that that track to exercise the automated testing.

Feel free to email me if you have any questions, otherwise I look forward to meeting you in Madrid!

cheers…

## 2.17 Adding extensions to a resource

https://fhirblog.com/2017/03/29/adding-extensions-to-a-resource/#more-160036

MARCH 29, 2017 [LEAVE A COMMENT](https://fhirblog.com/2017/03/29/adding-extensions-to-a-resource/#respond)

So a colleague of mine asked me this morning if the [clinFHIR](http://clinfhir.com/) scenario builder supported modifier extensions.

If you’re not familiar with these, they are extensions that actually change the meaning of the resource to which they are attached. For example, if you wanted to indicate that a patient did not have a particular condition, then you could attach a ‘negation’ extension to the Condition to indicate that you looked for – but did not find – that particular condition. Or, a statement that indicates that a patient is not taking a particular medication.

Now, it’s obviously really important that a recipient of that resource understands what the extension means – it would be clinically unsafe to ignore it – and so it is represented in FHIR as a ‘modifierExtension’. You can read more about them in the [spec](http://hl7.org/fhir/extensibility.html#isModifier).

显然，资源的接收者了解扩展的意义非常重要。忽略临床的不安全性，在FHIR中表示为modifierExtension。可以在规范中阅读更多的内容。

(As an aside, it should be noted that the ‘proper’ way to deal with negation in general is one that is not really well established yet, and there are a number of different ways that this can be done. I don’t want to go there in this post!)

The reason my colleague wanted to know about this, is that they wanted to be able to represent “No breathing difficulties” – a negated Observation – in a training course they are giving tomorrow.

My initial response was that at the moment clinFHIR doesn’t support modifierExtensions – the approach I have been taking is that you first create a profile which contains the possible extensions for a conformant resource, and clinFHIR uses that to generate the User Interface that allows the user to create the resource instance. No real reason that these couldn’t be modifierExtensions – I just hadn’t got around to it yet.

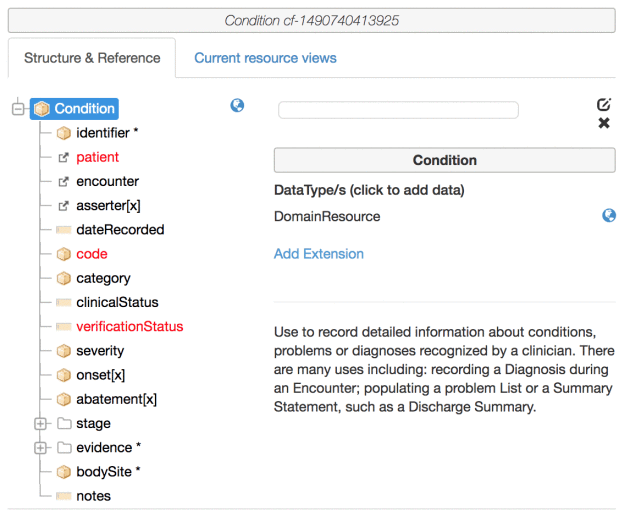
目前，不支持modifierExtension的时候，一直采取的方法是：首先创建一个profile，该profile包含一个符合资源的可能扩展，并且clinFHIR使用profile来生成用户界面，来允许用户创建资源实例。这些不能修改扩展。

But the question got me thinking. Why do you have to create a profile first? Why not simply allow the user to add any extension – normal or modifier – to a resource they are building?

So I added the functionality – or at least a step towards it.

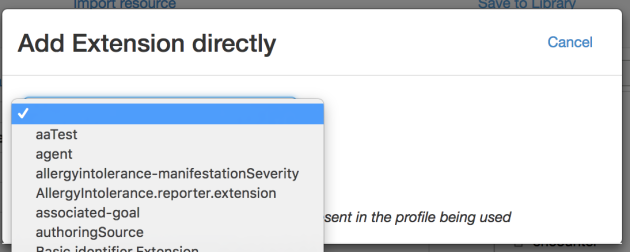
If you load the Scenario builder and select a resource type, you will see a tree of the elements within the type to the right hand side. Selecting an element in the tree displays its details and allows you to enter a value for it. If you click on the ‘root’ element (‘Condition’ in the example below) you will now see a link labelled ‘AddExtension’). Oh, and you do have to actually click on the node – it doesn’t appear automatically.

如果加载场景构建器并选择一种资源类型，则将在右侧显示类型中的元素树。在树中选择一个元素后则显示细节，并允许输入值。点击root元素，可以看到标记为AddExtension的链接。



If you click on that element, you’ll get a dialog box that allows you to add an ‘ad hoc’ extension to the resource you are currently building. There’s a dropdown control with a list of all applicable extension definitions for that element on the current conformance server (they are ones where the ‘[context’](http://hl7.org/fhir/structuredefinition-definitions.html#StructureDefinition.context) of use includes the resource type you are editing). Here’s a screen shot:

单击该元素，将得到一个对话框，允许为正在构建的资源添加“ad hoc”扩展，有一个下拉菜单没包含当前一致性服务器中该元素的所有适用扩展定义的列表。

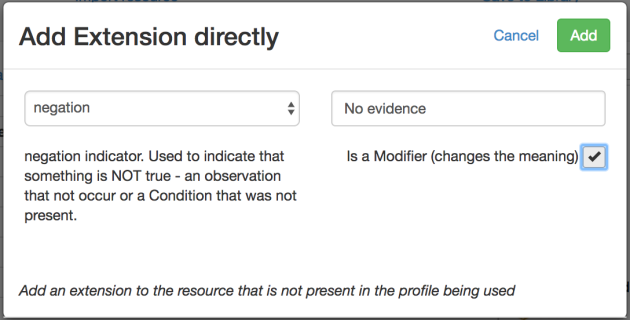


Select the one you want, and an input form appears to the right of the dropdown. This allows you to:

* Enter a value for the extension. Right now, this is only a string value but eventually it will display a datatype appropriate to the extension selected.
* Indicate if the extension is a modifier extension

选择所需的，下拉菜单右侧显示一个输入表单。这允许您：

* 输入扩展名的值。现在，这只是一个字符串值，但最终将显示一个适合所选扩展的数据类型。
* 指示扩展名是否是修饰符扩展名



Once you’ve entered a value, an ‘Add’ button appears to the upper right. Clicking that button closes the dialog, and adds the extension to the resource – which you can then see in the Json or Tree view of the resource as shown here:

输入值后，右上方会出现一个“添加”按钮。单击该按钮将关闭对话框，并将扩展名添加到资源中，然后可以在资源的Json或Tree视图中看到，如下所示：



I do have to put more work into this – as I said above the datatype should really come from the extension definition, and at the moment the extension can only be applied to the root of the resource, but hopefully it should be enough for my colleague to complete the training tomorrow!

数据类型应该真的来自扩展定义，目前扩展只能应用于资源的根源

And do note that this does not mean that profiles are not needed – far from it! But the goal of clinFHIR is to allow people to learn FHIR, and also for the designers of resources and profiles to be able to make modifications ‘on the fly’ to test out ideas – so it seems a worthwhile addition.

注意，这并不意味着不需要配置文件

## 2.18 A complex scenario…复杂的场景

https://fhirblog.com/2017/03/28/a-complex-scenario/#more-159998

MARCH 28, 2017 [LEAVE A COMMENT](https://fhirblog.com/2017/03/28/a-complex-scenario/#respond)

[Bob Milius](https://www.linkedin.com/in/bobmilius/) has kindly allowed me to share an example of a more complex genetics based scenario that he has built (by hand!) – you can download it from [here](https://fhirblog.files.wordpress.com/2017/03/fullbundlecollection_hla-xml.zip).

To load it into [clinFHIR](http://clinfhir.com/), follow the steps I described in [this post](https://fhirblog.com/2017/03/26/import-resources-into-clinfhir-scenario-builder/) and you can then view it as a graph. Here’s an image:



(I expanded the size of the graph by clicking the ‘hide selector’ link in the navbar – this gives more space to the graph).

Viewing a graph of this size does have a few challenges, and there are a couple of links immediately above the display that can help this a bit.

* *Hide patient* will remove the patient resource (and references to it) from the graph.
* *Show direct references to {resource type}* will only show the selected resource, and all those resources with a reference either to or from it. You can still select resources and see the details in the right pane

Once in the Scenario Builder, you can modify it as required – deleting & removing references as you need to.

一旦生成场景生成器，则可以根据需要修改它。——根据需要删除和删除引用

So I hope that’s interesting. The other way to share these scenarios is through the library, but currently there’s an issue with this particular scenario that I have yet to resolve. In any case, some people may prefer to share scenarios in this way.

BTW – if you want to create a bundle of a scenario you’ve created, select the ‘*description*‘ tab in the middle pane, then the ‘*Json*‘ tab below that. This will show the json bundle for the current scenario. You can select the json in the page and then directly copy it. (I could create a download link to make it easier if people want that).

## 2.19 [Internet etiquette](https://fhirblog.com/2017/03/27/internet-etiquette/)

## https://fhirblog.com/2017/03/27/internet-etiquette/

MARCH 27, 2017 [2 COMMENTS](https://fhirblog.com/2017/03/27/internet-etiquette/#comments)

Just a very short post to the author of the site that is copying my posts (and that of others). While I’m flattered you think my work worthy of duplication,  there are more polite ways to recognize this than just a complete copy – including asking me first!

There’s no obvious way to contact you on your site, so hopefully this post will come to your attention.

Please contact me directly – my email is david dot hay twenty five at gmail.com so we can work out how you can do this in a more polite fashion.

2.20 Import resources into clinFHIR Scenario Builder

https://fhirblog.com/2017/03/26/import-resources-into-clinfhir-scenario-builder/#more-159949

MARCH 26, 2017 [1 COMMENT](https://fhirblog.com/2017/03/26/import-resources-into-clinfhir-scenario-builder/#comments)

Just a short note to let you know that I’ve added the ability to import resources directly into the [clinFHIR](http://clinfhir.com/) Scenario Builder (people have asked about this for a while). There are 3 formats currently supported:

* A single Json resource
* A Json bundle
* An XML bundle

只需要一个简短的说明，知道，已经添加资源直接导入到clinFHIR场景生成器。目前支持3中格式：单个JSON资源、单个JSON bundle、单个XML bundle。（注意：不支持单个XML资源。）

(Note that a single XML resource is not yet supported – just place it in a bundle if this is a problem right now)

To access this, load the scenario builder and click the ‘Import resource’ link in the middle of the navbar. You’ll get a dialog into which you can paste the text of the resource or bundle you want to import. You’ll get an error if you paste something invalid, but otherwise the resource/s should be imported into the Scenario Builder.

加载场景构建器，单击导航栏中的“inport Resource”，则显示一个对话框，可以在其中粘贴要导入的资源或者bundle的文本。如果粘贴无效，则会出现错误，否则将资源导入场景生成器中。

Note that the Xml import uses the currently selected Conformance server to perform the conversion from Xml to Json (suggested by [Bob Milius](https://www.linkedin.com/in/bobmilius/)) by POSTing the bundle to the Bundle endpoint as XML, and then reading it back as Json. I’ve tested using HAPI STU-3, but the other servers should work provided that they support the Bundle endpoint, and both Json & XML. This also means that the bundle must have a ‘[type](http://hl7.org/fhir/bundle-definitions.html#Bundle.type)‘ value of *collection*.

注意：XML导入使用当前选定的Conformance服务器通过一Bundle的方式将Bundle发布到其端点，然后以JSON的形式读取，从XML转化为JSON。我已经使用HAPI STU-3进行了测试，但其他服务器应该工作，只要它们支持Bundle端点，以及Json和XML。这也意味着bundle必须有一个collection的“ [type](http://hl7.org/fhir/bundle-definitions.html#Bundle.type) ”值。

You can include an id on your resource/s if you wish (but NOT on the bundle) – but if you do it’s your responsibility to ensure that it will be unique in the scenario you are building.

For bundles, the best strategy seems to be to include an entry.fullUrl with a uuid for each resource, and use that as the target of the references from other resources. The uuid will remain as the id for the resource in the Scenario Builder, and the references between resources should persist into the scenario. (This may cause some issues when saving the scenario to the Data Server – I need to look into that).

对于捆绑包，最好的策略似乎是包含一个entry.fullUrl与每个资源的uuid，并将其用作来自其他资源的引用的目标。在场景生成器中，uuid作为资源的id来保存，资源之间的引用应该持续到场景中。

## 2.21 The FHIR EndPoint resource FHIR端点资源

https://fhirblog.com/2017/03/14/the-fhir-endpoint-resource/#more-159937

MARCH 14, 2017 [LEAVE A COMMENT](https://fhirblog.com/2017/03/14/the-fhir-endpoint-resource/#respond)

In the [previous post](https://fhirblog.com/2017/03/13/provider-registries-in-fhir/), we talked about how FHIR could support a provider registry – the resources that would be involved and the types of query that you could use to access data within the registry. I was talking with [Brian Postlethwaite](https://www.linkedin.com/in/brianpos/) (who is one of the authors of these resources, being a co-chair of the Patient Administration committee within HL7) and he pointed out that I didn’t mention the [Endpoint](http://build.fhir.org/endpoint.html) resource.

Let’s rectify that.

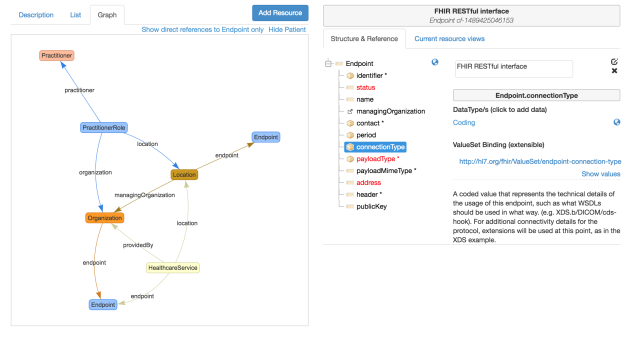
The Endpoint resource is new to STU3. Its function is stated as “*The technical details of an endpoint that can be used for electronic services, such as for web services providing XDS.b or a REST endpoint for another FHIR server. This may include any security context information*.”

端点资源是STU3的新功能。功能称为“可用于电子服务的端点的技术细节，例如：为另一个FHIR服务器提供XDS.b或REST端点的web服务”。这可能包含任何安全上下文信息。

So the idea is that once you’ve located your provider, the Endpoint tells you know how you can electronically communicate with that provider – eg to send them a referral (If you’re after a human style contact, then you’d use details from the ‘*telecom*’ property that a number of the resources have). As you may have gathered, a lot of what FHIR is about is discoverability of things – as well as how to share them.

一旦找到了供应商，Endpoint则会提供与供应商进行电子通信的方式。例如：发送一个referral。正如收集的那样，FHIR很多关于事物的可发现性。

In the following image, we’ve added a couple of Endpoints to the [clinFHIR](http://clinfhir.com/) scenario that we worked with before, and selected one them so we can see what properties it has available.



Some points to note.

* A single endpoint can, conceivably be shared by more than one resource instance – in the example above, both the Organization and the HealthcareService use the same endpoint. This will be implementation specific of course.
* The *connectionType* property indicates what sort of connection this is – FHIR REST server, XDS interface, direct interface, with the *address* property having the actual url (the format will depend on the type of interface of source). The connectionType [binding](https://www.hl7.org/fhir/terminologies.html#strength) is extensible – which means that you should use these values if they apply, but can add others if you need to. [Carrier pigeon](https://en.wikipedia.org/wiki/IP_over_Avian_Carriers)s for example…
* There are other properties to indicate payload type, the public key for secure conversations and so forth.

上面的实例中，定那个端点可以被多个资源实例共享，但是组织和医疗服务使用相同的端点。这是具体的实现。

connectionType属性指示了连接的类型，这是FHIR REST服务器、XDS接口、直接接口，具有实际url的地址属性。connection type的绑定类型是可扩展的。意味着如果需要，则可以添加其他的值。

其他属性可以指示有效的载荷类型，用于安全对话的钥匙。

By the way – do note also that selecting a property in the tree view of the resource type in clinFHIR will show the description of that property, and if it is coded will allow you to view the ValueSet that is bound to it.

注意; 在临床资源中的资源类型的树视图中选择一个属性将显示该属性的描述，如果它被编码将允许您查看绑定到它的ValueSet。

So – a short post, but does round out the Provider Directory discussion a bit further.

## 2.22 Provider registries in FHIR

https://fhirblog.com/2017/03/13/provider-registries-in-fhir/#more-159864

MARCH 13, 2017 [13 COMMENTS](https://fhirblog.com/2017/03/13/provider-registries-in-fhir/#comments)

So we’ve had some interest in New Zealand concerning Provider registries – locations where the details of providers and the services they provide can be maintained and discovered. There are a lot of potential uses for such a registry, but the two that we’ve been discussing are locating targets for referrals, and locating providers by specialty as part of Care Pathways.

Typical query use cases could include:

* Find a Provider by name (or other personal detail)
* Find a Provider by specialty – eg a physiotherapist
* Locate a Service provided by a facility like a hospital

典型的查询用例可能包括：

按名称查找提供者；通过展业寻找提供者；找到像医院这样的设施提供的服务。

There are lots more of course, but these will get us started.

So looking at FHIR, there are a number of resources that we’re going to need.

首先看FHIR，我们需要一些资源。

(Note that we’ll use the[STU3 candidate version](http://hl7.org/fhir/2017Jan/index.html) from January this year as this is the version currently on the public HAPI3 server which we’ll use for this. There are a number of differences planned for the final release of STU3 – most notably the removal of role information from the Practitioner resource – but these won’t affect this discussion.)

Here are the resources of interest (the links are to the current build – I’ll update them to the final STU3 version once that is released):

The [Practitioner](http://build.fhir.org/practitioner.html) resource represents a person involved in the delivery of healthcare – a doctor, nurse, physio, receptionist – pretty much any human (or animal for that matter). This is a human Provider.

Practitioner资源表示代表参与医疗保健服务的人：医生、护士、理疗师和接待员等以及几乎任何人，这是一个提供者。

The [PractitionerRole](http://build.fhir.org/practitionerrole.html) resource. This represents the role/s that a Practitioner can be in – details of the role, where they perform it, the times they are available and so forth. This is a new resource and was split off from the Practitioner specifically to support this kind of use. (See [Brian Postlethwaites blog](https://brianpos.com/2016/10/16/practitioner-role-vs-practitionerrole/) for more discussion on this point, and why the change was made which was driven by practical experience at Connectathons). Note that a single Practitioner can have multiple roles

PractitionerRole资源。表示practitioner的角色细节、执行方式、可用时间等方面的角色。这是一个新的资源，并从practitioner中分离出来，专门用来支持这种使用。注意：一个practitioner可以有多个角色。

The [HealthcareService](http://build.fhir.org/healthcareservice.html) describes an actual service (rather than a person) that is provided by some organization. For example a Cardiology outpatient clinic at a hospital, or a primary care service. This is a service Provider.

HealthcareService描述了一个由组织提供的实际服务（而不是一个人）。例如：在医院的出院门诊服务或者医院的初级保健服务。这是一个服务提供商。

The [Organization](http://build.fhir.org/organization.html) represents the grouping of people or companies that are delivering care. It’s a rather abstract thing that isn’t the actual place where care is delivered (that’s the location) – rather the entity that organizes the care (hence the name I guess)

Organization表示提供护理的人员或者公司的分组。这是一个相当抽象的事情，并不是实际护理的地方，而是组织护理的实体。

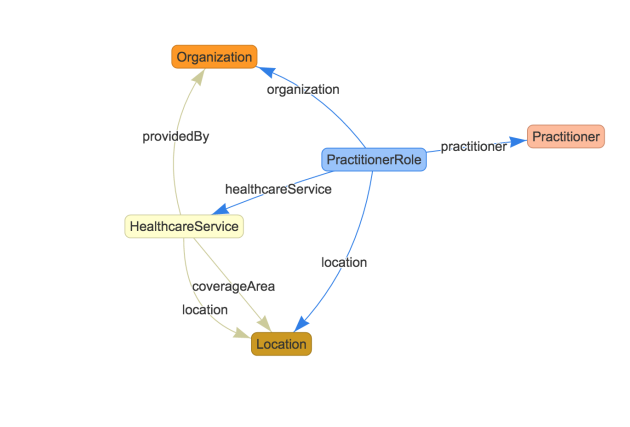
The [Location](http://build.fhir.org/location.html) is the actual place where care is delivered – the physical hospital or Primary Care/ambulatory care clinic.

Location是实际护理的地方，实体医院或者初级保健门诊诊所等。

Note that the term ‘Provider’ applies both to people (Practitioners) and services (HealthcareService). We’ll use the specific FHIR term in the following discussion.

注意：“Provider”适用于人（或者从业者）以及服务（HealthcareService）。将在下面的讨论汇总使用具体的FHIR术语。

Here’s an image of how the resources inter-relate:这是一个资源如何关联的形象。



This was generated using the Scenario Builder in [clinFHIR](http://clinfhir.com/). I set the Conformance and Data server to the public HAPI3  server and added a single instance of each resource type to the scenario. Then, I set the various references between resources, added some data to them and saved the lot to the Data server so I could check the queries. There are references to other resources of course, and the diagram does not indicate the multiplicity of the references – but it’s easy for you to do this yourself if you want to experiment further.

将一致性和数据服务器设置为公共HAPI3服务器，并向场景中添加每种资源类型定那个实例。然后，在资源中设置各种引用，并向他们添加一些数据，将数据保存在数据服务器汇总，以便查询。当然还有其他资源的引用。

As you can see there are 2 main resources that pull everything together – the PractitionerRole (which lets us find individual Practitioners) and the HealthcareService, which is about Services rather than people.

正如图中所示，两个主要资源将所有东西连在一起：PractitionerRole和HealthcareService（这是关于服务而不是人）。

Let’s think about what queries we’ll need to meet our Use Cases.

需要考虑，需要什么查询来满足用例。

Before we do this, just a reminder that each resource type has a number of search parameters defined in the specification. A server is not obliged to support all of them (it indicates which ones using the [CapabilityStatement](http://hl7.org/fhir/2017Jan/capabilitystatement.html) resource), and can support others (which it documents using the [SearchParameter](http://hl7.org/fhir/2017Jan/searchparameter.html) resource)

在这样做之前，只是说明了每个资源类型在规范中定义了多个搜索参数。服务器没有义务支持所有的这些（使用CapablityStatement资源指示这些），并且还可以支持其他的内容（使用SearchParameter资源来记录）

**Find a Practitioner by name 通过名字查找一个从业者**

To start with, let’s assume we know the Practitioners name, and just want their personal details (like phone number or email). In this case we’re just querying the Practitioner resource. Here’s a sample query we could use: 首先，我们知道Practitioner.name，需要知道从业者的详细信息（像：电话号码或邮件）。在这个例子中，仅仅查询Practitioner资源。

<http://fhirtest.uhn.ca/baseDstu3/Practitioner?name=welby>

It will return all the Practitioners whose name is Welby. Simple. And, of course, we could use any of the defined search parameters that are supported by the registry.

将返回name=welby的所有从业者。当然，可以使用注册表支持的任何定义的搜索参数。

Note that this query will not return any role based information (this is one place where the final STU3 version will differ from the candidate version that we are using) – see the link to Brians post above for more information on why this was done.

注意：此查询不会返回任何会给予角色的信息。

**Find a Practitioner with a given specialty 寻找具有特定专长的从业者**

Moving on, let’s think how we can find Practitioners by specialty.

Because the specialty is associated with the PractitionerRole and not the Practitioner, our query will be on PractitionerRole. (Note that a single Practitioner can have multiple PractitionerRoles of course). Here’s a query that will return the Practitioner roles for cardiology (where 394579002 is the SNOMED term for cardiology).因为专业与PractitionerRole相关，而与Practitioner无关。（注意：一个从业者可以有多个PractitionerRoles）。这是一个查询，将返回心脏病学的从业角色。

<http://fhirtest.uhn.ca/baseDstu3/PractitionerRole?specialty=394579002>

(Note that you should really specify the system when doing a lookup on coded elements – so the more accurate query would be:注意：对编码元素的查找，应该指定真正的系统。因此最准确的查询就是：

[http://fhirtest.uhn.ca/baseDstu3/PractitionerRole?](http://fhirtest.uhn.ca/baseDstu3/PractitionerRole?specialty=http://snomed.info/sct|394579002)

[specialty=http://snomed.info/sct|394579002](http://fhirtest.uhn.ca/baseDstu3/PractitionerRole?specialty=http://snomed.info/sct|394579002)

but I’ll leave that out to keep the queries more understandable.

This will return a bundle of PractitionerRole resources – useful, but we really want the names of the Practitioners. We could do separate queries for each Practitioner of course, but we could also take advantage of the [\_include](http://hl7.org/fhir/search.html#revinclude) search parameter. The following query will return the PractitionerRoles, but will also include the referenced Practitioners in the bundle as well:

这将返回一系列PractitionerRole资源。当然可以对每个从业者执行单独查询，但是也可以使用\_include搜索参数进行查询。下面的查询将返回PractitionerRoles，但是将包括在bundle中引用的从业者。

[http://fhirtest.uhn.ca/baseDstu3/PractitionerRole?](http://fhirtest.uhn.ca/baseDstu3/PractitionerRole?specialty=394579002&_include=PractitionerRole:practitioner)

[specialty=394579002&\_include=PractitionerRole:practitioner](http://fhirtest.uhn.ca/baseDstu3/PractitionerRole?specialty=394579002&_include=PractitionerRole:practitioner)

and why stop at the Practitioner? Let’s get the Organization and Location as well:

[http://fhirtest.uhn.ca/baseDstu3/PractitionerRole?](http://fhirtest.uhn.ca/baseDstu3/PractitionerRole?specialty=394579002&_include=PractitionerRole:practitioner&_include=PractitionerRole:organization&_include=PractitionerRole:location)

[specialty=394579002&\_include=PractitionerRole:practitioner&](http://fhirtest.uhn.ca/baseDstu3/PractitionerRole?specialty=394579002&_include=PractitionerRole:practitioner&_include=PractitionerRole:organization&_include=PractitionerRole:location)

[\_include=PractitionerRole:organization&](http://fhirtest.uhn.ca/baseDstu3/PractitionerRole?specialty=394579002&_include=PractitionerRole:practitioner&_include=PractitionerRole:organization&_include=PractitionerRole:location)

[\_include=PractitionerRole:location](http://fhirtest.uhn.ca/baseDstu3/PractitionerRole?specialty=394579002&_include=PractitionerRole:practitioner&_include=PractitionerRole:organization&_include=PractitionerRole:location)

This will return all the needed resources in a single query (assuming server support of course).

将会返回所需的资源。

**Find services 查找服务**

If we want to locate services rather than people, then a query on HealthcareService is required.

如果需要定位服务而不是人。则需要对HealthcareService进行查询。

There are a number of different coded elements that we could use:

* serviceCategory
* serviceType (eg the specialty of the service)
* specialty

as well as other query parameters such as

* location
* providedBy
* coverageArea

so the queries used are going to be very dependent on the actual implementation.

可以使用许多不同的编码元素：服务种类；服务类型（例如：服务的专长）；专长

以及其他的查询参数：location、providedBy、coverageArea。所使用的查询将非常依赖于实际的实现。

For the purposes of this post, let’s assume we’ve used the service type as the element to query on – eg Cardiology clinic or Physiotherapy service. The following query will return all the HealthcareService resources where the type is Cardiology. 假设已经使用服务类型作为查询的参数，例如：心脏病诊所或者物理治疗服务。以下的查询将返回type=cardiology的所有HealthcareService资源。

http://fhirtest.uhn.ca/baseDstu3/HealthcareService?

servicetype=http://hl7.org/fhir/service-type|165

Note that the ValueSet (and hence system) used for the serviceType is different to the specialty ValueSet used by Practitioner.

注意：HealthcareService的serviceType的值集是不同于Practitioner中specialty中的值集。

Of course we can use the \_include search parameter to include referenced resources in the query – location and providedBy (Organization) would seem two likely candidates.

当然，可以使用\_include查询参数查询包含引用的资源。location和provideBy是两个可能的候选项。

**Last thoughts**

So we’ve taken a – very – superficial look at how we could implement a Provider registry in FHIR. The design of a real system will be a lot more complex of course, and careful thought will need to be given on which relationship are important.

I suspect that the main issues in implementing such a registry will not be technical, but rather around governance – who has the right to update and access this information.

One thing that did stand out for me as I was preparing this post was how there are a number of coded elements with ValueSets that will need to be chosen and defined for a particular implementation – such as PractitionerRole.service or Healthcareservice.category (there are lots of others, and don’t even think about extensions!)

有一些需要为特定实现选择和定义的ValueSets的编码元素，例如PractitionerRole.service或Healthcareservice.category（有很多人，甚至不考虑扩展！）

This is, of course, where profiling comes in to play…

Cheers…

BTW – if you’re interested in a ‘real’ Implementation guide for a Provider Directory, check out the [Argonaut Provider Directory](http://build.fhir.org/ig/Healthedata1/Argo-PD/) . (Note that this is the development version – I’ll replace with the ‘released’ version when that is ready).

## 2.23 Building a profile in the new clinFHIR 在clinFHIR中建立profile

https://fhirblog.com/2017/03/10/building-a-profile-in-the-new-clinfhir/#more-159819

MARCH 10, 2017 [2 COMMENTS](https://fhirblog.com/2017/03/10/building-a-profile-in-the-new-clinfhir/#comments)

So I had a question from someone who was wanting to develop some profiles using [clinFHIR](http://clinfhir.com/) and not sure of the best way to approach this. I’ll write in more detail later on, but as there has been a substantial change in process and User Interface from the previous version, I thought I’d just do a quick post about how the new process is intended to work.

使用clinFHIR开发profiles，不是一个最好的办法。

Because the focus is really on helping folk new to FHIR rather than being a comprehensive profiling tool (use [Forge](https://fhir.furore.com/forge/) for that), the starting point is a Logical Model, from which the real profiles can be generated. To do this, there are some constraints in the way that the logical models are created – and the generated profiles themselves cover only a small subset of the profiling functionality that FHIR supports.

起点是一个逻辑模型，从中可以生成真实的配置文件。所以，在创建逻辑模型的方式上存在一些限制，生成的profile本身包含FHIR支持的分析功能的一小部分。

There are a number of clinFHIR modules that you can use for this. These include:

* The **Logical Modeller** – builds the model and generates the profile (StructureDefinition resource)
* The **Scenario Builder** – allows you to assemble resources based on profiles (and logical models for that matter) to represent a particular Use Care case
* The **Extension Definition builder** will create a – well – Extension Definition (actually a StructureDefinition resource
* The **CodeSystem builder allows** an ‘ad-hoc’ CodeSystem resource (with associated ValueSet) to be built. The idea is to allow you to build a ‘quick and dirty’ ValueSet that you can bind to an Extension Definition as part of the design process (again, clinFHIR is intended as a training & analysis tool). See [here](https://fhirblog.com/2017/02/20/clinfhir-and-the-codesystem-resource-stu-3/) for more detail.
* The **ValueSet builder** for simple SNOMED based ValueSets.

可以使用clinFHIR的许多模块：

逻辑模块：建立模型并生成profile（StructureDefinition资源）

场景构建器：允许基于profile（此事件的逻辑模型）的装配资源来代表特定的使用案例。

[CodeSystem builder](http://clinfhir.com/codeSystem.html) ：允许建立一个“临时”的codesystem。这个想法允许构建一个快速的值集，可以将其绑定到扩展定义作为设计过程的一部分。（再次clinFHIR旨在作为培训和分析工具）。

valueSet builder是基于ValueSet的简单的SNOMED。

[Here’s a link](https://fhirblog.com/2017/02/20/new-ui-for-clinfhir/) to a bit more detail, and [here](https://fhirblog.com/2017/01/19/webinars-on-clinfhir-scenario-builder-and-logical-modelers/) are some videos

So the overall process goes something like this (assuming you are wanting to profile a single resource).

* Select the servers (I use HAPI STU-3 myself, others should work but this is the one I test on).
* Create a new logical model, with a model type of ‘Single Resource’. Select your base resource and check the ‘copy elements into model’ checkbox. This will create a new Logical Model that has all the fields from the core resource.
* In the designer, remove the elements you are not supporting and add a new element for each ‘extra’ property you want to save (this will become an extension)
* For each of the extra elements, set the FHIR mapping path (bottom left of the edit dialog) to an extension path (eg Encounter.extension) and a ‘find’ button will appear that will allow you to select the extension definition. You can type in the url directly if you know it, but it’s better to select it. If necessary, you can save the model, use the Extension Definition builder to create it, then come back and select it.
* Once all the changes have been made and saved, select the ‘Model’ tab to the upper right, and then the ‘Generate Profile’ link which is at the lower right. All being well (for example all elements in the model have a FHIR mapping path), the app will generate a standard FHIR profile based on the logical model (with a name derived from the logical model name)
* You can then use the Scenario Builder to build resource instances based on that profile.

整个过程如下：

选择服务器（使用HAPI STU-3）；

创建一个新的逻辑模型，模型类型为“Single Resource”。选择基本资源并检查“ 元素复制到模型”的复选框。这将构建一个新的逻辑模型，其中包含资源的字段；

在设计器中，删除不支持的元素，并为要保存的每个“extra”属性添加一个新元素（这将是扩展）。

对于每个额外的元素，将FHIR映射路径设置为扩展路径，并将出现一个查找按钮，将允许选择扩展名的定义。可以直接输入url。如果需要，可以保存模型，使用“扩展器创建”，然后返回并选择它。

一旦进行所有更改并保存，请选择右上方的“模型”选项卡，然后选择右侧的“生成配置文件”链接。一切都很好（例如模型中的所有元素都有一个FHIR映射路径），该应用将基于逻辑模型生成一个标准的FHIR配置文件（具有从逻辑模型名称导出的名称）。

然后可以使用场景生成器根据该配置文件构建资源实例。

If you want to create a new ValueSet to bind to a coded element – either an extension or a core element, then you have a couple of options.

如果要创建一个新的值集绑定到一个编码元素中,不论是扩展还是核心元素，都有 几个选项。

The ValueSet builder is good for building simple SNOMED based ValueSets, and the CodeSystem builder is useful for ‘ad hoc’ code systems (it automatically builds an associated ‘select all’ ValueSet), good when you just want to put something together to illustrate a point, or to list the potential values and have someone else find them in SNOMED. Note that there are much more advanced tooling available for building ValueSets – like [Ontoserver](https://ontoserver.csiro.au/docs/4.1/api-fhir.html) for example

ValueSet builder是基于ValueSet构建简单的SNOMED，codesystem builder对于“ad hoc”代码系统非常有用（它自动构建一个关联的“选择所有”ValueSet）。注意：有更多先进的工具用于构建ValueSet。

You don’t need to use clinFHIR to create the extension definition (or you shouldn’t), but you’ll need to upload them to the Conformance server yourself.

不需要使用clinFHIR来创建扩展定义（或者不应该），但是需要自己将其上传到conformance服务器中。

As always this is a work in progress – I’ve only just got the basic functionality working so there’s a way to go before it is robust enough for use (with the May ‘Clinicians on FHIR’ event at the next [Working Group Meeting](http://www.hl7.org/events/working_group_meeting/2017/05/index_7.cfm) being the deadline) but I did want to let people know the development direction. The scenario builder in particular right now will choke on complex extensions at the moment, and support for the more complex resource types (like CarePlan) need some attention (though it will now save resources to the Data Server).

和往常一样，这是正在进行的工作。只是获取了基本的功能。目前场景构建器遏制复杂的扩展，并且支持更复杂的资源类型。

I intend to add a bit more capability to the profile generation – in particular simple slicing – but it will never be a match for [Forge](https://fhir.furore.com/forge/)! I also intend to connect the Extension definition builder to [Simplifier](https://fhir.furore.com/simplifier-net/) – so that Extension Definitions you create can be easily saved there, and also to enhance the ‘find an extension definition’ capabilities – this will be really important moving forward to minimise the creation of extension definitions that represent the same thing.

打算添加一些更多的功能，在配置文件中生成，特别是简单的slicing，但是永远不会匹配Forge。打算将扩展定义构建器链接到simplifier中。以便创建的扩展定义可以轻松地保存，并且还可以增强“查找扩展定义”功能。这将是非常重要的，以最小化扩展创建代表同样东西的定义。

But I think (hope) this is going to represent the ‘final’ version of clinFHIR (from an overall functionality / UI perspective)!

This is really short description, I’ll do a longer post – or maybe a video – when the dust has settled on the FHIR QA process!

Comments welcomed…

## 2.24 clinFHIR and the CodeSystem resource (STU-3)

https://fhirblog.com/2017/02/20/clinfhir-and-the-codesystem-resource-stu-3/#more-159742

FEBRUARY 20, 2017 [1 COMMENT](https://fhirblog.com/2017/02/20/clinfhir-and-the-codesystem-resource-stu-3/#comments)

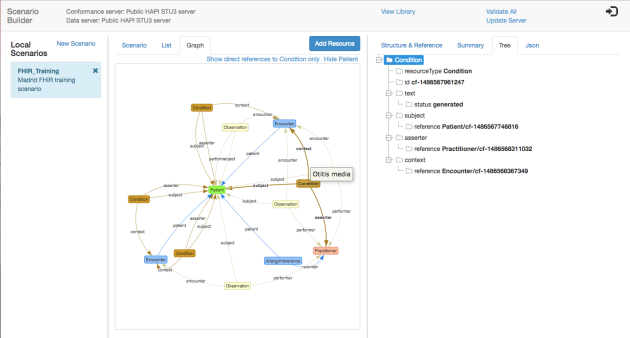
As you’ll probably be aware, a lot of my focus in recent months has been in updating [clinFHIR](http://clinfhir.com/) to help non-technical folk like clinicians understand what FHIR is, and how they can get involved in FHIR related projects. Here’s a [White Paper](https://orionhealth.com/nz/knowledge-hub/white-papers/fhir-for-clinicians/) I did for Orion Health on the subject…

There are a couple of core components that have been developed to assist this.

开发几个核心组件来协助这一点

The **Scenario Builder** lets you create and link together resources that shows how FHIR can represent given clinical scenarios (There are often multiple ways  ). The idea is that you have a scenario (e.g.  representing a consultation) and you create resource instances that represent the various components of that scenario (such as an [Encounter](http://build.fhir.org/encounter.html), a [ClinialImpression](http://build.fhir.org/clinicalimpression.html), [Observations](http://build.fhir.org/observation.html), [Conditions](http://build.fhir.org/condition.html) and so forth). The builder can show the linked resources in a variety of different formats, the most helpful being a ‘graph’ of interconnected resources like this:

场景生成器允许创建和链接资源，显示FHIR如何代表特定的临床场景。这个想法是一个场景，并创建代表该场景的各种组件的资源实例（例如：就诊、临床印象、观察、症状等）构建器可以以各种不同的格式显示链接的资源。



The **Logical Modeler** allows you to, in effect, create your own resource types and then use them as the ‘template’ to build a resource in the Scenario builder (rather than the core resource definitions from the spec). The intention is to use the Logical models to define the extra elements needed (or remove those not needed) to support a specific scenario in a way that is quick and easy, and then once you’re sure that you have the bits you need, you can then use those models as the basic of generating the actual FHIR profiles – either automatically or manually using a tool like [Forge](https://fhir.furore.com/forge/).

逻辑模型允许创建自己的资源类型，然后作为“模板”构建的情景构建器（而不是从规范的核心资源中定义）的资源。目的是使用逻辑模型来定义所需的额外元素（或者删除那些不需要的元素），以便以一种快捷方式支持特定的场景，然后确定需要的位置，然后，可以使用这些模型作为生成实际FHIR配置文件的基础。

So I can imagine a process like:

1. Gather requirements
2. Create Logical Model/s (based on a Core resource type)
3. Use them in Scenario Builder to build resource instances to test against real scenarios. You can combine logical models and Core resource types in the same scenario of course.
4. Repeat 2 & 3 as required (perhaps clarifying requirements as you go)

过程为：收集要求；创建逻辑模型（基于核心资源类型）；在场景生成器中使用它们构建资源实例，以针对真实场景进行测试。也可以将逻辑模型和核心资源类型结合在一起；根据需要重复2和3 。

When completed, use as the basis for generating the real FHIR artifacts.

完成之后，作为真正的FHIR工件的基础

But one thing that is missing from this is the ability to create ‘example’ values for coded elements (such as a diagnosis or an observation), and when working with clinicians it’s actually quite important that the examples are as real as possible.

但是，有一件事就是能够为编码元素创建“example”值（例如：诊断或者观察）。当与临床医生合作的时候，这些示例值尽可能真实是非常重要的。

I did build a simple ValueSet editor (it’s still there) that allows you to create a ValueSet from SNOMED terms, but the problem is that it only really works with SNOMED – and navigating that terminology is not simple if you are unfamiliar with it, so I felt it may not be the best choice, especially when working in a ‘real-time’ analysis situation.

创建了一个简单的ValueSet编辑器（它仍然在那里），允许您从SNOMED术语创建一个ValueSet，但问题是它只能与SNOMED一起使用 - 如果您不熟悉该术语，那么导航该术语不简单我觉得这可能不是最好的选择，特别是在“实时”分析的情况下工作。

When I noted that the [clever terminology folk](http://build.fhir.org/terminology-module.html)have created a specific resource in STU-3 to represent an ‘ad-hoc’ CodeSystem, I realized that this could be an alternative way for a user to quickly create a set of example values to help define the purpose of a given element that was fully compliant with the FHIR infrastructure – so operations like [ValueSet/{id}/$expand](http://build.fhir.org/valueset-operations.html#4.7.16.1)will still work – which is important in the scenario builder. I’m not suggesting that this is how you’d create the ‘real’ ValueSet, but it is a way to help understand and document requirements.

注意到在stu-3中创建特定资源的术语folk中表示一个“ad-hoc” codeSystem，意识到用户快速创建一组示例值以帮助定义的FHIR基础中完全兼容的给定元素（在场景构建器中是非常重要的），可以使用ValueSet/{id}/$expand 操作。

So the process becomes:

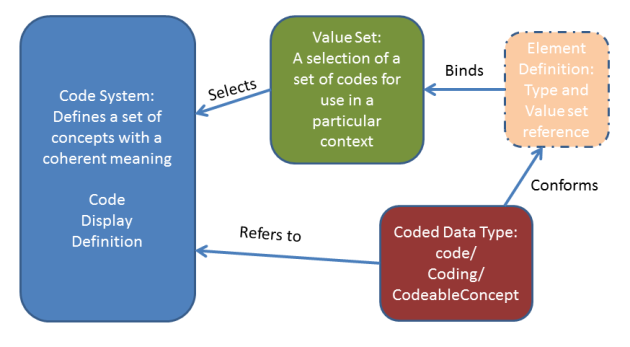
1. Gather requirements
2. Create Logical Model/s (based on a Core resource type)
3. Build CodeSystems (or ValueSets) to represent example values for coded data, and bind to the element in the Logical Modeler
4. Use in Scenario Builder to test against real scenarios
5. Repeat 2,3 & 4 as required – again validating requirements

这个过程为：收集需求；创建逻辑模型（基于核心资源类型）；构建codesystem（或ValueSet）来表示编码数据的示例值，并绑定到逻辑建模器中的元素；在场景生成器中使用以测试实际的场景；需要重复2、3和4，再次验证要求。

When complete, use as the basis for generating the real FHIR artifacts.

Before we take a look at the clinFHIR CodeSystem builder, a reminder about the relationship between CodeSystem, ValueSet and the binding from the Element definition in the model. (We also talked about this [just recently](https://fhirblog.com/2017/01/10/extending-a-required-valueset-binding/)) Here’s the picture we use: (from [the spec](http://build.fhir.org/terminology-module.html))

clinFHIR CodeSystem构建器之前，提醒了有关CodeSystem，ValueSet与模型中Element元素绑定的关系。



In summary we will:

* Create a CodeSystem that has all the example values we want
* Create a ValueSet that contains (or refers to) all of those values, which will be the ‘binding’ between the element definition and the CodeSystem
* In the Logical modeler, bind the coded value to the ValueSet
* When the profile generation is working, maybe generate the profile

总而言之：

创建一个具有所需所有示例值的codesystem；

创建包含所有这些值的ValueSet，这将是元素定义和codesystem之间的绑定；

在逻辑模型构建器中，将编码值绑定到ValueSet中。

当配置文件生成工件的时候，可能会生成配置文件。

So there’s 2 aspects to discuss – creating the CodeSystem and ValueSet resources, and then binding the ValueSet to the model. Let’s consider these in turn.

可以在两个方面讨论，创建codesystem和ValueSet资源，然后将值集绑定到模型中。轮流考虑一下。

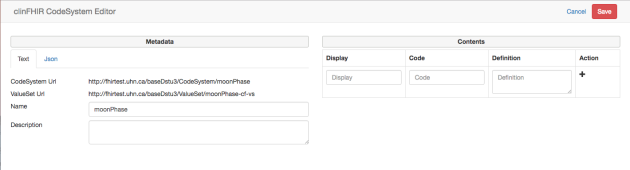
**Creating the CodeSystem**

Start [clinFHIR](http://clinfhir.com/) and select the CodeSystem Builder.

Click the ‘new’ button in the top right hand corner. You’ll get a dialog where you can enter a name (with no spaces) and then click the ‘Check’ button to ensure it is unique on the Terminology server. The name will form part of the url for the CodeSystem according to the pattern: *[server]/CodeSystem/[name]*, and the builder will also automatically create the matching ValueSet with a url of *[server]/ValueSet/[name]-cf-vs*that include the entire CodeSystem

点击右上角的“New”按钮，将收到一个对话框。可以在对话框汇总输入名称（无空格），然后单击“check”按钮，以确保在术语服务器中是唯一的。该名称将根据以下模式构成codesystem的URL的一部分：*[server] / CodeSystem / [name]*，构建器还将自动使用*[server] / ValueSet / [name]*的URL自动创建匹配的ValueSet *- cf-vs*包括整个CodeSystem。

Enter a title (short sentence) and a description then click save. You’ll get a screen like this:



You can now enter the values you want in the ‘contents’ pane to the right. For each value (technically we call this a concept) you enter:

* Display (required) – what shows to the user
* Code (required) – that actual code that is stored. Must be unique in the CodeSystem.
* Definition (optional) – a more detailed description of the concept.

您现在可以在右侧的“内容”窗格中输入所需的值。对于每个值（技术上我们称之为一个概念），您输入：

* 显示（必填） - 显示给用户
* 代码（必需） - 存储的实际代码。在CodeSystem中必须是唯一的。
* 定义（可选） - 概念的更详细描述。

You can re-order concepts using the arrow buttons. The order doesn’t matter in FHIR, but can make a difference for humans! 可以使用箭头按钮重新排列概念

At the top right is a Save button which will save both CodeSystem & ValueSet to the Terminology server. It appears when you make changes. 右上角是一个保存按钮，它将CodeSystem和ValueSet保存到术语服务器。当您进行更改时会显示。

Here’s an example with some (sort of) real data in it:



And that’s it! There’s also an option to load existing CodeSystems (It is quite interesting to see what has been defined in the spec), and to edit CodeSystems that were created by clinFHIR.

还有一个选项可以加载现有的CodeSystems（看看规范中定义的内容是非常有趣的），还可以编辑由clinFHIR创建的CodeSystems。

**Binding the ValueSet to an element in the Logical Modeler**

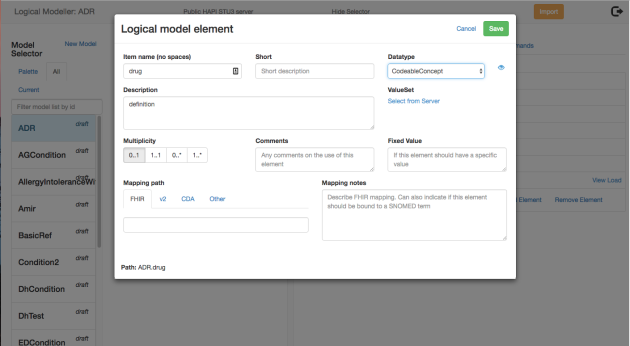
Start clinFHIR and select the Logical Modeler. 启动clinFHIR并选择逻辑建模器。

Create or select a Logical Model. 创建或者选择一个逻辑模型

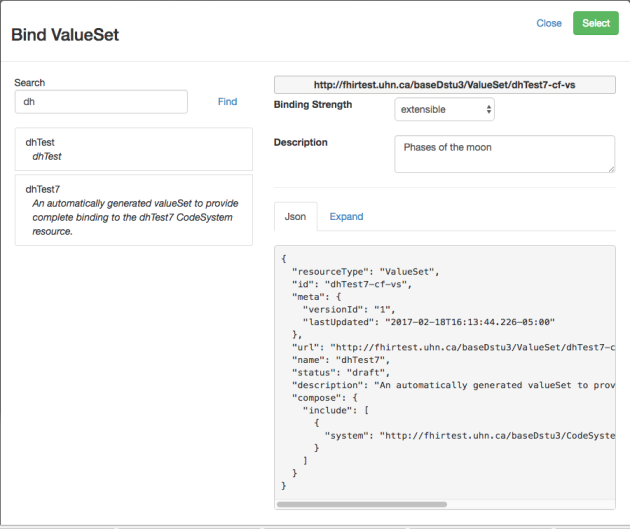
Enter a new element (or edit an existing) and set the datatype to CodeableConcept. Just under the datatype will be a link labelled ‘Select from Server’. Click this to select the ValueSet to bind.

输入一个新的元素（或者编辑已经存在的元素）并设置数据类型为CodeableConcept。数据类型下方将是一个标记为“从服务器选择”的链接。单击此按钮选择要绑定的ValueSet。

Here’s an image of the element edit screen:



And here’s the ValueSet selector:



The ‘Expand’ tab will display all the contents of the ValueSet (In this case it will pull in the values from the CodeSystem). “expand”选项卡将显示ValueSet的所有内容。

Select and save, and you’re done.

So there you go – a simple way to create example lists for coded data. Note that there’s nothing stopping you creating ValueSets directly – this is just a quick way to keep an analysis going.

一个简单的方法来创建编码数据的实例列表。

Note that the CodeSystem is new in STU-3. I will modify the builder to be able to create STU-2 ValueSets as well (In STU-2 the concepts are inside the ValueSet rather than being in a separate resource), but haven’t figured out the best way to do that just yet….

## 2.25 New UI for clinFHIR

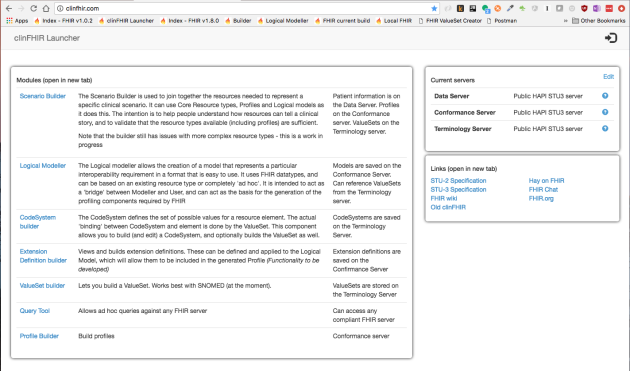
https://fhirblog.com/2017/02/20/new-ui-for-clinfhir/#more-159716

FEBRUARY 20, 2017 [4 COMMENTS](https://fhirblog.com/2017/02/20/new-ui-for-clinfhir/#comments)

If you’ve used [clinFHIR](http://clinfhir.com/) in the last 24 hours then you’ll have noticed a new User Interface. This is a recognition that clinFHIR is now really a number of ‘co-operating’ modules rather than a single application, so the previous UI wasn’t really the best any more.

In this post I’ll describe the UI (Actually I call it a ‘Launcher’ as it’s really a start point for the various modules).

Here’s a screen dump of the new UI:



The main parts of the UI are as follows. UI的主要部分如下：

At the top right is the log in icon. If you’re already logged in then mousing over will show the email, otherwise click to log in. The only purpose of logging in is to help manage updating of FHIR artifacts – for example only the author of a scenario is able to update it. This is a work in progress (as is the whole app actually  ) but will be rolled out across all modules eventually.

右上角是登录图标，如果已经登录，则鼠标悬停将显示电子邮件，否则单击以登录。登录的唯一目的是帮助管理FHIR工件的更新。例如：只有方案的作者才能更新它。这是一个正在进行的工作（实际上是整个应用程序），但最终在所有模块中推出。

To the left is a list of all the modules in clinFHIR – I’ll describe them in a minute.

在左边是一个列表中所有的模块

To the top right is a list of the 3 server roles that clinFHIR recognizes. Clicking the edit button will allow you to change them, and clicking the question mark will test that the server is on-line and available. By default, I tend to use the HAPI server for all server roles.

右上角是clinFHIR识别的三个服务器角色的列表。单击编辑按钮允许更改，并单击?将测试服务器是否在线并可用。默认情况下，倾向于将HAPI服务器用于所有服务器的角色。

To the bottom right is a list of helpful links – I’ll likely add to this list as time goes on – ping me if you have a suggestion. There’s also a link to the previous version of clinFHIR if you need it.

右下角是有用的链接列表，随着时间的推移，可能会将此添加到此列表中。

So, now a description of the modules.

The **Scenario Builder** lets you assemble a collection of the resources that shows how FHIR represents given scenarios. It can use the core resource types and logical models as the basis of creating these scenarios, and I’m working on supporting profiles. There are still some gaps in functionality, but this is the main ‘builder’ app moving forwards, and will replace the current resource builder.

场景生成器：可以组装资源的集合，显示FHIR如何表示给定的方案。可以使用核心资源类型和逻辑模型作为创建这些场景的基础，努力支持profile。功能上仍然存在一些差距，但是主要的“builder”APP 向前移动，将替换当前的资源构建器。

The **Logical Modeler** is used to create a model that represents some clinical ‘thing’ and is intended to be used when interacting with clinicians to capture clinical requirements. It can be used in 2 main ways:

* You can build a model that is based on an existing resource type and will become a FHIR profile. The plan is that eventually this will be an automated process – create the model, enter mappings to the core resource and Extension Definitions, and the Profile can be directly created
* You can build a model that is not based on an existing resource, but rather just captures requirements. The thinking is that you can use this during analysis, and then create the actual profiles as a secondary action.

逻辑建模：用于创建一个标识临床“东西”的模型。旨在与临床医生进行交互时候，使用捕捉临床要求的模型。可用于两种主要的方式：

构建基于现有资源类型的模型，并将称为FHIR profile。该计划是一个自动化过程：创建模型，输入映射到核心资源和扩展定义，并且可以直接创建profile。

可以创建不基于现有资源的模型。但是只是捕获需求。这个想法只能用于分析过程中，然后创建市级的profile作为辅助操作。

The **CodeSystem builder** is new, and allows you to build – well – [CodeSystem](http://build.fhir.org/codesystem.html) resources. These are new in STU-3 (they were split off from the [ValueSet](http://build.fhir.org/valueset.html) resource). I think that these will be very useful when building logical models in particular, and will do a post of them shortly.

编码系统构建器：这个功能是新的，允许建立一个codesystem资源。这些在STU-3中是新的（它们从[ValueSet](http://build.fhir.org/valueset.html)资源中分离出来）。我认为这些在建立逻辑模型时非常有用，并且很快就会发布。

The **Extension Definition builder** is used to create the definitions for Extensions (actually [StructureDefinition](http://build.fhir.org/structuredefinition.html) resources of course). These are ‘extra’ elements that are added to resource types to represent data specific to your Use Case.

扩展定义构建器：用于创建扩展的定义（实际上是StructureDefinition资源）。这些是添加到资源类型中表示特定用例的数据的“额外”元素。

The **ValueSet builder** will build simple ValueSets. I’m not sure how important this component will be for clinFHIR now that the CodeSystem builder has been developed – the ValueSets are quite simple and there are much more comprehensive tools available (or becoming available). Note that the CodeSystem builder will automatically create the ValueSet that is needed when binding between a coded element in a resource and the CodeSystem.

值集构建器：将创建一个简单的值集。不确定这个组件对于clinFHIR的重要性，因为已经开发了codesystem构建器：ValueSet非常简单，还有更多全面的工具可用。注意：codesystem构建器将自动创建在资源中的编码元素与codesystem之间绑定时候所需的ValueSet。

The **Query tool** allows you to make ad-hoc queries on any FHIR server. It will display the servers’ conformance (now called [CapabilityStatement](http://build.fhir.org/capabilitystatement.html)) and also a pick list of the search parameters supported by that server.

查询工具：允许FHIR服务器中的额ad-hoc查询。将会显示服务器的conformance（现在称为CapablityStatement）以及该服务器支持的搜索参数的选择列表。

The **Profile Builder** builds simple profiles. I’m not sure about the future of this component – once I get the bugs ironed out of the Logical Modeler to the point where it can generate profiles, then it may become deprecated. In any case, the [Forge](https://fhir.furore.com/forge/) tool is much more comprehensive.

profile构建器：建立一个简单的profile。不确定这个组件的发展。一旦将逻辑建模中的错误归结为可以生成配置文件的地方，则可能会被启用。在任何情况下，Forg工具更为全面。

So there you go – I hope you continue to find it useful and welcome any feedback!

## 2.26 FHIR QA

https://fhirblog.com/2017/02/03/fhir-qa/#more-159702

FEBRUARY 3, 2017 [LEAVE A COMMENT](https://fhirblog.com/2017/02/03/fhir-qa/#respond)

We’re looking for volunteers to help out with QA (Quality Assurance) for the STU3 FHIR specification.

The QA process commences on Monday Feb 27, and finishes on a fortnight later on Sunday March 13. What we do is to render the part of the FHIR spec that is for QA into a single MSWord document, and each person is assigned a section to review. The document is altered with change tracking enabled, and after the review period is over the document will be given to the appropriate committees and changes applied.

Note that the process will focus on non-substantive issues such as readability, formatting, grammar, spelling and graphical layout.  The only substantive issues that will be considered are those that are clearly typos (e.g. incorrect spellings of element names).  That said, QA participants are welcome to raise change requests through the usual channels as they go as initial feedback for consideration as we start work on FHIR Release 4.

If there’s a particular part of the spec that you’re interested in, then let me know and I’ll do my best to assign that part to you, but it may not be possible if we want complete coverage (Or course there’s nothing stopping you reviewing more than your allocated part!)

The more hands the better, so if you can help out – or know of anyone else who may be able to, then drop me a line at “david dot hay25 at gmail.com” (or get them to) and I’ll add them to the list.

## 2.27 [Move to New Zealand!](https://fhirblog.com/2017/02/03/move-to-new-zealand/)

FEBRUARY 3, 2017 [6 COMMENTS](https://fhirblog.com/2017/02/03/move-to-new-zealand/#comments)

If you’re thinking of moving to New Zealand (and, of course, everyone wants to), then have I got a deal for you!

<https://www.bayleys.co.nz/Listing/Auckland/Auckland/Mt-Eden/1630149>

(Sorry, blatant misuse of a FHIR blog!)

## 2.28 Videos on clinFHIR Scenario Builder and Logical Modelers

https://fhirblog.com/2017/01/19/webinars-on-clinfhir-scenario-builder-and-logical-modelers/#more-159641

JANUARY 19, 2017 [5 COMMENTS](https://fhirblog.com/2017/01/19/webinars-on-clinfhir-scenario-builder-and-logical-modelers/#comments)

So in a fit of enthusiasm I offered to do a short demo of [clinFHIR](http://clinfhir.com/) at the [CATonFHIR](http://interopen.org/events#catonfhir) event in the UK in a couple of days. As it turns out, the reality of the timing is that this would be at 2am my time (New Zealand). Of course I’d be happy to do this to support FHIR (and have done it before), but when the organizer [Philip Scott](https://www.linkedin.com/in/philip-scott-b4627211/) suggested I record a video that they could show instead of a live performance I jumped at the chance!

And having done one, a second wasn’t that hard…

So: a couple of videos:

The [first video](https://vimeo.com/199927777) (20 minutes) is a demo of how to use the new Scenario Builder (I [posted about it](https://fhirblog.com/2017/01/09/simple-builder-library-and-documents/) earlier). It’s improved a bit since that post as it’s being used for the Clinicians On FHIR event at the [HL7 Working Group Meeting](http://www.hl7.org/events/wgm012017/) that’s being held on San Antonio at the moment. It still needs work, but is coming along nicely.

The [second](https://vimeo.com/200105501) (which is 10 minutes) talks about how to use the Logical Modeler to create models derived from the core Resource types in the spec that can then be used in the Scenario Builder. The idea is to have something that allows you to quickly create the model and then ‘try it out’ in the Scenario Builder without having to first create a profile (which always takes a bit of time).

I’ve got this idea that non-technical people (especially Clinicians) could use the combination of the 2 tools to rapidly document the requirements for a specific scenario, and then more expert users could generate the FHIR conformance artifacts required – profiles, extension definitions, Implementation Guides etc.

In fact, with a bit of planning, there’s no reason why these couldn’t be generated automatically – at least for the simple scenarios…

Anyway, I hope you find the videos helpful.

## 2.29 Extending a required ValueSet Binding 扩展所需的值集绑定

https://fhirblog.com/2017/01/10/extending-a-required-valueset-binding/#more-159582

JANUARY 10, 2017 [1 COMMENT](https://fhirblog.com/2017/01/10/extending-a-required-valueset-binding/#comments)

One of the issues we come across quite a lot at [Orion Health](https://orionhealth.com/) is where we are creating a read FHIR interface to an existing data service with elements in the FHIR resource that are coded and have a ‘[required](http://hl7.org/fhir/terminologies.html#required)’ binding to a [ValueSet](http://hl7.org/fhir/valueset.html) (which means that you must only use one of the values in the ValueSet) – and the data we are mapping from has a different value. [AllergyIntolerance](http://hl7.org/fhir/allergyintolerance.html) is a particular culprit in this regard as it has a number of required bindings, but it has come up most recently in [MedicationStatement](http://hl7.org/fhir/medicationstatement.html).[status](http://hl7.org/fhir/medicationstatement-definitions.html#MedicationStatement.status) where we have values that are different to the ones in STU2 – the FHIR version we are using.

Now, the solution is actually quite straight forward. In the resource instance you choose the value from the ValueSet that most closely matches the missing one, and then create an extension on that element (not at the resource root) that contains the actual value you want to represent.

现在，解决方案其实很简单。在资源实例中，可以从ValueSet选择与缺失值最为匹配的值。然后在该元素上（而不是资源根上）创建一个包含要表示实际值的扩展。

For example the following snippet shows how we want to represent ‘stopped’, and the closest match is ‘completed’

{

“resourceType”:”MedicationStatement”,

…

"status": "completed",

"\_status": {

"extension" : [

   {

     "url": "http://fhir.hl7.org.nz/dstu2/StructureDefinition/ms-status",

     "valueString": "stopped"

   }

]

}

…

}

(Note that in JSON the extension is the element name preceded by an underscore – see [the spec](http://hl7.org/fhir/json.html#primitive) for more details)注意在JSON中，扩展名是前缀为下划线的元素名称。

In this way, a consumer that doesn’t understand our extension can still safely process the resource, while one that does can potentially do specific actions.

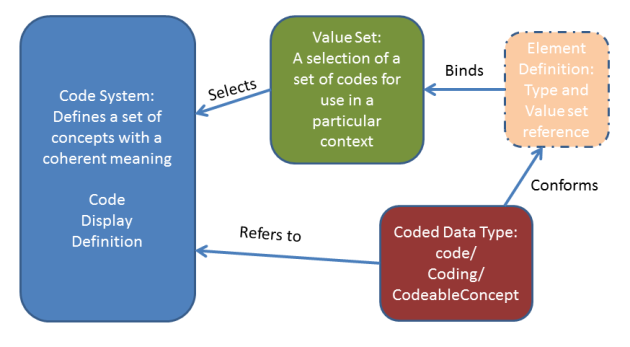
So now we need to create the Extension Definition (a [StructureDefinition](http://hl7.org/fhir/structuredefinition.html) resource of course) – and it will be a coded value, so we’re also going to need a ValueSet with the set of possible codes.

所以我们需要创建扩展定义（StructureDefinition资源），而且是一个编码值。所以我们也需要一组可能的编码值集。

To figure out how we do that, we need to take a short digression into terminology.

为了清楚如何做到这一点，需要对术语进行简要的研究。

The [following diagram](http://build.fhir.org/terminology-module.html) is taken from the most recent version of the spec (at the time or writing), and shows the relationship between the core terminology resources:



Looking at these resources:

* The Code System is ‘master list’ of codes (or concepts) that we can use – for example SNOMED or LOINC or one of the HL7 defined ones.
* The ValueSet is the sub-set of those codes from that CodeSystem for our particular context of use (in this case the MedicationStatement resource)
* The Element Definition will be part of our Extension Definition and will ‘bind’ the element to the ValueSet.

这些资源：

Codesystem：是我们可以使用代码（或概念）的主列表。例如：SNOMED或LOINC或者其中一个HL7定义的代码。

ValueSet：是来自codesystem代码的子集，用于特定的上下文（本例中为MedicationStatement资源）

ElementDefinition：是扩展定义的一部分，将元素绑定到ValueSet中。

(It’s actually a wee bit more complex than that as a single ValueSet can reference content from more than one Code System – a minor detail…)一个ValueSet可以引用来自多个代码系统的内容

So what we need to do for our Extension Definition is to first create the ValueSet, and then create the Extension Definition which will refer to that ValueSet. But wait – where will the ValueSet get its values from? In our case we are defining values that aren’t in an existing Code System.

我们首先需要做的扩展定义就是创建ValueSet，然后创建一个引用ValueSet的ExtensionDefinition。但是应该从哪里获取ValueSet的值呢？在例子中，定义了不在现有代码系统中的值。

Unfortunately the answer is slightly different in STU2 and STU3.

* In STU2 we define the codes directly in the ValueSet resource (using a unique url that we create as the ‘system’). 在STU2中，我们直接在ValueSet资源中定义代码（使用我们创建的唯一url作为“系统”）
* In STU3 we first create a [CodeSystem](http://build.fhir.org/codesystem.html) resource that contains the codes we want, and then refer to that CodeSystem from the ValueSet. 在STU3中，我们首先创建一个包含我们想要的代码的[CodeSystem](http://build.fhir.org/codesystem.html)资源，然后从ValueSet中引用该CodeSystem

Why was it changed? Well, it allows for better re-use of the CodeSystem we create – if we need to use it again in a different ValueSet, then we don’t need to duplicate all the codes, which is always a hassle. (And incidentally, this is one of the big advantages of the FHIR approach – we’ll get a much better standard at the end, even if it is extra work for the early adopters).

**So let’s look at STU-2 first.**

The ValueSet will look something like this:

{

"resourceType": "ValueSet",

"url": "http://hl7.org/fhir/ValueSet/ms-code ",

"name": "Extra values for MedicationStatement.code",

"codeSystem": {

   "system": "http://fhir.hl7.org.nz/CodeSystem/ms-code",

   "concept": [

     {

       "code": "stopped",

       "display": "Stopped",

       "definition": "Indicates that the medication was stopped before it was completely given"

     }

   ]

}

}

Notes:

* The url property is the ‘cannonical url’ that is globally unique and identifies this particular ValueSet.
* The codeSystem element contains the new codes/concepts
* The codeSystem.system element is another canonical url that will identify the codesystem (rather than the ValueSet)
* concept is an array of the actual code/concepts that are in this ValueSet.
* The example is not complete – only pertinent fields have been included, and there’s only a single concept.

注意：url属性是全局唯一且指示具体值集的“cannonical url”。

codesystem元素包含新的codes或者concept

codesystem.system元素是指示codesystem的“canonical url”（而不是值集的url）

concept是在值集中的实际code或concepts的数组。

这个例子是不完整的。

And the Extension Definition will have a binding to this ValueSet using the [binding.valueSetUri](http://hl7.org/fhir/elementdefinition-definitions.html#ElementDefinition.binding.valueSet_x_) field (so it doesn’t matter where the ValueSet is actually stored). If we do know the actual server where the ValueSet is stored, and are confident that it will be available forever then we could use the binding.valueSetReference field, but the uri seems safer to me.

如果我们知道实际的服务器存储了ValueSet，并确信它将永远可用，那么我们可以使用binding.valueSetReference字段，但是uri对我来说更安全。

**Now let’s discuss the situation in STU-3**

Overall the principle is exactly the same, the only difference is that the definition of the code system has been pulled out to its [own resource](http://build.fhir.org/codesystem.html), and the ValueSet then refers to that resource rather than defining it inline. This is exactly how we refer to other code systems like SNOMED & LOINC, so much more consistent than the STU2 way.

总体来说，原则是完全相同的，唯一的区别就是codesystem从ValueSet中单独出来。

Here what the CodeSystem would look like:

{

"resourceType": "CodeSystem",

"url": "http://fhir.hl7.org.nz/CodeSystem/ms-code",

"concept": [

 {

       "code": "stopped",

       "display": "Stopped",

       "definition": "Indicates that the medication was stopped before it was completely given"

   }

]

}

and here’s the ValueSet:

{

"resourceType": "ValueSet",

"url": "http://hl7.org/fhir/ValueSet/example-inline",

"name": "Extra values for MedicationStatement.code",

    "compose": {

       "include": [

           {

               "system": "http://fhir.hl7.org.nz/CodeSystem/ms-code "

           }

       ]

   }

}

Notes:

* In the ValueSet, the compose.include element has a ‘system’ value only – meaning that the entire CodeSystem with the url of <http://fhir.hl7.org.nz/codesystems/ms-code> is included. If we wanted to, we could specify a subset of those code/concepts – just as we normally do.
* The examples are incomplete.

注意：在ValueSet中，compose.include元素仅仅具有“system”值，这意味着包含“ <http://fhir.hl7.org.nz/codesystems/ms-code>”的url的整个codesystem都包含在内。如果我们想要，则可以指定这些编码的一个子集。

这个例子是不完整的

So there you go. Easy as…

## 2.30 Scenario Builder: Library and Documents

https://fhirblog.com/2017/01/09/simple-builder-library-and-documents/#more-159543

JANUARY 9, 2017 [1 COMMENT](https://fhirblog.com/2017/01/09/simple-builder-library-and-documents/#comments)

In the [last post](https://fhirblog.com/2017/01/09/building-a-set-of-resources-in-fhir/) we talked about the new builder component that was developed to help people (especially clinicians) develop sets of resources to represent clinical scenarios. There are a couple of features we didn’t have time to discuss then – the Library and Documents – so let’s talk about them now.

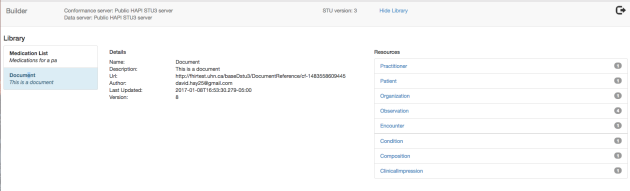
When you develop a set of resources, they are saved in a bundle on your local computer (actually in the browser storage). The problem with this approach is that you have to use the same browser to edit them – and you cannot share them with others. To get around this, we have the concept of the Library of sets.

当开发一组资源的时候，将保存在本地计算机的bundle中。这种方法的问题是必须使用相同的浏览器来编辑他们，不能与他人共享。

The idea is that you create and edit sets in your local browser, and periodically ‘publish’ them to a FHIR server (the library) from where they can be downloaded on to any browser and edited. This also means that others can view your work – it would be great to have a whole collection of ‘sample’ sets for people to download and examine – hence the idea of a library.

这个想法是在本地浏览器中创建和编辑sets，并定期将它们发布到FHIR服务器中，从那里将它们下载到任何浏览器并进行编辑。

To access the Library, click on the ‘View Library’ link in the navbar – you’ll see a display similar to this: 访问library，可以单击导航栏中“View Library”链接



To the left is a list of sets in the library – regardless of who published it. Selecting a set shows a summary of the set (name, description, author, resources in the set etc.) To download a set, click the download icon (a cloud with an arrow) and a copy will be downloaded to your local machine where it can be viewed and edited.

左侧是library中的列表，无论谁发布了它。选择一个集合将显示集合的摘要（名称，描述，作者，集合中的资源等）要下载集合，请单击下载图标（带箭头的云），并将副本下载到本地计算机，它可以被查看和编辑。

Technically, the set is stored as a bundle in the content property of a [DocumentReference](http://hl7.org/fhir/documentreference.html) resource on the Data Server. (Right now the contents are stored inside the DocumentReference itself – later on we’ll likely move the content to a [Binary](http://hl7.org/fhir/binary.html) resource and just reference them from the DocumentReference, as the current approach won’t scale well).

技术上讲，该set作为bundle存储在data server上的DocumentReference资源的content属性中（现在内容存储在DocumentReference本身中，稍后将内容移动到binary资源中，并且从DocumentReference中引用它们）

To add a set to the library, there’s a link in the navbar when a set is being edited that will copy the current set up to the library (creating a new version if there is one already there).

为了将set添加到library中，当编辑set的时候将当前set复制到库中，导航栏有一个链接。

Right now, anyone can access and update the library – later on we’ll add a bit more security – and also tidy up the [versioning](http://hl7.org/fhir/http.html#versioning) and [resource contention](http://hl7.org/fhir/http.html#concurrency) capabilities.

现在，任何人都可以访问和更新库。稍后增加一些安全性，并且还可以整理版本和资源内容的功能。

So now let’s talk about [documents](https://fhirblog.com/?s=documents).

In FHIR, a document is just a bundle of resources with a special resource – the [Composition](http://hl7.org/fhir/composition.html) resource – that has the information ‘about’ the document (analogous to the CDA header). There are a number of [other rules](http://hl7.org/fhir/documents.html) about what’s in a document, how it should be rendered and so forth but fundamentally it’s just a [Bundle](http://hl7.org/fhir/bundle.html) of resources.

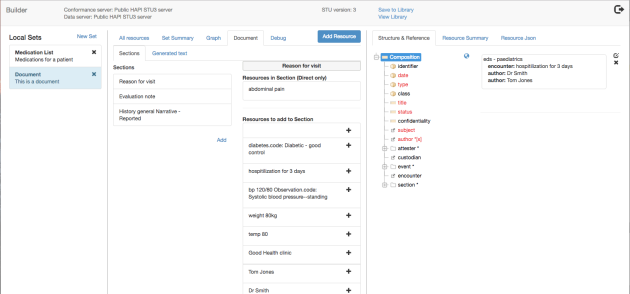
在FHIR中，Document只是一个具有特殊资源的资源梱束——Composition资源，具有有关文档的信息（类似CDA头文件），还有一些其他规则的文档，但是根本上指示一个bundle资源。

I’ve done a number of posts on documents – [here’s a summary](https://fhirblog.com/2013/12/13/fhir-meet-cda/), though a bit out of date as it uses an Atom feed as the container rather than a Bundle, but the fundamentals are still correct. Refer to [the spec](http://build.fhir.org/documents.html) for the most up to date details.

已经做了文档的帖子，这是一个总结。因为它使用Atom feed作为容器而不是一个Bundle，但基本原理仍然是正确的。请参阅[规格](http://build.fhir.org/documents.html)了解最新的详细信息。

In the clinFHIR simple builder (I really need a better name ! ) adding a Composition resource to a set will automatically invoke the document features – represented as a new tab in the middle pane. Here’s a screen dump (the Composition resource has been selected which is why it’s detailed in the right hand pane):

在clinFHIR简单的构建器（我真的需要一个更好的名称！）将组合资源添加到一个集合将自动调用文档功能 - 在中间窗格中表示为一个新的选项卡。这是一个屏幕转储（组合资源已被选中，这是为什么在右侧窗格中详细说明）：



To build a document, you add sections to the document (under the hood this will add Composition.section elements) and then add resources to the section (references from Composition.section to the resource).

为了建立一个文档，可以向文档中添加section。

To add a new section, click the Add button below the section list, then select the section code from the drop down list (the list contents come from the [ValueSet defined in the spec](http://hl7.org/fhir/composition-definitions.html#Composition.type) of course).

Then, select the section in the middle pane and you’ll see 2 lists to the right of the section:

* Resources that are in the section (directly referenced by the section)
* Resources that could be added to the section – ie the others.

然后，选择中间窗格中的部分，您将看到该部分右侧的2个列表：部分中的资源（由本部分直接引用）；可以添加到该部分的资源 - 即其他资源。

To add a resource to the section, click the plus (+) icon to the right of the resource display in the lower list (mousing over the resource will show the id and the complete text).

You can also re-order or remove the resources in the section using the arrow icons that appear when you mouse over the resource in the upper list. Similarly, you can re-order the sections themselves.

要向该部分添加资源，请单击下表中资源显示右侧的加号（+）图标（鼠标悬停在资源上将显示ID和完整文本）。

您也可以使用上方列表中的资源上方显示的箭头图标重新排序或删除该部分中的资源。同样，您可以重新排序这些部分。

The graph view will be updated according the resources referenced by the section.entry elements.

将根据section.entry元素引用的资源进行更新。

The reason why you might want to re-order sections and resources is that the tool will automatically generate the document text for you – and will show that text in the ‘Generated text’ tab.

想要重新排序章节和资源的原因是该工具将为您自动生成文档文本，并将在“生成的文本”选项卡中显示该文本。

There’s a specific set of rules for [rendering a FHIR document](http://hl7.org/fhir/documents.html#presentation):

1. First the Composition narrative
2. Next the subject narrative
3. Finally the section.text narratives

有一套特定的规则来呈现FHIR文档：首先是Composition叙述性文本；然后是subject叙述性文本；最后是section.text叙述性文本。

The [narrative for a resource](http://hl7.org/fhir/narrative.html#Narrative) is the textual representation of the resource. All resources can have a narrative (it’s the text element in the resource), and it’s up to the resource creator to decide what text should be in there. In this tool, the narrative for any resource is a combination of text directly entered by the user, and a textual representation of any structured data entered (which can be manually edited if necessary).

资源的Narrative是资源的文本表示。所有资源都可以由一个叙述，由资源创建者决定应该在哪个文本中。在这个工具中，任何资源的叙述都是由用户直接输入的文本和输入的任何结构化数据的文本表示的组合（可以在必要时手动编辑）。

When generating text, the following rules are used:

* The composition narrative is rendered
* The subject narrative is rendered (generally this is a Patient, but any resource can be a subject)
* The narrative for each section  is created by iterating through each of the sections in order, and then each of the referenced resources (section.entry) in order.
  + If the referenced resource is a [List](http://hl7.org/fhir/list.html) resource, then the narrative is the narrative of the List resource, plus the narrative of all the resources that the List references
  + If the referenced resource is not a List, then the narrative is rendered

生成文本时，使用以下规则：

构成叙事呈现

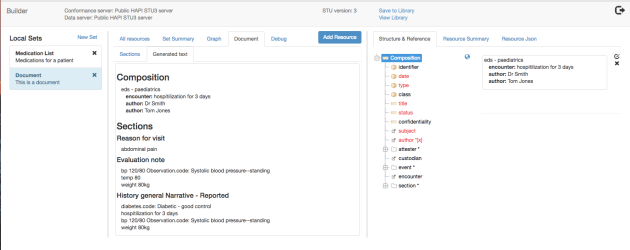
subject叙述被渲染（通常这是一个病人，但任何资源都可以是一个主题）

每个部分的叙述是通过依次循环遍历每个部分，然后依次引用每个引用的资源（section.entry）来创建的。

如果引用的资源是List资源，则叙述是List资源的叙述，加上List引用的所有资源的叙述

如果引用的资源不是列表，则会呈现叙述

And here’s a rendered document:



So that’s the new builder tool. Feedback welcomed!

## 2.31 Building a set of resources in FHIR

https://fhirblog.com/2017/01/09/building-a-set-of-resources-in-fhir/#more-159489

JANUARY 9, 2017 [4 COMMENTS](https://fhirblog.com/2017/01/09/building-a-set-of-resources-in-fhir/#comments)

One of the primary goals for [clinFHIR](http://clinfhir.com/) is to help people who are new to the standard understand how it works – and increasingly these are clinicians whose interest is less in the technology and more about how FHIR can be used to represent the clinical information they wish to exchange.

clinFHIR的主要目标之一就是帮助那些刚接触fhir标准的人来了解其工作原理。

While the current app does allow this, it has been aimed more at the people actually developing the resources than the casual user, and so can be time consuming to develop sets of resource instances that represent real world scenarios.

The component described by this post (called a simple builder – though a better name is needed!) is intended to allow someone completely new to FHIR to build sets of resources that represent clinical scenarios, to help them understand how the resources can be linked together – rather like [Lego](https://en.wikipedia.org/wiki/Lego) is used to build a complete model.

此帖子中描述的组件（0称之为简单的构建器）旨在让一个全新的FHIR创建代表临床场景的资源，帮助他们了解如何将资源链接在一起，就像乐高一样建立一个完整的模型。

Before we get into the details – there are a number of caveats with the current version.

* The resources that you create are not saved as individual resources on the server (yet). Rather they are packaged in a bundle and saved as a group. This means that the only references you can create are to resources within the set you are working on, and you cannot query for them directly off the server.
* There are limits to the structured details that can be entered against each resource. Because it’s intended as a training tool, the focus has been on the ‘bigger picture’ of creating the ‘web’ of resources, rather than the individual details of the resource, so in many cases you just enter the text that describes the resource, rather than the coded elements.
* The only resource types you can select are the ones defined in the spec – profiled resource types and logical models are not (yet) supported.

在详细介绍之前，有些注意事项：

创建的资源不会作为单个资源保存在服务器中，而是bundle在一起作为一组来保存。这意味着可以创建的唯一参考是正在处理的集合中的资源，不能直接从服务器查询他们。

对于每个资源可以输入的结构化细节有限制。这是一个培训工具，故重点放在创建资源的“网络”的更大的图像上，而不是资源的个别细节中。在很多情况下，只需要输入描述资源的文本，而不是编码元素。

可以选择的唯一资源类型是在规范资源类型中定义的资源类型，而不是支持逻辑模型。

The intention (if there is interest) is to remove the limitations – it’d be great to store the resources individually on the server for example, and support profiles, logical models and resource updating. It just takes time  ).

Let’s work with an example – representing a patient’s medication list, which is something [we’ve talked about](https://fhirblog.com/?s=medication%20list) before. The resources that we’ll need include:

* The [Patient](http://hl7.org/fhir/patient.html) who is the subject of the list
* The [Practitioner](http://hl7.org/fhir/practitioner.html) who created the list (it could also be the Patient in real life of course)
* A [List](http://hl7.org/fhir/list.html) resource to represent the list itself
* Multiple [MedicationStatement](http://hl7.org/fhir/medicationstatement.html) resources for the individual medications.

例如：表示患者的药物清单，需要包括资源：

Patient：药物清单的对象

Practitioner：创建药物清单的医生

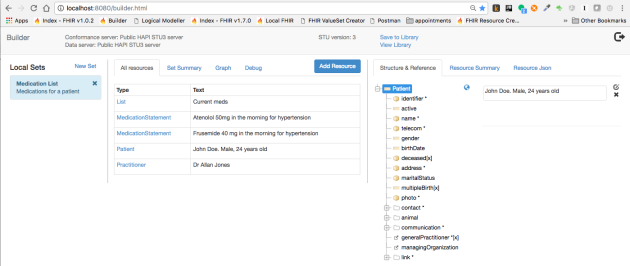
List：表示药物清单的资源

MedicationStatement：个别药物的药物说明。

To start the builder, load [clinFHIR](http://clinfhir.com/) and log in (You can easily create an account based on your email address). Select the servers to use (I generally use the HAPI STU-3 servers for data & Conformance and Grahames server for terminology). Click the ‘gear’ icon to the upper right and select the builder from the menu that appears. The builder will open in a new page.

要启动构建器，加载clinFHIR并登陆。选择要使用的服务器（通常使用通常使用HAPI STU-3服务器的数据和一致性和Grahames服务器的术语）。点击右上角的“齿轮”图标，然后从出现的菜单中选择构建器。

Here’s what the front screen looks like:



At the top of the screen is a navbar with the name of the current FHIR servers in use, some Library commands (that we’ll talk about later) and an icon for the user to the upper right – it is display only at this stage.

There are 3 main panes:3个主窗格

* To the left is the list of sets that you are working with (you can have multiple sets at any one time). 左侧是正在使用的set列表
* In the middle are a number of views of the currently selected set. These are:
  + A list of all the resources in the set
  + A summary of the set (url, name, description, author and so forth)
  + A ‘graph’ view showing the references between resources
  + A ‘document’ view that appears when you are creating a FHIR document (more on this later).

中间是当前选定set包含的相关内容的视图：

集合中所有的资源列表；集合中的摘要（url、name、description、author、等）

集合中资源之间引用的图形；创建FHIR文档时候出现的“文档”视图

* To the right are the details of the currently selected resource – whether selected in the list of graph view. It also has 3 tabs:
  + Structure and Reference where you can add specific elements to the resource including (especially!) references to other resources
  + Resource Summary which shows the current references and allows you to remove them
  + A JSON representation of the resource for those interested in such things.

右侧是当前所选资源的详细信息，无论是否在graph view 列表中选择，需要由三个选项卡：

结构和引用。可以向资源中添加特定的元素，包括对其他资源的引用；资源摘要，显示当前的引用，并允许删除他们；

Create a new set by clicking on the ‘New Set’ at the top of the left pane. You’ll be prompted for a name for the set, and a description of its purpose. After saving, the new set will appear in the left hand list. (btw – in testing, I noted that sometimes it doesn’t appear but does after the page is refreshed. I’ll fix that eventually…)

单击左侧的“New Set”来新建一个集合。系统将提示输入该集合的名字，并说明其用途，保存后，新建的集合将显示在左侧的列表中。

To add a new resource, click on the ‘Add Resource’ button at the top of the middle pane. The right pane then shows a drop-down of all the resource types defined in FHIR. Select the one you want (say, Patient) and then enter a textual description of the resource (name, date of birth, gender etc.) into the text box that appears.

添加新的资源，点击中间窗格顶部的“add Resource”按钮。然后右侧窗格显示FHIR汇总定义的所有资源类型的下拉列表，选择想要的资源，然后在出现的文本框中输入资源的文本描述（例如：姓名、出生日期、性别等）

Finally, click the ‘Add’ button to add the resource to the set. After a brief delay (while the app retrieves the resource definition from the Conformance server) the details of that resource type will be displayed, including a tree view of the resource structure.

最后，点击“Add”按钮，将资源添加到集合中。短暂的延时之后，将显示该资源的详细信息。包括资源结构的树状视图。

Repeat the process for all the resources identified above. You should end up with a display similar to the screen shot above.重复上面的过程。

To ‘wire up’ the references between the resources, you select the resource that the reference is from, then the source element within that resource, and finally the target resource. You’ll notice that when you select an element that is a reference type, the app will show a list of all the types of resource that that element can reference, plus a list of all the resources in the set that match any of those types.

要“连接”资源之间的引用，可以选择引用来自的资源，然后选择该资源中的源元素，最后选择目标资源。注意; 当您选择一个引用类型的元素时，应用程序将显示该元素可以引用的所有资源类型的列表，以及该集合中与任何这些类型匹配的所有资源的列表。

For example, select the List resource, then the subject element. You’ll see to the right of the tree:

* The textual description of the resource. To the right of the text field are icons that will allow you to edit the resource – or to remove it from the set.
* A list of the potential resource types (Patient, Group, Device etc.). Each type has a ‘globe’ icon that will load the specification for that type in a separate page.
* A list of the potential resources currently in the set that can be referenced from that element. Clicking the ‘plus’ symbol to the right of the resource actually creates the reference.
* The description of the purpose of that element.

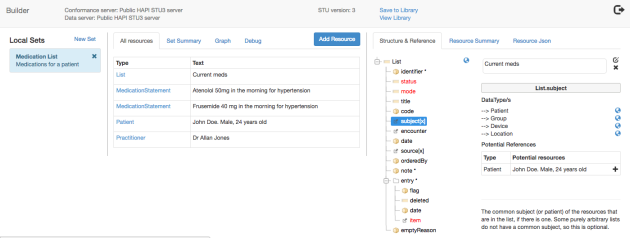
例如：选择List资源，然后选择subject元素，则树的右侧将显示：

资源的文字说明。允许编辑资源的图标或者从界河中删除图标

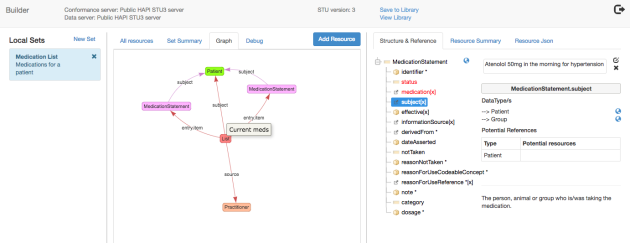
潜在资源类型的列表。每种类型都有一个“globe”图标，将在单独的页面中加载该类型的规范。

集合中当前可以从该元素引用的潜在资源列表。单击资源右侧的“+”实际创建引用

Here’s a screen shot:



And that’s all there is to it! If you select the graph view in the middle pane, you’ll see a line drawn between the referenced resources (from source to target). The graph updates in real time as you create the references, which is kind of fun! Here’s what the Medication list looks like after all the references have been established: 如果在中间窗格中选择图形视图，您将看到在引用的资源（从源到目标）之间绘制的一条线。在创建引用时，图表会实时更新，



To remove a resource, select the Resource Summary tab in the right pane, which will show all the resources that are linked to/from this resource. There’s an ‘X’ icon against each reference that will remove that reference.

要删除资源，请选择右侧窗格中的“资源摘要”选项卡，该选项卡将显示链接到此资源的所有资源。每个引用都有一个“X”图标，将删除该引用。

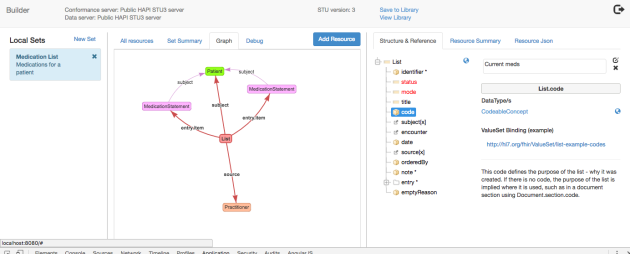
If you want to enter the data for an actual element, select the element in the Structure pane. This will show the data element types that can be used for that element, and clicking on the datatype will display a dialog where the value can be entered. This is much the same as in the clinFHIR resource builder, though in a modal dialog. Note that not all datatypes are currently supported.

如果要输入实际元素的数据，请在“结构”窗格中选择元素。这将显示可用于该元素的数据元素类型，单击数据类型将显示可以输入值的对话框。尽管在一个模式对话框中，这与临床资源构建器中的相同。请注意，当前不支持所有数据类型。

If the datatype is coded and there is a ValueSet binding (which should always be there) then you can browse the ValueSet – again, much like clinFHIR.

如果数据类型被编码，并且有一个ValueSet绑定（应该永远在那里），那么你可以再次浏览ValueSet，就像clinFHIR一样。

Here’s a sample screen shot:



And a reminder of the caveat around entering structured data – this doesn’t work all that well for more complex resource types (like care plan) – with deeply nested structures. It really only works properly for elements directly off the root of the resource. This is a work in progress.

提醒您注意输入结构化数据 - 这对于更复杂的资源类型（如护理计划）而言并不适用 - 具有深层次的结构。它只能直接从资源根的元素正常工作

The sets you create (and you can have as many as you want) are saved locally (in the browser) as you make updates so there’s no explicit ‘save’ option needed. You switch between different sets simply by selecting the set in the left pane.

To save the sets you create to the FHIR server (actually the Data Server) you use the ‘Library’ function. We’ll talk about that – and creating proper FHIR Documents – in the next post.

创建的集合（您可以拥有所需的数量）在本地（在浏览器中）保存，因为您进行更新，因此不需要明确的“保存”选项。您可以通过选择左侧窗格中的集合来切换不同的集合。

要将您创建的集合保存到FHIR服务器（实际上是数据服务器），请使用“库”功能。我们会在下一篇文章中讨论这一点，并创建适当的FHIR文档。

## 2.32 [Presentations from Amsterdam devdays](https://fhirblog.com/2016/12/19/presentations-from-amsterdam-devdays/)

DECEMBER 19, 2016 [LEAVE A COMMENT](https://fhirblog.com/2016/12/19/presentations-from-amsterdam-devdays/#respond)

<https://vimeopro.com/user12740828/hl7-fhir-developer-days-2016-amsterdam>

enjoy!

## 2.33 Challenges with Lists

https://fhirblog.com/2016/12/15/challenges-with-lists/#more-159459

DECEMBER 15, 2016 [6 COMMENTS](https://fhirblog.com/2016/12/15/challenges-with-lists/#comments)

So one of the challenges we’re facing at work at the moment is Lists – more specifically ‘current’ lists – eg the medications a patient is taking right now, their problem list and set of allergies.

面临的挑战之一就是list：更具体的说就是“当前”列表，例如：患者正在服用的药物列表，症状列表和过敏集。

A bit of background.

Our application is really a combination of HIE (Health Information Exchange) and EMR (Electronic Medical Record). We have a Clinical Data Repository (CDR) that collects information from many different sources, and them maps them to a set of canonical models. The main source of information is v2 messages, from which we extract ‘Event’ based information like encounters, problems, allergies & medications as well as Document based – CDA & CCDA being the most common (and yes, combining data across paradigms carries it’s own challenges!)

真正的应用是HIE和EMR的结合。临床数据存储库（CDR）可以从不同的来源收集信息，并将它们映射到一组规范模型中。信息的主要来源是V2消息，从中来提取“Event”信息，例如：就诊、问题、过敏和药物以及基于CDA的文档。

In addition, we have our own applications – such as Portals, Reconciliation/Curation and Case Management systems that also contribute data to the repositories.

此外，还有自己的应用程序，例如：门户、案例管理系统等，也可以向存储库提供数据

One of our key strategies is to make all this data available to consumers via FHIR based API’s, so we need to think about just what API’s we’re going to support. We’ll consider this in the context of medications, though the same applies to problems and allergies

主要的策略就是通过基于FHIR的API将所有这些数据提供给消费者，因此我们需要考虑支持的API。会在药物的上下文中考虑这一点，同样适用于问题和过敏。

Making event based data (eg prescriptions or dispensing) is straightforward – someone tells us about a dispensed medication, we store it in the CDR, and them make it available via a REST query to the [server]/MedicationDispense endpoint. I’ve glossed over the security/privacy issues of course – as well as dealing with duplicated data – but conceptually this is quite straightforward.

基于事件的数据是直接的。例如：有关药物的分发，将其存储在CDR中，并通过REST查询将其提供给[server]/MedicationDispense的端点。当然，掩盖安全或隐私问题以及处理重复数据。

But it’s not so simple when we want to supply an endpoint for the current list of medications.

但是对目前的药物清单提供端点并不是很简单的事情。

You’d think it would be reasonably straight forward: each event has a period over which it applies (for prescription it’s how long to take the medication for, for dispense you can work that out from the amount given), from this you can work out what meds are ‘active’ at a point in time and there’s the list! (Again, glossing over a few details, but you get the point).

每个时间都有一个适用的时期（对于处方是多长时间服用药物，为了免除你可以从给定的金额工作），从这可以工作在某个时间点出现什么药物是“活跃的”，并有列表！（再次，掩饰了一些细节，但你得到了点）。

Of course, such a list will always be approximate – we may not have all the events for example, or some may not have all the data we require in those events – so it’s much better for a human to be able to confirm that the list is correct – a ‘reconciled’ or ‘curated’ list. And, indeed, we have functionality that does that, but there are a few wrinkles.

当然，这样的列表总是大致的 - 我们可能没有所有的事件，或者有些可能没有我们在这些事件中需要的所有数据 - 所以人类能够更好地确认列表是正确的

For a start, we can receive medication lists from external systems as well as events – that’s what the medication data in CDA documents is – what do we do if the lists from different sources for the same patient is different? (surely not, you say …  )

And what happens if we get or create a list, and then subsequently receive an event (e.g. dispense) for a medication not on that list? What is the ‘current’ list then?

So we’ve come to the conclusion that there is no single way that we can expose the ‘current’ list – instead we need a number of endpoints, each slightly different in what it supplies.

我们没有一种可以公开“当前”列表的方法，而是需要一些端点，每个端点在它所提供的内容上略有不同。

a) Firstly a simple RESTful query against each resource type – in this case MedicationOrder (prescriptions), MedicationDispense (dispenses), MedicationAdministration (actual taking of the medication) and MedicationStatement (from the CDA’s and other lists we receive) that returns a simple bundle containing the resources. We’ll provide various searches to facilitate the query of course, and likely some ‘aggregate’ operation that supplies all medication related resources.

[server]/MedicationOrder?patient={patientId}

首先对每种资源类型进行简单的RESTful查询，这种情况下MedicationOrder (prescriptions), MedicationDispense (dispenses), MedicationAdministration和MedicationStatement（来自CDA和接收的其他列表）返回一个简单的包含资源的bundle。我们会提供各种搜索，以便查询，并可能提供所有药物相关资源的一些总体操作。

b) Next, queries against the [List](http://hl7.org/fhir/list.html) endpoint that will return the ‘actual’ lists we have received – eg from CDA or our own List creator. This will return a bundle containing List resources – each of which represents a single list – plus the medication resources to which the list refers (likely MedicationStatement). This will be something like:

[server]/List?subject={patientId}&code=http://loinc.org|57828-6

接下来，针对List端点查询将返回收到的“实际”列表。例如：来自CDA或自己的LIST创建者。这将返回一个包含List资源的bundle，每个资源都代表一个列表，加上该列表引用的药物资源（可能是MedicationStatement）

c) And finally an endpoint that represents our ‘best estimate’ for the current list. There will be internal business logic that assembles the list – for example it could take the most recent ‘curated’ list, remove meds whose activity period has expired and add any new medications we receive since the list. The details of this are to be determined! We’ll expose this list like this (it will only have MedicationStatement resources in it):

[server]/MedicationStatement?patient={patientId}&\_list=$current-medications

最后一个端点表示当前列表的“最佳估计”。将内部业务逻辑组合列表。例如：采用最近的“curated”列表，删除已过期的药物，并添加列表中收到的新药。确定这个细节，并公开列表。（仅仅是MedicationStatement资源）

Although this lets us proceed, it’s not the most elegant of solutions so I’m interested in any comments that you may have – positive or negative! I’m sure that we aren’t going to be the only organization facing these questions.

虽然可以继续下去，但是这并不是最好的解决方案。

Interestingly, the spec also [discusses this issue](http://build.fhir.org/list.html#query) and comes up with similar conclusions. [This page](http://build.fhir.org/lifecycle.html) also talks about lists. (both links are to the current build BTW so may change)

And finally, [there’s a track](http://wiki.hl7.org/index.php?title=201701_Medication_Track) at the [next connectathon](http://wiki.hl7.org/index.php?title=FHIR_Connectathon_14) exploring this topic. Unfortunately I can’t be at that one, but will be interesting to see what comes out of it…

## 2.34 Using workflow to track comments 使用工作流来跟踪注释

https://fhirblog.com/2016/12/04/using-workflow-to-track-comments/#more-159425

DECEMBER 4, 2016 [LEAVE A COMMENT](https://fhirblog.com/2016/12/04/using-workflow-to-track-comments/#respond)

So there’s a requirement that’s come out of the UK profiling work (among others) to be able to gather comments about logical models as part of getting feedback – particularly from Clinicians.

The overall process would go something like:

1. Gather the requirements for a particular purpose – such as representing a medication list.
2. Create the first draft of a Logical model that represents that requirement
3. Circulate the model amongst interested participants – clinicians, informed consumers etc and gather their feedback
4. Update the model based on that feedback
5. Repeat steps 3 & 4 until consensus (or close to it) has been achieved
6. Create the FHIR artifacts (Profiles, ValueSets, Extension Definitions etc.) that represent that model

整个过程为：

收集特定目的的要求，例如：表示一个药物列表

创建表示该要求的逻辑模型的初稿

将模型传递给感兴趣的参与者——临床医生、消费者并收集反馈意见

基于该反馈的更新模型

重复步骤3和4，知道达成共识

创建该模型的FHIR工件（Profile、ValueSet、ExtensionDefinition）

(This is a slightly abbreviated version of the process that I proposed earlier, and focuses on the modeling aspects of meeting the requirements.There are, of course, other issues to consider such as exchange paradigms, but for now let’s focus on the model.)

So from the perspective of changes required to clinFHIR we have:

* The ability to notify someone that there is a model that they can view and review.
* A way for them to record their comments – and view the comments of others
* A way for the editors to track who has commented – and who has yet to comment so that they know when it’s feasible to update the model based on feedback (step 4 above)

从临床观察所需的变化来看，我们：

通知某人可以查看和检查模型的能力

记录其意见

跟踪评论的方式

This sounds like a form of workflow to me, so I thought it was a good opportunity to experiment with the new FHIR workflow resources and see if I could do the whole thing (or most of it) using FHIR artifacts.

听起来是一种工作流，所以认为这是尝试新的FHIR工作流资源的好机会。

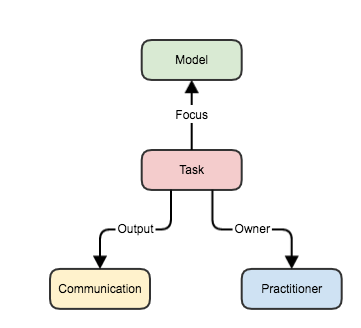
STU3 has proposed major changes to workflow (which you can read about [here](http://build.fhir.org/workflow-module.html)) – the most important (to my mind) being the definition of a ‘[Task’](http://build.fhir.org/task.html) resource that co-ordinates the various activities around completing the workflow.   This supports quite a number of different workflow patterns – including our one.

STU-3已经提出了工作流的改变，定义了Task资源，协调完成工作流的各种活动，支持很多不同的工作流模式。

After a bit of thought, I came up with the following pattern of use. When a model review is desired, a Task instance is created that has the Model as it’s ‘focus’ property, the Practitioner who is to perform the review as the owner, and a Communication resource that represents the contents of the review. (I did consider using an Observation resource – or embedding the review as the Task.note property, but Communication did seem the best candidate for now.)

当需要模型审查的时候，创建一个具有Task实例，将model作为task.focus的内容，Practitioner执行审核的从业者，以及表示检查内容的Communication资源。

Here’s what it looks like:



With this infrastructure in place, it becomes possible for the authors of a model to track who has made comments (and who has not) for any given model simply by retrieving all the tasks that are focused on the model, and then retrieving the communication resources that contain the comment. The task can either be created ‘up front’ by the model author, or a reviewer can create a task at the same time as they enter the comment.

这种基础设施。模型的作者可以通过检索所有关注model的task，然后检索Communication资源，来跟踪为给定的模型发表了评论的人。该task可以由模型作者创建，或者审阅者可以在输入注释的同时创建一个task

The details are all hidden behind the UI of course, so let’s see how this plays out in practice.

这些都是隐藏在界面之后。

Start [clinFHIR](http://clinfhir.com/), and select the servers (right now, you need to have the same server for data and conformance, so select the public HAPI-3 server for both). Select the Logical Modeler option from the gear menu to the upper right.

启动clinFHIR，选择服务器。从右上角的齿轮选项中选择逻辑建模。

In the modeler, create the first draft of the model (making sure you save it) and then click the ‘Share’ button to the upper right. This will create a url that is a ‘shortcut’ to the model Copy the Url from the dialog box.

在建模中，保存创建模型的第一稿（并保存它），然后单击右上角的“share”按钮，这将创建一个url。

Then send the url to anyone you want to perform a review. They can enter it into a browser and the modeler will be displayed with the model selected (and the selection pane hidden to reduce screen clutter).

然后将url发送给执行审查的任何人。可以将其输入到浏览器中，并且将使用所选模型显示建模者（并且隐藏选择窗格以减少屏幕杂乱）。

If they are logged in, then to the right of the screen is a tab panel labeled ‘Comments’. Selecting that tab will display a table of all the comments that have been entered so far. (If not logged in, then it’s easy to do so using the login icon – and they can also create a new user account if needed).

如果已经登录，则屏幕右侧是标签为“Comment”的面板。该选项卡显示目前为止的所有注释的表格。

To enter (or update) a comment, select the ‘Add Comment’ tab. There’s a button to click that will enable the comment screen (technically, this is where the task is created). After this a text box is displayed where the comment can be entered. This is actually a simple ‘markdown’ editor to allow more advanced users to create a formatted note. There are buttons in the display that will do the formatting, or they can use the text shortcuts:

* #(space)<text> for a heading (also ## <text>, ### <text> & so forth)
* \*(space)<text> for a bulleted item
* \*\*<text>\*\* for bold (note no spaces)

要输入评论，选择“Add comment”标签，点击按钮后将启动注释的显示屏。之后显示一个文本框，可以进行输入注释。其实是一个简单的“Markdown”编辑器，允许高级用户创建格式化的音符。显示屏中有可以进行格式化的按钮，以及文本快捷方式。

There’s a preview option so you can see what it looks like.预览选项，看到样子。

Click the Save link (bottom right of the box) to save.单击保存，

You may wonder why we set it up that anyone can add a comment, and that the tasks get created automatically on request rather than being pre-assigned. The reason is that we want to make it as simple as possible for people to contribute, and giving them a link that they can paste into a browser is really simple

You might also think that this is a bit overkill for such a simple process – and you’re right, but it’s always good to have an opportunity to learn new stuff – and it does pave the way for more complex workflow in the future, such as the editor creating the Tasks and the reviewer operating off a ‘todo’ list, which could extend to the other artifacts such as ValueSet contents.

So this is very much a first cut of the reviewing functionality. Enhancements could include:

* Right now anyone can edit the models. Preferably, only model editors should be able to make changes to avoid confusion – though it would be nice for a reviewer to be able to make a copy of the model so they can show the editors what they think the model should look like (I think this can be done using FHIR’s versioning ability – though need to think through how that will work)
* The display of comments is rather crude – being a simple table of content and email. This is unlikely to scale well with a lot of reviewers – or with verbose reviews! We also need to allow users to enter their name as well as the email so the display is friendlier.
* Only a single comment per model is supported. Possibly multiple comments – maybe related to a specific model element – will be required
* We need to support multiple iterations of comment. This could require task status’s to be updated (to indicate that the task is complete) and new ones to be created. The display of old comments also requires some thought.
* It may be desirable in some cases to limit the people that can provide feedback – in that case a page where the editor can select the people they wish to perform the review might be needed (though the people must have already registered with clinFHIR for this to work).

## 2.35 Basic Pregnancy Logical Model

https://fhirblog.com/2016/11/11/basic-pregnancy-logical-model/#more-159354

NOVEMBER 11, 2016 [5 COMMENTS](https://fhirblog.com/2016/11/11/basic-pregnancy-logical-model/#comments)

So there was an [interesting conversation](https://chat.fhir.org/#narrow/stream/implementers/topic/FHIR.20Resource.20for.20Pregnancy.20Episode) in the FHIR chat just recently about representing maternity information in FHIR. It was pointed out by [Lloyd](https://www.linkedin.com/in/lloyd-mckenzie-6b6681) that some elements that seem simple – like the LMP (Last Menstrual Period date) – are actually more complex than that, they are more properly [Observations](http://hl7.org/fhir/observation.html) made by someone, at some date, with a degree of certainty etc.

最近讨论在FHIR中表示产妇信息的资源，似乎很简单的一些元素，例如LMP（最后一次月经），实际上比较复杂，由某人在一定的阶段以及具有一定的确定性的适当的Observation。

So what this means is that a single logical model – like a summary of pregnancy – is going to wind up being represented in FHIR by a number of different resources, presumably ‘bound together’ in some way (eg  a bundle of some sort Document/Message – or just a [Composition](http://hl7.org/fhir/composition.html) with appropriate references) and described by an Implementation Guide.

考虑到一个逻辑模型（就像怀孕的总结一样）将由一组不同的FHIR资源来表示，大概以某种方式“束缚在一起”（例如：一些Document或Message或者只是具有适当引用的Composition），并由实施指南描述。

Given that I’m working on a logical modeler in clinFHIR, this seemed like something to address. Here’s where I’ve got to so far.

I started by getting the requirements for a basic logical model from my colleague [Peter Jordan](https://www.linkedin.com/in/peter-jordan-44146418) (now the [chair of HL7 New Zealand](http://[8/11/16%201:24:39%20pm%5d%20Dealla%20Smith:%20This%20is%20the%20copy%20that%20will%20go%20on%20the%20advert,%20good%20point%20about%20phones%20catching%20fire%20though%20%5b8/11/16%201:29:23%20pm%5d%20David%20Hay%20%F0%9F%94%A5:%20Those%20pictures%20worked%20great!%20Thanks...%20%5b8/11/16%201:29:41%20pm%5d%20David%20Hay%20%F0%9F%94%A5:%20btw%20are%20you%20in%20tomorrow?%20Wenesday%20you%E2%80%99re%20generally%20at%20home%20aren%E2%80%99t%20you?%20%5b8/11/16%201:30:19%20pm%5d%20Dealla%20Smith:%20I%27m%20coming%20in%20for%20the%20prep%20before%20the%20webinar%20%5b8/11/16%201:31:40%20pm%5d%20David%20Hay%20%F0%9F%94%A5:%20good%20oh%20-%20see%20you%20then%20(I%E2%80%99m%20at%20home%20-%20I%20have%20that%20clinfhir%20preso%20I%20showed%20you%20-%202%20hours%20from%206am!)%20%5b8/11/16%201:33:50%20pm%5d%20Dealla%20Smith:%20Sounds%20great!%20I%20won%27t%20stay%20for%20the%20actual%20presentation,%20but%20will%20get%20there%20before%20%5b9/11/16%2010:22:25%20am%5d%20Dealla%20Smith:%20High%20res%20version%20of%20todays%20presentation%20%5b9/11/16%2010:22:40%20am%5d%20Dealla%20Smith:%20BIG%20FINAL%20FHIR%20Webinar%20APAC_%204%20Nov_FINAL.pptx%20%5b9/11/16%2010:23:28%20am%5d%20Dealla%20Smith:%20FHIR_API_300x250_v2.gif%20%5b9/11/16%2010:23:53%20am%5d%20Dealla%20Smith:%20Could%20you%20please%20paste%20this%20into%20your%20browser%20and%20watch%20the%20animation,%20it%20is%20going%20to%20be%20used%20in%20APAC%20%5b9/11/16%2010:24:09%20am%5d%20Dealla%20Smith:%20Let%20me%20know%20what%20you%20think???%20%5b9/11/16%2010:28:48%20am%5d%20David%20Hay%20%F0%9F%94%A5:%20I%20guess%20the%20phone%20is%20coming%20to%20the%20FHIR!%20%5b9/11/16%2010:29:26%20am%5d%20David%20Hay%20%F0%9F%94%A5:%20but%20overall%20I%20like%20it...%20%5b9/11/16%2010:29:27%20am%5d%20David%20Hay%20%F0%9F%94%A5:%20but%20fiora%20should%20be%20consulted%20as%20well%E2%80%A6%20%5b9/11/16%2010:30:07%20am%5d%20Dealla%20Smith:%20Sure%20will%20do,%20glad%20you%20like%20it,%20we%20are%20trying%20to%20promote%20your%20new%20white%20paper%20%5b10/11/16%202:08:56%20pm%5d%20Dealla%20Smith:%20https://www)), and this is what he gave me:

|  |  |  |
| --- | --- | --- |
| patient | Patient() | 1 |
| lead maternity carer | Participant() | 0…1 |
| LMP – last menstrual period | date/time | 0…1 |
| date of conception | date/time | 0…1 |
| gravida | integer | 0…1 |
| parity | integer | 0…1 |
| EDD By Dates | date/time | 0…1 |
| EDD By Scan | date/time | 0…1 |
| EDD Agreed | date/time | 0…1 |

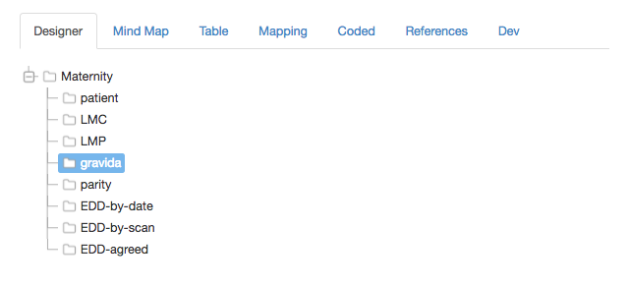
(he did note that this is a very basic model!)

So based on the chat – at least the LMP and EDDs are going to be Observations rather than a simple dateTime (or date) property. To enable representing this in the model, I made a few changes to the clinFHIR logical modeler, and created the beginnings of a Maternity model.

至少LMP和EDD将是Observation而不是简单的额dataTime属性。为了在模型中表示这一点，在clinFHIR中创建一个Maternity

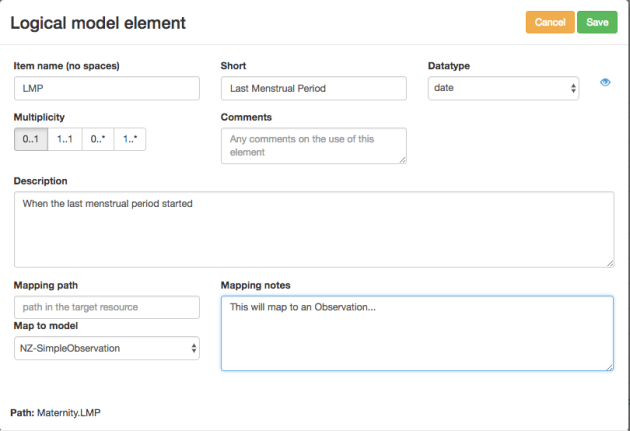
(You can look at this directly in [clinFHIR](http://clinfhir.com/) by setting the conformance server to the SNApp server, then selecting ‘Logical Modeler’ from the ‘gear’ menu to the upper right. You should see a Maternity model in the list to the left. I can’t guarantee that this will always be present of course – but there are plenty of screenshots in this post).

Here’s the designer view of the model:



When creating (or editing) a model element, there’s now an option in the bottom left labeled ‘Map to Model’ which allows you to indicate that this element is actually going to be represented by another model (presumably one that is represented as a profile on a single resource – though I gues it doesn’t need to be) – rather than a single datatype. To keep it simple, you still indicate the datatype of the element (as that is what will make sense to a Clinician). Here’s a screenshot:

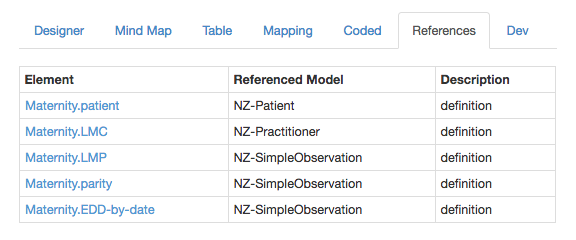
当创建或者编辑model元素的时候，左下角有一个标签为“映射到model”的选项，允许指示该元素实际上将由另一个模型表示在一个资源中而不是一个单一的数据类型。为了保持简单，仍然需要指出元素的数据类型。



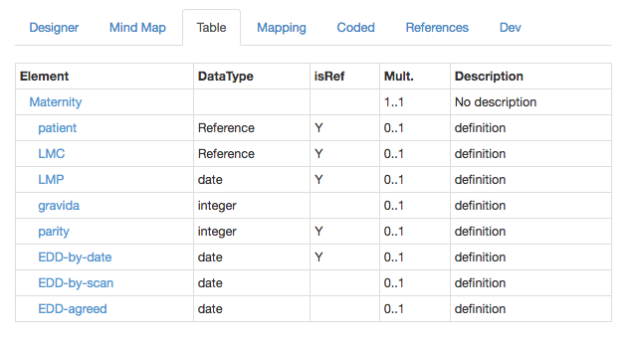
(The Mapping path is the path in the base profile – this needs more thought)

As you can see, I’ve indicated that this element is actually going to be an ‘instance’ of NZ-SimpleObservation – a model that profiles an Observation for representing a single value.

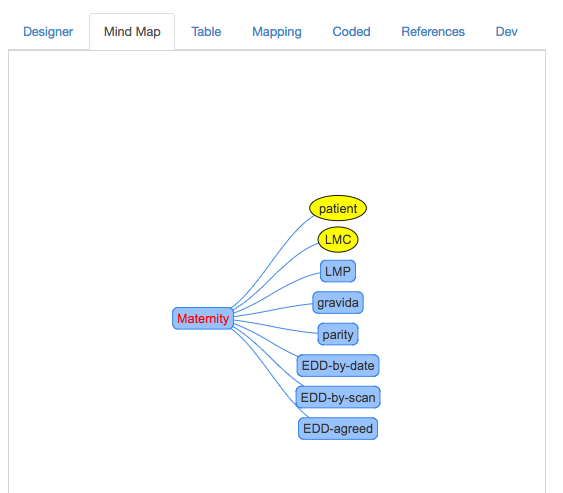
Then, I updated the ‘References’ tab to show these references, as well as the ones that are ‘real’ reference types:



and an additional column to the Table to be explicit about ‘referenced’ elements:



Oh, and here’s the mindmap:



This exercise does bring home the fact that we (or at least a particular jurisdiction) need a ‘library’ of ‘core’ models that can be used to create these larger artifacts – similar to [openEHR’s](http://www.openehr.org/) archetypes or [CIMI](http://www.opencimi.org/), but more constrained, rather  than the concept of a maximal dataset.

There’s still a long way to go – and the UI definitely needs more attention – but I think that it shouldn’t be impossible to generate the real profiles (and the Implementation Guide) from this model – maybe they could be simple resources that a tool like [forge](http://fhir.furore.com/Forge) could then pick up and refine. I’m keen on exploring Grahames [Mapping language](http://build.fhir.org/mapping-language.html) for this…

And we need to decide up front what this model will represent – for example a FHIR [Document](http://hl7.org/fhir/documents.html) or [Message](http://hl7.org/fhir/messaging.html) exchanged between partners would seem to make sense here (which would mean that the ‘base’ model would be a Composition…)

And it should also be straightforward to build sample resources from the models (as well as from the real profiles)

So there you go – moving forwards! More examples are really going to help, so ping me if interested in working on this…

Btw – I’ll likely be showing this capability as part of my presentations at the upcoming [developer devdays](http://fhir.furore.com/devdays) if you’re going to be there…

2.36 clinFHIR Profiling Walk Through

NOVEMBER 6, 2016 [2 COMMENTS](https://fhirblog.com/2016/11/06/clinfhir-profiling-walk-through/#comments)

So I’m in the middle of getting ready for the Furore  [devdays](http://www.fhirdevdays.com/) event – part of the preparation being a writing series of posts on the various stages we’ll be exploring as part of that event.

Quite co-incidentally, Mark [Braunstein](https://www.linkedin.com/in/mbraunstein) from [Georgia Tech](https://www.linkedin.com/edu/school?id=18158) asked me to do a presentation for their FHIR class which is happening in a couple of days, so I thought it a good idea to write this walk through of the complete process from end to end – from a requirement to a profiled resource instance.

We’ll return to the more detailed consideration of the steps after this.

Background

Here are the overall steps we’ll follow:

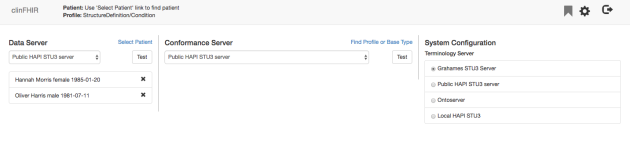
1. Gather requirements
2. Build Logical Model/s
3. Build / find valuesets
4. Create Profiles
   1. Build / find Extension Definitions
   2. Build Profile
5. Build Example resources

Pre requisites

There are a couple of things to set up first.

* Load [clinFHIR](http://clinfhir.com/).
* Login using the icon to the upper right. You just need an email for now – I promise it’s only for use within the app!
* Servers: We’ll use the public HAPI STU3 server as the data and the profile server, and Grahames STU3 server as the terminology server. However, this should work on any STU3 server (and probably STU2 though I haven’t tested that yet)

Here’s a screen shot of the configuration:

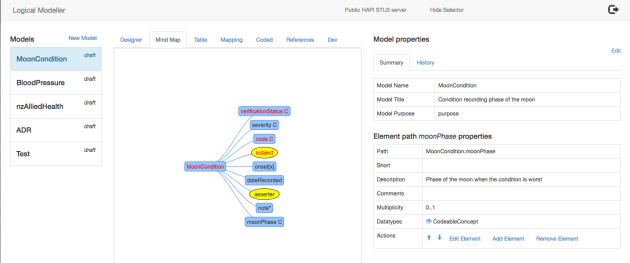


Requirements

We want to add a profile that allows us to record the phase of the moon when a condition is at its worst (I’ve been watching Harry Potter again). There’s a fixed set of values for the phase of the moon: new, waxing, full, waning. We’ll also take the opportunity to remove some of the elements from Condition that we don’t want to support.

Logical Model

Here’s the logical model:



And the steps to produce were:

1. Create new model
2. Model type = single resource (shows the ‘Initial Content’ tab)
3. Initial content from Condition resource
4. Complete other fields and save

This creates a model that is basically a simplified copy of the core resource type that we can then modify in the modeler. Here is [more detail](https://fhirblog.com/2016/11/05/more-on-fhir-logical-models/) on the Logical Modeler.

Analysing the model, we realize we’ll need:

* An Extension Definition for an element in Condition representing the phase of the moon
* A valueSet that has the permissible values for that extension
* A profile on Condition that includes the extension definition, and removes some elements we don’t want.

ValueSet

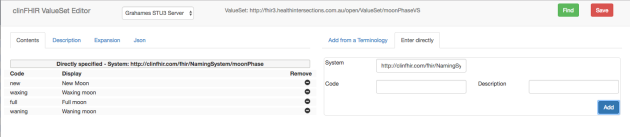
We’ll need to create a valueSet for the values of the phase of the moon. This will be bound to the extension definition that we’ll create next and will be stored on the terminology server (Grahames in our case). (In real life we’d go looking for an existing one that matches, but we’ll assume that we’re the first to think of this).

Note that we have to create the ValueSet first, as it needs to be present when we create the binding in the Extension Definition.

Steps to produce:

1. From the main clinFHIR screen click the gear icon and select ‘ValueSet Editor’. The Editor screen is displayed (Note that it is in a separate browser tab to clinFHIR).
2. Click the ‘New’ button on the top right. A dialog appears where we can enter the name of new ValueSet and a description.
3. Enter the name (see the bottom of this post for thoughts on naming conventions) and description, then the ‘check name’ button that appears. A save button should appear upper right. Click it.
4. Click the ‘Enter directly’ tab to the upper right. Then enter the possible values into the input boxes one at a time clicking ‘Add’ for each one. Set the ‘system’ value to ‘http://clinfhir.com/fhir/NamingSystem/moonPhase’
5. When all the values have been entered, click the ‘Save’ button to the upper right.

Here’s a screen shot of the ValueSet after it has been saved.



For a ‘real’ ValueSet we’d likely choose values from a proper Terminology like SNOMED, but for the purposes of education this approach is fine.

If we wanted to, we could always go back to the logical modeler and bind the ValueSet there as well. In practice, this is something that we’re likely to do commonly as the Logical Model is refined

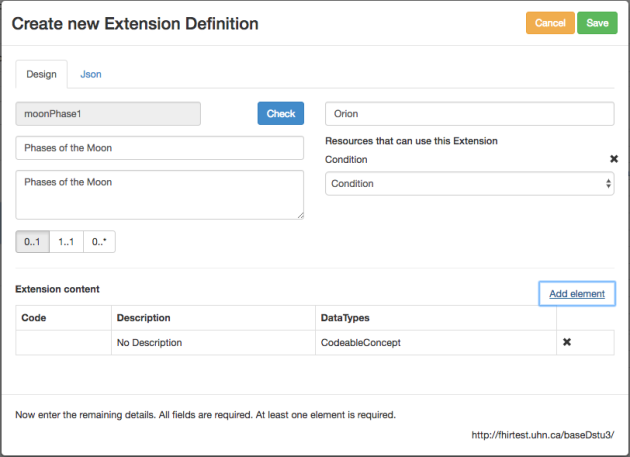
Extension Definition

The Extension definition defines the new element we are adding to the Condition. It will be a simple extension (one value only) and will be of type CodeableConcept, bound to the ValueSet we just created with a binding strength of preferred.

The steps:

1. From the main clinFHIR screen click the gear icon and select ‘Extension Definitions’. The ED Explorer screen is displayed.
2. Click the ‘new Extension Definition’ button on the top right
3. Enter a name into the dialog and click ‘check’ to make sure the name isn’t already being used. The Save button will appear (upper right) if it is new.
4. Select ‘Condition’ from the ‘Resources that can use this extension’ drop down, and fill in the description fields (short & long).
5. Click the ‘add element’ link (lower right). A dialog appears titled ‘add child element’
   1. Set the Datatype to CodeableConcept. A link labeled ‘Bind ValueSet’ appears. Click it.
   2. Another dialog appears titled ‘Bind Valueset’. Enter the name of the Valueset you created above and click ‘find’.
   3. Your ValueSet should be displayed. Click the select link alongside it, then the ‘select’ button to the upper right.
   4. The ‘Add Child’ dialog is re-displayed. Click the ‘Add’ button to the upper right. The ‘Create new ExtensionDefinition’ dialog re-appears.
6. Click the ‘Save’ button (upper right) to save the new Extension Definition.

Here’s a screen shot of the new Extension Definition just before clicking Save.



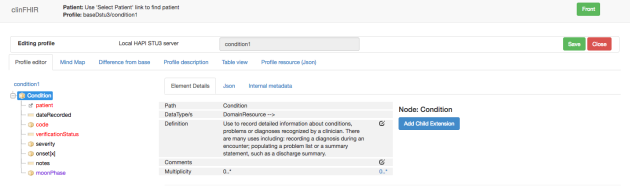
Profile

The profile is going to be the ‘template’ that we’ll use to create the Resource Instance from (ie the real resource that is attached to a patient).

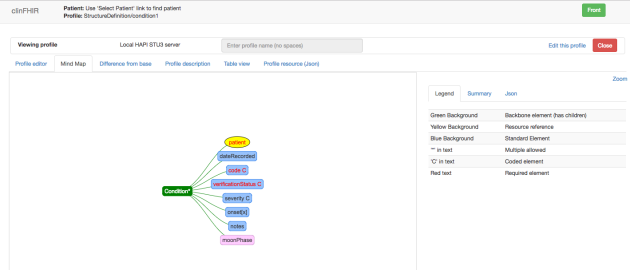
The steps:

1. From the main clinFHIR screen click the gear icon and select ‘Profiles’. The Profile Explorer screen is displayed.
2. Enter your name (or something else) as the publisher. Click the ‘magnifying glass’ icon. There probably won’t be any profiles.
3. Click the ‘new Profile’ button on the top right
4. Enter a name into the dialog and click ‘check’ to make sure the name isn’t already being used.
5. Set the ‘Resource Type being profiled’ to ‘Condition’. The Save button will appear (upper right)
6. Enter the descriptions then click save.
7. The profile explorer is displayed. Click the ‘magnifying glass’ icon. The new profile is listed in the left. Click it.
8. The details of the profile are displayed on the right, along with buttons at the bottom. Click the ‘Details/Edit’ button.
9. The profile details screen is displayed. Click the ‘Edit this profile’ link (upper right).
10. Click the Condition node that is on the top of the tree. A description of the node appears in the right, along with a button labeled ‘Add extension’. Click it.
11. A dialog should appear with the list of extensions appropriate for this resource type, including yours. Select it in the list, then click the Select button to the upper right. The profile screen is re-displayed, with the new extension at the bottom.
12. Remove elements you don’t want to support by selecting them in the tree, then clicking the ‘remove from profile’ button that appears.
13. When complete, click the Save button (upper right) to save the profile.

Here’s the tree view of the profile:



and the mindmap (note that the mindmap is not updated as the profile is updated – I had to save the profile and re-edit it to get it. I’ll fix that…)



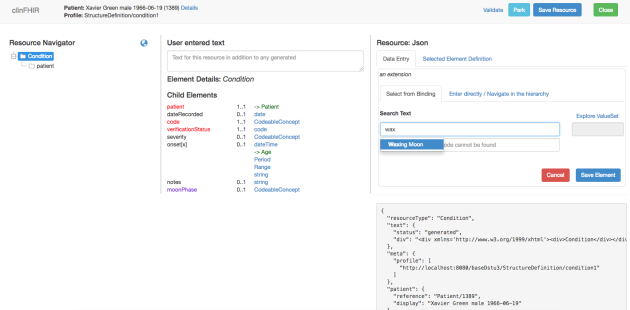
Build example resource

Finally you are ready to build an example resource using this profile.

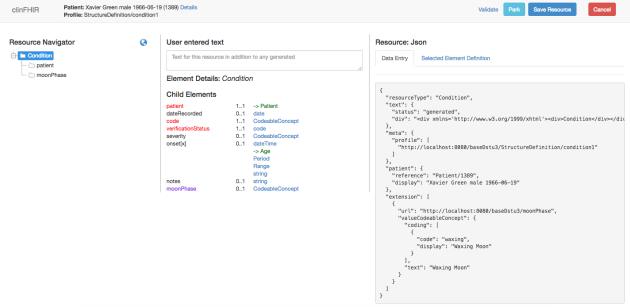
Steps:

1. Re-load clinFHIR. In fact, it’s often a good idea to do this every now & then.
2. The profile you created should be in the middle list. Click to select it.
3. Select a patient from the left list (or create a new one). The ‘New Resource Instance’ button should appear to the upper right.
4. Click the ‘New Resource Instance’ button. The resource builder screen should be displayed, with the profiled resource it it.
5. Click on the moon phase extension datatype. The CodeableConcept editor should appear in the upper right. Entering the first 3 letters of any of the values from the ValueSet should trigger the autocomplete. Allowing you to select from the list of possible values. You can also click the ‘Explore ValueSet’ link to look at the whole ValueSet. The ‘Compose’ tab in the dialog shows the permissible values. (The Expand tab will not work).

Here’s a screen shot of the resource editor with the profile selected and a value for the extension being selected.



And here’s the resource instance itself. Note that the instance has a profile tag indicating that it claims conformance to that profile.



And that concludes the walk through. I noted a number of things as I did this:

* There is not a lot of consistency between the different components – a result of the organic growth of clinFHIR. At some point, we’ll need to give it some UX attention, which means that over time the details of this walk through will change.
* If you have odd things happening then re-load the page. ClinFHIR is quite complicated under the hood and can get confused. This can happen especially when you are modifying a profile – use the ‘clear Profile cache’ from the gear menu to clear everything out and re-select the profile (in this walkthrough you’d look for profiles on Condition and yours should be in the list).
* It’s easy to forget where the different resources are – especially if you use different servers. Remember:
  + Data server has the resource instance
  + Conformance server the extension definition and the profile (they are both StructureDefinition resources)
  + Terminology server has the ValueSet
* It’s also easy to get confused between the names of ExtensionDefinitions and Profiles – as they are the same resource on the same server. I suggest a naming convention like {your name}{name of object}{object type} – eg dhMoonPhaseSD for the structure definition and dhMoonPhaseProfile for the profile.
* There are a lot of moving parts here! Do let me know through the FHIR chat of any issues you uncover.

And a reminder that clinFHIR is not a fully featured authoring tool like Forge – it’s primarily a training tool, which means that functionality is confined to a specific sub-set (especially in profiling) and that it works best with resources created with the tool. (and can be temperamental at times…)

If you do try this out, do let me know of any errors or inconstancies you find.

2.37 More on FHIR Logical Models

NOVEMBER 5, 2016 [4 COMMENTS](https://fhirblog.com/2016/11/05/more-on-fhir-logical-models/#comments)

So a couple of weeks ago I posted on a new component that has been added to [clinFHIR](http://clinfhir.com/) – a [tool that will generate Logical Models](https://fhirblog.com/2016/10/17/logical-models-in-fhir/) – models of something that uses the FHIR infrastructure but doesn’t generate ‘real’ resources. The idea is that it is used as a requirements gathering tool when when interacting with clinicians to describe what information and processes are needed to meet some interoperability related need.

Since then, we’ve added a few new capabilities to the tool, and also further refined how it can work with the other clinFHIR components to provide an ‘end to end’ solution that starts with an idea, and ends with the various FHIR resources (profiles, valueSets, extension definitions etc) that will be needed. This post starts a ‘mini series’ that describes how this could work (and do note that this is just my idea – there are lots of other ways that this could be done).

I’ve also got in the back of my mind the [FHIR Devdays](http://www.fhirdevdays.com/) in a couple of weeks where I’m leading a clinician related track – this will help flesh out what we’ll be doing there.

Let’s start with the new functionality in the modeler.

* There’s the beginnings of a way to capture comments against a model. It’s pretty basic right now – at the upper right is a tab labeled ‘Comments’ which shows a set of threaded conversations about the current model. You have to be logged in to add a comment (and the email is displayed for each comment) but anyone can read.
* The mind map now looks more ‘mind mappy’ with the elements of the model arranged horizontally from left to right. Otherwise it works the same – it’s automatically generated from the model, and selecting an element will show the details and allow you to edit it or add children.
* There’s a new tab labeled ‘Coded’ that displays all the elements that are bound to a ValueSet (code, Coding, CodeableConcept) and the url of the ValueSet – if you’ve specified that. The idea is to help identify – and find or build – the ValueSets you need in your model.
* The tab labeled ‘References’ also displays a subset of the elements in the model – those that are references to another model. I’m imagining that many of the models will actually become a profile on a single resource (though some won’t be) and so you’ll start by creating some of these ‘common’ models first (like patient and clinician’s) and then refer to them from subsequent models.
* The json and treeData views have moved to be sub-tabs of a ‘Dev’ tab. These views are only really needed for those interested in the internals of the tool.
* The tool should work against any STU-3 server (at the time of writing there’s an issue with Grahames server, but HAPI seems to work fine)

(There are screenshots of some of these tabs at the bottom of this post)

It also occurred to me that the modeler could be of value to a couple of other types of user:

* Those who are building new resource types. These do need to be developed in the proper tooling (based around Excel spreadsheet models currently) to be included in the specification build, but being able to quickly show and modify a proposed resource type might be helpful in the early days. And the ability to make comments might also be useful…
* There’s a proposal for [custom resources](http://wiki.hl7.org/index.php?title=FHIR_Custom_Resources) in FHIR – resources that use the infrastructure but are not part of the FHIR specification. This is a somewhat controversial proposal at the moment, as the ability to share these custom resources is very limited compared with the official ones – but there are scenarios under which they could be useful. For example, you could build an EMR that uses custom resources internally, converting to official ones during exchange (and Grahame has proposed a [mapping language](http://hl7.org/fhir/2016Sep/mapping-language.html) that you could use for this).

But back to the original purpose of the modeler – collecting requirements for creating proper FHIR profiles. How would the overall process work? I can imagine a number of steps.

The **first step** is to document the business requirements – what are you wanting to do. There will likely be aspects of workflow and/or process involved in this step, but the parts we are focusing on here is the information that needs to be shared – and there could potentially be more than one model we need. Domain knowledge is required for this step, and there isn’t (currently) any tooling in clinFHIR to help.

The **next step** is to use the logical modeler to create a structured representation of the information to share – the model. This could really be done by anyone – you do need to be familiar with the [FHIR datatypes](http://hl7.org/fhir/datatypes.html), and ideally the basics of FHIR (especially referencing between common models/resources) but really domain knowledge is the most important.

You also need to decide if you are modeling what will become a single resource (eg an allergy intolerance) or a collection of resources – like a list, document or message. In the latter case you might want to split the design up into a number of referenced models to make the next stage simpler.

This step can take some time as getting agreement from multiple people is not always straightforward – hopefully the ability to make comments against the model will help here.

There are a number of outputs from this stage.

* The models themselves will become profiles – both constraints and extensions on resource types
* The ValueSets we’ll need for the coded elements (which is why there’s a separate tab in the modeler for these). We’ll consider ValueSets in more detail in the next post.

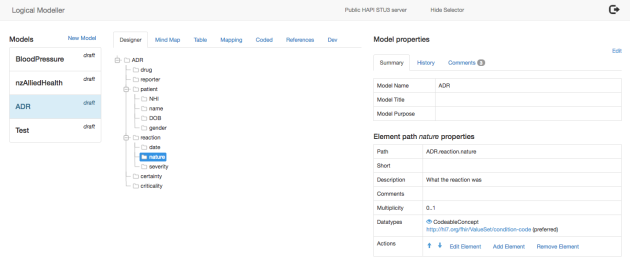
The **final (almost) stage** is to create or locate the FHIR artifacts – ValueSets, Profiles and Extension Definitions that represent the models in FHIR-space. Right now this will largely be a manual process using tooling (either in [clinFHIR](http://clinfhir.com/) or [Forge](http://fhir.furore.com/Forge)) and will need to be done by someone who understands FHIR reasonably well. However, it may be possible to automate the process (possibly using Grahames [mapping proposal](http://hl7.org/fhir/2016Sep/mapping-language.html)) by constraining how the modeler works – this needs more work though. And do note that Forge is much more comprehensive than clinFHIR. It may be that you use clinFHIR in the early stages to get the artifacts ‘mostly’ right (or exemplars of what you want), then Forge to complete them.

My guess is that there will be a ‘feedback loop’ at this point right back to the original model as the process of mapping to FHIR resources will likely produce suggestions that may not have been considered in the first stage – after all, a lot of people have contributed to the FHIR resources! So there will be one or more ‘cycles’ of development until the FHIR artifacts are finalized. There’s currently no direct link between the model and the FHIR profile – something to work on.

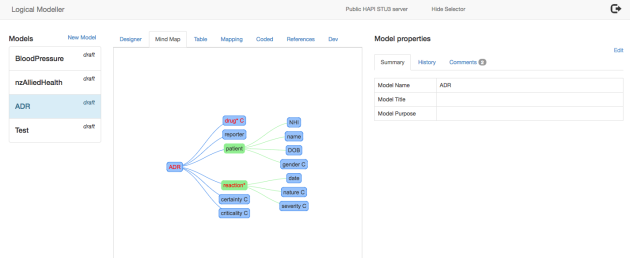
So I think that’s enough for a single post. In the next installment, we’ll take a logical model (probably Adverse Drug Reactions) and think about how we’ll generate the real FHIR artifacts from that model.

In the meantime, here are screen shots of the main parts of the modeler as a reference.

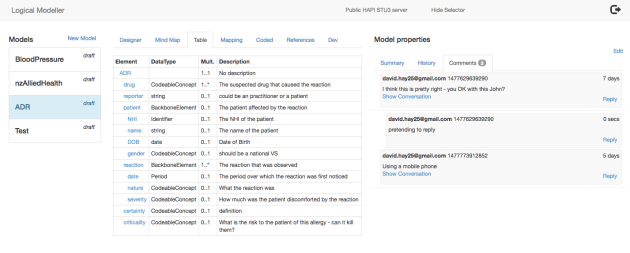
The main screen with the ADR Model selected



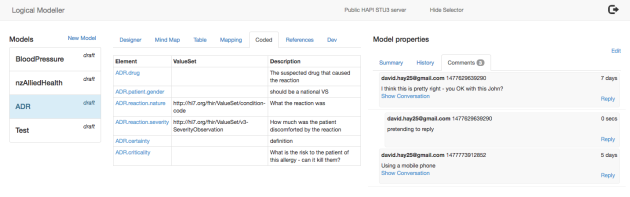
The mind map: (colours & icons still need attention)



The table view – with the comments tab selected



Coded elements summary



2.38 Setting up your own FHIR server

OCTOBER 19, 2016 [2 COMMENTS](https://fhirblog.com/2016/10/19/setting-up-your-own-fhir-server-for-profiling/#comments)

I’ve been having some conversations with a group that are wanting to use [clinFHIR](http://clinfhir.com/) to view profiles – and create sample resources from them – but they are using [Forge](http://fhir.furore.com/Forge) to create them (good choice!) and so the question becomes about how to put them where clinFHIR can access them.

Before we get into the detail, lets take a step back and look at what we need to achieve to support clinFHIR in this way.

ClinFHIR has the concept of ‘server roles’. We’ve [talked about them](https://fhirblog.com/2015/09/07/server-roles-in-fhir/) before, but in summary there are 3 of them:

* The Patient/Data server that holds the Patient resources, and all of the ‘non-conformance’ resources – those that represent clinical data and supporting ones such as Practitioner or Organization
* The Conformance (or Registry) server that holds the ‘definitional’ resources like StructureDefinition (Profile and Extension definition), NamingSystem and a few others
* The Terminology server that has both ValueSet resources, and also exposes a number of terminology services (like $expand) that clinFHIR uses to populate drop downs from a ValueSet.

(Note that this is just how clinFHIR works – it’s not a FHIR requirement – but it does support the idea of a single place to place the ‘infrastructural’ resources that multiple data servers in an environment can then use).

And also note that these are ‘roles’ – they could all be performed by the same physical server if needed.

Here’s clinFHIR configured to use HAPI STU-3 as the Patient/Data server, and Grahames STU-3 server as the Conformance (‘My Profile’) and Terminology servers. (Note that all servers should be on the same FHIR version).



This distribution of roles has an impact when creating Profiles. Generally you’re going to be creating at least 2 types of resource, that will be hosted on 2 of these server roles – StructureDefinition resources (Profile and Extension Definition), which will go on the Conformance server, and ValueSets that will go on the Terminology server.

So: if we’re setting up our own infrastructure to perform profiling we’re going to need a server (or servers) that can fulfill at least the Conformance and Terminology roles. As always we have a number of options, but the one we’re going to use is the HAPI ‘Command Line Interface’ (CLI) application produced by the clever Canadians who built the HAPI library – they’ve taken the library and wrapped it up into a complete server that anyone can download and use for free – how cool is that!

***Just to emphasize that the server we’re going deploy is for testing purposes only – it has no security! You should definitely not upload any Personal Health Information to this server, and if you have a need for reliability you should consider hardening it – which is outside the scope of this post.***

So here are the steps to set up your own FHIR server using HAPI. I’m going to assume that you are comfortable with the [command line](https://en.wikipedia.org/wiki/Command-line_interface) – or know someone who can help you out if you get stuck.

First, create a hosted server in the cloud. I use [DigitalOcean](https://www.digitalocean.com/) as you can do the whole thing within a  few minutes and they have options starting from $5/month (though you really want a reasonable amount of RAM so the $20 / month version is probably advisable *\*\*Update – you do need the $20 version to get 2Gig of RAM\*\**). This will give you an [Ubuntu](https://www.ubuntu.com/) server in the cloud with an IP address and a password (though you can set up certificated access which is both more convenient and more secure). Note that the server must have Java 8 installed.

Next login to the server from the command line of your computer using [SSH](https://en.wikipedia.org/wiki/Secure_Shell) eg:

ssh [root@clinfhir.com](mailto:root@clinfhir.com)

Note that we’re logging in as the root user in this case. There are more secure ways of doing this, but it keeps it simple for now.

Now we need to download the HAPI CLI application to the server. A good place to put this is in the /opt directory, so execute the following commands to move to that folder and download the file (we’ll use [WGET](https://www.gnu.org/software/wget/) to do so)

cd /opt

wget https://github.com/jamesagnew/hapi-fhir/releases/download/v2.0/hapi-fhir-2.0-cli.zip

This will download version 2 of the CLI into the folder – you can get the latest version from the [hapi download page](https://github.com/jamesagnew/hapi-fhir/releases).

Next unzip the downloaded file:

unzip hapi-fhir-2.0-cli.zip

You are now ready to start the server. There are a number of ways of doing this – I’ve used the following command which seems to work most of the time, but Java can be a dark art at times (at least for me…)

java -jar hapi-fhir-cli.jar run-server

All going well, you will see a number of informational lines scroll by, ending in a declaration that you now have a fully functional dstu-2 server listening at the endpoint [ip]/baseDstu2:8080

But what if you want STU-3? Or a different port? Well, the server actually has a number of command line switches (they are documented in the [hapi page](http://hapifhir.io/doc_cli.html)) to change the port, set the FHIR version & so forth. This command line will start the server in STU-3 mode, and make it run in the background so that it doesn’t stop when you close the connection:

nohup java -jar hapi-fhir-cli.jar run-server -f dstu3 --allow-external-refs &

(I usually create a small [shell script](https://en.wikipedia.org/wiki/Shell_script) to save typing)

update: note that “–allow-external-refs” is added to the end of the command (you have to scroll horizontally to see it) as otherwise the CLI won’t accept resources that reference external entities. (Thanks to [Philip Scott](http://www.linkedin.com/in/philip-scott-b4627211) for discovering this – and other omissions in this post!)

Do note that although you can use either STU-2 or STU-3, you shouldn’t use the same instance for both on the same server as weird things will happen. (I found this out the hard way).

The server is currently empty, but it would be convenient to load the standard resources if it is going to act as a Conformance/Terminology server. This can easily be done as follows.

With the server running, open a new SSH connection to the server (by opening a new terminal window and re-issuing the SSH command), change to the folder where HAPI is installed and enter the following command:

hapi-fhir-cli upload-definitions -t http://localhost:8080/baseDstu3

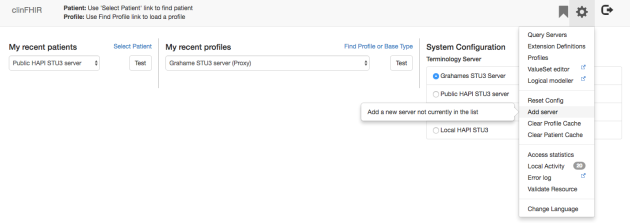
or

java -jar hapi-fhir-cli.jar upload-definitions -t http://localhost:8080/baseDstu3

(but replace ‘http://localhost:8080/baseDstu3’ with the base url of your own server)

This will download all the conformance resources (StructureDefinition, ValueSet) from the main FHIR site to your local server.

So now you can point clinFHIR at your new server, select any of the standard resource types and view their profile. You can also create sample instances. To do this, you’ll need to tell clinFHIR about your new server – you can do so using the ‘Add Server’ option off the main menu:



And here is where you enter the server details:



…making sure you enter the correct Url of your server. (After entering the server url, click the ‘test’ button that appears to the lower right – that will check that the server is a valid FHIR server, and will display a ‘Add’ button if it is – ie it returns a [Conformance/CapabilityStatement](https://fhirblog.com/2016/10/19/renaming-the-fhir-conformance-resource/)resource. You also need to select the FHIR version – though as I write this I realize that the app should be able to determine this from the CapabilityStatement – I’ll fix that later!)

(btw – this ‘add Server’ functionality only works on the Browser where you issued the command – and will be deleted if you need to reset the config. Something else for me to fix  )

There’s one last thing you might want to do, which is to install the SNOMED files on your server. This will allow the server to expand SNOMED based ValueSets when acting as a Terminology sever. The process is described [here](http://hapifhir.io/doc_cli.html) – but do be aware that there are licensing issues relating to the use of SNOMED, so make sure there are no issues before you do this.

Now that your server is all set up, you’re good to go with loading your own profiles on to it. There are (as always) a number of options for doing this.

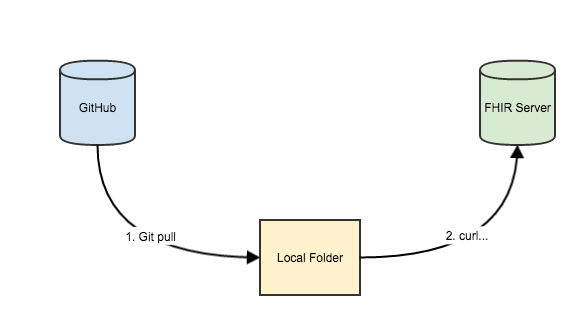
The easiest is to simply use a REST client of some sort to copy them. You can either use a REST client with a User Interface (such as [POSTman](https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop?hl=en)) and copy/paste the resource file into the tool then upload it – or use [CURL](https://curl.haxx.se/) which is a command line client that has been around for ever. Here’s an example of its use:

curl http://fhir.hl7.org.nz/baseDstu2/StructureDefinition/ohCondition --upload-file ohCondition.structuredefinition.xml

It will upload the file ‘ohCondition.structuredefinition.xml’ in the current folder to the FHIR server at <http://fhir.hl7.org.nz/baseDstu2/StructureDefinition/ohCondition>

(What I did was to create a shell script in the folder where the files are stored with a command for all the files I want to upload, then it’s easy to upload them just by executing the script. Remember to [make the script executable](http://askubuntu.com/questions/229589/how-to-make-a-file-e-g-a-sh-script-executable-so-it-can-be-ran-from-termina)!).

If you are using gitHub, then just create a local copy of the files by cloning the repo and execute the upload script after using ‘git pull’ to make sure you have the most recent copies of the files). Here’s a diagram of how it could work…



One ‘gotcha’ to be aware of, is that when you upload a resource to the HAPi server, it will check that any resources it references are already on the server, rejecting the resource if it does not. (I’m not 100% sure that I agree with this, but it is the behavior). The practical impact of this is that there is a distinct order of upload to follow (this can all be in the same script of course).

1. All ValueSets referenced by any of the resources (if your server will be a Terminology Server)
2. The StructureDefinitions that represent extension definitions
3. The StructureDefinitions that represent profiled resources

(Another possible ‘gotcha’ is to make sure that Forge generates snapshots in the profiles…)

So there you go. A FHIR server of your very own for profiling!

Addendum: If you have issues with this installation, there is an [active support group](https://groups.google.com/forum/#!forum/hapi-fhir) that you can use.

2.39 Renaming the FHIR Conformance resource

OCTOBER 19, 2016 [1 COMMENT](https://fhirblog.com/2016/10/19/renaming-the-fhir-conformance-resource/#comments)

Just a short note to let people know of a significant change that has occurred in FHIR since the Baltimore Working Group meeting in September.

Perhaps surprisingly, there’s only one actual requirement of a RESTful FHIR server – that it hosts a resource that describes what its capabilities are. It returns this resource when you call the metadata endpoint of the server like this:

<http://fhir3.healthintersections.com.au/open/metadata>

Previously called the [conformance resource](http://hl7.org/fhir/conformance.html), it describes the version of FHIR that it supports, the RESTful endpoints, profiles, messaging endpoints and a host of other stuff.

However, the word ‘conformance’ has come to mean a lot more as FHIR has evolved, so its name has been changed to ‘[CapabilityStatement’](http://hl7-fhir.github.io/capabilitystatement.html) to reflect that.

It has the same structure (or pretty much so I think) and serves the same purpose – it’s just a rename.

Currently only Grahames server has this change (so the link above will return it) but you can expect the other STU-3 servers to support this soon.

You have been warned!

2.40 Logical models in FHIR

OCTOBER 17, 2016 [6 COMMENTS](https://fhirblog.com/2016/10/17/logical-models-in-fhir/#comments)

So one of the capabilities that has been added to FHIR reasonably recently is the capability to use the conformance infrastructure (like StructureDefinitions, ValueSets etc) to produce logical models – rather than defining resource types and such like.

The purpose of the logical model is to gather requirements from clinicians and other folk that can subsequently feed into the design process for profiles and Implementation Guides. While it’s possible to create profiles directly during this process (and one of the goals of clinFHIR is to support this) – there’s no doubting that it does require some understanding of FHIR and the structure/purpose of the existing resource types, so the Logical Model ‘bridges the gap’ by documenting the requirements in a computable fashion without placing any constraints on the model structure.

Because it is computable, it makes the task of generating the formal profiles that much simpler (and recall that a single Logical model may result in multiple profiles – in fact it’s closer to an [Implementation Guide](http://hl7.org/fhir/implementationguide.html)) – and also enables the creation of specialized tools like the clinFHIR Logical Modeler that this post describes to create the model in the first place. It also makes it simpler to generate examples, which are both helpful for understanding the domain, but also for testing the model.

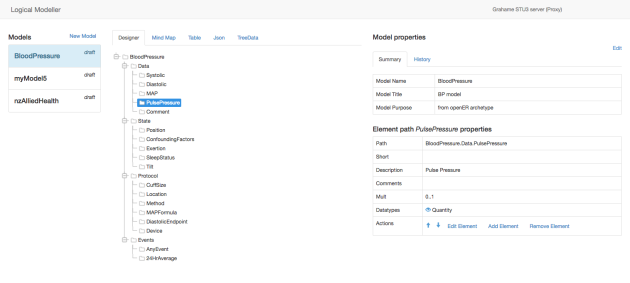
So the overall flow from requirement to Implementation Guide would be as follows:

1. Build the Logical Model. This requires significant interaction with the clinical and business community – multiple meetings and refinement of the model, but should require minimal knowledge of FHIR (though a certain amount is needed – like an understanding of the datatypes). An output of this stage would be the Logical Model and the ValueSets required (or at least their identification)
2. Generate the Implementation Guide that has all the FHIR artifacts such as Profiles and the ones generated in the first step (e.g. ValueSets). This step will require better understanding of FHIR – but at the ‘business’ level rather than deep technical level so should be well within the capabilities of Business Analysts and motivated clinicians.

With that background, let’s take a look at the new  [clinFHIR](http://clinfhir.com/) Logical Modeler.

*(Currently it will only work against Grahames STU-3 server (as it uses some of the newest conformance resource change) so if you select that server as the conformance server, you’ll find a link to the modeler in the ‘gear’ menu.)*

When the modeler loads, it will read all existing models from the server and present them in the list. Selecting one of them will display the main screen:



(This model is the [openEHR](http://www.openehr.org/) Blood Pressure archetype adapted (mostly) to the logical model structure).

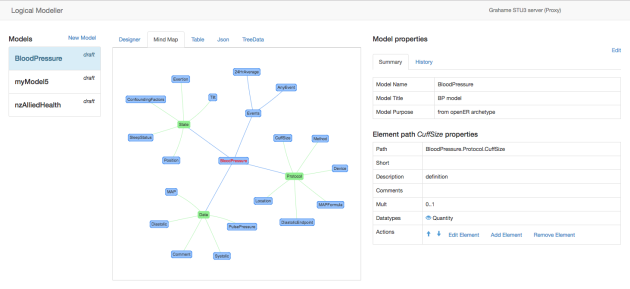
There are 3 main panes:

* To the left is the model selector – currently this will show all models on the server
* In the middle is the editing pane.
* To the right is the details pane – the model and the currently selected element.

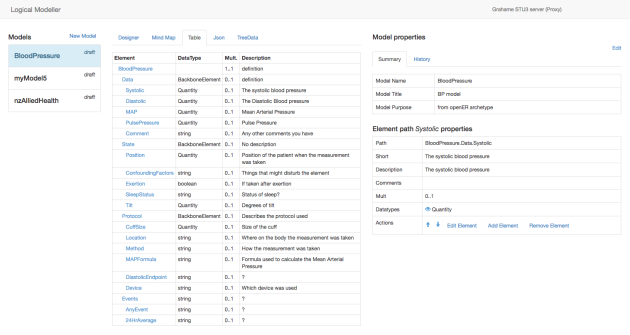
Most of the action is in the editing pane, which has 5 tabs:

* The *Designer* shows a tree view of the model. Selecting an element will show the details of that element in the details pane. The tree can be expanded and collapsed as required.
* The *Mind Map* (technically it’s more of a graph, but I’ve been told that clinicians will better understand what a Mind Map is than a graph) gives a hierarchical representation of the components of the model. Like the designer, you can select a node and see the details in the detail pane. You can click and drag nodes, and move/zoom the graph as well.
* The *Table* lists each element in a row. It’s similar to the designer, but is a more compact representation. Again you can select an element to see the details of that element.
* The *Json* tab shows the StructureDefinition resource that holds the details of the Model. Useful if you want to know the details of the FHIR representation, but intended for geeks rather than ordinary folk.
* And finally the *TreeData* tab is really only for me during development of the tool. It will probably disappear in due course.

Here’s what the MindMap looks like:



And here’s the Table:



As described above, the **Details tab** has 2 main parts.

At the top are details about the overall model. This has 2 tabs:

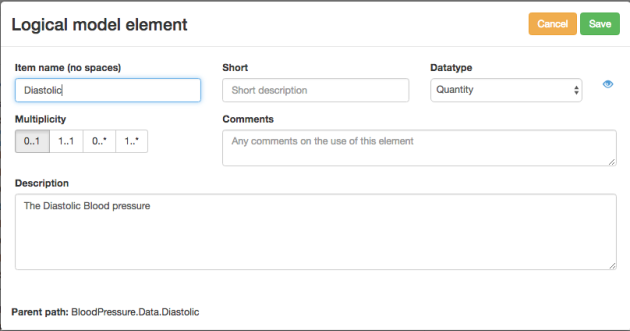
* The Summary tab has the model details. You can edit them by clicking the edit tab to the upper right (There’s not a lot of information right now – that will likely change).
* The History tab will show previous versions of the model. Each version shows when it was created. If you select one of these versions, then the details of that version are displayed in the Editing pane, and you can view the details as you would a Model being edited (though the editing options are disabled). There’s a notification above the Editing pane that tells you that you are viewing history, and gives you 2 options:
  + ‘*Back to Current*’ re-loads the Model you were editing before you viewed the history (and re-enables editing)
  + ‘*Revert to this version*’ will make this version the current one. It’s useful if you’ve gone down the wrong track and want to go back to a previous version.

Below this is the details of the selected element. It also has 2 parts:

* The value for the main elements (description, comments, datatype etc.)
* Action links that apply to this element. These include:
  + *Up arrow*. Moves the selected element (and its children) higher in the hierarchy (unless it is already at the top)
  + *Down arrow*. Moves the selected element (and its children) downthe hierarchy (unless it is already at the bottom). There still appears to be some flakiness about these movement options – they are surprisingly tricky to implement.
  + *Edit Element*. Display the edit screen for this element and allows you to make changes
  + *Add Element*. Allows you to add a child element to this one. Uses the element edit screen to get the details
  + *Remove Element*. Removes this element – and all the child elements as well.

Note that the action links will work for the different views in the Editing pane (Designer, Mind Map & Table) – and all views will be updated in real time with edits as they occur.

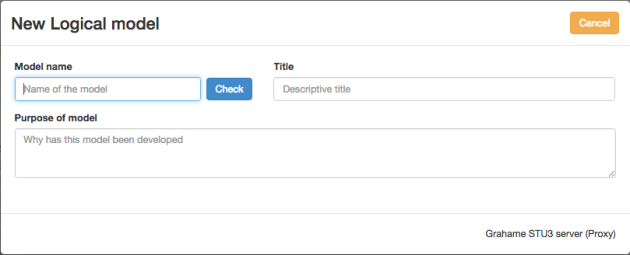
Here’s the new and edit element screen:



The ‘eye’ icon to the right of the dataType dropdown will show the details of that datatype in the spec.

If you make a change to the model, a ‘Save’ button will appear in the Designer tab. Clicking that will save a new version of the model, but you can carry on editing. As you can always revert to a previous version, this means you can do ‘what if’ changes, and then easily revert. If there are errors during saving, then an error tab will appear in the Editing pane. More advanced users can use that to fix errors. It is suggested that you save frequently.

To create a new model, click the ‘New Model’ link in the left pane and the model details screen is displayed:



Enter a name for the model – it needs to be a single word and not too long, then click the ‘Check button’. ClinFHIR will make sure that there isn’t already a model with that name – if there is you’ll get a warning, if not then the Save button will appear and you can enter the remaining information and save.

So there’s probably a few more things to do before this becomes a useful tool.

* The UI needs refinement – which will probably only become apparent after use. For example, it’d be nice to be able to hide the model selector to give more space to the editor tabs, and the mind map definitely needs some UX love.
* Each element can only have a single datatype – we need to allow multiple types. There are other elements we need to capture as well – for example the ‘units’ for a quantity, or the system for an Identifier
* We need the ability to reference between models – that way you could have common models that can be re-used like a custom patient or provider.
* A library of common sub-models to jump start design. These would be incorporated into the model.
* Generating the Implementation Guide from the model. This is a biggie, and unlikely to ever be fully automated, but there are ways to make it easier. For example you could indicate in the model what conformance artifact (and path) that it would map to, and then use Grahames [declarative mapping](http://wiki.hl7.org/index.php?title=201605_Declarative_Mapping) tooling to generate the IG (including profiles and extension definitions).
* The ‘security’ definately needs enhancement. Currently anyone can update any model so we probably need to think in terms of ‘projects’ that ‘own’ a model and can update it. Likely we’ll have the concept of ‘editors’ – specific people who can udpate models, but it would be nice if anyone can copy a model and then adapt that to their needs. Even nicer would be the ability to ‘merge’ models – picking branches from multiple models to create a new one.
* The ‘social’ aspect could also use a bit of attention. While I’m imagining that you could use the tool in a meeting to document the discussion, it would be nice for people to make comments at any time from anywhere. Some kind of threaded conversation attached to a model element would be nice.

2.41 Supporting SNAPP: Accessing New Zealand Medicines Terminologies With FHIR Terminology Services

OCTOBER 16, 2016 [2 COMMENTS](https://fhirblog.com/2016/10/16/supporting-snapp-accessing-new-zealand-medicines-terminologies-with-fhir-terminology-services/#comments)

A guest post from [Peter Jordan](https://www.linkedin.com/in/peter-jordan-44146418) – author of the Patients First Terminology server that will be available for use at SNApp, and frequent HL7 FHIR Connectathon attender…

Take it away Peter…

As those developing healthcare applications for the local market are well aware, and overseas developers preparing for the [‘SNAPP’](https://confluence.ihtsdotools.org/snapp) event at the upcoming [SNOMED conference in Wellington](http://www.snomedexpo.org/), are becoming aware, New Zealand has its own medicines terminology. Interacting with this terminology – broadly known at the NZ Universal List of Medicines ([NZULM](http://www.nzulm.org.nz/)) – is one of the many challenges facing SNAPP participants.

In common with the self-deprecating slogan used to advertise a local soft drink, NZULM happens to be ‘world famous in New Zealand’, but little known outside the ‘Shaky Islands’. Therefore, in the interests of levelling the playing field for all contestants (and there is prize money at stake here), this post is a brief introduction to NZULM and how one may interact with it using a Terminology Services API based on…no surprises to readers of this Blog…FHIR.

While NZULM shares some similarities with SNOMED CT (SCT-compatible identifiers, fully-specified names and preferred terms), to date it has not been designed as a SNOMED-CT national drug extension and therefore, not instantly accessible through the Implicit Value Set syntax that FHIR offers for accessing SNOMED CT.

In fact NZULM consists of several components, the main one – particularly for this purpose – being the NZ Medicines Terminology (NZMT). In FHIR parlance, one might describe this as the [Code System](http://hl7-fhir.github.io/codesystem.html). In relational database terms, this consists of a large number of tables, but the key objects are those designed to hold the seven distinct “product concepts” – collectively known as the “Seven Boxes”:

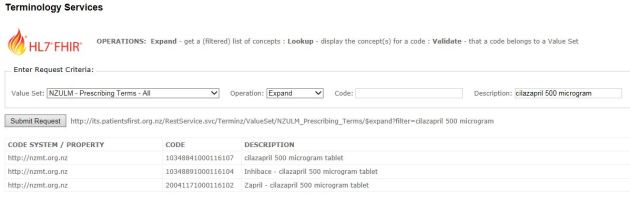
|  |  |
| --- | --- |
| **MP** | **Medicinal Product**  Generic substance.  Eg: Paracetamol.  May refer to a complex product containing multiple substances.  Eg: Paracetamol + Codeine. |
| **MPUU** | **Medicinal Product Unit of Use**  Generic substance formulation including dose size and dose form.  Eg: Paracetamol 50mg Tablet. |
| **MPP** | **Medicinal Product Pack**  Generic substance pack size. Generally treated as an Abstract concept.  Eg: Paracetamol 50mg Tablet, pack of 20. |
| **TP** | **Trade Product**  A branded product  Eg: Panadol  Eg: Panadol Night and Day (complex pack) |
| **TPUU** | **Trade Product Unit of Use**  A branded product formulation including dose size and dose form.  Eg: Panadol 50mg Tablet |
| **TPP** | **Trade Product Pack**  A branded product formulation with pack size.  Eg: Panadol 50mg Tablet, pack of 20 |
| **CTPP** | **Containered Trade Product Pack**  A branded product formulation with specified pack size and container  Eg: Panadol 50mg Tablet, pack of 20, Blister Pack |

In addition, the monthly distribution includes several combined views of these tables, the most significance of which – for SNAPP anyway – might be that containing all the unique available prescribing terms, generic and trade, with their associated concept identifiers and a flag to indicate whether the term is generic or trade.

Now – short of downloading grabbing the entire NZULM package and digesting copious amounts of additional documentation – how does one interact with this complex terminology without attempting to become an overnight expert? The answer offered here centres on the value proposition for terminology services – delegate most of that responsibility to specialist software and interact with a relatively simple, standards-based, API. In particular, the NZ Terminology Server ([Terminz](http://its.patientsfirst.org.nz/RestService.svc/Terminz/)) built by [Patients First Ltd](http://www.patientsfirst.org.nz/).

For a simple introduction to Patients First Terminology Services a simple interactive test page is provided here: <http://itp.patientsfirst.org.nz/Home/Terminology>

1. Choose a Value Set  
2. Select an operation  
3. Enter a description (expand) or code (lookup and validate)  
4. Click Submit Request…view results and the request URL (below)



For API access, assuming some knowledge of the basics of [FHIR](http://hl7-fhir.github.io/index.html) and the [Terminology Service Section](http://hl7-fhir.github.io/terminology-service.html) of the Specification, here are some basic URLs…

// Base URL

<http://its.patientsfirst.org.nz/RestService.svc/Terminz/>

// request a Conformance Statement

<http://its.patientsfirst.org.nz/RestService.svc/Terminz/metadata>

// request definitions of all the available Code Systems

<http://its.patientsfirst.org.nz/RestService.svc/Terminz/CodeSystem>

// request definitions of all the available Value Sets

<http://its.patientsfirst.org.nz/RestService.svc/Terminz/ValueSet>

The list of [Value Sets](http://hl7-fhir.github.io/valueset.html) contains an entry for each of the 7 NZMT Boxes tabulated above, plus an additional 3 that provide ‘generic only’, ‘trade only’ and ‘all’ concepts from the Prescribing Terms Table. These can be used at the target for requesting [Expansions](http://hl7-fhir.github.io/valueset-operations.html#expand) (e.g. for populating dropdowns controls on a user interface) or to [Validate](http://hl7-fhir.github.io/valueset-operations.html#validate-code) that a particular Code (concept identifier) belongs in a particular value set.

For example…

// [BASE ValueSet URL]

http://its.patientsfirst.org.nz/RestService.svc/Terminz/ValueSet/NZULM\_Prescribing\_Terms/

// Expansion filtered on all terms containing ‘cilazapril’

[BASE ValueSet URL] $expand?filter=cilazapril

// Validate that this code & code system combination belong to the Value Set

[BASE ValueSet URL] $validate-code?system=http://nzmt.org.nz&code=10348841000116107

Requests to [Lookup](http://hl7-fhir.github.io/codesystem-operations.html#lookupl) codes operate across the NZMT Code System as a whole (i.e. the “Seven Boxes” of NZMT, which includes those contained in the Prescribing Terms Value Sets)…

// [BASE CodeSystem URL]

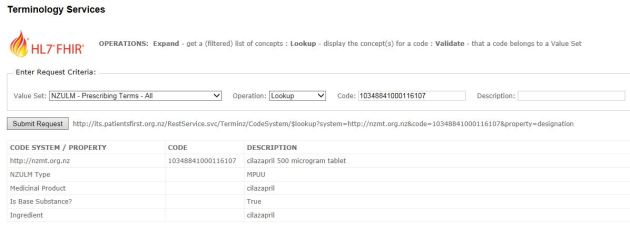
<http://its.patientsfirst.org.nz/RestService.svc/Terminz/CodeSystem/>

// Lookup this code in the passed code system and return any designations

$lookup?system=http://nzmt.org.nz&code=10348841000116107&property=designation

Note that the last request also asks for the designations which, in addition to the code system, code and description, will return the following information…

* NZULM Type…which of the Seven Boxes the code belongs to
* Medicinal Product…The name of the generic substance linked to this concept
* Is Base Substance…Whether the generic substance is the base substance
* Ingredient…The active ingredient(s) in the generic substance



2.42 Contained resources – MedicationStatement

OCTOBER 3, 2016 [LEAVE A COMMENT](https://fhirblog.com/2016/10/03/contained-resources-medicationstatement/#respond)

Continuing our ‘mini-series’ on the [SNapp](http://www.snomedexpo.org/snapp) event, one thing you may come across when consuming MedicationStatement resources are ‘Contained’ resources. [We’ve talked](https://fhirblog.com/?s=contained)about them before, but in the interest of having all the SNapp information in the same place let’s take a look at what they are.

Before we do, let’s take a look at t[he definition](http://hl7.org/fhir/medicationstatement.html) of the MedicationStatement resource in the spec. In particular, note that there are 2 ways for the resource to refer to the actual medication it is describing – medicationCodeableConcept and medicationReference (remember that when there is more than one possible datatype for a property, the property name incorporates the datatype).

screen-shot-2016-10-03-at-11-17-27-am

And note that the multiplicity is 1..1 – you must have one (and only 1) of them.

The medicationCodeableConcept makes sense where all the information you want to record about the medication is found in the terminology that the code refers to. Here’s an example of it in use:

...

"medicationCodeableConcept": {

   "coding": [

     {

       "system": "http://snomed.info/ct",

       "code": "108551001",

       "display": "carvedilol"

     }

   ],

   "text": "carvedilol"

},

...

The medicationReference allows you to refer to a specific resource that represents the medication, and can have a lot more detail. This is useful when the terminology does not have all the data that you wish to record, but is undeniably more difficult to use as the client needs to retrieve the ‘referred to’ resource to get the full details, and it needs to be served up from somewhere. For this reason, many implementations will ‘contain’ the Medication resource inside the MedicationStatement. Here’s an example:

...

"contained": [

   {

     "resourceType": "Medication",

     "id": "med1",

     "code": {

       "coding": [

         {

           "system": "http://snomed.info/ct",

           "code": "7947003",

           "display": "aspirin 81 mg"

         }

       ],

       "text": "aspirin 81 mg"

     },

     "isBrand": false

   }

],

"status": "active",

"medicationReference": {

   "reference": "#med1",

   "display": "aspirin 81 mg"

},

...

(btw – I got these examples from the sample data that [clinFHIR](http://clinfhir.com/) generates, as [previously discussed](https://fhirblog.com/2016/09/29/supporting-the-snomed-snapp-hackathon/). Aspirin is represented as a contained resource)

The contained resource can have any of the values that the resource would normally have (including extensions) but not text, and you can’t nest contained resources (ie a contained resource can’t have its own contained resources). You can, however, have any number of contained resources within the parent.

Note that the ‘parent’ element has got a reference to the id of contained resource which is a simple string prefixed by a ‘#’.

Now, it is possible for individual implementations to be able to restrict this behaviour using profiles. For example. I could have a profile that only allows the CodeableConcept and removes the ability to make a reference to a Medication, whether contained or a discrete resource (Though you’d want to be really sure that there will never be information about the medication that is not in the terminology – so it seems a brave step to make!). The argument about whether any particular piece of information belongs in the terminology rather than the structure is one that can provoke passionate opinions!

Note also that all FHIR clients need to be aware of, and be able to resolve, contained as well as external resources

2.43 SNapp server

OCTOBER 2, 2016 [2 COMMENTS](https://fhirblog.com/2016/10/02/snapp-server/#comments)

Just a short note to say that I’ve set up a FHIR server for the [SNapp event](https://confluence.ihtsdotools.org/snapp) – based on the [HAPI CLI server](http://hapifhir.io/doc_cli.html).

The IP address of the server root is:

http://snapp.clinfhir.com:8080/baseDstu3/

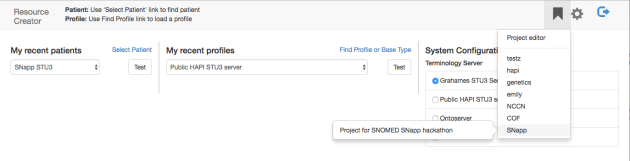
For example to get the conformance resource you’d enter:

http://snapp.clinfhir.com:8080/baseDstu3/metadata

So the recommended layout for SNapp is:

* Data Server: SNapp (STU3)
* Conformance Server: Public HAPI (STU3)
* Terminology Server: Grahames Server (STU3)

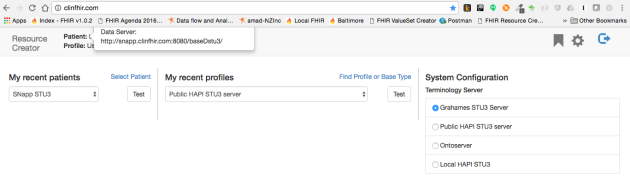
Which quite nicely shows how the different server ‘roles’ could work in a real environment. I’ve created a ‘project’ to select them for you:



Just a note that if you’ve already used [clinFHIR](http://clinfhir.com/) then you’ll need to reset the config to see the SNapp server in the list as the screen shot below shows (and restart the app after you’ve done that – just to be tidy).



Oh – and also worth noting that if you want to see the URL of your currently selected Patient & Conformance server, just mouse over the label in the nav-bar – like so:

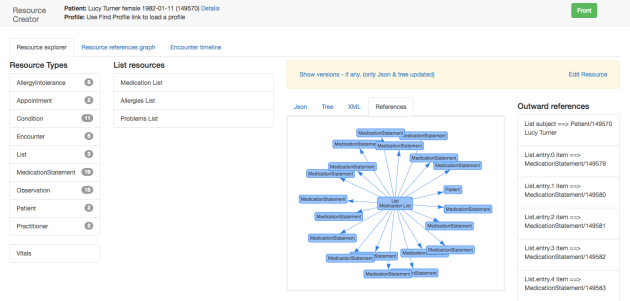


2.44Supporting SNapp: Lists of medications

SEPTEMBER 30, 2016 [6 COMMENTS](https://fhirblog.com/2016/09/30/supporting-snapp-lists-of-medications/#comments)

In this post we’re going to look at representing the Medication List using a [List](http://hl7.org/fhir/2016Sep/list.html) resource as part of supporting the [SNapp](http://www.snomedexpo.org/snapp) event at the [SNOMED conference in New Zealand](http://www.snomedexpo.org/) next month. As we discussed in the [last post](https://fhirblog.com/2016/09/30/managing-the-medication-list-in-snapp-part-1/) – this is a preferable, albeit more complex way than just using queries against MedicationStatement resources.

To understand how this works, take a look at this image:



It’s a screen dump from clinFHIR that shows how the List resource in the middle of the graph refers to the individual MedicationStatement resources that are part of the List (as well as the patient). Yes, I know, I really need to work on those colours! Oh, and BTW take a look at [this post](https://fhirblog.com/2016/09/29/supporting-the-snomed-snapp-hackathon/) to see how you can create your own sample patients on a FHIR server

What you can see is that you still need the individual MedicationStatement resources that represent each medication, but you now have an additional resource that adds the extra information about the list – e.g.

* When the list was created, and by whom.
* Type of list eg a comprehensive review of all mediations, or just updating it in the context of a specific event (an antibiotic for a fixed period)
* Medications that have been stopped in this review (you can indicate why they were stopped using an extension)
* Notes about the list

And we can be explicit about stating that the patient is not taking any medications (as far as we know), plus versioning of the list as it changes.

So lets see how we can do that.

To retrieve an existing list, you have a few options. (We assume that you have the patient Id – 149632 in this case).

You could get all the lists for a patient, then find the medications list (it is the one where the List.code value is ‘medications’ (remembering it’s a CodeableConcept):

<http://fhirtest.uhn.ca/baseDstu3/List?subject=149632>

You could get a bit smarter, and include the type of list in the query:

[http://fhirtest.uhn.ca/baseDstu3/List?subject=149632&code=http://hl7.org/fhir/list-example-use-codes|medications](http://fhirtest.uhn.ca/baseDstu3/List?subject=149632&code=http://hl7.org/fhir/list-example-use-codes%7Cmedications)

(Note that we’re being a good FHIR citizen and including the system in the query. You don’t have to, but it’s recommended. And note also that even though there’s only a single list you always get back a bundle – no surprises!) – Actually, if there multiple matching lists, they’d all be included…

Of course, these queries only return the List resource – it’s up to you to retrieve the actual contents of the list – boring! [\_include](http://hl7.org/fhir/2016Sep/search.html#include) to the rescue:

[http://fhirtest.uhn.ca/baseDstu3/List?subject=149632&code=http://hl7.org/fhir/list-example-use-codes|medications&\_include=List:item](http://fhirtest.uhn.ca/baseDstu3/List?subject=149632&code=http://hl7.org/fhir/list-example-use-codes%7Cmedications&_include=List:item)

So now – in a single call – we get the List and all the resources it refers to. Note that the syntax for \_include is based on the search parameters – not necessarily the path in the resource, and it works because this is a query – we’re not GETting a single resource.

There are even more options:

* An [operation $find](http://hl7.org/fhir/2016Sep/list-operations.html#2.31.15.1) on List
* [Current](http://hl7.org/fhir/2016Sep/lifecycle.html#lists) Lists

But as these aren’t imlemented by HAPI, we won’t discuss them further.

What if the patient doesn’t have a medication List? Well, to create a new List manually, there are 3 steps.

1. Locate the patient Id as we did in the last post.
2. Create the individual MedicationStatement resources – 1 for each medication, including the reference to the patient and save each one on the server (POST to the /MedicationStatement root). The server will return the id of the newly created resource in either the *Location* or *Content-Location* response header. (the spec is a bit ambiguous at the moment I’m afraid).
3. Create a List resource that references those resources in its entry element, set the value of the ‘code’ property to ‘medications’ (It’s a CodeableConcept actually), the patient property to the patient and any other properties you need to then POST to the server.

(In step 3, you set the ‘type’ of the list (List.code) to indicate that it’s a medication list – more details [in the spec](http://hl7.org/fhir/2016Sep/lifecycle.html#lists).)

As you can see, this is all a bit tedious. It does work, but involves a fair amount of client side work – something we try to avoid with FHIR. However there is an alternative – and that is to use the [transaction](http://hl7.org/fhir/2016Sep/http.html#transaction). The flow is similar, except that rather than sending each resource to the server, you instead place them in a bundle and send the bundle to the root of the server. The server will then save each resource, updating the references appropriately and return the completed bundle to you.

So here’s the new set of steps.

1. Locate the patient Id as we did in the last post.
2. Create a bundle, setting the type to *transaction*
3. Create a List resource and add it to the bundle
4. Create the individual MedicationStatement resources – 1 for each medication, including the reference to the patient and make up an id. It doesn’t matter what the id is, so long as it is unique in the context of the bundle as the server will replace it.
5. Add the resource to the bundle
6. Update the List resource (still in the bundle) by adding an entry.item.reference to the List that has a reference to the id you just created.
7. POST the bundle to the server root.

The server will then perform the processing and return another bundle containing the newly created resources. Much easier! This is exactly the process that [clinFHIR](http://clinfhir.com/) uses when creating the sample data – if you put a ‘sniffer’ across the HTTP connection you will see the bundles going back and forth. You can also check out the code if you want – search for the function buildMedicationList() in the [github source code](https://github.com/davidhay25/FHIRSampleCreator).

As you can imagine, this can get quite complicated, and not all servers will support it (those that do – including HAPI – will declare so in their conformance resource).

One thing to be a bit careful of in the scenario above, is that a new List (and MedicationStatement resources) will always be created. What do you do if there is already a List (of the correct type) – if you just do the above, then the patient will have 2 lists – hardly what we are after. (Actually, this is a question of data integrity – and really important. We don’t have time to discuss right now, but will return to it at some point)

There are a couple of solutions

* You could check first (using the examples above) if there is a already a list, and if so update it (more on that in a moment)
* You could use the [conditional update](http://hl7.org/fhir/2016Sep/http.html#2.42.0.10.2) mechanism on the List – specifying the patient and list type as the identification criteria.

Most of the time you’re going to want to retrieve the list first (so you know what changes to make) so let’s think about updating the list.

Updating is not really all that hard (conceptually at least).

You retrieve the current List resource and all the MedicationStatement resources (\_include is your friend). We’ll start by doing this manually

1. Make whatever changes to MedicationStatements you need – updating the status to completed (or whatever) or creating a new one. Update the server as you go.
2. Update the List resource to reflect the changes. You may add a new entry to reflect an additional medication, or remove one that is no longer current. In the latter case, you could leave it in the List, but set the deleted property to true so you can be explicit about the fact that you’ve removed it. You can also set the ‘flag’ property to give more details, or use an extension.
3. Update the List resource on the server

As with creating a new list, you can also use a transaction. You’d follow the steps above, but place the updated resources in the bundle and then send the bundle once at the end. You wouldn’t bother with resources that are unchanged (although you could if you wanted to). For changed resources, remember to use the conditional update facility of the bundle (request.ifMatch) so the original resource is updated rather than creating a new one. We discussed this in the previous post, and the principles are the same.

Retrieving the history of changes to the list is also straightforward.

First, you’ll need the id of the List – perhaps by:

[http://fhirtest.uhn.ca/baseDstu3/List?subject=149632&code=http://hl7.org/fhir/list-example-use-codes|medications&\_include=List:item](http://fhirtest.uhn.ca/baseDstu3/List?subject=149632&code=http://hl7.org/fhir/list-example-use-codes%7Cmedications&_include=List:item)

Next, you can get the history of changes to the List:

<http://fhirtest.uhn.ca/baseDstu3/List/149655/_history>

This will give you a bundle containing all the previous versions of the List with that id.

Then you can GET the version you want:

<http://fhirtest.uhn.ca/baseDstu3/List/149655/_history/2>

Unfortunately, \_include won’t work when you’re retrieving a specific version of a resource (it’s not a query) so you’ll need to retrieve each MedicationStatement separately. Oh well, you can’t have everything…

The last thing we’ll discuss is how to absolutely assert that the patient is not on any medications. This is done by simply creating a List resource and setting the emptyReason property.

So that should be enough to get going. The purpose of these 2 posts is to support [SNapp](http://www.snomedexpo.org/snapp) – so for those participating (or anyone else for that matter) feel free to reach out with any questions. I’m not planning any further posts, unless reasons arise why I should do so…

But – as mentioned in the previous post – we have been completely ignoring the security and privacy considerations in these discussions, as well as integrity issues – e.g. multiple medication lists. They are worth discussing – and we’ll do so in the not too distant future.

Oh – and just to be clear – the url’s in this post are correct at the time of writing (ie the patient and list exist on the HAPI server. This will not always be the case, and sooner or later the resources will be removed and the URL’s will break. But hopefully you can figure out how to correct them (Hint: find the patient id, luke…). Sorry.

2.45 Managing the Medication List in SNapp, part 1.

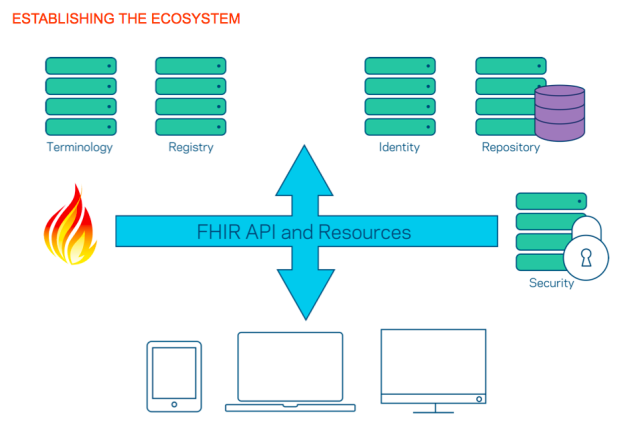
SEPTEMBER 30, 2016 [2 COMMENTS](https://fhirblog.com/2016/09/30/managing-the-medication-list-in-snapp-part-1/#comments)

So, in the [last post](https://fhirblog.com/2016/09/29/supporting-the-snomed-snapp-hackathon/) we talked about setting up a FHIR server, and populating it with some sample data. Let’s turn our attention now to how we would interact with that server in the context of a ‘Medication List’, and we’ll use the [STU-3 release candidate](http://hl7.org/fhir/2016Sep/index.html) (as used at the recent Baltimore connectathon) as the basis of the discussion, though this should work just as well in older versions.

*(Our discussion will generally apply to other lists – such as Allergies and Problem lists – though it’s not uncommon for there to be multiple Problem Lists for a patient that are ‘clinician focussed’ e.g. the list of significant problems from the perspective of a respiratory physician tends to be different to that from the perspective of an orthopedic surgeon. But I digress…)*

First, lets think about the overall architecture of the solution that we’ll be working in. As this is in support of the SNOMED SNapp, we’re really talking about the proposed ‘NZ-EHR’ – a single national source of health related data accessed freely by all that have the rights to do so. (For the purposes of this discussion we won’t consider security aspects – which in practice will be a biggie of course. Similarly we won’t consider privacy – we’ll assume that a caller has access to the entire list – again, not something likely in real life).

Here’s a picture that shows the overall ecosystem (Taken from an [Orion Health](http://orionhealth.com/) presentation)



The repository is in the upper right hand corner. Systems at the bottom of the picture will access and update the list as required.

So how do we represent the List of medications? We’ve discussed that before a [number of times](https://fhirblog.com/?s=list) – and most of the comments are still reasonably true, so let’s make this post more of a practical exercise.

There are quite a few different ways that you can represent a medication list in FHIR. This is deliberate as it is intended to be useable in a wide variety of contexts – and countries – and so needs to be adapted to many different environments. So we’ll mention the options, but at the same time call out the ones that would seem to make the most sense in the New Zealand context – and ones that participants at the SNApp event might choose to use.

*(Though, as another side line, we’re a bit early in the process to be making firm recommendations . As we get (hopefully) closer to the reality of a single EHR in New Zealand we might want to hold one or more connectathons to test out the various possibilities before coming up with a firm proposal – and an*[*Implementation Guide*](http://hl7.org/fhir/2016Sep/implementationguide.html)*that describes it)*

First, which resource will we use to represent an item (medication) in the list? There are a number of [medication related resources](http://hl7.org/fhir/2016Sep/medications-module.html), but we’ll settle on the [MedicationStatement](http://hl7.org/fhir/2016Sep/medicationstatement.html) as the one that we’ll use. Of course applications that are updating the list may get data from other sources like MedicationOrder or MedicationDispense, but in updating the list they can easily create a MedicationStatement, so let’s go with that.

The key properties of the resource for our use include:

* Medication – the actual drug we’re referring to. This can either be a [CodeableConcept](http://hl7.org/fhir/2016Sep/datatypes.html#codeableconcept) (to an underlying terminology) or a reference to a [Medication](http://hl7.org/fhir/2016Sep/medication.html) resource if we need to record more details about it. Commonly this will be a [contained](https://fhirblog.com/2013/10/10/fhir-contained-resources/) resource, though there are other options.
* Dosage information
* The period over which the statement applies. This is useful to distinguish between ‘regular’ medications like anti-hypertensive medications and ‘short term’ medications such as a course of antibiotics.
* Why the medication is being taken
* General notes

Next – how to represent the list? Well there are a couple of options.

The one we’ve talked about most in the past is using the [List](http://hl7.org/fhir/2016Sep/list.html) Resource. This has a number of advantages:

* We can indicate when the list was created, and by whom.
* We can be explicit about medications that have been discontinued (Using an extension to indicate why)
* We can be explicit that the patient is not taking any medications.
* We can have ‘types’ of list – e.g. following a comprehensive review of all mediations, or just updating it in the context of a specific event (an antibiotic for a fixed period)
* We can add notes to the list
* We can easily support different versions of the list

But the primary disadvantage of the List is that it is more complex to implement – and to update – compared to the alternative option of just querying for ‘active’ MedicationStatement resources (where ‘active’ is based both on the status and the period).

So, for now, we’ll support both, but we’ll probably want to think about this a bit more in the future (and test it at connectathon!).

So – on to more practical stuff. For the examples below, I created a test patient against the public HAPI STU-3 server. But do be aware that the actual resources may be cleared when the server is re-set, however as described in the [previous post](https://fhirblog.com/2016/09/29/supporting-the-snomed-snapp-hackathon/), creating another test patient is trivial.

First up, we’ll need the patient id. (We can do without it, but let’s keep it simple for now).

The name of the patient I created was Nina Thomas, so the following query gave me all the patients with a name of ‘Thomas’, from which I found the id I wanted – 149632.

<http://fhirtest.uhn.ca/baseDstu3/Patient?name=Thomas&_count=50>

(As an aside, the search parameters I could have used are documented [here](http://hl7.org/fhir/2016Sep/patient.html#search) – I think HAPI supports all of them). Note also that I’ve included the \_count parameter. By default HAPI will only return 10 matches, with paging links if there are more than 10. I’m lazy, so I prefer not to have to traverse the pages – but you should watch out for the [paging links](http://hl7.org/fhir/2016Sep/http.html#paging) and take appropriate action if they are present.

So I could retrieve the Patient resource directly like so:

<http://fhirtest.uhn.ca/baseDstu3/Patient/149632>

and, of course, I could use the ‘Accept’ header to indicate the format I wanted – *application/fhir+xml* or *application/fhir+json*. (I find json easier to deal with as I’m a javascript developer – but either is just fine).

Now we have the patient Id we can retrieve the list.

**Let’s start by assuming that we’re not using the List resource – but simply working with MedicationStatement resources.**

First, let’s just get the current MedicationStatement resources for the patient.

<http://fhirtest.uhn.ca/baseDstu3/MedicationStatement?patient=149632&status=active&_count=50>

This will return a bundle of MedicationStatement resources (up to 50) that are active. I figured out the correct query parameters from [the spec](http://hl7.org/fhir/2016Sep/medicationstatement.html#search). Note that I still need to examine the *effctivePeriod* property to be sure that the MedicationStatement is still current (It could be active, but expired for example).

What if I want to update the ‘list’? Maybe remove a medication that the patient is no longer taking?

Well, I could always [delete](http://hl7.org/fhir/2016Sep/http.html#delete) a MedicationStatement that is no longer true. HAPI will keep a [history](http://hl7.org/fhir/2016Sep/http.html#history) of the resource so I can always get it back, but it’s not a very tidy solution. Instead I’ll update the status. Looking at the spec, I note there is a status of ‘*completed*’, which sounds like the one I want ([Here’s](http://hl7.org/fhir/2016Sep/valueset-medication-statement-status.html) a complete list – It has a binding strength of [*required*](http://hl7.org/fhir/2016Sep/terminologies.html#required), so I must use one of these ones).

So the process will be:

1. Retrieve the MedicationStatement I want to update
2. Change the status property
3. Update it on the server.

Retrieving the MedicationStatement is straightforward – it’s a simple GET:

<http://fhirtest.uhn.ca/baseDstu3/MedicationStatement/149666>

Changing the status to ‘*completed*’ is simple, and then I just PUT it back using the same query but with a method of PUT, and the resource in the body of the call. You’ll also need to set the Content-Type header.

Now when I query for active MedicationStatements for that patient, this one will not be in the list.

But hold on,  what if someone else updates the MedicationStatement between the time I retrieve it and the time I update it? Won’t this overwrite their changes?

Well, yes it will. The server will keep a version (providing it supports versioning that is) but that can be cold comfort. However, the specification does provide a mechanism for managing this [resource contention](http://hl7.org/fhir/2016Sep/http.html#concurrency) by using ETags and the ‘If-Match’ header (kind of like [Optimistic Locking](https://en.wikipedia.org/wiki/Optimistic_concurrency_control)). It’s up to the server to decide whether to support this – HAPI doesn’t as it is a dev server, so we won’t bother with it, but it’s good to know it’s there.

This complexity also explains why some vendors won’t support updating in this way – there are often significant business processes around updating medication lists. If a server doesn’t allow this kind of update, it can declare so in its [conformance resource](http://hl7.org/fhir/2016Sep/conformance.html#5.3.1.1).

Creating a MedicationStatement is reasonably straightforward.

1. Create the resource
2. Set the ‘patient.reference’ property to the patient id
3. POST to the type root (eg <http://fhirtest.uhn.ca/baseDstu3/MedicationStatement>)

The server will assign an id to it, and create the link to the patient. After that, performing the search query will include the new resource.

So that’s what’s involved in updating the list assuming a simple collection of MedicationStatement resources. However, as mentioned above, the List resource offers significant advantages at the cost of some complexity.

We’ll consider that in the next post.

2.46 Supporting the SNOMED SNAPP hackathon

SEPTEMBER 29, 2016 [7 COMMENTS](https://fhirblog.com/2016/09/29/supporting-the-snomed-snapp-hackathon/#comments)

So there’s a ‘hackathon’ (I actually don’t like that word) associated with the [SNOMED conference in Wellington next month](http://www.snomedexpo.org/). They’re calling it a [‘SNAPP’](https://confluence.ihtsdotools.org/snapp) event (SNOMED App) – which is much nicer!

As part of supporting this event, HL7 New Zealand is going to stand up a server that can be used for the ‘My List of Medicines’ challenge. (The other challenge – Machine Mapping – is a bit esoteric for me).

So there are a couple of things we need to do.

* Stand up a FHIR server for participants to use that can respond to appropriate requests (we’ll talk about what they will be in the next post).
* Populate that server with the sample data given on the [web site](https://confluence.ihtsdotools.org/snapp/resources/my-list-of-medicines-dataset)

So the first step is straight forward. We’ll get a server in the cloud (I use [Digital Ocean](https://www.digitalocean.com/) myself) and install the [HAPI CLI server](http://hapifhir.io/doc_cli.html) on it.

To do this, download the zip file for the server (there’s a link on the [documentation](http://hapifhir.io/doc_cli.html) page) and expand it into a folder. To run it, the following command from the command line works for me:

java -jar hapi-fhir-cli.jar run-server --allow-external-refs -f dstu3

This will start the server in STU3 mode (or the release candidate), allowing StructureDefinitions that refer to external urls. I usually create a small shell script to make this easier to run.

Next task is to install the various conformance resources (StructureDefinitions for the base resource types and ValueSets). To do this, leave the server running and open a new window. Enter the following command.

java -jar hapi-fhir-cli.jar upload-definitions -f dstu2 -t http://localhost:8080/baseDstu2

All going well, you should now have a fully featured FHIR server populated with the base conformance resources. Excellent! (And it was free!)

(By the way – you can also set up your new server to be a terminology server with a complete installation of SNOMED – we won’t do that now, but maybe a topic for another post.)

Now to populate the server with the sample data. After some thought, I decided to update clinFHIR so that when you create a new patient and ask for sample resources to be created, it will add the ones from the SNAPP site – demo data is always helpful. So all you need to do is to tell clinFHIR about your new server, set it as the data server, and create a new patient with examples. Here are the steps:

Start [clinFHIR](http://clinfhir.com/).

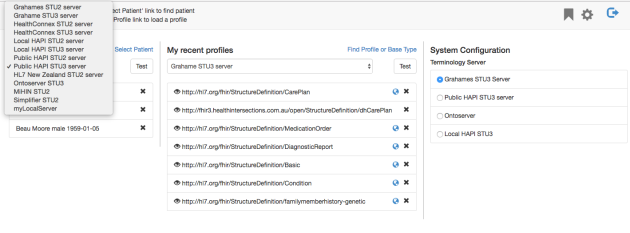
From the ‘gear’ menu at the upper right, select the ‘add server’ option. Enter a name , the location of the server and the FHIR version as shown in the following screen shot:



(Note the trailing ‘/’ in the url – that’s important).

Click the ‘test’ button that appeared on the lower right. This will attempt to retrieve the conformance resource from that server. If it succeeds, then you’ll get a message and an ‘add’ button will replace the ‘test’ button. Clicking that will add the new server to list of servers that can be accessed by this browser. (The definition is saved in the local browser cache right now – I’ll probably update that at some time to be associated with a logged in user).

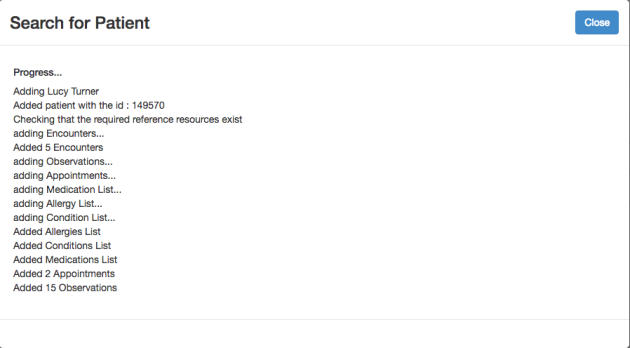
Now you can dropdown the list of servers in the ‘my recent patients’ and be able to select the new server (It’s at the bottom of the list).



To create a patient, click the ‘Select Patient’ link, and in the dialog that is displayed select the ‘Add new Patient’ link to the lower left.

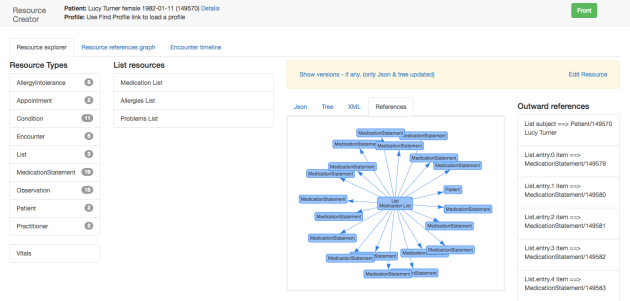
This will change to a screen where there is some sample data (retrieved from an on-line service) – you can just accept that or add your own demographics. Leave the ‘Generate Samples’ checkbox checked. Now click ‘Save’.

After a while (maybe 5-10 seconds) you’ll get a display like the following.



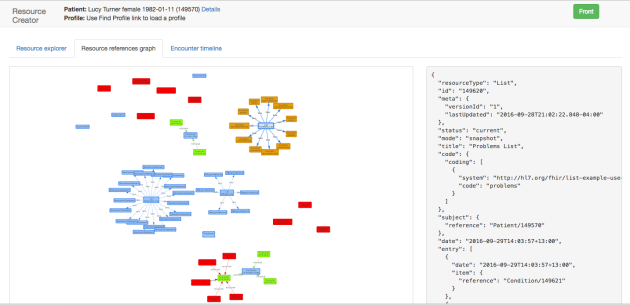
Click ‘close’ to return to the front page.

You can now view the resources that have been created for the patient. Click the ‘Details’ link (upper left) alongside the patient name, and the following screen is displayed:



(I’ve selected the List resource type, then the Medication List instance, and finally the ‘References’ tab to get a graphical view of the Medication List that was created.) I’ve [blogged before](https://fhirblog.com/category/clinfhir/) about this stuff so won’t go into that here (though do note that some of the older posts may no longer be correct –clinFHIR changes rapidly!)

Before we go – click on the ‘Resource References Graph’



This will show all the resources for this patient (excluding the Patient). You can zoom and move the display to show the resources you want (remember that these are resource instances – real resources created for a patient).

So that’s all for now – we’ve set up a FHIR server and loaded it with sample data from the SNAPP site.

In the next post we’ll take a look at what kind of queries we can use when we’re ‘SNAPPing’ with SNOMED (sorry, couldn’t help myself)…

And if you’re a guru with colour – reach out (as you can see, I need help with that  )

2.47 Baltimore Working Group Meeting

SEPTEMBER 25, 2016 [2 COMMENTS](https://fhirblog.com/2016/09/25/baltimore-working-group-meeting/#comments)

Well, the Baltimore Working Group Meeting is finished – and what a time it was!

For me the WGM was all about Connectathons – one way or another.

Saturday and Sunday was the technical Connectathon where we have different streams that exercise various aspects of the specification. We had around 180 participants – which is a record number, but we coped pretty well I thought – and many thanks to the HL7 staff who work so hard behind the scenes to make it all work.

There were the usual issues – for example we had a problem accessing ‘gotomeeting’ during the Sunday presentations which meant that Ewout and I were scrambling about with microphones and computer connections to make it all work, but in the end it worked out OK (And there’s still a task on me to upload all the presentations – soon, I promise!)

On the Tuesday in the General Session I was surprised – and honored – to be called up to the front of the room and [receive thanks](https://fhirblog.com/2016/09/25/baltimore-working-group-meeting/David+Hay) for my work on [clinFHIR](http://clinfhir.com/), along with a nice bottle of wine and a signed copy of Grahames book. It’s something I do gladly to help the community, both the designers of resources and clinicians new to FHIR (an increasingly important audience I believe) and the effort pales in comparison with what other team members do –especially [Grahame](https://www.linkedin.com/in/grahame-grieve-952637), [Lloyd](https://www.linkedin.com/in/lloyd-mckenzie-6b6681), [Ewout](https://www.linkedin.com/in/ewoutkramer) and [Josh](https://www.linkedin.com/in/joshua-mandel-88347235) – but I sure didn’t refuse the wine! (And it explained why a colleague was so insistent on a breakfast meeting that morning . Turned out he was in cahoots with Grahame to make sure I was there! You can remain anonymous, but you know who you are [Peter](https://www.linkedin.com/in/peter-jordan-44146418)!)

Actually, it was a good meeting for the FHIR community – both [Brian](https://www.linkedin.com/in/brian-postlethwaite-91b9414) & [Claude](https://www.linkedin.com/in/claudenanjo) received the Volunteer of the Year award – well deserved.

Then on Friday we had the Clinicians on FHIR event, where the clinical folk use clinFHIR to ensure that FHIR will meet the clinical needs of the community by using the tool to create resources representing various clinical scenarios. For me, this is at the same time the most interesting and the most stressful event of the meeting, as it completely relies on clinFHIR so any bugs are painfully evident! Much of the week (especially evenings and early mornings) is spent enhancing clinFHIR and swatting bugs, so Friday is an anxious time!

And, surprisingly, not everyone uses it in the exact same way that I do – how strange…

But it’s a wonderful experience – and one that I’m really pleased to be able to attend.

And a reminder to all that at the [devDays in November](http://fhir.furore.com/devdays) we’re going to have a clinical stream aimed at Clinical and Business Analyst folk who want to learn more about FHIR. We’re still working on the details, but it will be a combination of tutorials and practical exercises in the aspects of FHIR most important to clinicians – like profiling – so I urge you to come along if at all possible.

So now I can relax – for a little while…

2.48 FHIR and Version 2: the Location

SEPTEMBER 20, 2016 [2 COMMENTS](https://fhirblog.com/2016/09/20/fhir-and-version-2-the-location/#comments)

So last week  [Simone Heckman](https://www.linkedin.com/in/simone-heckmann-16a96179)  (CTO of [Gefyra GmbH](http://www.gefyra.de/)) wrote a piece in the FHIR chat in response to a query about mapping data from an HL7 v2 message into FHIR – specifically about representing the location of a patient as shown in the PV1 segment of the message into FHIR resources. It was a great response so I asked her if I could repeat it here and she was kind enough to not only say yes, but also beef it up a bit!

Over to you Simone…

It’s all about Location, Location, Location!

Even though the design of FHIR has always acknowledged the requirement for compatibility to other Standards from the HL7 family, mapping HL7 Version 2 to FHIR (though pretty straightforward at first sight) poses some challenges, when going into detail.

**PID-Segment maps to Patient resource**

**OBX-Segment maps to Observation resource**

**PV1-Segment maps to Encounter resource**

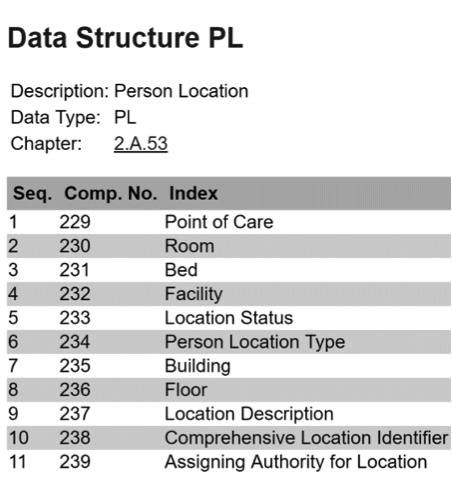
so far, so obvious.

A couple of days ago, someone asked in the FHIR implementer’s chat about how to map PV1.#3 to FHIR. And the answer to that question is not as simple.

Quick explanation for all readers who are not deeply familiar with Version2:

PV1.#3 refers to the 3rd field of the Patient Visit segment (PV1), which is used to convey the current location of the patient.

The data type of this field is “PL”, a list of up to 11 subfields describing different attributes of the patient location. Here’s what it looks like:



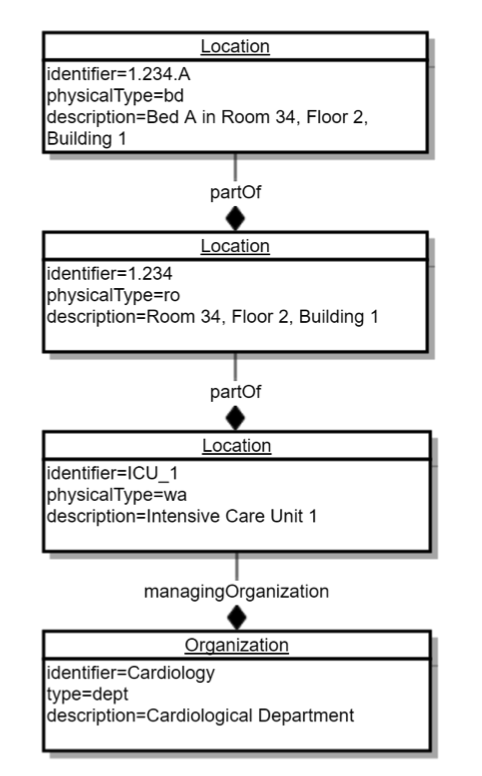
However, what’s usually being communicated by V2 systems mostly boils down to something like this:

PV1|I||**ICU\_1^1.234^1.234A^CARD**|…

The example message above holds information about the ward (ICU\_1), the room (1.234 ) and bedplace (1.234A) the patient has been assigned to and as well as the department he’s associated with (CARD).

When translating V2-PV1 into FHIR-Encounter, suddenly there’s no longer a 1:1 relationship between segment and resource, since FHIR considers all physical locations and parts thereof as individual resources, that may have a hierarchical structure among them. Also, FHIR differentiates between physical locations (room, bed, ward) and their managing organizations (department).

So the FHIR representation of the PV1.#3 content in the example looks something like this:



It has to be noted though, that there is not always a 1:1 relationship between a ward and a department. Wards are sometimes interdisciplinarily shared between departments (especially ICUs), so the association between a case and a department has to be a direct reference, rather than an indirect one via location.

The obvious question when translating V2 to FHIR is: “*Do I really need to create all of these resources and references just to convey that tiny bit of information in PV1.#3***?”**

Well, probably not.

It really depends on your use case. Or rather on:

***“How much does your FHIR server care about Locations?“***

**Scenario 1: it cares very much**

If the server is supporting a location aware application (e.g. patient transportation), creating Location resources from your V2 client doesn’t make much sense. Keep in mind, that all you know about the Location is its identifier ( eg “1.234A”). In this case, the locations relevant to the server, will probably have been created manually or imported from another data source to include information like building, floor, wing, accessibility and maybe even geo coordinates, and the servers job is to find the correct one to reference.

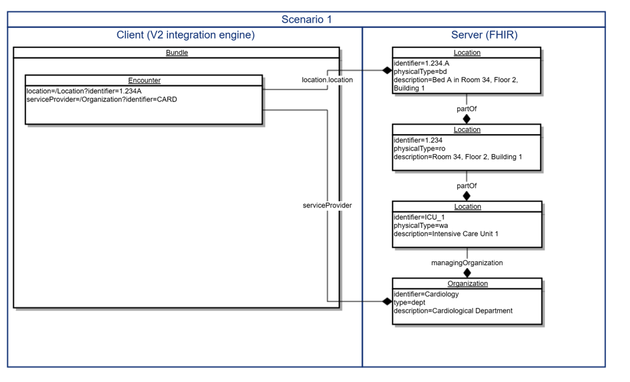
Under the assumption that the server is already aware of all the locations it serves, the challenge for the V2 mapping is to reference the proper instance. Without knowing the server assigned url (based on the resource Id) for the bed place “1.234A”, the integration engine can either search the server for a location that fits the identifier and then reference Encounter.location.location to the corresponding url or – much easier – use a c*onditional reference* (a recent [enhancement](http://hl7.org/fhir/2016Sep/http.html#2.42.0.16.2) to the STU3 specification). Conditional reference moves the responsibility for searching for a matching resource and retrieving the proper url from the client to the server.

When using conditional references, instead of a url, the client provides a matching criteria, in our bedplace example: Encounter.location.location=/Encounter?identifier=1.234A

It has to be noted however, that conditional references will fail, if the server finds either more or less than one resource that matches the criteria. So this mechanism will only work if the strings provided in PV1.#3 constitute unique identifiers for the respective locations and the server uses the same identifiers as the V2 system.

Since all other information (in which room is the bedplace located, which ward does the room belong to) are already known to the server, they do not need to be included in the mapping. It will suffice to provide a reference to the lowest node in the hierarchy.

If the information of which organization is responsible for the encounter is also required by the server, it is encouraged to provide this information in the Encounter.serviceProvider reference. As noted above, wards are often not statically tied to just one department.



**Scenario 2: It cares very little**

If the FHIR application has no need for detailed patient location information (e.g. a lab system), the information required can often be reduced to department and ward, since these attributes are often used for filtering, statistics or billing, whereas room and bedplace are irrelevant.

Also, the server will not need to know any details about the locations other than their designation.

Such servers will probably not be prepopulated with Location resources but rather add them at runtime. In this case, any Bundle created from the V2 message should at least include the required resources for ward (Location) and department (Organization) and reference them from the Encounter resource by their locally assigned uuid.

Whether PUT or POST is the http method of choice, depends on whether the FHIR system may update or change these resources and which of the system is considered to be the “source of truth” for the location information.

If users of the FHIR system are allowed to edit Location resources and enter additional information, http PUT from the V2 system will perpetually override their changes and reset the Resources to what the V2 system sends.

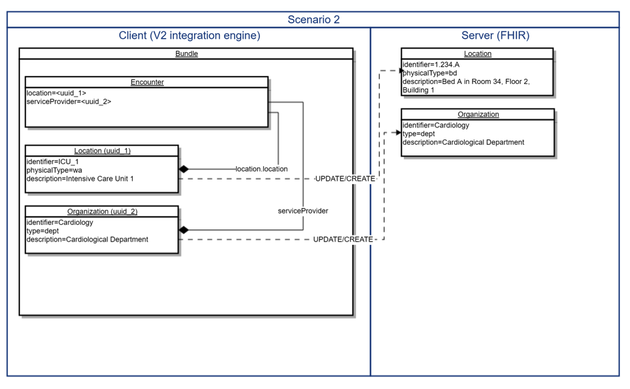
In this case, to maintain the changes on the FHIR server, a conditional create would be the method of choice. With this mechanism, the FHIR server will only create a new Resource if the condition does not match. If a match is found, no update will be performed.

If the sending system however is considered the “master” of all location information, and overriding changes on the FHIR server is explicitly desired to keep the systems in sync, a conditional update is the better choice. This mechanism will update the matching resource or create a new one, if no match is found.

Of course, there is a large grayscale between scenario 1 and 2 and many use cases may be situated somewhere in between. However, FHIR has all the mechanisms required to cover that area.

And as always, the words of Grahame the Wise are true: “Complexity has to go somewhere, you can’t make it go away.”

Scenario 1 represents the classic FHIR approach, shifting the complexity of managing locations to the server side. But especially in the early days of FHIR adoption, the V2 system may represent a complex EHR system, while the FHIR server my just be a very simple FHIR implementation to deliver patient demographics to FHIR based devices or apps (like the FHIR equivalent of a DICOM worklist). The interaction between EHR and FHIR server will then presumably rather resemble scenario 2 with the complexity scale tipping to the client side. Since this particular client however is not a typical FHIR app but rather an agent acting on behalf of the EHR, the general idea of moving complexity towards the bigger fish in the pond remains intact.



Read more on

* [Conditional Reference](http://hl7-fhir.github.io/http.html#2.42.0.16.2)s
* [Conditional Update](http://hl7.org/implement/standards/fhir/http.html#2.1.0.10.2)
* [Conditional Create](http://hl7.org/implement/standards/fhir/http.html#ccreate)
* [Resolving references in Bundles](http://hl7-fhir.github.io/bundle.html#references)

(note that some of these links are to the ‘work in progress’ version of FHIR. When the STU-3 version is finalized, they will be in a similar location (sorry!) but not necessarily the same.

2.49 Terminology services

SEPTEMBER 16, 2016 [LEAVE A COMMENT](https://fhirblog.com/2016/09/16/terminology-services/#respond)

Just a short post to call peoples attention to terminology services in FHIR. [Rob Hausam](https://www.linkedin.com/in/robert-hausam-a273aa7) (who co-ordinates the [Terminology track](http://wiki.hl7.org/index.php?title=201609_Terminology_Services_Track_Proposal) at the connectathon) made the excellent comment in the [Zulip chat](https://chat.fhir.org/#narrow/stream/terminology) that terminology is a ‘cross-cutting’ concern that can be integrated into many different applications – and tracks at [connectathon](http://wiki.hl7.org/index.php?title=FHIR_Connectathon_13).

There are a number of terminology services defined – [this link](http://hl7.org/fhir/2016Sep/terminology-service.html) (and the others in this post) is to the STU-3 candidate version that we’ll be testing at Baltimore so go take a look to see what you can use. Once the spec passes ballot it will be at the http://hl7.org/fhir/index.html location

As an example of using them, [clinFHIR](http://clinfhir.com/) uses a couple of these services – let’s take a look at the [ValueSet expansion](http://hl7.org/fhir/2016Sep/terminology-service.html#4.6.3) service. This service ‘expands’ a ValueSet to return all the concepts within it. This is needed because a ValueSet will generally contain ‘rules’ defining the contents rather than listing all the options.

More usefully, when calling the service you can define a filter that will restrict the returned to only those whose description matches the filter – really useful for ‘auto-complete’ lists – which is what clinFHIR uses it for.

There are a number of different ways that this service can be invoked (and this can vary between servers). Regardless of the method you use, you will need the URL (which is globally unique) of the ValueSet you are interested in. To do this, take a look at the resource definition in the spec. A coded element will have a link to the ValueSet description in the ‘description & Constraints column. [Here’s the link](http://hl7.org/fhir/2016Sep/valueset-condition-code.html) that I found for condition.code – you can see that the URL for this ValueSet is <http://hl7.org/fhir/ValueSet/condition-code>. The process is different for profiles of course – but that’s a discussion for another day.

Now that you have the URL, the simplest way to call the service is to call it with the URL in an identifier parameter, plus the filter. Strictly speaking the filter parameter is optional – but you should use it for anything other than small ValueSets.

Here’s a sample call:

http://fhir3.healthintersections.com.au/open/ValueSet/$expand?identifier=http://hl7.org/fhir/ValueSet/condition-code&filter=asthma

All going well, you’ll get back the ValueSet resource with the ‘expansion’ node populated with the matching concepts – these can then be used for your User Interface.

But if the server doesn’t support this syntax then you need to find actual resource id on the server and call it slightly differently.

First: query the server for the ValueSet based on the URL. For example:

<http://fhir3.healthintersections.com.au/open/ValueSet?url=http://hl7.org/fhir/ValueSet/condition-code>

This will return a Bundle of ValueSet resources (it’s a query remember) with – hopefully – only a single resource. You can then get the id of the ValueSet – in this case it is ‘*condition-code*’

Now we can perform the expansion using the alternative syntax:

<http://fhir3.healthintersections.com.au/open/ValueSet/condition-code/$expand?filter=asthma>

The result is the same as the first one – a ValueSet with the expansion node populated.

So there you are – it doesn’t get much simpler than that!

2.50 Projects in clinFHIR

AUGUST 23, 2016 [LEAVE A COMMENT](https://fhirblog.com/2016/08/23/projects-in-clinfhir/#respond)

One of the things that the Resource Builder in [clinFHIR](http://clinfhir.com/) does is to allow you to use different servers for the various ‘[roles](https://fhirblog.com/2015/09/07/server-roles-in-fhir/)‘ that are needed. These include:

* The Conformance server – holds resources like StructureDefinition and NamingSystem
* The Terminology server – ValueSets and the Terminology operations required by the different applications within clinFHIR
* The Data server – where patient (and other) data is kept.

This isn’t an official part of the spec, but is a useful way to think about the main ‘components’ of a FHIR solution.

While flexible, this can make it somewhat more complicated to use, as you need to select the right combination of servers to suit a particular need.

To make this a bit easier, I’ve added the notion of ‘projects’ to clinFHIR. The purpose of a project is to allow you to select a combination of servers, profiles and sample patients and then make it easier for you to share that combination with others.

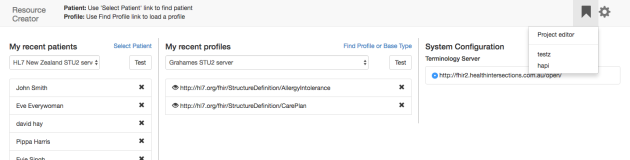
Indeed, part of the motivation for this is for my colleagues in the UK who are going to use clinFHIR to help them visualize profiles that are created by [forge](http://fhir.furore.com/Forge) and stored in [simplifier](https://simplifier.net/). By creating a project that ‘wraps up’ the profiles, the servers where they are stored and some sample patients it should make it easier when they get together to decide what profiles work best in the UK.

Another motivation was the need to support the ‘clinicians on FHIR’ event that we hold at each Working Group Meeting  – commonly the participants will prepare samples and profiles in advance, so I was imagining that projects could help help group those artifacts together.

A caveat before I start – this is quite basic functionality (I wrote it over the weekend) so please bear that in mind when you use it. In particular all the projects are world visible and (to some extent) world editable. I’ll add better security if it’s needed.

To create or use a project, there’s a new bookmark icon in navbar at the top right.  Clicking on that icon will display a menu dropdown with the currently defined projects and a link to the editor (so we’re going to have trouble if people create too many!)

The image below shows the menu selected:



If you click on the ‘project editor’ link you’ll see a screen where projects can be created, edited and removed (and if you do create a ‘test’ project – please remove it when you are done). Clicking on the ‘new’ link allows you to enter some simple information about the project:

* The name (this will appear in the menu, so keep it short)
* A description (this will appear when you mouse-over the item in the menu – make it reasonably complete – but not too much)…

Note that the servers are automatically selected – they are the ones that were selected when the screen was invoked, *so make sure you select them prior to creating a new the project*.

When finished click the save button (which will only appear after there is a name and a description) and then the ‘front page’ button to the upper right. The front screen will be re-displayed and when you display the project list, your new project will appear. (You do need to select it from the menu – even after creating it).

You can also use the project editor to view existing projects – it will show the profiles and patients that have been added to it (we’ll see how to do that in a minute).

To actually use a project, select it from the project list. The front screen will then:

* display a banner at the top to indicate that a project is active
* change the servers to the ones from the project and deactivate them (they can’t be changed while a project is active)
* display the patients and profiles added to the project (for a new project they will be empty).

In the banner at the top is the name of the project and a link to the right that when clicked, will toggle clinFHIR into a mode where any changes you make to the lists of profiles and patients will be reflected in the project – ie selecting a profile will automatically add it to the project.

So all you need to do is to click the link, and then set up the project the way you want it. The purpose is to make sure that updating a project is a deliberate – rather than an accidental – act.

A couple of other minor changes I made to clinFHIR help support the project concept:

* Patients and Profiles can be removed from the project by clicking the ‘x’ to the right side of the name
* When selecting a profile, you can now select a profile from a different server than the currently selected conformance server. You need to select the server and the id of the profile on that server. This was added so that you can keep the conformance server set to – say – Grahames or the HAPI server which have all the standard profiles and valuesets needed by the rest of clinFHIR, but select a profile on – say – simplifier (which the UK folk are using)

So there you go. Hopefully this will make it easier to set up and share specific scenarios in clinFHIR. Let me know what you think – and any bugs or enhancements!

And just a reminder that currently all projects are visible – and editable – by anyone. If there are large numbers of projects then right now it will become unwieldy. I’ll improve functionality to manage this – likely a way to users to ‘subscribe’ to specific projects rather that the current ‘free for all’ – but for now, just be aware of this and thoughtful of others.

2.51 What is SMART and why should you care.

JULY 8, 2016 [10 COMMENTS](https://fhirblog.com/2016/07/08/what-is-smart-and-why-should-you-care/#comments)

This was actually a summary of [SMART](http://docs.smarthealthit.org/) that was intended to wrap up [a series](https://fhirblog.com/category/smart-2/)I wrote a few months back, but I forgot to post it! So, better late than never…

Introduction

Most people who are active in the healthcare interoperability space will have heard of [FHIR](http://hl7.org/fhir/index.html) – the ‘next generation’ interoperability standard from HL7 that is implementer focused – reusing many of the concepts already familiar to developers from other domains such as Resources to represent common healthcare concepts such as Allergies, Medications and Problems and a simple REST based API made popular by some of the major internet players such as Google, Twitter and Facebook.

FHIR has the support of many of the larger vendors – Epic, Cerner, Allscripts and Orion Health as well as national bodies such as ONC in the US, the NHI in the UK, HINZin New Zealand and NEHTA in Australia.

But there’s a new acronym making the rounds at the moment – SMART (standing for Substitutable Medical Applications and Reusable Technologies) that is also generating a lot of interest. It isn’t yet as wide spread as FHIR, but the same organizations are taking a close look at it.

*Put simply, SMART adds a layer of security in front of FHIR interfaces by leveraging OAuth2 for Authentication and Authorization, OpenID Connect for user Identity and standardizing the process of negotiating access to information and operations between client and server. It also describes a process by which an EHR application can launch an external application preserving context, and providing safe access to the data within the EHR.*

Details

SMART was originally proposed as a standard to enable specialist external applications (applets) to interact securely with the data in an EMR system. The vision was that by standardizing the communication between the two it would be possible to create an ‘ecosystem’ of apps and data sources in much the same way as mobile applications can interact with on-device resources such as contacts lists and camera.

When first developed it used its own definitions for content and API, but once FHIR gained traction in the market place it switched to use FHIR for both of those.

The problem that it was trying to solve is that there is a great deal of clinical information that is collected within EMR systems that is only available to the EMR vendor. Any application development that used that data could only be done by the application vendor, reducing the possibility of developing specialized and innovative applications. By splitting the roles of data consumer and data supplier, it becomes possible for vendors to develop specialist applications that can be run against different data sources – something called [‘sidecar’ applications](https://fhirblog.com/2015/07/10/fhir-smart-and-sidecar-applications/)by Gartner analyst Wes Richell.

However, since originally described, it has become a more general standard for securing access to FHIR resources.

SMART focuses on 3 key areas – leveraging other standards to do so.

* Providing a way for a client to identify themselves and to negotiate access to the data in the EHR. This leverages the [OAuth2](http://oauth.net/2/) standard.
* Representing the information that is being exchanged, and the manner in which it can be accessed. This uses [FHIR Resources](http://hl7.org/fhir/resourcelist.html) (specifically utilizing FHIR profiles for the details) for the content, and the [FHIR REST API](http://hl7.org/fhir/http.html) as the query mechanism.
* Providing a mechanism by which an external app can be ‘launched’ from an EHR – preserving the context (current patient and user).

Identifying the client & negotiating access

There are a number of ‘roles’ that are important in understanding SMART (these come from the underlying OAuth2).

* The *Resource provider* is the supplier of the data – the system that wishes to make the data available via FHIR interfaces to a client
* The *Authorization Server* is the component that identifies and authorizes both the client and the user wishing to access the data. It can be part of the same application as the Resource Provider – but it doesn’t have to be.
* The *App* is the actual application that is requesting the information (eg a smartphone app)
* The *User* is the person who is using the app

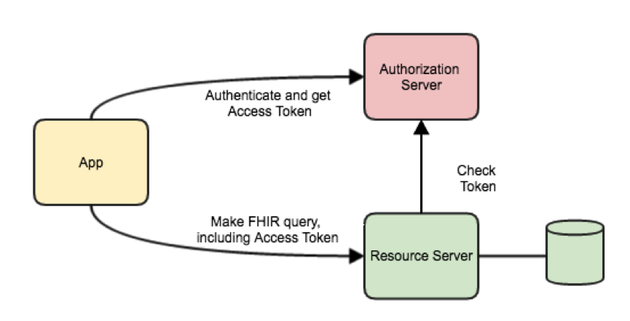
While the implementation details are complex (and can vary from site to site even within the SMART standard) from a high level it’s quite straightforward.

A pre-requisite is that the app must be registered with the Authorization Server, which supplies it with an identification key.

When the user uses the app to request some data:

1. The Authorization Server checks that it knows the app (the app has to supply the key that identifies it). This allows the Authorization Server to customize its behavior based on the capabilities of the app.
2. The Authorization Server then identifies the user of the app. This can be by requesting a Username/Password, or some other mechanism that the Authorization Server trusts.
3. Next the Authorization Server determines what data, and what functions on that data (read, insert, update, delete) the user is allowed. Again, the actual details of how that is done varies according to the implementation. This is expressed as the ‘scope’ of the access, and SMART describes how the content of the scope is expressed. One of the possible scopes is for the app to read the user details (name, date of birth, address etc.). Somewhat confusingly, this is also referred to as the user ‘Identity’
4. Having identified the user (but only that they are a valid user – not the ‘Identity’ details) and determined what data they can access, the Authorization Server then issues the app with a token (called the *Access Token*) that the app can then use when requesting the data.
5. Finally the app makes a FHIR request against the Resource Server, including the Access Token in the request. The Resource Server checks that the token is valid and that the user is allowed to make the request (implementation dependent) and if so, returns the requested data (or processes any updates).

Here’s a diagram of this process:



You’ll note that a lot of the details of the process can vary according to the implementation. For example, the Authorization Server could place the agreed scope in the Access Token and encrypt and sign it, or Access Token could simply be some random key, and the Resource Server uses it to check each request with the Authorization Server.

The client doesn’t care – SMART will work regardless of the servers implementation details, which is the whole purpose of the standard.

The API and the Data content

The other big thing that SMART provides is related to the data content – the FHIR resources. While FHIR provides descriptions of each resource (for example what information is inside an Allergy), it does so in a very general way. Specific implementations then ‘adapt’ that resource to meet their specific requirements – a process called ‘profiling’ FHIR. Although not necessary for interoperability, it is helpful that both client and server recognize the same profile and so SMART defines a standard profile.

In fact, this aspect of SMART is undergoing development at the moment – currently there is a ‘SMART profile’, but in the US this is moving towards being [Data Access Framework](http://hl7.org/fhir/daf/daf.html) (DAF). There is also work on adapting DAF for other countries – specifically Canada at the moment – so it is likely that SMART will allow a client to specify the profiles it supports as part of the negotiation process.

Launching an app

SMART describes a couple of ways of launching an app – from within an app (the EHR launch) and an externally launched app (Standalone launch). The standalone launch is really just an external app requesting access via the external API as described above.

The ‘EHR launch’ describes how the EHR can start a SMART aware app, preserving the current EHR context – the current user and patient (if selected). In brief:

1. The user invokes a function to launch a previously registered app (eg clicks on a link in the User Interface).
2. The EHR stores the context information somewhere, creating a token (the *launch token*) that refers to it. Alternatively the EHR could include the context within the token encrypting and signing it.
3. The app then authenticates in the same way as described above.
4. After authentication, the app includes the launch token with each call to the Resource Server. The Resource Server uses the token to retrieve the context of the call – either by using the token as a key to some internal store or by decrypting it – depending on how it was initially created.

Note that like the other aspects of SMART, most of the server side implementation details can vary between implementations. As long as the interface between client and server is preserved it doesn’t matter to the client.

Future growth of SMART

SMART (like FHIR) is an evolving standard, and so is likely to change and extend as it is implemented internationally. The areas where we are likely to see evolution are:

* Becoming the ‘de facto’ security standard for FHIR interfaces. FHIR itself doesn’t prescribe a security mechanism, leaving that to the implementer. Given SMARTs use of widely accepted standards, it is likely that it will become the ‘normal’ security mechanism
* Supporting different profiles. As described in this paper, SMART is already moving in the direction of allowing the client to describe the profiles it supports.
* [CDS Hooks](https://github.com/cds-hooks/cds-hooks/wiki) – an exciting new development that seeks to standardize how an EHR can invoke Decision Support capability in the course of its usual operation. This standard – if it becomes widely adopted – could make it easier for a provider of Decision Support services to be utilized by different EMR/EHR systems – another important aspect of exposing advice generated as part of the Precision Medicine initiative.

2.52 Dev days in November

JULY 8, 2016 [4 COMMENTS](https://fhirblog.com/2016/07/08/dev-days-in-november/#comments)

In the past I’ve talked about the [FHIR Developer days](http://fhir.furore.com/devdays) hosted by our friends [Furore](http://fhir.furore.com/) in November in Amsterdam – generally with a slightly envious tone. This year – thanks to my employer [Orion Health](https://orionhealth.com/) – I’m able to attend! (I’m going to get into the group picture at last  )

Since I’m going to be there and I have a clinical background, the Furore folk suggested that I might want to lead a track specifically for Clinicians and Business Analysts who want to learn more about FHIR – so this post is all about putting some ideas out to get feedback.

Perhaps you are a Clinician or Business Analyst employed by a vendor to assist with product development – or maybe you play a leading role in implementations at, say, a hospital or some other healthcare facility and need to advise on how it should be configured for maximal clinical value.

Even if not currently involved in FHIR implementations, you are very likely to be in the near future, and so need to have an understanding of how FHIR works – and what you need to understand to be able to make it meet your own scenarios.

We haven’t locked down the programme (which is one reason I’m writing this – to get ideas from you about what you’d like us to do), but we’re imagining a program that combines theory and practical. Something like:

Some presentations on FHIR (from a clinical perspective):

* Why you’d use it and what are the expected benefits in doing so (so you can speak knowledgeably to decision makers about it)
* The fundamentals of FHIR – specially resources and datatypes
* A review of all the resources currently available – especially the clinical ones
* Using coded data, ValueSets and the use of Terminology – including FHIR based terminology services
* Using FHIR to share data – whether by real-time API, Documents, Messages
* Decision support and Quality measures
* Profiling – the basics, why it’s there and what it does
* Standards that build on FHIR – like [SMART](http://smarthealthit.org/smart-on-fhir/) and [CDS-Hooks](https://github.com/cds-hooks/cds-hooks/wiki)
* How FHIR is developed and how to get your ideas into the specification
* Engaging with the community – how to get help – and help others

And whatever else we think would be of interest to a clinician.

And we’d make sure we worked in with some of the other tracks that could be of interest – such as the  [logical modeling and openEHR](http://www.fhirdevdays.com/track-logical-models-fhir-profiling/) one that Grahame is giving.

We’d go over some of the tooling that is available to help learn FHIR – basic tools like XML & JSON editors plus REST clients to test out servers – but especially [clinFHIR](http://clinfhir.com/) that was built with that purpose in mind.

There will be some exercises to help you get familiar with the tooling – and also to help cement some of the concepts we talk about during the presentations (don’t worry – we’ll break it all up into digestible pieces!).

Then we’ll spend some quality time on [Profiling](http://hl7.org/fhir/profiling.html), as this is the area that you are likely to be the most involved in during an implementation or a development project. We’ll talk about the capabilities – and limitations – of the profiling infrastructure and the tooling currently available to find and build profiles. We’ll talk about all the different artifacts we’re going to need in real life – the profiles themselves (StructureDefinition) plus other resources such as ValueSets, NamingSystem, Implementation Guides and so forth.

We’ll also take a look at [Forge](http://fhir.furore.com/Forge) which is the professional profile builder built by our hosts Furore as well as [Simplifier](https://www.simplifier.net/) – where the conformance resources can be saved.

There will be some practical exercise where we take a clinical Use Case, define the profiling artifacts we need (perhaps using openEHR as an analysis tool to do so, or just a spreadsheet), build and deploy them, and then create some real resources based on those profiles.

We’re also investigating the possibility of ‘pairing up’ clinicians and developers to build some real applications just to show how easy this can be.

So what do you think?

There will be plenty of opportunities to dive into the depths of FHIR if you want to by attending any of the other tracks, all the FHIR experts will be there for your questions and it will be a great networking opportunity to meet with like-minded people – what’s not to like?

(Oh, and there will also be beer! )

Please feed back any comments and/or suggestions you may have either as comments to this post or directly to my email – david dot hay25 at gmail.com and I’ll see you in Amsterdam!

2.53 More FHIR on the pi…

JUNE 21, 2016 [4 COMMENTS](https://fhirblog.com/2016/06/21/more-fhir-on-the-pi/#comments)

So this is in the nature of the ‘I’m doing it this way just because I can!’

The background is that I want to create a series of time lapse pictures from a property overlooking a tidal river that doesn’t have internet access, and to be able to access those images remotely. (The property is in Waipu – rural New Zealand!)

I’m planning on using a [raspberry Pi](https://www.raspberrypi.org/) (with a camera) to take the photos, then upload them to a server from where I can generate the video sequence. (Because I have to use a cellular connection to access the net, I can’t host the images directly off the pi – which would have been my preferred option).

My original thought was to set up a simple hosted server and database (likely node.js and mongoDb) and have the pi regularly send the images there using HTTP, and then I thought – why not use FHIR? The server side then becomes easy as I can use an instance of a reference server (likely the [HAPI CLI server](http://hapifhir.io/doc_cli.html)) to receive and store the images. There was an added bonus that I could get to play with python a bit – and see how it handles sending FHIR resources.

Now, I’m the first to admit that this is hardly the intended use of FHIR – and that there are much better solutions you could use, but the opportunity to experiment with the pi as a FHIR client is certainly useful as I can imagine that as a real use case – imaging the pi acting as a home-based hub sending patient observations from devices to a FHIR server in the cloud.

As it turned out, it was really easy to do this – at least in a prototype fashion.

Here the code for the python script (and I’ll use [cron](https://en.wikipedia.org/wiki/Cron) to call it on a regular basis):

import json

import base64

import requests

import time

from datetime import datetime

import picamera

from sense\_hat import SenseHat

sense = SenseHat()

url = "http://fhir.hl7.org.nz/dstu2/Media"

fileName = "images/pic"+str(time.time())+".jpg"

# colours for the sense hat

colour\_red = [255, 0, 0]

colour\_white = [255, 255, 255]

colour\_blue = [0,0,255]

colour\_green = [0,128,0]

colour\_yellow = [255,255,0]

# image dimensions

image\_height = 620

image\_width = 480

sense.clear(colour\_yellow)

with picamera.PiCamera() as camera:

   camera.resolution = (image\_height,image\_width)

   camera.start\_preview()

   # Camera warm-up time

   print('Waiting for camera')

   time.sleep(2)

   camera.stop\_preview()

   camera.capture(fileName)

sense.clear(colour\_blue)      #to indicate picture saved

# load the file and send

with open (fileName, "rb") as myfile:

      pic = myfile.read()

# code for FHIR object

      reference = {}

      reference['reference'] = "Location/hayWaipu"

      extensions = []

      extensionLocation = {}

      extensionTime = {}

      extensionLocation['url'] = "http://fhir.hl7.org.nz/StructureDefinition/locationOfMedia"

      extensionLocation['valueReference'] = reference

      extensions.append(extensionLocation)

      extensionTime['url'] = "http://fhir.hl7.org.nz/StructureDefinition/locationOfMedia"

      extensionTime['valueDateTime'] = datetime.now().strftime("%Y-%m-%dT%H:%m:%S")

      extensions.append(extensionTime)

      data = {}

      data['extension'] = extensions

      data['resourceType'] = 'Media'

      data['type'] = 'photo'

      data['deviceName'] = 'rpi'

      data['height'] = image\_height

      data['width'] = image\_width

      content = {}

      content['data'] = base64.b64encode(pic)

      content['contentType'] = 'image/jpg'

      data['content'] = content

      json\_data = json.dumps(data)

headers = {'content-type':'application/json+fhir'}

print('Sending picture to ' + url)

r = requests.post(url=url, data=json\_data, headers=headers)

if (r.status\_code == 201):

      sense.clear(colour\_green)

      print(r.headers['location'])

      print('Image saved')

else:

      sense.clear(colour\_red)

      print(r.json())

It’s actually longer than it needs to be as I decided to use the ‘sense hat’ as a visual indicator of where the script was in the overall process (I had it lying around – a simple LED might be better) – although the fact that it has a number of other sensors (temperature, humidity etc.) does suggest that I could collect other observations.

It starts by importing the various libraries it needs (I’m using the [requests](http://docs.python-requests.org/en/master/) library to make the actual REST call) and then activates the camera, waits a couple of seconds and takes a photo – saving it to a file with a name constructed from the current time (that way I have a local copy of the images – though I’ll need some way to ensure they don’t fill up the pi!)

Next it constructs a Media resource to contain the actual image as a base64 encoded attachment.

I added a couple of extensions – one to record the date the image was taken, and the other so I could have it reference a Location resource (the closest existing match was ‘specimen’ which didn’t feel quite right…). I still have to create the extension definitions (StructureDefinition resources) but I will!

So this all works, though there are a few enhancements that come to mind:

* Better logging to assist troubleshooting is likely to be required when I deploy for real
* The use of the sense hat is a bit overkill – perhaps a multi colour LED that does the same thing and maybe a small LCD screen showing the last operation?
* Rather than storing the image as a base-64 encoded attachment (with associated overhead), I might store it as a binary resource referenced from the Media – or maybe an Observation might be better?
* I could add other observations (temperature, humidity & such like)
* I should really shift the resource creating stuff to a separate function – maybe a separate module, just to keep the main script tidier

Anyway, a fun little experiment exploring FHIR on the raspberry pi!

2.54 Resource reference visualization in clinFHIR

JUNE 14, 2016 [2 COMMENTS](https://fhirblog.com/2016/06/14/resource-reference-visualization-in-clinfhir/#comments)

I was skyping with my colleague [Viet Nguyen](https://www.linkedin.com/in/viet-nguyen-053a836) over the weekend about some improvements I was adding to clinFHIR – the resource viewer now generates a more ‘user friendly’ display for some of the resources (Encounter, Observation, Condition). I showed him what I had done, and asked him what else might be useful and he said:

*This maybe pie in the sky – a network type diagram with different colored icons representing conditions, encounters, labs, etc. that allows the user to visualize the relationship between conditions and related resources. This is the type of “breaking out of the document paradigm” approach I’d like to pursue.*

As it turns out, I’ve played a bit with that kind of functionality in the past using some javascript libraries, so I thought it might be worth having another look at. A bit of googling found this amazing library – [vis.js,](http://visjs.org/) so I set to work!

To see the results, start up [clinFHIR](http://clinfhir.com/) and select the updated Resource Builder. Once the app is displayed you need to select a patient server (ie where the patient data resides). The functionality should work with any server, but it’s good to have some data to work with, so there are a few options:

* Create them manually using the resource creator. Doable, but tedious
* Use clinFHIR sample creator. Click the ‘Select Patient’ link, then the ‘Add new Patient’ link and create a new patient with samples. The public HAPI STU-3 server is a good choice for this
* Use a server with a good range of sample data.

For this post I took the latter approach – using ‘MiHIN’ as the Patients provider. These guys are working on sample data which is going to be really important as we continue to exercise FHIR in real-world scenarios. (You may need to select ‘reset config’ from the gear menu to the upper right, then re-load the page to see this Server – sorry about this – I need to fix some caching issues).

Select a patient (I used Karen Florence Smith) and then click on the ‘Details’ link in the top navbar against her name to display the resource viewer.

The ‘resource viewer’ now has 3 tabs:

* Resource Explorer
* Resource References Graph
* Encounter timeline

Lets’s look at each of these.

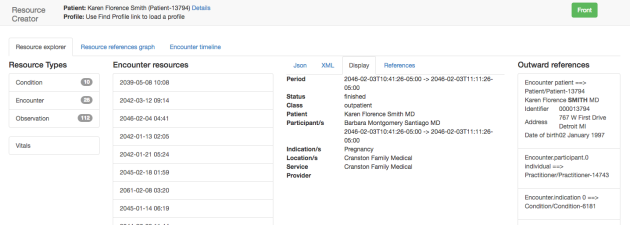
The *Resource explorer* is the original viewer that displays the types of resource that a patient has, then a specific resource instance of that type and allows you to view the details of that resource in Json or Xml, download it and view the referenced resources. If you select a resource (an Encounter is a good one) you’ll see that there is a new tab labeled ‘References’ in the resource detail section.

Clicking on that tab will display a graphical rendition (technically a network or graph representation) of that resource with all the other resources that it references, plus the resources that reference it. Here’s a screen shot:



Now, this is the same information that was already displayed as text on the right side, but the picture is easier to understand – and to get your head around what we mean by resource references. You can drag the resources around if they aren’t clear and the others will move out of the way. Like the textual representation, clicking on any of the resources in the graph will put that resource ‘in focus’ – showing its references as well, thus allowing you to ‘surf’ the interrelationships between resources. Oh  and sorry about the colours – if any of you have a User Interface background, I’d appreciate help!

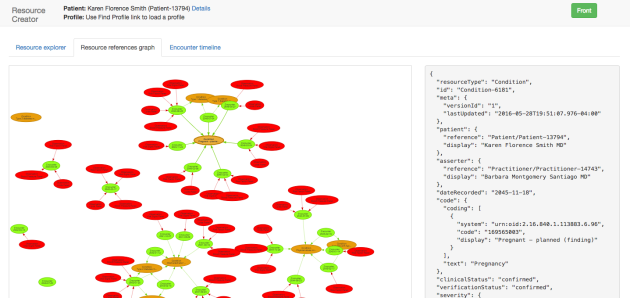
There’s also a ‘Display’ tab that will render a more user friendly display for some of the resources (currently Encounter, Observation & Condition). I intend to expand this to include other resources in the future. This image shows an encounter. Note that the display will not necessarily show all the data in the resource – use the Json or XML tabs to see the complete picture. (Right now, extensions aren’t displayed for example)



(By the way, we tend to use the word ‘Resource’ rather ambiguously – especially in the context of profiling. In this post, a ‘Resource’ is actually a resource instance – ie an actual set of data for a specific patient.)

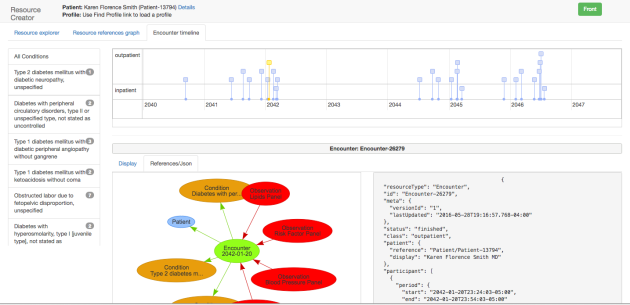
The *resource references* graph will display a graph of all the resources for a patient, along with their references. The initial display is small if there are many resources – you can zoom the display (2 fingers vertically on my mac trackpad) and scroll (click and drag) – I’m not sure how that works on other computers – will update the post when I find out. Like the previous graph you can drag a resource around to make the display clearer.

Heres a screen dump:



Selecting a resource in the graph shows the Json representation to the right but doesn’t otherwise affect the display. By the way – note that the Patient resource is not shown in this graph – most resources will have a reference to the Patient, but including it completely skews the graph, so I took it out…

The last area where the library is (currently) used is in the 3rd tab –the  *Encounter timeline*. This is different to the others as it is intended to represent a timeline of encounters (oddly enough) – and to filter those by condition. Here’s a screen shot:



As you see there are a number of components to the view.

The timeline itself is in the upper right of the screen. It is organized into rows by encounter class, and can be zoomed and scrolled like the other charts.

Below the timeline is the details of a selected encounter (selected by clicking on the encounter in the timeline – the yellow one in the example above). It shows:

* A graph of the references of the selected encounter (this is the same as you would see in the resource browser). Clicking on a resource will display the Json to the right
* The Json representation of the encounter
* The summary display (on a separate tab – not shown above)

To the left side is a list of all the patients Conditions, with a count of the number of encounters where that condition is referenced (as an ‘Encounter.indication’ reference). Selecting any of them will filter the timeline to show only the encounters referencing that condition.

There are a number of issues/enhancements to complete:

* Extensions are currently ignored in the display view and in the references
* Only a few of the resources have a detail view
* If you select resources in the graph in multiple succession too quickly, the app seems to hang for a while – I suspect there is an issue in the event handler that I need to look into
* The colour scheme is – well – basic. (One of my Orion colleagues called it a ‘Christmas’ display!) As mentioned above, if you know about colour – I’d love a better palette!

2.55 ValueSet editor in clinFHIR

MAY 30, 2016 [2 COMMENTS](https://fhirblog.com/2016/05/30/valueset-editor-in-clinfhir/#comments)

I’ve been working on the profiling abilities of clinFHIR recently. As I’ve said before, although there is the official tool for creating profiles – [Forge](http://fhir.furore.com/Forge) – I think there is a place for a simple profiling tool primarily aimed at clinicians, with the goal to help them understand how FHIR profiling works.

One of the big things that keeps coming up is ValueSets – and how to create them.

As a short recap – recall that the purpose of profiling is to take the core resources and make them more suitable for real-world Use Cases by adding new elements (extensions) and removing the ones that are not needed. As part of this process you often want to specify a particular set of values for coded elements that is different to the one in the spec and the ValueSet is the mechanism that you use to specify those. The problem is that there isn’t currently any widely available tooling to create ValueSets (outside of the tooling used to build the specification itself) – especially ones for clinicians to use, so over the weekend I decided to write a simple ValueSet Editor.

Like the profiling in clinFHIR – the ValueSet Editor is intended to allow the creation of simple valueSets rather than as a comprehensive tool. At the moment, a number of the functions are really geared towards SNOMED – although it should work with other terminologies (if supported by the terminology server). It will allow you create a ValueSet and to edit them – but won’t allow you to edit other ValueSets – eg ones from the spec – for fairly obvious reasons. But you can make a copy of these and edit those if you want to use them as a starting point for your own.

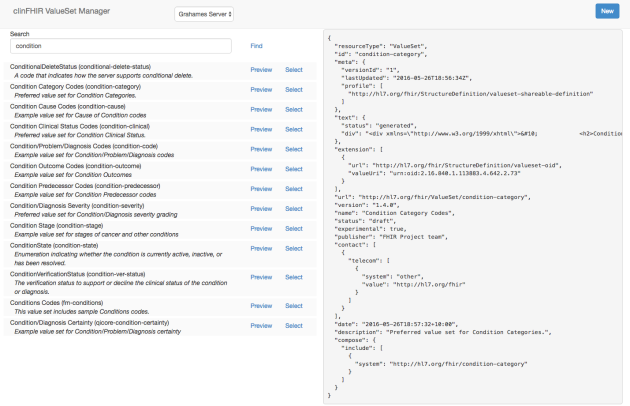
Having [selected the editor](http://107.170.196.80/valuesetCreator.html), the first thing to do is to select the Terminology server where you want the ValueSet to be saved. This is displayed in the menu bar at the top of the screen. The editor does require that the server implement a number of the STU-3 terminology services – namely:

* [Concept lookup](http://hl7.org/fhir/2016May/codesystem-operations.html#lookup) – used when viewing the hierarchy of concepts
* [ValueSet expansion](http://hl7.org/fhir/2016May/valueset-operations.html#expand) – used to find a concept by name.

(Having said that – at the moment the only server that supports v3 of terminology services that I am aware of is Grahames server. But hopefully there will be more available shortly!)

Once the Terminology server has been selected, you can either view existing ValueSets on that server (either just to have a play with them or to use them as the basis for your own ValueSet) or go straight ahead and create your own from scratch.

To view an existing ValueSet, enter some filter text (like ‘condition’) into the search box and click the ‘find’ link. Matching ValueSets (based on the ValueSet name) will be listed and you can either preview them (which shows the Json representation of the ValueSet resource) or select them – which will take you to the editor – though in a View mode initially. Here’s a screen shot with a ValueSet previewed.

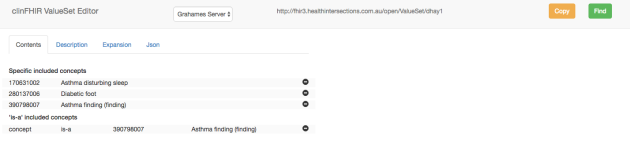


The Editor is where the action is, and is displayed when you ‘select’ a ValueSet (or create/edit one as we will see shortly.

On the left side are 4 tabs.

* The *Contents* tab shows the concepts that are included in this ValueSet. There are 2 types that are supported by the tool – either a list of specific concepts, or a filter that describes a SNOMED ‘is-a’ relationship. (We’ll see how to find these in a minute)
* The *Description* tab shows information about the ValueSet – the description, requirements, publisher and the like. You can update this data when creating a new ValueSet (or editing an existing one) – otherwise it’s read only.
* The *Expansion* tab allow you to perform an expansion against the ValueSet – this is the same that occurs in the resource builder – you enter in some search text and the Terminology server then ‘expands’ the contents of the ValueSet – finding all the concepts whose display matches the text you entered. This is a really great way of testing the ValueSet. Note that the expansion occurs against the copy of the ValueSet on the server, so if you make any changes then save them first!
* The *JSON* tab simply displays the Json view of the ValueSet – same as the ‘preview’ when performing the search.

Here’s an example:



This shows one of my test ValueSets containing 3 specified concepts, plus an ‘is-a’ concept – which we’ll discuss in a minute.

The right hand side (which is empty in the example above) is where you can add concepts to a ValueSet you are able to update – so let’s take a look at how to do that. There are 2 ways: You can make a copy of an existing one, or create a brand new one from scratch. The copy function is convenient – but remember that clinFHIR only supports a small subset of the overall capabilities of ValueSets so if you select one of the more complex ValueSets then there will be functionality that you cannot modify. You’ll be able to see it in the json view, and it will still work (ie expand) – but your ability to make changes is limited.

Both copy and create are activated from buttons in the top right hand side of the menu bar.

* The ‘*new*‘ button is displayed from the front page
* The ‘*copy*‘ button is  displayed when an existing ValueSet is being viewed.

In both cases you need to enter a name for the new ValueSet. This name needs to be unique on the terminology server as it will also be the ‘id’ of the ValueSet resource. Enter a name into the dialog that appears after clicking the button, then click the ‘check’ button. If the name is new, then a ‘select’ button appears in the dialog at the top right – otherwise you’ll get an error and need to re-try.

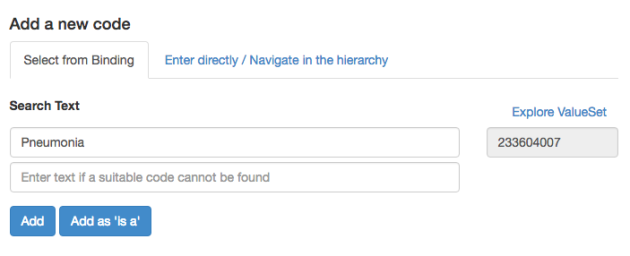
Once a valid name has been selected, then you will be returned to the editor, and this time the ‘add concept’ pane to the right will be displayed.

There are 2 ways to add a new concepts to the ValueSet – and 2 roles that they can fulfill.

* The first role a concept can fulfill is as a simple list of codes. The ValueSet is comprised of that list and you will be able to select one of them in the resource builder
* The second is where the concept acts as an ‘is-a’ filter. When expanded, the list can include the concept and any of its children

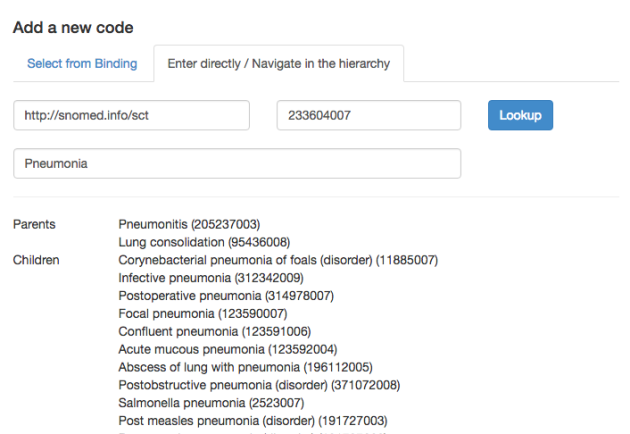
To select a concept you can either use the ‘select from binding’ tab or the ‘enter directly’ tab – or a combination of the two. The ‘select from binding’ option is the same as in the resource builder – the terminology server will locate all matching concepts using the expansion against the bound ValueSet. (More on that later).

Heres a view with asthma selected (and note that the selection will auto-populate if you pause for a second – just like the Resource Builder):



The second tab is rather more interesting. This tab allows you to enter the code directly (it is pre-filled when you select a concept from the first tab) – but as well as the details of the selected concept, it will also display the parents and immediate child concepts as well. In addition, the parent and children concepts are clickable – selecting them will then display their immediate relations thus allowing you to traverse up and down the hierarchy.

Here’s a view of the second tab – filled in with values from the first – ie ‘pneumonia’



To add the selected concept to the ValueSet – click on either the “Add” or the “Add as ‘is-a'” buttons. The difference is:

* The ‘*Add*‘ button adds that single concept only. If the ValueSet as empty then it would only have a single value.
* The “*Add as ‘is-a’*” button adds the concept as an ‘is-a’ filter. This means that the concept *and all its children* become part of the ValueSet. In the example above, then the concept ‘Abscess of Lung with pneumonia’ would be a member of the set.

The screen shot above also shows  quite nicely shows how (in SNOMED at least) a concept can have multiple parents – ie it is an ‘is-a’ of multiple things.

So there you go – a simple ValueSet editor, developed to help with Clinician profiling. In the next post, I’ll talk about how it fits in with the enhanced clinFHIR profiling (I just have to finish that first  )

Hope you find it useful!

Some known issues & notes:

* The tool will automatically add a contact of ‘clinfhir’ to all ValueSets – which is how it recognizes the ones it has created, and will allow to be edited.
* The search works on the name in the ValueSet, so make sure you enter something sensible for that value (on the ‘description’ tab in the editor). It will default to the name entered when the ValueSet is created – which will also be the id on the server.
* When you start the editor, the server name might be ‘undefined’. It will still work OK, but to correct this, start up the [Resource Builder](http://107.170.196.80/resourceCreator.html), and select ‘reset config’ from the gear menu to the upper right. If you re-load the ValueSet editor, then the name should be ‘Grahames Server’.
* If you’re creating a new ValueSet from scratch – it’s best to reload the application first – just to clear all the internal variables (there are a few gremlins left to swot)
* Every now and the Grahames server gets a ‘deadlock’ error from the database. If you get an error message that mentions this, just try again.
* When you are looking up a concept based on the name (like in Resorce Builder) the tool uses the ValueSet [*http://hl7.org/fhir/ValueSet/condition-cause*](http://hl7.org/fhir/ValueSet/condition-cause) as the base – because that just references all of snomed. This is a temporary situation, as I intend to allow you to specify the Root concepts as a way of restricting the search – I just haven’t had a chance to do that yet.

2.56 Modifier Extensions in versioning (maybe)

MAY 20, 2016 [3 COMMENTS](https://fhirblog.com/2016/05/20/modifier-extensions-in-versioning/#comments)

So I think I’ve come across my first real place where using a modifier extension makes sense. *(actually, not quite – see the note at the bottom of the post, but the discussion is hopefully still interesting)*

Before we get into that, a short recap on what a modifier extension is. As most people working with FHIR will be aware, the specification aims to keep the individual ‘packets’ of data (Resources) as small and simple as possible, by providing a fixed core of elements that ‘most’ people are currently sharing and an extension mechanism that allows individual implementers to add the extra elements that they may need for specific Use Cases – a process called Profiling.

Now, when you add a new element you can’t expect that a recipient will understand what it means. You can provide a definition of the extension (in fact you have to) but that still doesn’t mean that the recipient will know to do with the extra information you are adding. Most of the time that doesn’t really matter – if you add an extension for ‘eye colour’ to Patient, and the recipient doesn’t store eye colour then no harm is done.

But sometimes you can’t be that sanguine about it. Suppose you wanted to assert that a patient did not have a given Condition and created a ‘hasNotGot’ extension to use, then a recipient MUST understand what you are trying to say – otherwise they may think you are saying that the patient does have the condition. To cover this scenario, FHIR has the concept of [modifier extensions](http://hl7.org/fhir/extensibility.html#modifierExtension) – extensions that potentially change the meaning to the resource and which a recipient must have a strategy to deal with.

The spec states:

Implementations processing the data in resources SHALL check for modifiers anywhere they may appear, and if a modifier extension is present, SHALL do one of these things:

* understand the impact of the extension when using the data
* refuse to process the data
* carry a warning concerning the data along with any action or output that results from processing the data to inform users that it has not fully understood the source information

It also states:

Implementers should avoid the use of modifier extensions where possible. Any use should be carefully considered against its possible downstream consequences. However, implementers are often forced into these situations by the business arrangements around the use of resources, so this specification creates a framework for handling these cases. If modifier extensions are present, an application cannot safely process the resource unless it knows what the extension means for its own use of the data.

So, on to our Use Case.

The application that we’re developing acts as an ‘aggregator’ of information – it collects information from a number of sources (in this case Allergies) both inside and outside the Enterprise and makes that available to users.

What we want to be able to do is to support [versioning](http://hl7.org/fhir/http.html#versioning) of resources – the ability to record changes to a resource instance and to make that available to the user. This is optional in FHIR (though recommended) and is declared by a server in its conformance statement on a ‘per type’ basis – so a server declares that it supports version for some resource types – and potentially not for others.

This creates a particular problem for us as not all of the source systems are able to support versioning – and it’s really an ‘all or none’ – the client consuming our service doesn’t care about our internal architecture and if we say we support versioning, then they will expect that all resources we expose across that interface can have a history and that they can retrieve that history and any specific version

We bounced this around a few times and looked at a number of options. We could:

* Require that all sources support versioning – but then we could potentially cut ourselves from useful data
* Do the versioning ourselves – but that could be quite a lot of work, and potentially incorrect as we won’t know every change that occurred
* Only support versioning on the resources that have a version – but that could be just plain wrong! You can easily envisage scenarios where a clinician acted on information that subsequently changes and needs to be aware of that change.

So in the end we decided that best solution was a variant of the last option – we’ll add a modifier extension to those resources that cannot be versioned to indicate that any history is not available through the API– at least we can be explicit to the user that there is data that **may** have changed – we just can’t tell them through the interface – they’ll have to go to the logs (or the System Administrator will) if a history of changes is important to them.

If a client asks for a version history or a specific version then we’ll return a 404

It does add an extra burden on to the client – but it does mean that we can offer versioning over most of the resources.

So not the most satisfactory solution – but one that should work… Oh, and if you have a better alternative (or a problem with this approach) then we’re happy to hear it!

*As an addendum to the post, both Grahame & Mark commented (thanks!) – Grahame suggesting that simply returning a 404 in the case of an unavailable version would be sufficient to alert the user to the possibility of a missing version – ie go look elsewhere if this is important to you. So no Modifier extension needed in this case – which is actually kind of nice – and great to get the feedback!*

2.57 Using clinFHIR for profiling

MAY 10, 2016 [LEAVE A COMMENT](https://fhirblog.com/2016/05/10/using-clinfhir-for-profiling/#respond)

One of things we’ve discussed many times in the past is [profiling](http://hl7.org/fhir/profiling.html) FHIR – taking the base resources and adapting them to specific Use Cases by adding extra fields (extensions) or restricting/removing base fields.

The community has provided tooling in the form of the [forge tool from Furore](http://fhir.furore.com/Forge) for creating these profiles, but to do this properly requires a certain amount of planning and work, and there is a place for being able to quickly create a profile to try something out. This is kind of like clinical modeling – you have some data you wish to add to a resource, but you’re not sure where the best place to put it is – or you want to quickly create a profile that others can then comment on.

To meet this need, the latest version of [ClinFHIR](http://clinfhir.com/) now has the facility for creating simple profiles, that can subsequently be used by the resource creator to create sample resources. These profiles are stored in a FHIR server in the usual way, so that anyone can create a sample resource from that profile to see what it looks like.

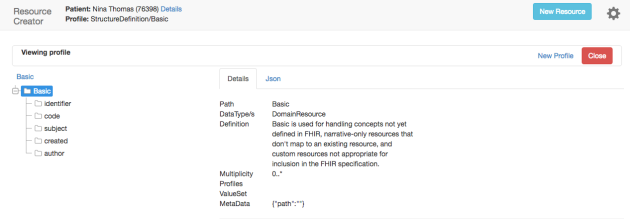
To start, load the resource builder from the main [clinFHIR](http://clinfhir.com/) site. The first thing to do is to make sure that you’re referring to the correct servers. A simple way to do this is to click the gear icon to the top right, and select ‘Reset Config’ from the list. This will set the following default servers: (Note that this selection might change in the future)

* Data Server (for Patients and Data) to the HAPI STU-3 server
* Conformance Server (for the Profiles) to Grahames STU-3 server
* Terminology Server (for Terminology services) to Grahames STU-3 server.

Of course you can set these to any of the configured servers’ – just be sure that they are at the same STU (Standard for Trial Use) level. (Incidentally there are also options on the gear menu to clear the cache of profiles and patients, which is handy when the lists get a bit long).

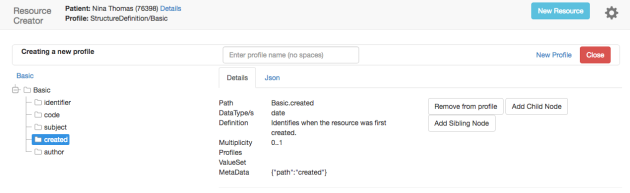
Next you need to select the base resource (actually the profile that defines the base resource) you want to modify. This is done by clicking the ‘Find Profile’ link in the second column on the front page ([here](https://fhirblog.com/2016/04/24/clinfhir-resource-builder-version-2/) is a post describing the new version of clinFHIR) and then selecting the appropriate resource type. Once you’ve done that, the profile will appear in the list of profiles in the middle column.

You’ll notice what looks like an eye immediately to the left of the profile name. Click on the eye, and the front page will be replaced by a tree representation of the profile. The following image shows what it looks like:



You’ll note that the tree view (which is quite simple in this profile) is to the left, and in the middle is a tab control currently showing some details of the currently selected node – and which will be updated as you select nodes.

At the top right is a link ‘New Profile’. Clicking on that link indicates that you want to create a profile based on the one selected (in this case the Basic resource) and does a couple of things as shown in the image below:



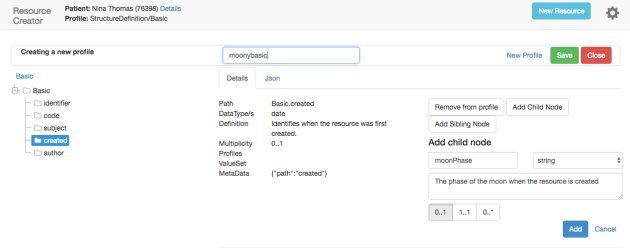
At the top middle of the page is the name of the profile that will be created. Enter a name into that box. Technically, this will become the id of the profile you create against the currently selected Conformance server, so clinFHIR will check that that name is not currently in use – warning you if it is.

It also displays a series of buttons to the right of the details. These buttons have the following functions.

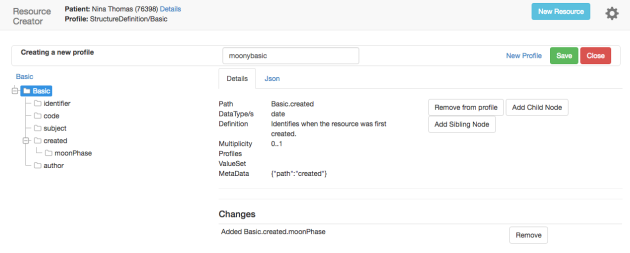
*Remove from profile* will delete the currently selected node from the profile. In the screen shot above, this will remove the ‘created’ node from the profile.

*Add Child Node* and *Add Sibling Node* both add a new node (or element). The ‘Add child node’ adds the new node (which will be saved an as extension) ‘underneath’ the current node – in other words, the extension will be applied to the current node. ‘Add sibling node’ adds the extension at the same level as the selected node – in the example above this would be an extension on the root of the profile.

Clicking the ‘Add child node’, displays a small form allowing you to enter the key information about the new node (extension) as shown here:



So we’re adding an extension that will allow us to enter the phase of the moon when the resource is created. It will be a simple string, and there can only be one of them- and it’s optional. Click ‘Add’ to add it to the profile, and you’ll get a screen like this:



Note that the new node appears under the existing ‘created’ node.

Note also that at the bottom of the screen is a list of the changes you have made (additions and deletions), with the option to reverse that change.

Now click on the ‘Save’ button (which appears only after you entered a name – with no spaces), and with a bit of luck you’ll get a dialog stating that the profile has been saved.

Technically, what happens is that ClinFHIR will automatically create a new extension definition (actually a StructureDefinition resource) for each new element you have added, and then save the new profile with the correct references to the extension definitions.

Close the dialog to return to the front page, and you’ll find your new profile in the list of profiles.

Selecting a test patient and that profile (rather than clicking on the eye)  will display the [resource builder](https://fhirblog.com/2016/04/24/clinfhir-resource-builder-version-2/) with that profile as shown here:



You can create the resource in the usual way – and don’t forget to use the ‘Validate’ option to satisfy yourself that the resource is valid.

Some caveats.

* The intention of this is to make it simple to visualize what a profile will look like so it just adds new extension definitions for each new element. Generally you would look for an existing extension definition that matched what you wanted to add (the moon phase in this example) as we really want re-use of extension definitions. (Future enhancements will add this).
* The information you can add about the extension is quite limited. I’ll probably add a couple more items – like the ability to specify the ValueSet binding for a coded element – but it will never be a full featured tool like [Forge](http://fhir.furore.com/Forge).
* When you specify a resource reference you can’t specify a resource type/s
* Currently you can only add simple extensions. We’ll look to add complex ones in a later release (and update the resource builder to recognize them as well).
* Right now, the functionality is brand new so testing has been minimal. We’ll squash bugs as fast as we can, but it will take people using the tool to find them. If you do get an error that let me know (preferably with a screenshot and description of what you were doing) and re-start the application.
* The functionality has only been tested with STU-3 servers – and those servers can have issues at the moment as we work on the STU-3 release. Let me know if you run into problems.

On the task list for things to do:

* Allow select an existing extension definition
* Specify binding for coded elements
* Support complex extensions
* Allow to specify resource types in Resource reference
* Add support for all datatypes

So there you have it – feedback welcomed (and appreciated)!

2.58 Internationalizing clinFHIR

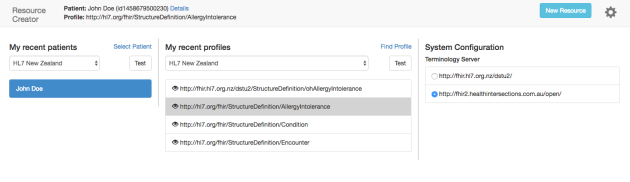
APRIL 30, 2016 [LEAVE A COMMENT](https://fhirblog.com/2016/04/30/internationalizing-clinfhir/#respond)

I’m supposed to be finishing off my presentation (I’m giving the ‘[FHIR for developers](http://www.hl7.org/events/wgm052016/tutorials.cfm)’ tutorial at the [Montreal WGM](http://www.hl7.org/events/wgm052016/?ref=banner)) and [Mary-Anne](https://www.linkedin.com/in/mary-ann-boyle-0b1318a) will kill me if it’s in late – but I was chatting with a colleague from [HL7 Russia](http://www.hl7.org.ru/en/), and got a bit sidetracked…

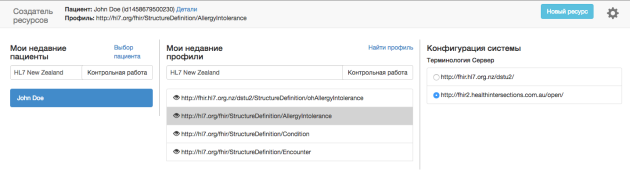
Apparently they like some of my posts (which is flattering) and are translating them into Russian (also flattering). They also like using [clinFHIR](http://clinfhir.com/) as an educational tool (even more flattering) but wondered if it could be ‘internationalized’ – ie the User Interface elements translated into Russian (the resource content would remain English of course). Now this is the second time that someone has asked for this (the last one was from [Lithuania](https://www.hl7.org/documentcenter/public_temp_E8B6297F-1C23-BA17-0C21D09CF61DDB36/wg/java/20150510_AID_Jan_Jasinski_Lithuanian%20NHR_on%20FHIR.PDF)), so I thought why not?

It actually turned out to be quite straight-forward. Angular has support for Internationization (or at least there are plug-ins that will do that) so it’s just a matter of using ‘placeholders’ for the UI text which get rendered in the language of choice.

So here’s what the front page of the new Resource Builder looks like in English…



And in Russian…



You select the language by clicking the gear icon to the upper right and selecting the ‘Change Language’ option (actually the UI re-renders in real time as you change languages, which is kind of cool!). It will remember the language you chose (in the browser) – but can always be changed back if you are experimenting.

I’ve done partial translations for English, Russian and Dutch (to honour [Furore](http://fhir.furore.com/) and all the effort they’ve put into FHIR). I was going to do one for Australian (or [Ocker](https://en.wikipedia.org/wiki/Ocker)), but for some reason Google didn’t offer that…

This is a work in progress, so:

* The implementation is a bit crude – the translations are baked in so need to be pulled out to separate files for maintainability
* Not all of the UI elements are currently translated
* Because some of the translated words are longer than their English equivalents the UI does get mucked up a bit
* I used Google for the translations – better than nothing, but can doubtless be improved (my Russian colleague was very polite…).

But it is a start!

I should have the code tidied up some time after the WGM, so leave a comment (or contact me directly) if you’d like to contribute a translations file and I’ll give a shout when ready.

Now, back to that presentation…

2.59 clinFHIR Resource Builder: version 2

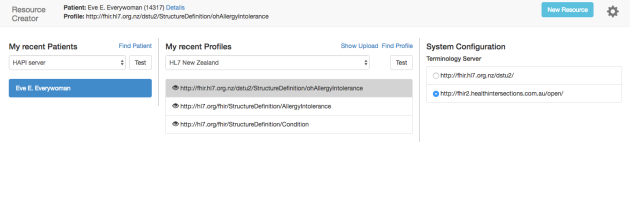
APRIL 24, 2016 [1 COMMENT](https://fhirblog.com/2016/04/24/clinfhir-resource-builder-version-2/#comments)

In the past few months I’ve been working on the [clinFHIR](http://clinfhir.com/) resource builder that we use at the ‘Clinicians on FHIR’ events at the Working Group Meetings.

In fact, this really became a complete re-write of the application, as there were a number of issues that I just couldn’t fix in the current version – especially around the rendering and creation of extensions. This has resulted in some changes to the User Interface, so the purpose of this post is to describe those changes when creating and viewing resources. It’s not quite complete, but ready for people to give it some user testing.

The app itself can be accessed from the main [clinFHIR.com](http://clinfhir.com/) site – just click on the ‘resource builder’ link. (Note that the app is hosted on a different server at the moment- I’ll move that later, though the link will always be up to date).

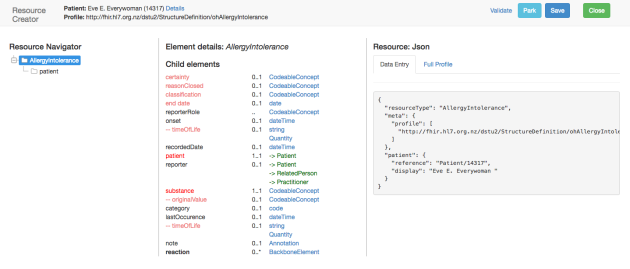
Once invoked, the ‘front page’ of the app will be displayed – it looks like this:

The idea is that you select the patient in the left pane, the profile you wish to use to build the resource against in the middle pane, and terminology server on the right – though that will generally default to the best choice. It is quite possible to use different servers for patient (plus the resources for that patient) and the profile on different servers, so long as they are on the same STU version. You’ll be warned if you select from different versions, though the tool won’t prevent you from proceeding. In the example above, the HAPI server is being used for patient and data, the HL7 NZ to access profiles, and Grahames server for terminology.

Both panes work in the same way – you select the server from the drop down (currently there’s a hard coded list) and then a patient or a profile from that server. (There’s also a ‘test’ button that allows you to make sure that the server you select is actually working – these are test servers, so no guarantees). The tool will remember your previous choices, so you can easily re-select them – these are saved in the local browser cache, and are not user specific. (To clear the local cache – which you have to do if the underlying profile is changed – click the ‘gear’ icon to the upper right and select the clear cache option).

Patient selection is by name (I’ll add the ability to create a new patient shortly) and Profile selection uses the same dialog as previously – you select the resource type you want, and then either that type directly or a profile on that type.

Having selected Patient and Profile, a button labeled ‘New Resource’ will appear in the top right of the screen. Clicking on that button will load the actual resource builder UI. Here’s a screen shot:



This screen also has 3 panes.

* To the left side is a tree view that acts as a navigator to the resource. As you add elements to the resource, they will be added to the tree.
* In the middle pane is a display that will show the value of the currently selected element, and all of the ‘child’ elements of that element. When the resource is first loaded, the root of the resource is selected in the navigator, and the middle pane will show the direct children of the root.
* In the right hand side is the pane where the json view of the resource is displayed, and where the data entry screen for an element is displayed when an element datatype is selected in the middle pane. This is the same as the previous version.

To add an element to the resource, first select the ‘parent’ you want to add it to in the navigator, and then select the element itself in the middle pane. The display in the middle is very similar to the previous version, with one big exception. If the datatype is ‘BackboneElement’ (like the reaction in the example above) then it is a node that can have children, rather than a value. Clicking on one of these will add a node to the navigator and then display the possible children in the middle. Multiple child elements can be added in this way (if the underlying profile permits).

Any other datatype will result in a data entry form being displayed in the upper right hand side – exactly the same as in the previous version.

Note that extensions are displayed ‘in-line’ with the other elements (extensions are normal in FHIR remember) with only a different colour to distinguish them. ‘Certainty’ is an example above. If the extension has a a couple of ‘minus’ signs (–) to the left, then it is an extension against the element directly above it rather than against the resource root. Note that at the moment only ‘simple’ extensions are supported – not complex ones containing ‘child’ extensions.

So you build your resource in this way:

1. Select parent node in the navigator
2. Select element from the middle pane
3. Enter data in right pane
4. And repeat

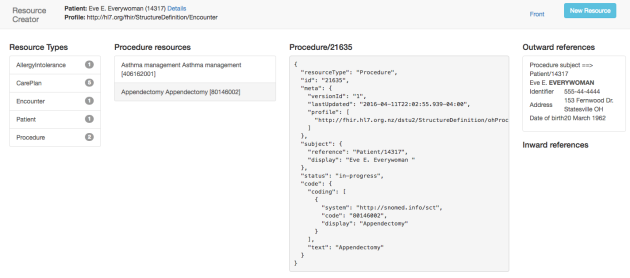
You can also now delete an element by selecting it in the navigator, then clicking the ‘delete’ button in the middle pane.

The parking function still also works – though you do need to be in the ‘build’ screen with the patient selected before you invoke it.

To save your resource (which will go to the same server as you selected the patient from), there is a ‘Save’ button to the upper right of the navbar. Alongside the save button is a link labeled ‘validate’. This is handy, as it will tell you if the resource is valid – ie if it will be accepted by the server (saves disappointment later  ). This link actually used the [validate operation](http://hl7.org/fhir/resource-operations.html#validate) on the server – it sends a request like:

POST [server]/[resource type]/$validate

Once saved, the resource can be viewed in the Resource Viewer – which you access by clicking the ‘details’ link next to the Patients name on the left side of the navbar. This is similar to the screen in the previous version – the only real difference being that the inward and outwards links are in a column to the right of the display rather than either side (there is less room to play with). Here’s a screen shot:



To return to the builder screen click the ‘New Resource’ button. The ‘Front’ link returns to the front page, though you’ll lose any new resource you may be building.

So that’s a quick overview of the new builder – it still has some rough edges and bugs that I’ll address between now and the Working Group Meeting, but do have a play with it and let me know of any errors you find. You can email me directly – david.hay25 at gmail.com –  but a better choice might be the [stream in FHIR Chat](https://chat.fhir.org/#narrow/stream/clinFHIR) (where all the cool kids hang out  ).

There are some other capabilities of the new tool that I’ll describe in some upcoming posts – notably viewing a profile and ad-hoc querying of a FHIR server.

I’m also planning a post describing how to create and upload profiles using the community tools – [forge](http://fhir.furore.com/Forge) and clinFHIR extension builder.

2.60 Montreal Connectathon

APRIL 8, 2016 [3 COMMENTS](https://fhirblog.com/2016/04/08/montreal-connectathon/#comments)

I would imagine that most readers of this blog are aware of the Connectathons that we hold at the beginning of each Working Group Meeting. These events are critical to the evolution of FHIR as an ‘implementer friendly’ standard, so we love to have as many people present as possible!

Each event is divided into a number of ‘streams’, with each stream exploring some aspect of FHIR – almost always some change that is being proposed to the spec – it’s the opportunity to find out if the proposal is something that should be incorporated into the main spec – or whether it doesn’t really add value. The significance of this is that the Connectathon is based on a ‘ballot candidate’ of FHIR – not the current version. So the version that will be tested at the next Connectathon is the [candidate for STU3](http://hl7.org/fhir/2016May/index.html) which we’re hoping to be able to release at the end of the year.

We’ve got a decent number of streams this time around, so I thought it might be a good idea to list them here. If there’s one that interests you – or that has some impact on work that you are doing – then it will really pay to be there (and hopefully the person who makes the funding decision for you to attend will appreciate that!)

So here are the tracks at [this WGM](http://www.hl7.org/events/wgm052016/?ref=banner) – in alphabetical order. Each track links to the proposal, which describes the track in more detail – this page is just a summary.

[CCDA on FHIR](http://wiki.hl7.org/index.php?title=201605_C-CDA_on_FHIR_Connectathon_Proposal). How to create a FHIR document that is analogous to the CCDA standard. This work leverages the [DAF](http://hl7.org/fhir/daf/daf.html) (Data Access Framework) Implementation Guide and has as the long term goal the ability to convert between CCDA and FHIR documents.

[CDS Enablement Services](http://wiki.hl7.org/index.php?title=201605_CDS_Enablement_Services). This stream explores how to apply SOA principles to support Clinical Decision Support (CDS) within the FHIR framework. Note that we have 3 CDS streams this time – see the next stream and CQF on FHIR.

[CDS hooks](http://wiki.hl7.org/index.php?title=201605_CDS_Hooks_Connectathon_Track_Proposal). This is a proposal from the talented [Josh Mandel](https://www.linkedin.com/in/joshua-mandel-88347235)that describes a standard way for applications to call CDS services. It leverages [SMART](http://docs.smarthealthit.org/) and more details are [here](https://github.com/cds-hooks/cds-hooks/wiki). (We’ve actually done a [few posts recently](https://fhirblog.com/tag/smart/) on SMART if you’re interested).

[Conditional References](http://wiki.hl7.org/index.php?title=201605_Conditional_Reference_Connectathon_Proposal). One of the difficulties in a messaging paradigm is how to indicate the reference between resources – eg the Patient to which an Encounter refers. There are ways of doing this at the moment using [conditional creates](http://hl7.org/fhir/http.html#ccreate), but it’s a bit klunky, and this stream investigates whether there are better ways of doing this.

[CQF on FHIR](http://wiki.hl7.org/index.php?title=201605_CQF_on_FHIR_Connectathon_Track_Proposal). This stream is a spike looking both at Decision Support and quality reporting. There’s a lot of detail and references in the proposal, so take a look if quality measurements are important to you.

[Data Access Framework (DAF)](http://wiki.hl7.org/index.php?title=201605_Data_Access_Framework_(DAF)). The [DAF Implementation Guide](http://hl7.org/fhir/daf/daf.html) is shaping up to be an important guide internationally. It is essentially a profile (or Implementation Guide) on FHIR that expresses the ‘meaningful use’ information set – but in a computable way. It forms the core of the CCDA work (see above) and is likely to be the guide supported by SMART. It’s also likely to be the ‘base’ guide that other countries start with and adjust to need their specific needs. (see the SMART-ca stream below). This stream exercises both client and server.

[Financial Management](http://wiki.hl7.org/index.php?title=201605_Financial_Track_Proposal). Of greatest interest in the US, nevertheless the ability of FHIR to be able to transmit financial and claims based information is going to be an important aspect of information sharing everywhere.

[FHIR genomics](http://wiki.hl7.org/index.php?title=201605_FHIR_Genomics_on_FHIR_Connectathon_Track_Proposal). Surely one of the hot topics today is how to share genomics information. The genomics workgroup have proposed a new resource ([sequence](http://hl7.org/fhir/2016May/sequence.html)) and a number of profiles on existing resources to support this need. These guys are really organized!

[Lab Order Services](http://wiki.hl7.org/index.php?title=201605_LabOrder). Explores the workflow around the ordering of lab tests. There’s another workflow track below – see ‘workflow’ that uses referrals as the use case.

[Patch](http://wiki.hl7.org/index.php?title=201605_PATCH_Connectathon_Track_Proposal). Explores the issues in ‘patching resources – ie where only partial updates to resources are submitted, rather than a whole new resource.

[Patient track](http://wiki.hl7.org/index.php?title=201605_Patient_Track_Proposal). The most important track in the whole event! (Of course, I’m not at all influenced by the fact that I’m the co-ordinator for this…) This is an ‘entry level’ track for those completely new to FHIR who want to find out who easy FHIR is to use. It’s the one track that is not really exercising the standard, but rather allowing people to ‘ease themselves’ into FHIR.

[Scheduling and Directory](http://wiki.hl7.org/index.php?title=201605_ServicesDirectoryAndScheduling_Track). This track exercises the ability of FHIR to manage appointments, but even more importantly the directory aspects that this involves – for example maintaining and locating services for referrals (which leads to an overlap with the workflow track below). You’ll make my friend [Brian Postlethwaite](https://www.linkedin.com/in/brian-postlethwaite-91b9414) very happy if you do this track…

[SMART-ca](http://wiki.hl7.org/index.php?title=201605_SMART-ca_on_FHIR). We referred to this earlier. This is taking the SMART standard and applying it to a different country – Canada (which is appropriate given that the WGM is in Montreal!). It exercises the DAF IG as well by looking at how to adapt it to a different country. SMART is shaping up to be one of the more important ‘extensions’ to FHIR, so the ability to support different Implementation Guides is going to be really important.

[Structured Data Capture](http://wiki.hl7.org/index.php?title=201605_FHIR_Structured_Data_Capture). Or the ability to use a standard set of forms to collect and record structured data using the Questionnaire and QuestionnaireResponse resources. In some cases these can be automatically converted into resources. [Lloyd](https://www.linkedin.com/in/lloyd-mckenzie-6b6681) is very fond of this track.

[Terminology](http://wiki.hl7.org/index.php?title=201605_Terminology_Services_Track_Proposal). The ability for FHIR to interact with terminology servers is critically important to the accurate sharing of clinical information – ‘semantic’ interoperability. This track exercises the [terminology services](http://hl7.org/fhir/2016May/terminology-service.html) defined in FHIR. This is [Grahames](https://www.linkedin.com/in/grahame-grieve-952637) baby – ’nuff said.

[Workflow](http://wiki.hl7.org/index.php?title=201605_Workflow). The workflow capabilities of FHIR are one of the major focus areas for this release of FHIR. We’re looking to move from an order/OrderResponse protocol, to one in which there is a [Task](http://hl7.org/fhir/2016May/task.html) resource which acts as the co-ordinator of the workflow process. In this track we take a referrals scenario, and see if the task is going to meet the business requirements. As mentioned above, there is overlap with the scheduling steam here.

So there we are! A record number of streams to exercise even the most jaded implementer.

I look forward to meeting you in Montreal…

2.61 SMART – Scopes and Profiles

MARCH 23, 2016 [2 COMMENTS](https://fhirblog.com/2016/03/23/smart-scopes-and-profiles/#comments)

In a [previous post](https://fhirblog.com/2016/03/16/smart-security/) we looked at some of the ‘security related’ aspects of SMART. In this one we’re going to take a closer look at what the ‘scope’ is, and make a couple of comments on the use of Profiles.

Scope is an AOuth2 term that represents the range of functionality requested by (and potentially granted to) a client application by the Authorization Server. For example an app that displays a person’s vitals might want to be able to read and create Observations.

The sequence when requesting a scope goes something like this:

* After passing authentication (so we know that the user has a valid account in the EHR), the client app requests to be able to perform a particular set of functionality, and represents this in the scope string (which contains any number of individual scopes separated by spaces). For example, either of the following scopes would represent the Observations example above (read and write on a patients observations):

Patient/Observation.read Patient/Observation.write

Patient/Observation.\*

Note that the scope doesn’t specify the specific patient – just that it intends to read and write observations. It is up to the Resource Server to decide if the authenticated user can do this against any specific patient.

* The Authorization Server determines if the client can do this. This is completely up to the particular implementation, but generally it will make this decision based on the EHR capability, the User and possibly the selected Patient if this was an EHR launch. Part of the workflow may involve asking the user if this is OK – like when you load a new app on to your phone.
* The access granted (if any) is then returned to the client (in the access token) as well as being stored in a way that the Resource server can access it when deciding whether to respond to a request from that client. This could be by storing a copy somewhere else where the Resource Server can access it, or the Authorization Server could sign the Access token – or some other mechanism. So the app knows in advance what the Resource Server will likely allow it to do and can adjust its UI accordingly.

OAuth2 does not define the contents of the scope – it simply states that that is the mechanism by which access can be negotiated between client and server. So what SMART does is to [define a pattern for the use of scopes](http://docs.smarthealthit.org/authorization/scopes-and-launch-context/), which is one of the neat things about SMART, as otherwise every implementation would be doing something different – hardly helping interoperability!

As an aside, it’s interesting to note that the functions of scope overlap with the [conformance](http://hl7.org/fhir/conformance.html) resource that is usually used to describe the capabilities of a server (which resources it supports, which operations on those resources and other stuff), but it can equally be used by a client to state what its expectations of a server are.

It could be argued that it would have been more FHIR-like to use a conformance resource rather than scope. However, there are a few reasons why scope is the better choice in this scenario:

* There are some scopes that don’t have an analogue in conformance (e.g. a request that the EHR display a patient selection screen as described below)
* If SMART didn’t support scopes, then it would be diverging from OAuth2, which is not such a great idea.
* The scope is what can be presented to the app user during authentication  so they know what data the app is going to collect (and what actions it can potentially take against the EHR) which is an important part of the process

Refer to [the spec](http://docs.smarthealthit.org/authorization/scopes-and-launch-context/) for a detailed view of how scope works in SMART, but briefly there are a number of categories:

* Clinical Data
* Contextual Data
* Identity Data

Clinical data is further divided into:

*Patient specific* scopes (about a single patient) e.g. patient/Observation.read would be read access to the selected patients observations, patient/Condition.write to be allowed to create Conditions and so forth. The pattern for patient specific scopes is: **patient/:resourceType.(read|write|\*)**

*User level* scopes – information that the user has access to. Note that this not information *about* the user – it’s what they are allowed to see (This may be the same, but not necessarily.

For example:

* user/Appointment.read – read access to all the appointments that the user is allowed to view.
* user/Patient.read – if the client wants to select a patient directly.

Because there’s a whole page in the spec on scopes, we won’t repeat that here, but we will call out some of the specific ones of interest.

Selecting the Patient

If a client wants to have the ability to select a patient, then they do so by setting a particular scope. There are a couple of options:

* *launch/patient* means that the application is requesting that the EHR display a patient selecting form, and allow the user to select the patient. The EHR may limit the patients that a user can access, based on the user role.
* *user/Patient.read* means that the client application wishes to be able to select a patient from within the app (again, the EHR may limit who the current user can select)

Notes:

* The scopes above simply deal with the selection of the patient – other scopes (such as patient/\*.read) will be required to be able to actually do anything with the selected patient
* The EHR server has complete control over what it will allow. It may limit the patients allowed to be selected by the user or – for an EHR launched app – may only allow that patient to be selected
* There is no concept of placing the patient in context in the EHR – the client makes the appropriate queries which the EHR may execute.

The standalone app will definitely need one of these scopes if it is to be able to operate against specific patients. The EHR launched app may also need it, for example it may be launched without a patient selected.

Requesting a refresh token

As described in the previous post, an Access Token has a limited lifespan – commonly an hour. After this time, it is no longer valid, and the app would need to re-authenticate if it needs to continue to access the EHR data. The app can request the Authorization Server issue a ‘Refresh Token’ during the initial authentication process  – which will allow it to get a new Access Token simply by sending the Refresh Token to the ‘token’ end point of the Authorization Server. It does so by including one of a couple of scopes:

* online\_access – The Refresh Token is valid only if the app remains on-line (however that may be determined)
* offline\_access – The Refresh Token is valid even if the user goes off line. (generally the more useful)

Note that the Refresh Token also has a limited time within which it is valid – no longer than 24 hours is the recommendation. After that, the client will need to re-authenticate.

Find out about the current user

It’s important to note that in all of our discussions so far, we haven’t considered the identity of the user of that app. As long as they have a valid account in the EHR they have been allowed to access data (within the bounds of their access rights of course). For clinical users, that’s probably all that’s needed as they are accessing the details of other people and the whole EHR infrastructure is likely built around doing that in a secure fashion (respecting the patients privacy) but for Patient access it’s a bit different.

In general, a patient will only be able to access their own data. (In some cases of course this is not quite correct – there may be the ability to look up the data of other people like children or relatives, but in that case from a security model perspective they are acting just like clinicians – albeit with more limited functionality).

So what we need is a way for the user of the app to request access to themselves. This is where openID Connect comes into the picture.

The app puts a couple of scopes – **openid** & **profile** – into the authorization request, and if the Authorization Server accepts the request, then it will include an ID token along with the Authorization Token when it responds to the request.

There’s a slightly convoluted process for starting with the ID token and getting the information you need – which ends up as a url to a FHIR resource (Generally a Practitioner or a Patient). This is described at the bottom of the [scopes page](http://docs.smarthealthit.org/authorization/scopes-and-launch-context/) so go take a look there for details.

So that’s a brief look at scopes. As always, the definitive description is in the [spec](http://docs.smarthealthit.org/authorization/scopes-and-launch-context/). The other way in which a SMART client ‘negotiates’ with the Resource server is in the use of profiles – so let’s take a quick look at that.

Most people familiar with FHIR are familiar with the concept of Profiling. It’s the way that we take the defined resource (like an Observation), and make them suit a specific use case. They can:

* take elements out
* add new elements
* change how many of a given element can be present
* specify what terminologies and/or ValueSets coded element are bound to

and a bunch of other stuff.

Profiles are necessary because FHIR – by design – is a ‘platform’ specification. It is designed to be used anywhere in the world where healthcare interoperability is needed, and so there is a balance between being restrictive (‘this is the way to do it’) and permissive (‘just do what you need to do’).

But the amount of work the client has to do, is directly related to the ‘permissiveness’ of the profile – the more permissive it is, the more work they need to do.

To make this easier for implementers, SMART defines a [specific profile](http://docs.smarthealthit.org/profiles/) to use. Actually, it’s a wee bit more complicated than that, as details of the profile are being reviewed as part of the [Argonaut](http://argonautwiki.hl7.org/index.php?title=Main_Page) program – and likely to settle on the [DAF](http://hl7.org/fhir/daf/daf.html) (Data Access Framework), itself a work in progress.

Now, there’s a slight problem with that, as DAF is a ‘US realm’ specification, meaning that it is tailored towards US requirements. Other ‘realms’ (read Countries) will likely have slightly different requirements (though hopefully aligned with the US ones where that is possible). For example, one of the tracks in the next connectathon will be to use a [Canadian profile in SMART](http://wiki.hl7.org/index.php?title=201605_SMART-ca_on_FHIR) (and note that it specifically refers to using DAF as a base) .

My guess is that profiles are one of the areas where we are going to see the most change in SMART as more people start to use it. The security and the scope aspects will likely be common internationally, but the profiles will vary between countries. We will need a way for a SMART application to indicate what profiles it supports/needs which could well be using the conformance resource which can indicate supported profiles. Another possibility might be to use [Implementation Guides](http://hl7.org/fhir/implementationguide.html), which are a ‘layer above’ profiles.

We’ll also see ‘SMART security’ being else in other areas – like the rather exciting ‘CDS-hooks’ that is just starting up.

So that completes our ‘mini-series’ on the technical aspects of SMART. Hope you found it helpful! There’s a lot more to think about – I would recommend taking a look at [Argonaut](http://argonautwiki.hl7.org/index.php?title=Main_Page/Implementation) if you are serioussly interested in this stuff, it’s still in operation…

2.62 Using SMART to talk between systems

MARCH 16, 2016 [LEAVE A COMMENT](https://fhirblog.com/2016/03/16/using-smart-to-talk-between-systems/#respond)

A question I was asked was ‘Can SMART help the scenario where an EMR users wants to access data from another system for the patient they have in context’?

Take the situation where there is, say, an [HIE](https://www.healthit.gov/providers-professionals/health-information-exchange/what-hie) that contains information about a patient that is useful to share in care delivery. It might have the definitive list of the patients medications, all known prescriptions, or their allergies, or encounters – information of value to the clinician and exposed by FHIR interfaces. In New Zealand, it could be the proposed [national EHR](http://www.nzdoctor.co.nz/news/2015/october-2015/20/grand-plan-to-develop-a-single-national-electronic-health-record-for-kiwis.aspx).

There’s also a Primary Care (Ambulatory Care in the US) system ([EMR](https://en.wikipedia.org/wiki/Electronic_health_record)) that wants to be able to access that information in the context of their EMR. Now, you might say that the EMR should just consume those services – and it should. But that does require development on behalf of the EMR vendor, and does mean that all of the functionality needs to be provided by the vendor – we lose the flexibility and potential innovation that SMART ‘[sidecar](https://fhirblog.com/2015/07/10/fhir-smart-and-sidecar-applications/)’ applications provide.

For example some apps may be able to update the EMR data from the HIE data (if the EMR allows). There might be specialist apps – like chronic care apps – that are tailored to that condition and maybe calling on external decision support services that read data from both systems, and update the patients care plan based on that data.

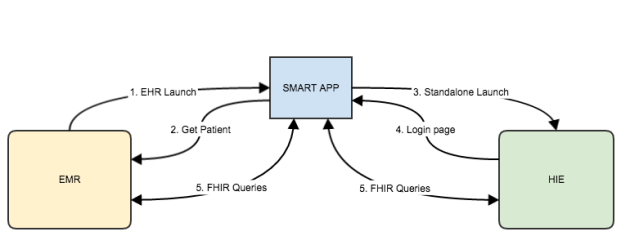
Can we use SMART to bridge the gap?

Well,  you can – and here’s how.

First some assumptions and caveats.

* This is really about clinicians accessing patient data.
* The clinician user needs to have an account in both systems. We’ll assume that it’s a standard username/password, though other variants are certainly possible. Life would be simpler if both EHR and EMR shared a common user identity source – but that’s not common at the moment.
* There’s some form of common patient identifier. Or – the HIE knows the identity used by the EMR. This isn’t an absolute requirement, but does make the Use Case more user friendly.
* The EMR supports ‘EHR launch’ – ie can launch a SMART app supplying the user and patient context.
* The HIE supports ‘standalone’ launch – ie a SMART app that is not launched from the HIE User Interface.
* There is a SMART app registered in both systems

Heres a picture:



And here’s how it would work.

1. The EMR user invokes the app, which authenticates the user in the [usual way](https://fhirblog.com/2016/03/13/implementing-smart-on-fhir-in-an-ehr/) for an EHR launch.
2. Because there was a patient selected, the Access Token from the EMR will have the patients id in the EMR. The app therefore calls the EMR FHIR Patient endpoint to retrieve the Patient resource. This has the patient identifier as a property.
3. The app then calls the authorize end point of the HIE (a standalone launch). The HIE responds by displaying a logon screen for the user.
4. The user logs in, and is presented with another Access Token from the HIE. (They now have 2 Access Tokens – one from each system).
5. And now the app has access to both systems. It has the patient identifier, and so can locate the patient in the HIE and the relevant clinical information.

At this point the skys the limit! – depending on the access rights that each system grants (which is one of the nice things about this approach – each system still controls the access rights of the user according to their own policies). The app can combine data from both systems, compare data from both systems, prepare summaries – whatever. It could reconcile the data between the two or perform a ‘sync’ – update the data in both systems.

And neither system had to do anything special – they just needed to implement SMART.

Of course, nothing’s perfect, and the need for the user to authenticate to the HIE each time is a bit tedious. However there is a solution to that as well.

Provided that the app is ‘private’ (i.e. can store a secret) then it could maintain a secure cache of Access Tokens – one for each EMR user. When the EMR user invokes the app, then the app could by-pass the HIE authentication step, and just present the Access Token for that user when querying the HIE. In this way, once the user has authenticated, they don’t need to re-authenticated (at least for the life of the Access Token -and there’s always the Refresh Token to make it last even longer).

Of course the solution above does require a significant amount of trust between HIE & EMR (at quite a number of levels  ). Saving and replaying the access credentials always makes security people twitch. Still, there’s no technical impediment that I can see.

In the next post, we’ll take a closer look at scopes

2.63 SMART – Security

MARCH 16, 2016 [1 COMMENT](https://fhirblog.com/2016/03/16/smart-security/#comments)

So in the [last post](https://fhirblog.com/2016/03/13/implementing-smart-on-fhir-in-an-ehr/) we went into some details concerning a specific SMART scenario – where a web based application is launched from the EHR (technically an EHR launch as described below). Let’s take a step back and take an overview of SMART as a whole, including some the other ways it can be used.

One way of looking at SMART is to think of it from 3 distinct perspectives:

The **Security Profiles / Authorization** part – how a client application can safely access patient specific data in the repository (safely in the sense that the Resource Server is satisfied that it is appropriate to release the requested data, and/or to accept updates to the identified user). This is the OAuth2 / OpenID Connect part. We’ll include the ‘modes’ of access in this discussion – what SMART refers to as the ‘Launch Mode’

The **Scope of access** – or what data and functions ares available to the client application. There’s actually a negotiation that occurs during authorization – the client application asks for access to a specific set of the patient data (Could be all, could be just observations etc.) and the operations it wants to perform against those resources, The Authorization Server decides if that information should be given to the current user.

This does have some overlap with the concept of privacy, though Privacy is more extensive. For example the Authorization Server may give a client access to Conditions, but the Resource Server may only supply a subset of those conditions, as some are sensitive and the policy engine restricts access.

This is also an OAuth2 concept.

The **Data and API part**. Having been authorized to access the patients data, what is the API to use to get or update that data, and how will it be represented (This includes the ‘bindings’ to terminologies).

This is the FHIR part, and we’ll need to think about FHIR profiles here.

Each one of these is a large topic, so we’ll consider security aspects here, and do the others later. Note this is by no means a comprehensive discussion on security – rather just calling out a few things of particular interest from the developer perspective in the context of SMART. (I am not a security expert – rather a developer with a desire to implement secure FHIR based applications).

Launch Mode

SMART defines two ways in which a SMART client can be initiated, and these have an impact on the details of how authentication is performed.

* The *EHR launch* assumes that the app is launched from within the EHR. For example there is a button (or some other link) within the EHR that will start the client application. In this mode, the user and (possibly) patient have already been identified, making the user experience more streamlined. This was the scenario that we went into detail [last time](https://fhirblog.com/2016/03/13/implementing-smart-on-fhir-in-an-ehr/).
* The *standalone* mode allows an externally started application to access the EHR data. In this mode the user will need to authenticate to the EHR, and will also need to be allowed to select the patient (the details of this ties in with scopes – more in the next post).

A web application could operate in either of these modes, while a mobile application will generally be in standalone mode (unless the EHR is running on the mobile device of course).

The big difference between the two is that in the EHR launch mode, we know the user (and possibly the patient), but in the standalone mode we don’t know either. This means that in the standalone launch there must be some way that the user (as well as the app) can authenticate themselves to the EHR. SMART doesn’t go into any detail about how that is to be done – it regards that as an implementation detail that can vary across different sites and OAuth2 offers a number of possibilities:

The simplest is that when the app is re-directed to the Authorization Server within the EHR (recall that it gets the url from the conformance resource), the Authorization Server requires the user to log in – possibly using the same login functionality as for a normal EHR user. All it needs to do is redirect to the login form, and for the login form to redirect back to the Authorization Server after a successful login. There won’t be a valid ‘launch’ token in the request so the Authorization Server can easily distinguish between standalone and EHR launched applications. (We might have glossed over some of the details  )

Refer to step 3 in the diagram in the [previous post](https://fhirblog.com/2016/03/13/implementing-smart-on-fhir-in-an-ehr/) for details.

An alternative could be to use an external Authorization server that the EHR trusts. The external Authorization Server performs the authentication and issues the authorization code and subsequently the access token. The Resource Server in the EHR then uses that access token to validate a request. There needs to be some way for the Resource server to ‘trust’ the access token – it could be signed with the Authorization Servers public key, or there could be a separate back channel between EHR and Authorization Server – it’s up to the implementation.

Another complication though is that even if the users identity has been ‘asserted’ by the trusted Authorization Server, the EHR (or Resource Server) still needs to apply privacy policies to the request (which can be down to the individual resource level) – and that implies either a common identity source for the user, or some way passing the user identity back to the Authorization Server.

These are issues that are being discussed in the community at the moment – the [HEART](http://openid.net/wg/heart/) project seems to be a focus for this, so this is a good time to get involved if you have these sorts of needs or an opinion on how they should be done.

But for implementers, the current SMART approach of having the Authorization Server as part of the EHR makes it easier to get started.

*It’s also worth pointing out that the standalone mode of SMART is really no different to the general case of an external client accessing a FHIR based API. In other words, assuming that you want to protect your API then SMART provides a standardized way to do this.*

HTTPS

Shouldn’t really need saying, but any exchange of real patient data must be over a secure connection – i.e. HTTPS (or [TLS](https://en.wikipedia.org/wiki/Transport_Layer_Security)). And note that in some jurisdictions this goes even further, requiring that any query that contains PHI (Personal Health Information) cannot use GET, but must use POST – the main reason this is to prevent that PHI being recorded in Server logs and Browser caches. FHIR offers the [\_search](http://hl7.org/fhir/search.html#2.1.1.2) query mechanism to support this.

CORS

This is not really related to SMART, but if you are writing an HTML based application, then you will likely be making HTTPS calls from the browser to the server, usually using javascript. This will generally violate the ‘[same origin](https://en.wikipedia.org/wiki/Same-origin_policy)’ policy and so the EHR server will explicitly need to support this – most commonly using [CORS](https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS). Fortunately, this is quite simple for a server to do – effectively it adds a few HTTP headers saying ‘it’s OK for code in pages served from other servers to call me’.

Public vs Confidential applications.

SMART divides applications into two categories depending on whether they are able to keep a token secure (this comes from the underlying OAuth2).

* A web app has the ability to be confidential if it saves the token in a safe area on the web server that is only available by server side code. If it stores the token in, say, a browser store then it is not confidential
* Public apps are all the others. In particular a mobile device is a public application as the device can be stolen

It’s not entirely clear what the effect of this difference is – other than the fact that the access token can last for longer. It does change the authorization workflow, so it seems simplest from the perspective of the server to assume that all clients are public.

Patient Access

Supporting Patient access to their data isn’t specifically a security issue, but it’s a question that is likely to increasingly asked and there isn’t yet a definitive answer.

A simplistic answer is that from the perspective of the EHR, the patient is simply another user – albeit one in a specialized role (and not one that the EHR has had to accommodate until recently). So provided that the patient has an account with the EHR in the appropriate role then it’s just a matter of authenticating in the same way as a clinician user. The EHR would likely restrict access to that persons records only (based on their role as a patient) plus any other security policies that may be in place (a common one being that access to lab data has to be ‘released’ by the orderer). More sophisticated functionality is possible of course.

But this is a model that is unlikely to scale. Does an EHR really want to have to create and maintain a user account for all of its patients? Seems unlikely. And users in general don’t like having lot’s of different accounts to remember – “why can’t I just use my google login”? This is where having an external Identity provider does make sense, and would provide a better user experience, but there are a lot of details to work out!

It’s certainly an area that is going to evolve over time.

General Notes

There are some [general notes on security](http://docs.smarthealthit.org/authorization/best-practices/) in the spec that is worth reading (and the page is likely to be enhanced over time).

And here’s a set of statements about apps that is short & to the point:

* Apps MUST assure that sensitive information (authentication secrets, authorization codes, tokens) is transmitted ONLY to authenticated servers, over TLS-secured channels.
* Apps MUST generate an unpredictable state parameter for each user session. An app MUST validate the state value for any request sent to its redirect URL; include state with all authorization requests; and validate the state value included in access tokens it receives.
* An app should NEVER treat any inputs it receives as executable code.
* An app MUST NOT forward values passed back to its redirect URL to any other arbitrary or user-provided URL (a practice known as an “open redirector”).
* An app should NEVER store bearer tokens in cookies that are transmitted in the clear.
* Apps should persist tokens and other sensitive data in app-specific storage locations only, not in system-wide-discoverable locations.

So that’s some notes on some of security aspects of SMART. If you are a supplier of a FHIR API, then it would make sense to align with this aspect of SMART as a step on the way to providing support for the standalone invocation pattern. Even if you don’t support the scopes components, using OAuth2 in this way makes sense.

Feel free to ask any questions here, or even better on the [FHIR chat](https://chat.fhir.org/#narrow/stream/smart) or [SMART support group](https://groups.google.com/forum/#!forum/smart-on-fhir). (and both have the history of previous questions & answers)

Oh – and in the previous post I should have included a link to the [Argonaut project](http://argonautwiki.hl7.org/index.php?title=Main_Page/Implementation) where this stuff is being worked on at the moment.

2.64 Implementing SMART on FHIR in an EHR

MARCH 13, 2016 [5 COMMENTS](https://fhirblog.com/2016/03/13/implementing-smart-on-fhir-in-an-ehr/#comments)

We’ve [talked about](https://fhirblog.com/tag/smart/) [SMART](http://docs.smarthealthit.org/) and [OAuth2](https://fhirblog.com/tag/oauth2/) before, but it was a little while ago and it was in the context of what SMART is about and how it worked (with the odd bit of sample code thrown in). This post takes a slightly different perspective by looking at SMART from the perspective of an EHR (Electronic Health Record) developer tasked with implementing a SMART interface to an EHR– what are they things that they need to consider adding SMART Interfaces ?

We’ll take a slightly roundabout way of doing that by starting with a brief overview and some key points about SMART, then diving into the details of the steps involved in a sample implementation. That’ll be enough for this post, then in a follow up we’ll take a closer look at the issues that our EHR developer will need to resolve. And later on, we’ll switch to the clients perspective.

So what is SMART?

Basically, SMART fulfills the use cases around supporting small, independent applications that want to provide specialized functionality in the context of an EHR (Electronic Health Record). The example often given is that of an application that displays pediatric growth charts. Launched from within the context of an EHR, it can pull data from the EHR to display charts of a childs development progress and potentially interact with other services to provide customized services such as decision support to the end user. In effect, it adds this functionality to that of the EHR to give a richer experience to the clinician – something that Wes Richell has called [‘sidecar’ applications](https://fhirblog.com/2015/07/10/fhir-smart-and-sidecar-applications/).

To do so, it takes a number of well accepted existing standards – both within healthcare and more generally in the Internet community, and creates a profile of those standards – a pattern of use – to meet that need.

These standards are:

* [FHIR](http://hl7.org/fhir/index.html). Describes the content of data being exchanged, and the [query API](http://hl7.org/fhir/http.html)to be used. FHIR is Health specific.
* [OAuth2](http://oauth.net/2/). An Authorization framework widely used outside of health – particularly allowing users to permit client applications (like mobile and web) to access their data. It permits the authentication (are you who say you are) and the authorization (what are you allowed to to) processes to be delegated to separate  ‘*Authorization Servers’* so that the application supplying the data (the *‘Resource Server’*) doesn’t have to do this – it can delegate that to the Authorization server.
* [OpenID](http://openid.net/connect/) Built on top of OAuth2, this supports user identity – personal information about the user to be given to the client application.

The huge value of this  standards based approach is that the specialized application can then work against any EHR that supports the SMART interface, in effect helping to establish the ecosystem of health care applications that will support innovation across the healthcare sector by creating a market place for application developers to exploit specific niches.

To understand how SMART works, we need to describe the main components – the building blocks – that we will use.

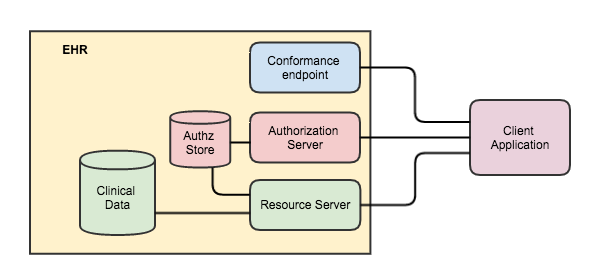
The *Client application* is the SMART enabled application that wishes to access data from the EHR – or to update it – in order to perform its specialized function (displaying growth charts in this example). While it uses (and potentially updates) EHR based data – there is nothing that prohibits it getting data from other sources, or calling on other sources (like immunization protocols) as it performs its functions.

The *EHR,*or *Electronic Health Record* is the system which holds the information that the client application wishes to access, and Is responsible for managing access to it as well as it’s overall integrity. It can be considered as the Custodian of the data. In many (but not all) of the Use Cases, the client application will be invoked from within the EHR. Part of the work in developing a SMART EHR interface is creating a way to actually invoke the client application – a process that we’ll talk about a bit later on

In SMART speak, the EHR has a number of sub-components that are separately defined in the underlying OAuth2 spec, but are managed by (and are really a part of) the EHR application. These are:

* The *Resource Server*. This is the component that holds the desired data and is responsible for delivering / updating it. In SMART, this is an implementation of a FHIR server. The more fine-grained privacy will likely be implemented by this component (Again, we’ll talk about this a bit later on).
* The *Authorization Server* is the component responsible for confirming that the user of the client application is who they say they are (a process called Authentication for the geeks out there), and for determining the precise access that a given user can have to a given patients records (Authorization).
* The *Conformance Endpoint* is part of FHIR, and allows a client to discover the capabilities of the FHIR Server.

Here’s a picture of how these components interact.



Note that making these components part of the EHR simplifies the security model in this version of SMART. Note that the terms ‘Authorization Server’ and ‘Resource Server’ are logical terms – physically they may be the same application.

It is also entirely possible that future versions will support a more distributed model, where – for example – the Authorization Server sits outside of the EHR, and manages the access rights to multiple EHRs. This is something that the base OAuth2 specification supports, but it is complex (and therefore expensive) to implement.

Small steps!

So, as promised, let’s jump right into the detail of how all this works in a specific Use Case. The example that we’ll use is where there is a vendor who has created the worlds best paediatric growth chart application. Their application is web based, and so can be started simply by calling a specific Url from within our EHR.

Our user (a clinician) is using the EHR. They will select a patient, and then click a button labeled ‘Growth Chart’. After a short (hopefully) delay, a window will open with the Growth Chart application User Interface populated with data from the EHR system. (Of course, the application could also open inside of the EHR so the user has no idea that the functionality is being provided by an external application, which would be even cooler…)

They interact with the application – viewing data, entering new information, making orders – and when finished close the window. All of the changes they have made are reflected in their EHR.

So what happened under to hood to make this happen? Well…

**Detailed steps**

We’ll now look at the sequence of events that made this happen. If you’re come here to find out what SMART can do, then you probably want to stop reading now. If you like the details, them read on…

There are a few prerequisites for all this to work properly. Essentially the client app needs to be initially registered with the EHR, providing:

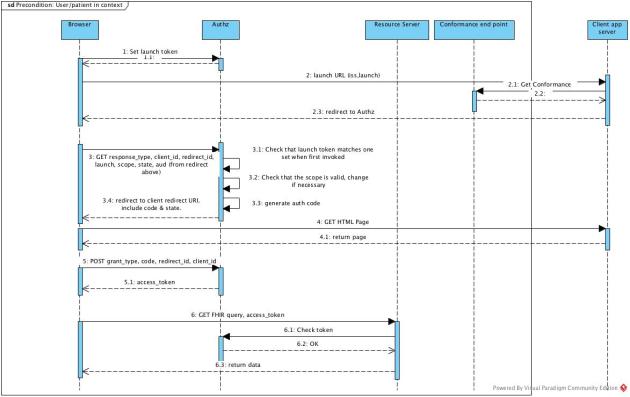
* A *client\_id* that uniquely identifies the client application to the EHR.
* The *launch\_url* for the client application has been supplied- so the EHR knows where to call the app.
* The *redirect\_url* for the application has been registered – so that the EHR can redirect after authentication has successfully completed.

(We’ll think about how all this might happen in a later post).

Also – for those looking at this who are already aware of some of the SMART concepts, the app is assumed to be a ‘public’ application – ie it is not capable of keeping a ‘secret’ token. This has implications on the workflow, which is why we need to be aware of this. We’ll talk about this a bit more in the next post, so you can safely ignore that for now.

Architecturally, this will be a [Single Page Application](https://en.wikipedia.org/wiki/Single-page_application) – running in the browser, perhaps using a framework like [Angular](https://angularjs.org/).

Now for the details. First a sequence diagram:



And the description of the steps – with the numbers corresponding to those in the diagram.

s

|  |  |
| --- | --- |
| 1,2 | The EHR user invokes the launch Url to start the application (eg create an iframe with the Url pointing to the client application launch Url). As part of this process (prior to creating the iframe) the EHR saves the current context (User, Patient etc.) in some common place where the Authorization Server can access it, with a ‘launch’ parameter (see below) below being the key to it. (There are other ways that the EHR can talk to the Authorization Server, but this will likely be a common one)  The HTTPS call to the client application includes the following required parameters:   * **iss** – (short for issuer) the url of the base FHIR endpoint of the EHR. This allows the client to query for FHIR data – including the conformance resource, which holds the location of the *authorization* and *token* end points (more to follow). * **launch** – an opaque token that is the key into the common store recording the current EHR context (patient, user). (btw – there are a number of different ways of doing this – for example it’s also feasible that the launch token actually contains the state directly if strongly encrypted and signed) |
| 2.1 | The client application (which is hosted on a web server, of course) retrieves the servers conformance resource (which it can locate using the value of the *iss* parameter – ie {iss}/metadata). The Conformance resource will contain the *authorization* and *token* endpoints (as well as lots of other goodies). These are represented in the Conformance resource as standard extensions. |
| 3 | The client application then redirects the browser to the authorization endpoint in the EHR passing in a number of parameters.   * **response\_type** – fixed to the value ‘*code’* * **client\_id** – the id of the client. This is the one created when the client app was registered with the EHR * **redirect\_uri** – the uri that the EHR will redirect to after successful authentication. This must be the same as one that the client app set at registration time * **launch** – the opaque token that it was given by the EHR when first invoked (and is the key to the context) * [**scope**](http://docs.smarthealthit.org/authorization/scopes-and-launch-context/) – a string that describes the access rights that the client application needs to operate properly. This is a string (containing a number of words) in a specific format that we’ll consider in the next post. But for example   + *patient/\*.read* asks for permission to to be able to read all the data for a single patient.   + *patient/Condition.write* asks to be allowed to create conditions.   + *offline\_access* and *online\_access* are included if the client wants to get a refresh token. This will allow them to get a new access token after their current one expires. The difference between the two scopes is that the online\_access is only valid if the user remains on line (it’s not specified how to determine that).   + There are 2 scopes from openID that are significant here. If the values ‘openid’ and ‘profile’ are present, then the client application is requesting that they receive details of the current user – we’ll also think about that in a future post * **state** – an opaque token set by the client. The server (the EHR) will include this token when redirecting after successful authentication * **aud** – the url of the resource server. This is the same value as the *iss* parameter that the EHR included when it invoked the client app. |
|  | Authz server: Authenticate the request  The authorization server then authenticates the request. If the value of the *launch* parameter matches the one that the EHR set when first invoking the client app, then it can assume that it is the registered user (assuming sufficient entropy in the launch token).  (If this is not acceptable from a policy perspective, then there will need to be a separate step where the user re-authenticates, eg by user name & password. This may not be an attractive user experience). |
|  | Authz server: Authorize the request  The Authorization Server then decides whether it will grant the application the requested access rights (as listed in the scope parameter). It knows the user and the patient (from the original context of the call keyed by the *launch* token from the common store) so can apply privacy policies.  It may grant more or less that what was requested. The scope granted will be included in the Authorization token that is eventually issued (see below) so the client knows what it can do, but the Authorization Server will keep a copy locally so it can be supplied to the Resource Server when the client eventually makes a request.  It is also possible that the EHR request further information from the user at this point. For example:   * If the user client app is requesting access to sensitive data the EHR may require a ‘break glass reason’. * If the user is a patient, then the EHR may wish to be certain that the patient is comfortable with the app getting that data.   Remember that it is the app that is requesting the access – not the user of the app – and there’s nothing stopping the app storing that data it receives from the server elsewhere, so these steps may be necessary.  Note that if the client application has requested access to the user profile (by setting the *openid* and *profile* scopes), the Authorization Server will also check whether those are permitted. |
|  | Authz server: Respond to the request  Assuming that Authz server is satisfied that the user can have at least some access to the data, it will redirect to the clients *redirect\_uri* passing 2 parameters:   * **Code** – a short lived token (string) that is created by the Authz server that the client will use to exchange for a longer lived access token. Referred to as the authorization code, it’s a key to some internal store in much the same way as the *launch* token was. * **State** – the same token that the client supplied (so it can check that the return is valid)   Of course, the Authorization server may decline the request for a number of reasons. These aren’t yet detailed in SMART, I asked [Josh](https://www.linkedin.com/in/joshua-mandel-88347235?) and he pointed out that looking at the [OAuth2 spec](https://tools.ietf.org/html/rfc6750#section-3.1) we can see that there are a number of possibilities depending on the nature of the error:   * Missing token: HTTP Status code401 * Expired/bad token: 401 * Good token, but not appropriate for the requested data: 403   These messages should also come with a WWW-Authenticate Response Header response header, and an error code (see the link above) |
| 4,5 | The browser will retrieve the actual HTML page for the application, and once it has loaded will make an AJAX request to retrieve the Access Token from the Authorization Server. (This means that the Authorization server will need to implement [CORS](https://en.wikipedia.org/wiki/Cross-origin_resource_sharing), to allow the cross-domain request).  (The Access Token is really the key to the kingdom, and has a number of properties that are described below, but the main one is the one that it must supply when subsequently making a call to the Resource Server to get patient information. This is how the Resource Server can check with the Authorization Server that the call is legitimate.  The authorization code is generally short lived (ie the Authorization server will only accept it as valid for a short time – maybe a minute) and it’s purpose is to inform the Authorization server that a longer lived access token (typically lives for an hour) can be issued to the caller.  It is the access token that the Resource Server will examine when responding to a request. It is possible for the Authorization server to ‘refresh’ an access token – ie issue a new one when the old one has expired without having to repeat the handshake above – this is described below.)  So: the client make a POST to the Authorization Server (using application/x-www-form-urlencoded) containing the following parameters:   * **grant\_type** – fixed to ‘authorization\_code’ * **code** – the authorization code that was given to the client in the previous step. * **redirect\_uri** – the same uri used in the initial query – step 2 * **client\_id** – the id of the client (as used in the original query, and how the Authorization Server knows which client application is making the authorization request.   The Authorization server checks that the code is still valid, and if so responds with an access token that is a JSON structure with the following properties:   * **access\_token** – the actual token (string) that the client will include when it makes a request to the Resource server. Confusingly, it has the same name as the whole structure! * **token\_type** – fixed to *Bearer* * **expires\_in** – how long (in seconds) that the access token is valid for. * **scope** – the authorized scope (in the same format that the client requested. This doesn’t have to be the same as what the client requested – the server may allow more or less than the requested * **state**– the opaque token that the client asked for in the initial request (see step 2) * **id\_token**– if the client application requested information about the user, and the Authorization Server allowed it then an id\_token will be included in the access token. * **refresh\_token**– the access token has a limited life span (an hour is recommended). The refresh token is used when the access token has expired to get a new access token. The refresh token also has a prescribed life span – 4 hours is recommended – which means that it must be used within 4 hours of issuing. This is only included if the client included either a *offline\_access* or *an online\_access* scope in their initial request (and the Authorization Server permitted it) * **Patient**– the id of the patient, if there was a patient in context when the app was launched. Other context parameters are possible. |
| 6 | Client makes FHIR requests on the resource server  Once the client has been authenticated, it can make standard FHIR queries against the data in the Resource Server. It passes the *access\_token* property of the access token in the *Authorization* header (somewhat confusing terminology IMHO). The resource server will validate that the token is current, and that the query is within the allowed scope before responding. It may also apply other privacy rules, depending on the specific configuration and data requested. |
|  | Refresh Access token  Not shown in the diagram is the ‘refresh token’ stage. When the Access Token expires, if there is a refresh\_token in the Access Token, the Client can get a new one by again calling the ‘token’ end point, passing in the refresh\_token and they will be issued with a new Access Token (probably). See [the spec](http://docs.smarthealthit.org/authorization/) for details. |

So that’s the detailed flow for a single Use Case – an EHR user invoking a SMART application which can securely access data from the underlying EHR (the Resource Server). Note that this is just one implementation – there are many others that could be used. SMART really focuses on the interface between EHR and Client – not the internals of how the EHR (or client) do their work.

Which brings us nicely to the question of libraries. As we’ve seen, Outh2 is a complicated specification – and it’s very easy to have bugs when writing code to support it, which is not a good thing – especially in the security implementation! There are many [libraries](http://oauth.net/2/) available that will perform these steps that have been thoroughly tested, and you are strongly advised to use one of them rather than trying to do it yourself.

And for clients, there’s also a [javascript library](http://docs.smarthealthit.org/clients/javascript/) provided by the SMART project team – plus others for [python](http://docs.smarthealthit.org/clients/python/) and [iOS](http://docs.smarthealthit.org/Swift-SMART/).

In the next post, we’ll take a step upward and consider some of the higher-level aspects of SMART – different Launch schemes, more about Scopes, Confidential vs Public applications and so forth. We’ll also think about the FHIR profiles that are used, and touch on User Details – the openID part of the specification.

And lastly, [here](http://docs.smarthealthit.org/) are the official SMART specifications. The [page on Authorization](http://docs.smarthealthit.org/authorization/) is well worth a read (most of the contents of this post came from reading that page – plagiarism at its best! – or the most sincere form of flattery…)

2.65 Building your own FHIR server

MARCH 6, 2016 [5 COMMENTS](https://fhirblog.com/2016/03/06/building-your-own-fhir-server/#comments)

Update: There is a more recent post about this [here](https://fhirblog.com/2016/10/19/setting-up-your-own-fhir-server-for-profiling/).

[This](http://www.openhealthhub.org/t/howto-build-a-health-database-and-fhir-api-server-in-15-mins-using-open-source/155) is an interesting post describing the use of the HAPI library to rapidly build a FHIR server. Another option using the same library is the [CLI Library](http://jamesagnew.github.io/hapi-fhir/doc_cli.html) which is even easier – a single file to download and run. (And can be used to populate the server with examples and standard profiles/valuesets)

I’m not sure whether these apps are suitable for production use (though the underlying libraries certainly are) – but they make it absurdly simple to get a FHIR server of your very own up and running so you can learn more about FHIR.

If you prefer .net, then the [Furore API](http://fhir.furore.com/Products#api) is available for you – and there are others available as well.

FHIR is becoming (has become?) the common language and API for exchanging healthcare information – a [Lingua Franca](https://en.wikipedia.org/wiki/Lingua_franca) if you will – and these apps and libraries are part of the reason why, by making it so simple to get started.

Oh – and don’t forget the [test servers](http://wiki.hl7.org/index.php?title=Publicly_Available_FHIR_Servers_for_testing) immediately available on the web…

Another part of the reason is the active community around FHIR, and it’s worth mentioning that the [new chat](https://chat.fhir.org/) is now running. It uses [Zulip](https://github.com/zulip/zulip), and replaces the Skype chats which were not keeping up with the volumes. There’s also a more classic [discussion group](http://community.fhir.org/) available, and the email based [list server](http://www.hl7.org/myhl7/managelistservs.cfm?ref=nav).

So no excuse not to get involved!

2.66 Opening up clinFHIR – An example generator

FEBRUARY 20, 2016 [LEAVE A COMMENT](https://fhirblog.com/2016/02/20/opening-up-clinfhir-an-example-generator/#respond)

I’d always intended that [clinFHIR](http://clinfhir.com/) would be an open source project.

It all started when we were planning the ‘Clinician Connectathons’ – now ‘[Clinicians on FHIR](http://wiki.hl7.org/index.php?title=Clinicians_on_FHIR)’ a year or so back, and realized that we needed some sort of tooling to support the events – tooling that would allow a user to create resources – and view the references between those resources – in a way that made sense to a clinician rather than a techie, and didn’t require them to understand the ‘on the wire’ formats of a resource (unless they wanted to).

After a few false starts, the current version was developed that seems to meet the need of the events. In fact, there are a number of different tools under the clinFHIR umbrella:

* The Resource Builder is the main one. It takes the actual definition of a resource (a [StructureDefinition](http://hl7.org/fhir/structuredefinition.html) resource) – and uses that to build a User Interface that allows a user to enter information compliant with that definition and to build a sample resource instance.
* A Resource Browser that lets you explore the references between resources for a patient
* The Profile Builder allows a user to build a simple profile. While the [forge tool from furore](http://fhir.furore.com/Forge) is the official profile builder tool, this builder is intended as an on-line tool to allow a clinical user to perform basic profile building – removing elements, adding extensions and the like – again with an educational focus.
* The Extension Browser adds a user friendly face to querying a FHIR registry for extensions and creating simple ones.

Focusing on the Resource Builder, I hadn’t really intended that it would be capable of working against any profile (at the time we had pretty specific needs) but as it turns out there is a need in the community for something like this – generating example instances of a profile (whether created by the internal tool or the official forge tool) as well as presenting a UI for users to help them evaluate profiles as they are being built are two of the potential uses I’ve come across.

Trouble is, this stuff is hard to do – and there is only so far that a single part-time developer can go with this kind of work, so I’m at a point where I could use some help.

The original code base is (to be kind) a bit ‘krufty’ – it started with DSTU-1 and also reflected my learning of both FHIR and the Angular JS toolset that I used and it is really not in any state for other developers to understand (as a few brave colleagues have already found out!).

So what I’ve done is to pull the Resource Builder stuff out into a separate repository, and made it part of a stand-alone SPA (Single Page Application) that calls a FHIR server directly – i.e. it doesn’t use Node.js as a back end server.

The Resource Builder is part of a small application that creates sample patients and data – you can create a patient with a set of resources (Condition, Observation, Encounter), and then use the Builder to add any other resource you want. I’ve also pulled across the Resource Browser so you can explore relationships between resources, and added a ‘Vitals’ Table (Observations that are considered vitals – like height, weight, blood pressure etc.).

The intention is that it will act as a ‘container’ to allow the builder component (implemented as an Angular directive) to be enhanced and debugged, so it can be re-used by others as well as the main clinFHIR application.

The source is available on github at <https://github.com/davidhay25/FHIRSampleCreator>.

Just download it (clone or zip) and then set it up behind an HTTP server to serve the various files and then navigate to sampleCreator.html in a browser. If you’re using a mac, then there’s a file in the root directory – startServer.sh that will start the simple python browser that comes with the mac.

There’s a certain amount of documentation in the app (there are a number of Markdown pages and code comments) – and I’ll continue to add to it over the next few weeks – I’ll also continue to enhance the UI to allow more control over the sample resources created .

As mentioned above, the app is a Single Page Application developed in AngularJS. It calls a FHIR server directly, which means that the server must support CORS – currently it works against Grahames server, the HAPI server or a local server on port 8080 (I use the excellent [HAPI CLI server](http://jamesagnew.github.io/hapi-fhir/doc_cli.html) myself), but I’ll enable the ability to add different servers shortly.

So, if you’re interested, then download the app and have a look around. If you would like to contribute to the project then drop me a line at my email – david.hay25 at gmail.com and we’ll work out the best way to move forward (This will be my first proper Open Source project so any advice from more experienced developers will be much appreciated!)

There’s an on-line instance of the app [here](http://107.170.196.80/sampleCreator.html) – I’ll do another post describing how it works shortly.

2.67 Where did that data come from?

FEBRUARY 18, 2016 [3 COMMENTS](https://fhirblog.com/2016/02/18/where-did-that-data-come-from/#comments)

This post grew out of a question from one of the Analysts at Orion Health. We’re in the process of embedding FHIR pretty deeply into our product stack – and part of that involves creating FHIR interfaces to our existing data repositories.

This particular repository takes data feeds from a number of sources – mostly in the form of v2 messages, but also including CCDA documents – and from them extracts clinical data of interest such as encounters, procedures, problems and so forth. Because of the varied source of the data, one of the data items in the existing output that is displayed to the user is where it come from – ie which facility and possibly which application. (As much as we’d like to get the clinician, this data is not generally available in v2 messages).

So they question they asked me was – ‘where does this stuff go in FHIR’?

At first, this looked like quite a simple question – the [provenance](http://hl7.org/fhir/provenance.html) resource is “*a record-keeping assertion that gathers information about the context in which the information in a resource was obtained*”

But as we dug into the details, the situation for us became a bit murky.

The connection between provenance and the resource it refers to is from Provenance to Resource. This is in keeping with the general pattern we have in FHIR that resource references tend to be from the ones created later to the ones created earlier – otherwise resources need to be updated, which adds significant complexity to the server (and the client). For example a Procedure refers to the Patient – not the other way around. You can imagine all the issues if you needed to update the Patient resource every time you created some data.

The provenance resource would need to be a separate resource in the bundle and in order for the UI to be able to display the source alongside the resource details, then the UI code would need – for each resource – to locate the appropriate Provenance and extract the source details from there. Do-able, but not attractive.

It would be made a bit easier if we ‘contained’ the Provenance resource inside the one to which it refers – but then the reference direction becomes an issue. A ‘container’ resource is meant to have a reference to the ‘contained’ resource, and so the direction is wrong.

And then the actual Provenance itself doesn’t really seem to fit our particular use case. The Provenance.agent node seemed the logical place to store that information, but that is a reference to another FHIR resource, which we are not specifically identifying (in theory we could – it is an [HD](http://hl7api.sourceforge.net/v251/apidocs/ca/uhn/hl7v2/model/v251/datatype/HD.html) datatype, but we aren’t). We could of course just put the text there in the display element… (And, as we will see, it isn’t really the correct place anyway!)

And there’s significant capability in Provenance that we don’t need.

I asked the question on the FHIR list – and got a great response and a comment from [John Moehrke](http://healthcaresecprivacy.blogspot.co.nz/) made the penny drop.

What we’re actually doing is to build a model from information in the v2 message. In the case of a Procedure for example, we’d get most of the data from the PR1 segment. All we want to do is to be able to say that ‘we were told this by the GoodHealth Clinic’ – and we get that from the MSH segment. We aren’t wanting to track back to the original v2 message.

Now, if we DID want to be able to do so, then a better approach would be:

* Save the v2 message somewhere, and establish a URI that can refer to it.
* Create the various derived resources – like the procedure above, which might also involve identifying the other resources that it references (like Patient, Organization whatever)
* Create a Provenance resource that:
  + refers to those derived resources (and there can be more than one of course) using Provenance.target.
  + refers to the original v2 message using Provenance.entity.reference.
  + indicate the facility from where the message came (which is what we want here) using Provenance.entity.agent (modeled as a FHIR resource of course)

For our specific requirements – and the timeframes we have – this isn’t really an option for us, so for the moment we’ll create an extension that we’ll use for all resources derived from v2 messages. But – now that we have a better understanding of how this should work – we can consider adding this functionality in future versions.

As mentioned above I’m deeply grateful for the input of those in the FHIR community – I hope that I got it all right!

Oh, and this is a reference that came up in the conversation to a [blog post](http://www.ringholm.com/docs/04350_mapping_HL7v2_FHIR.htm) from [Rene Spronk](http://www.ringholm.com/column/rs_last_en.htm) about v2 -> FHIR mapping in general.

2.68 FHIR messaging

JANUARY 21, 2016 [LEAVE A COMMENT](https://fhirblog.com/2016/01/21/fhir-messaging/#respond)

This was a messaging question just posted on the FHIR List (and [Stack Overflow](http://stackoverflow.com/questions/34896847/fhir-messages-for-scheduling-of-encounters/34904609#34904609))

*We have a scenario where we are evaluating the use of FHIR to transfer information on planned encounters from a EHR application (source application) to an internet portal application (destination application). The event that triggers this message transfer is that an encounter is planned, or that an planned encounter is canceled.*

*Quoting*[*https://www.hl7.org/fhir/messaging.html*](https://www.hl7.org/fhir/messaging.html)*:*

*In FHIR messaging, a “request message” is sent from a source application to a destination application when an event happens.*

*Some questions/issues:*

* *First of all: Is the use of FHIR messaging suitable for this scenario?*
* *If I understand the FHIR website correctly, the source application should send a “Bundle” to the destination application. Correct?*
* *The destination application needs various information on the scheduled encounter, such as the location and the referral request. Am I correct in assuming that these “extra” information elements should also be in the bundle, and that the “main” element should somehow reference these elements ?*
* *How should the main element reference the extra elements? Is there an XML example that shows “intra-bundle” references?*
* *If I read the spec correctly, the destination application is supposed to send a response message back. What should this response bundle contain to indicate that the message is received OK?*

And [Lloyd](https://www.linkedin.com/in/lloyd-mckenzie-6b6681) replied:

*Messaging is most appropriate mechanism when:*

* *You want to pass a number of resources as a single package*
* *You don’t need the overhead of a document (table of contents, tight rules over rendering)*
* *There’s a potential need for asynchronous communication and/or for routed communication*
* *You want to do something more complex than simple CRUD operations*
* *You want to use transports other than HTTP*

*However, you can choose to use messaging for any communication. I.e. No-one will call you non-conformant for choosing to use messaging even if another paradigm might be more ‘typical’.*

*Messages are Bundle resources where the first entry is a MessageHeader. The MessageHeader points to the focal resource(s) using the ‘data’ element. In this case, the focus would be the Encounter. Additional resources can also be conveyed. Typically you use a profile to provide guidance around what should be included in the bundle vs. not. (element.type.aggregation identifies whether something needs to appear in the bundle or not). The only rule is that when you trace all the relationships between the resources in the bundle, they form a totally connected network. However, the associations can be in either direction. I.e. It’s ok to include things that point to the Encounter, not just things the Encounter points to.*

*Within the Bundle, references can be full URLs, relative URLs (based on the URL of the referencing resource) or UUIDs. You can see an example message*[*here*](http://www.hl7.org/fhir/message-request-link.xml.html)*.*

*For a notification-type message interaction, your response message would typically just contain a MessageHeader whose .response would point to the identifier of the original message hand have code of “ok”. In other cases, a message response might contain data. E.g. the results of a create or merge, the response to a query, decision support information, etc.*

2.69 Cleaning up clinFHIR

JANUARY 7, 2016 [LEAVE A COMMENT](https://fhirblog.com/2016/01/07/cleaning-up-clinfhir/#respond)

Just a short post for people who may have become frustrated with errors in [clinFHIR](http://clinfhir.com/) when selecting patients in the Resource Builder.

What was happening is the clinFHIR keeps a note of all patients who have had resources made for them (by anyone) through the tool – to make it easier to find them later. The list is grouped by server – ie depending on the [Data Server](https://fhirblog.com/2015/09/07/server-roles-in-fhir/) you have configured you’ll get the appropriate list.

The issue was that the current servers out there are all test servers – their purpose is to help develop the spec and ensure fitness for purpose, and this means that periodically the contents may be deleted as the resource structure has changed (or for some other purpose). So clinFHIR winds up holding on to some ‘stale’ references – hence the errors.

So there’s now an option off the main menu (it’s a gear icon at the top right of the screen) called ‘Clean local db’ that will remove these stale items from the list of patients. You shouldn’t need to use it often – and I’ve just cleaned up the current lists – but if you start getting errors where clinFHIR says it can’t load patients that it has in its list, then it’s worth a try.

And do feel free to contact me with any other issues, suggestions or whatever.

2.70 clinFHIR demo

JANUARY 6, 2016 [LEAVE A COMMENT](https://fhirblog.com/2016/01/06/clinfhir-demo/#respond)

We just had a meeting to discuss the format for the next ‘Clinicians on FHIR’ meeting at the [January HL7 Working Group meeting](http://www.hl7.org/events/wgm012016/?ref=banner). A couple of the folk on the call were unfamiliar with the [clinFHIR](http://clinfhir.com/) tooling we use to support the event, so I did a short demonstration and offered to put a few links here to more details.

In addition, my colleague [Viet Nguyen](https://www.linkedin.com/profile/view?id=AAkAAAE59H8BbmWQm0BAOnhnG2i3NxhcxCjraVI) has kindly allowed me to share a recording of a demonstration that he gave – you can access it below.

* [A FHIR learning application for non-technical folk](https://fhirblog.com/2015/05/24/a-fhir-learning-application-for-non-technical-folk/)
* [Finding Patients seen by clinicians](https://fhirblog.com/2015/07/11/finding-patients-seen-by-clinicians/)
* [Creating and using FHIR profiles](https://fhirblog.com/2015/07/26/creating-and-using-fhir-profiles/)
* [Using clinFHIR to create a profile](https://fhirblog.com/2015/07/26/using-clinfhir-to-create-a-profile/)
* [Building resources from FHIR profiles.](https://fhirblog.com/2015/08/03/building-resources-from-fhir-profiles/)
* [Creating Lists in clinFHIR](https://fhirblog.com/2015/08/03/creating-lists-in-clinfhir/)
* [The clinFHIR Chat](https://fhirblog.com/2015/08/19/the-clinfhir-chat/)
* [Server roles in FHIR](https://fhirblog.com/2015/09/07/server-roles-in-fhir/)
* [Talking with a Clinician](https://fhirblog.com/2015/09/10/talking-with-a-clinician/)
* [Creating reusable scenarios using clinFHIR](https://fhirblog.com/2015/09/19/creating-reusable-scenarios-using-clinfhir/)

2.71 Orders in different servers

DECEMBER 22, 2015 [3 COMMENTS](https://fhirblog.com/2015/12/22/orders-in-different-servers/#comments)

So in the [previous post](https://fhirblog.com/2015/12/17/orders-in-fhir/) we talked about how we could support the workflow track in the upcoming Connectathon (specifically for diagnostic orders) using a RESTful paradigm. There was quite a flurry of activity on the implementers chat about how this should work – with a number of differing opinions expressed about whether a separate set of resources (the Order / OrderResponse) is required or whether the detail resources (eg DiagnosticOrder, MedicationOrder) are sufficient. This sort of discussion is exactly what Connectathons are intended to promote to that’s all good!

Nevertheless, for this Connectathon we need to proceed with the current arrangement, which will help ‘flesh out’ some of these issues.

In the current version of orders in clinFHIR, all the resources are saved on a single server – which will not be the case in a real world environment. At the very least the clinical record (the EHR equivalent) will be on one server, and the server that processes orders (often called order fulfillment) on another. We’ve talked about [Server Roles](https://fhirblog.com/2015/09/07/server-roles-in-fhir/) before and at that time we defined 4 of them:

* A Patient server that knows about patients (perhaps more generically an identity server because we’d likely want to do the same for Practitioner and other ‘[Identification](http://hl7.org/fhir/administration.html)‘ type resources)
* A data server that represents the clinical record – the EHR/EMR
* A Conformance server that holds profiling (StructureDefinition) and related resources
* A Terminology server for expanding ValueSets – and the other [Terminology services](http://hl7.org/fhir/terminology-service.html) defined in the spec.

So we now add a 5th:

* An ‘Order’ server that responds to orders. This might also be called a ‘fulfillment’ server – but we’ll stick with ‘Order server’ for now.

(And just a reminder that a single physical server may implement multiple roles, and that there can be more than server with a given role in any implementation.)

Lets think about what our requirements are going to be from the perspective of the Order Placer for simple order processing in this multi-server world.

* We want to be able to create an order in our EMR (Electronic Medical Record – Data server) and store a copy of that order in the patient record for reference.
* We need be able to ‘send’ the order to the Order server so it will be actioned. For example in the case of a blood test this would trigger an entire workflow from the taking of the blood sample all the way through to the creation of the [report](http://hl7.org/fhir/diagnosticreport.html).
* We need to be able find out when the order has been completed, and copy the outcomes (eg a DiagnosticReport) into the Local EMR. (An alternative might be to have a ‘distributed record’ where the report remains external to our EMR and just reference it when needed -but even then we’d want to know when it was available).

There are lots of other more detailed requirements – like being able to track the ‘state’ of the referral, be alerted if a report is not done in a reasonable time and so forth, but that will do for now.

Before we go into the details of how this could work in FHIR, we need to review a few important background concepts – especially around identity. We’ve [talked about this before](https://fhirblog.com/2014/01/21/fhir-a-question-of-identity/), but to re-iterate a few of the important points:

* Every resource (apart from [contained](https://fhirblog.com/2013/10/10/fhir-contained-resources/) resources) has a unique id – actually a [URI](https://en.wikipedia.org/wiki/Uniform_Resource_Identifier) – by which it can be referred to. This allows us to build a network of resources to describe clinical (and other) Use Cases. The id must be unique on a given server, and can be expressed either as relative to the server where it is stored (e.g. like /Patient/100) or absolute (eg http:/myServer/FHIR/Patient/100). This difference becomes important when resources are stored on different servers, as we will see.
* Most resources can also have an [identifier](http://hl7.org/fhir/datatypes.html#identifier). This differs from the id in that the former is a ‘structural’ thing used to locate and link resources, whereas the identifier is a ‘business’ level concept – like a Medical Records Number, Drivers license or Social security number.

Some consequences of this include:

* A resource representing a single ‘thing’ (like a patient) can potentially be stored on different servers, and so will have a different id on each server, but the identifier (and other properties) will remain the same on each (hopefully!). To know that 2 patient resources actually refer to the same person, you need to look at the identifiers (or other properties) – not the id.
* If a resource on one server wants to explicitly refer to a resource on another server, it must use an absolute id – the server plus the id on that server. If it doesn’t have that id, then it will need to locate it by using a query (say by identifier), and deal with any duplicates that may occur.

So let’s think about deployment scenarios.

For a start there will be at least an Order server and a Data Server. There might be a separate Patient server as well, but let’s assume that the Data and Patient roles are served by the same server so we can focus on the interaction with the Order server.

For a start, when we create an order, we’ll have a copy on the Data Server (for the patient record) and then make a copy to send to the Order server where it will be actioned by the fulfillment process (whatever that may be). So the order of events will be as follows:

1. Create the detail resource/s (eg DiagnosticOrder) locally. It will have a relative reference to the patient as it’s on the same server. (Of course, if there were a separate Patient server then it would need to be absolute).
2. Create and save the [Order](http://hl7.org/fhir/order.html) resource locally. It will have a relative reference both to the Patient and the detail resources.
3. Save a copy of the detail resource on the Order service by POSTing it to the server. The Order Server will assign a new id, which is returned in the REST response (in the [location header](http://hl7.org/fhir/http.html#create)).
4. Make a copy of the order resource on the Data Server. Replace the local relative reference to the detail resource with the one that the order server assigned above, and POST that to the order server. Note that the response from the Order server is the standard HTTP response – not an OrderResponse resource.

(We could have chosen to use the transaction capability and have the server do all the work – but we can’t assume that the Order server will support that).

Job done! – actually no, we need to decide how all of our resources are going to refer to the patient when we send an order to the Order server (as well as the Practitioner and any other referenced resource). There are (of course) at least a couple of options.

* The order that we sent to the Order Server can have an absolute reference to the patient on the Patient server. This is the simpler option, but assumes a deployment scenario where the Order server is capable of referencing the external Patient server to get the patient details – and most applications will likely want to have a local copy of the patient.
* We can send a copy of the Patient resource to the Order Server using the [conditional create](http://hl7.org/fhir/http.html#ccreate) option so it can either create a new Patient resource or find previous one if this is not the first order for that patient. Assuming we take this approach, then there is a step 1a. in the list above where we POST a copy of the Patient resource (after removing the id element) to the Order server with the identifier in the *If-None-Exist* header. Of course, we’re assuming that the Order server supports conditional create – if not, then we’ll have to:
  + Query for the Patient on the Order server
  + If found, then update the reference in the Order and detail resources before posting.
  + If not found, then POST the new Patient to the Order server, and update the Order & detail reference as above when the Order server responds.

So we’ve sent our order to the Order server. How do we retrieve the result? Let’s assume that we don’t have a notification service (or any other ability to push results) in place – and that we want to continue to use REST – that implies that we need to poll the Order server for [OrderResponse](http://hl7.org/fhir/orderresponse.html) resources. However, looking at the OrderResponse resource we see that there is no link to the patient – only to the Order (and the resulting – fulfillment – resources).

We need to query the Order sever for the OrderResponse using the Order Id that was assigned by the Order server when the order was saved, which implies that we need to save that id when the order server saves our order in step 4 above. You might think that adding a Patient reference to OrderResponse will resolve this – but that reference would be to the Order Servers copy of patient, not to our own (unless both were referring to a common Patient server of course).

An alternative way might be to add an identifier to the Order. The Order server will save that identifier in it’s copy, and provided that it supports chained extensions would allow us to make a call like:

/OrderResponse?request.identifier= abc123

Return the OrderResponse resources where the associated order (the request) has an identifier of abc123.

But we probably can’t count on the server supporting that, so the easiest way right now will be to add an extension to our own copy of the order resource which is a reference to the Order Servers copy of the Order. When we want to update our list, we retrieve the extension (which will have the id of the order on the Order server and execute the query:

/OrderResponse?request = abc123

It’s a bit klutzy as we’ll need to update our own copy after it has been saved on the Order server, but it reduces the expectations on the Order server.

Of course, once we have the OrderResponse, if it references any ‘fulfillment’ resources – like a DiagnosticReport – it’s a straightforward matter to copy them to the Data server (though we will need to look for and resolve any relative references to other resources plus copy any dependant resources (like an Observation), so it’s not THAT straightforward…)

So we do have a few Requirements of the Order Server:

* Able to store Order and detail resources
* Conditional Patient update (to avoid multiple copies of the Patient being created)
* Perform processing on receipt of the Order resource including:
  + Create OrderResponse resources
  + Create result resources
* Support the ‘request’ query on OrderResponse
* Possibly support the chained query for OrderResponse.request.identifier

So this will all work – but the issues we discussed do show that there is room for improvement! If you’re interested in this stuff, then attending the Connectathon will be a great way to contribute to what is likely to be a robust discussion!

2.72 Orders in FHIR

DECEMBER 17, 2015 [8 COMMENTS](https://fhirblog.com/2015/12/17/orders-in-fhir/#comments)

One of the streams at the [next FHIR Connectathon](http://wiki.hl7.org/index.php?title=FHIR_Connectathon_11) is[Lab Order Services](http://wiki.hl7.org/index.php?title=201601_LabOrderLabReport) – a track designed to exercise the workflow components of FHIR. Up until now, most of the focus of the team has been on developing the base resources and supporting infrastructure, so now that that’s in a good shape, one of the next focuses will be workflow. There’s a lot of thinking going on right now –  there’s a specific [workflow project](http://wiki.hl7.org/index.php?title=FHIR_Workflow) looking at this, and [Keith blogged](http://motorcycleguy.blogspot.co.nz/2015/12/why-should-healthit-workflow.html)about it quite recently so the current way in which FHIR manages workflow is very likely to change significantly – probably as part of the [2.1 version](http://wiki.hl7.org/index.php?title=FHIR_Ballot_Prep) scheduled to be released towards the end of next year.

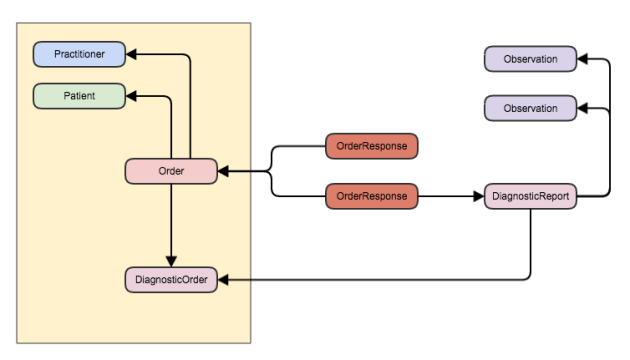
I was asked if we could use [clinFHIR](http://clinfhir.com/) as a client for exercising workflow at the Connectathon, so have spent a bit of time just recently thinking about how we could do that.

Before getting into the details, a quick review on how workflow is currently organized in FHIR (and remembering that this will change as a result of the workflow project.)

There is a clear separation between the ‘action’ (or actions) that is being requested and the information about what is required to complete that action. The action parts are represented by 2 resources ([Order](http://hl7.org/fhir/order.html) and [OrderResponse](http://hl7.org/fhir/orderresponse.html)).

* The Order specifies who is initiating the order, when it should be done and who it should be done on. It has references to the actual resources that represent what is being requested – eg a [MedicationOrder](http://hl7.org/fhir/medicationorder.html)(for a prescription), a [DiagnosticOrder](http://hl7.org/fhir/diagnosticorder.html) (for a lab test) or a [ReferralRequest](http://hl7.org/fhir/referralrequest.html) (for a referral). A single order can refer to any number of ‘information’ resources – and they needn’t be the same type.
* The OrderResponse resource then indicates the outcome (or outcomes) as the order is processed by the recipient (a process sometimes called ‘fulfilling’ the order). It has references to the Order that it is fulfilling and to the resources that represent the outcomes of the order.

Take a look at the picture below.



In the yellow rectangle are the resources that were created when the order was established. These are:

* The Order resource itself
* A Practitioner representing the person who made the order
* A Patient representing the target of the order
* A DiagnosticOrder that has the details of which tests to perform.

Then there are other resources that are created during processing/fulfilling the order.

A couple of OrderResponse resources – the first maybe indicating that the order was accepted by the lab and the second being generated when the result is available and also containing references to the DiagnosticReport that was generated with the results.

(Not shown in the diagram is that all of the resources – *with the exception of the OrderResponse* – have a reference to the patient. )

You can use this pattern through any of the supported paradigms. If you were using REST it could go something like this:

1. The client gets a reference to the Patient.
2. It creates and stores the DiagnosticOrder. (For simplicity we’ll assume that everything is saved on the same server – though in practice you might have one on the [Data Server](https://fhirblog.com/2015/09/07/server-roles-in-fhir/) as an ‘EHR’ reference in the patient record, and another on a ‘Workflow’ server where the action takes place). At this point the DiagnosticOrder is like any other resource.
3. It creates and saves the Order resource (referencing the DiagnosticOrder). This initiates the overall workflow.

We assume that the server (the ‘fulfiller’ of the order) is monitoring for the appearance of a new Order. The process that follows will depend entirely on how it is implemented, but could be something like this:

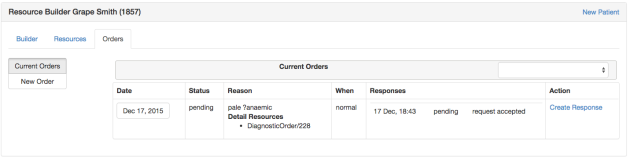
1. The server decides whether to accept the order (this may be automated or manual). It creates an OrderResponse to indicate this (setting the OrderResponse.status appropriately).
2. The order is actioned (eg blood is take from the patient and passed to the analyser which produces the DiagnosticReport/Observation resources). Potentially further OrderResponse resources are created at various steps so the client is aware of the progress of the workflow.
3. When the result is available, a final OrderResponse (with a [status](http://hl7.org/fhir/orderresponse-definitions.html#OrderResponse.orderStatus) of ‘completed’) is created with references to the Order and to the DiagnosticReport.

So how does the client know when the process is complete, or when significant actions occur? Well, that depends (again) on the implementation – it could be a polling process (a bit inelegant) or maybe a [subscription](http://hl7.org/fhir/subscription.html) mechanism that notifies the client in real time – it’s up to you.

So with that background, how can [clinFHIR](http://clinfhir.com/) help?

It already has the ability to create any resource (including from profiles – so long as they aren’t too complicated) so in theory it’s good to go. But it’s rather tedious to create all the relationship manually, so I decided that a specific orders interface to make it easier would be worthwhile.

If you load [this page](http://clinfhir.com/builder.html) and select a patient and then the Orders tab, you’ll get this interface:



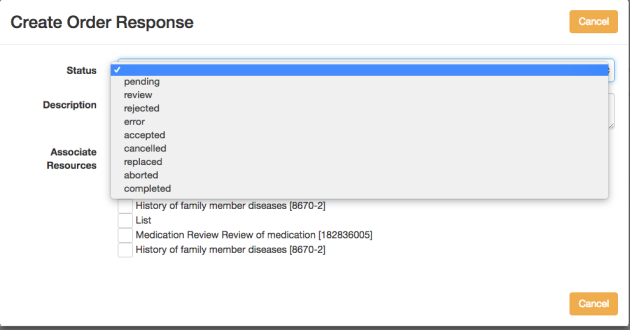
(Oh, as I was testing this I realized that clinFHIR doesn’t recognize when a patient has been deleted from the server, so if you get errors stating that the resource has been deleted, just try a different patient – or add a new one. I’ll tidy up that behavior soon).

There are 3 tabs:

* The Builder tab has the resource builder component – it’s the same one used in the main clinFHIR application
* Resources lists the resources for the current patient – again the same one as used elsewhere.
* The Orders tab is where you can manipulate orders, and looks like this:

The order display lists all orders for the current patient showing:

* The date the Order was created
* The current status. Technically, this is the status of the most recent OrderResponse associated with that Order as the Order itself doesn’t have a status. (If it did, then any process saving an OrderResponse would also need to update the Order – ie a transaction – that we try to avoid.
* The reason for the Order. This column actually has 2 items in it:
  + The reason itself (currently this is Order.reasonCodeableConcept.text – this needs to be improved as the reason can refer to an external resource)
  + A list of the detail resources associated with this order. Only the id of the resource is shown – we’ll see how to view the details in a minute…
* The ‘When’ column shows the order.when.text value – again, needs to be improved to show the timing element as well
* The responses column shows all the OrderResponse resources that reference this order. This has a sub-table in it showing the date, status and description of the response. If there are associated resources to the order (eg the DiagnosticReport) then they will be shown as well.
* The final column has a link that allows you to create a new OrderResponse against this order. In real life of course, this is created by the server processing the order – not the client, but it’s there so you can think about the full workflow process – and model the resources (or profiles) needed. Clicking on the link displays the following dialog:

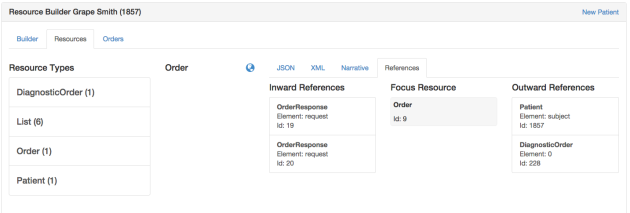


From here you can create a new OrderResponse resource, specifying the following fields:

* The new status of the order
* A description
* Associated resources. You would use this when you want to ‘complete’ the order. Before doing this, create the ‘complete’ resource in the Builder (maybe a DiagnosticReport with associated Observations) and then link to it from here.

(and just a reminder that you can do all of this from the Builder as well – this is just to make it a bit easier).

This is all rather complicated (actually, workflow IS complicated) and visualizing the end result is not easy. You may have noticed that the date column of the order is actually a button. Clicking on the button will display the Order in the Resources tab – where you can view all the relationships of the order. Here’s an example – the order has a couple of OrderResponse resources that reference it, and it has references to the Patient and DiagnosticOrder resource.

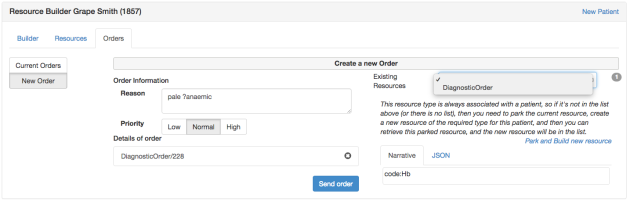


Remember that selecting any resource in the references tab will put that resource ‘in focus’ allowing you to view its contents (as XML or JSON) and the references that it has.

To create a new order you need to first create and save the ‘Information’ resource/s (eg DiagnosticOrder or MedicationOrder) and then create the Order to action it.

Use the ‘Resources’ tab to create the resource in the usual way – oh, and note that the Builder UI has been simplified a bit – to select a profile (which includes the base profiles for the resources), click the ‘Load new profile’ link, then select the base profile from the dropdown in the dialog that appears. You can then select that base profile directly or look for other profiles on that base profile (it makes sense when you do it!). This just reflects that in FHIR all resource definitions are, in fact, profiles (or [StructureDefinition](http://hl7.org/fhir/structuredefinition.html) resources to be precise).

To create the Order you can either use the builder, or create it from the orders tab. To do this, select the Orders tab then click the ‘New Order’ button to the left of the screen. Enter the reason for the order, the priority and the information resources, which will be in a dropdown to the upper right of the screen.



(To be honest, the UI is still a bit rough – it’ll be better by the Connectathon!).

Either approach achieves the same end – the Order resource, with links to information resources, is saved on the server, and will appear in the Orders list.

So that’s how clinFHIR can support the current orders workflow. It’s still a bit flaky – not all the components are properly set up, sometimes it doesn’t update correctly (just re-load the page if that happens) and remember that the whole workflow infrastructure in under review, so change is certain!

This is probably my last post before christmas, so I wish you all the compliments of the season and hope that you will enjoy the company of family and friends over the holiday break. Next year is going to be wild!

2.73 [Should I use FHIR in production?](https://fhirblog.com/2015/11/25/should-i-use-fhir-in-production/)

NOVEMBER 25, 2015 [LEAVE A COMMENT](https://fhirblog.com/2015/11/25/should-i-use-fhir-in-production/#respond)

A question was asked today on the Implementer’s chat that has been asked many times before: “Should I be using FHIR in production today?”

There are a number of answers to this – ranging from the fact that quite a lot of people actually are, through to some comments in [Grahame’s keynote](https://vimeo.com/146477763) at the [Amsterdam devdays](http://fhir.furore.com/Events/DevDays2015) concerning “Working Interoperability” and talking about the resource maturity model.

But I thought the most succinct one was this from James ([HAPI creator](http://jamesagnew.github.io/hapi-fhir/))

the point that i’ve always made when trying to make the “net benefit” argument is that even if you’re using FHIR in a completely closed environment, the alternative is to do your own data modeling and API design. no matter how good a job you do of that, you still have to do the work, and you’ll still never come up with something as robust as FHIR since FHIR has many peoples’ implementation experience baked into it. add to that the fact that FHIR is well documented (so you don’t need to do that part either) and you have a very compelling case.

Posted here so I can find it when the question is next asked!

2.74 Slicing extensions with FHIR

NOVEMBER 24, 2015 [LEAVE A COMMENT](https://fhirblog.com/2015/11/24/slicing-extensions-with-fhir/#respond)

So recently [I wrote](https://fhirblog.com/2015/11/04/creating-and-finding-fhir-extension-definitions/) about a simple ‘[extensions editor](https://fhirblog.com/2015/11/04/creating-and-finding-fhir-extension-definitions/)’ that we created to allow our [Orion Health](https://orionhealth.com/) developers to create Extension Definitions (actually [StructureDefinition](http://hl7.org/fhir/structuredefinition.html) resources of course) to describe the extensions they were creating when developing FHIR interfaces. I assumed that they were all ‘simple’ extensions – that is an extension that just had a single value – rather than extensions that were nested inside other extensions. Wrong!

These are  defined using ‘Slicing’, – so I needed to find out more about that works, and then implement it.

Slicing is one of the more complicated parts of FHIR (and I don’t fully understand it yet!). At its heart it’s all about taking resource elements that can appear more than once – and then being being more prescriptive about what the list contents can be in a particular profile. (That’s a gross over-simplification of course – check out [the spec](http://hl7.org/fhir/profiling.html#slicing) for more details).

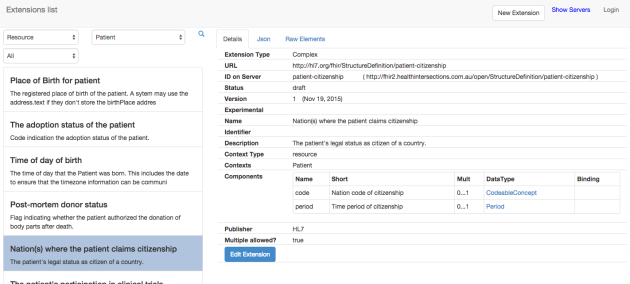
An example in the spec is that of a Blood Pressure – where you take the Observation.component element, and say that for this profile (technically a profile on Observation) we only want 2 component elements – one for the systolic and the other the diastolic – and defines the codes that should be used.

So in the profile (StructureDefinition) you first define a thing called a ‘discriminator’ – how to be able to tell the slices apart. The discriminator defines what the ‘unique’ property is in the slices.

Next, you create a group of elements for each slice that defines the contents of each slice – the value of the discriminator, the data type, multiplicity and so forth. There are a number of [examples in the spec](http://hl7.org/fhir/profiling-examples.html) that show how this works.

For extensions the discriminator should (?must) be the url – so that’s one thing settled, but I still wasn’t completely clear about what I needed to do, so I decided to take a look at the extensions that were defined in the spec, and use those as a guide. I updated the [extension editor](http://clinfhir.com/extensions.html) component in [clinFHIR](http://clinfhir.com/) to show these extensions.

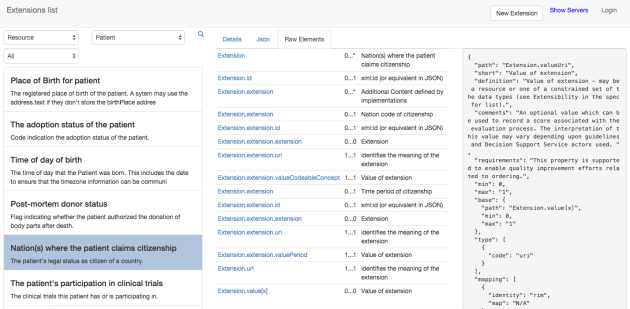
Take a look at the following screen shot.



It shows the detail page from the extension ‘patient-citizenship’ – we can see that the extension is to be applied to the Patient resource, and that there are 2 ‘child’ components – one called code (of type [CodeableConcept](http://hl7.org/fhir/datatypes.html#codeableconcept)) and another called period (of type [period](http://hl7.org/fhir/datatypes.html#period)). So this extension is to allow a patient to claim citizenship of a particular country over a given period. (Incidentally, the CodeableConcept ought to be bound to a ValueSet – not that I’m criticizing, mind…   )

btw – the ‘official’ description of this profile can be found [here](http://hl7.org/fhir/extension-patient-citizenship.html) – all of the profiles defined in the spec are found from a link at the top of each resource description.

But what does this look like under the hood? How are the ‘child’ slices defined? Click on the link that says ‘Raw Elements’ and you get this page:



This view shows all the elements in the [StructureDefinition.snapshot](http://hl7.org/fhir/structuredefinition-definitions.html#StructureDefinition.snapshot) property – ie all the elements contained in this profile – an extension that has 2 child properties. There are quite a few elements here, but they do make sense (after a little while!). Lets walk through them. (If you are really interested then you can start the extensions editor, select this profile and check it out for yourself.

* The first element (path = Extension) indicates that the extension is at the ‘root’ of the resource – as opposed to an element within the Patient resource.
* The next element (Extension.id) is provided to allow internal referenceing of this element from within the StructureDefinition. I’m unsure when this is used.
* The third element (Extension.extension) – which is the one showing in the right hand box – is the discriminator. You can see at the top that this is set to ‘url’ (always the case with extensions, but can be different in other profiles – as it is in the Blood Pressure example).
* After this there are 2 sets of 5 elements each. These describe each slice. Each slice contains the following:
  + Extension.extension. In other words, the ‘child’ element is an extension on the parent extension
  + Extension.extension.Id. Similar to the Id on the ‘parent’ extension, this is to allow references to the element from within the profile.
  + Extension.extension.extension. This would be where an extension on the child would sit – i.e. an extension nested to another level. This is perfectly legitimate in FHIR – though it’s starting to get quite complicated – pity the poor client that needs to make sense of it! In any case, note that the multiplicity is 0..0 – in other words we are explicitly stating that this is not permissible in this profile.
  + Extension.extension.url. This is the unique value for this slice (recall that the discriminator is ‘url’). If you look inside it (using clinFHIR) you’ll see that there is a property called ‘fixedUri’ whose value is ‘period’. In other words, a conformance resource will have the ‘url’ value for this element set to the value ‘period’
  + Finally an element with the path Extension.extension.valuePeriod. So the datatype for this child element is a period.
* Next up is an element Extension.url. This is the url of the ‘parent’ extension and also has a fixedUrl property – in this case set to <http://hl7.org/fhir/StructureDefinition/patient-citizenship> – the url for this whole extension
* And finally an element with the path Extension.value[x]. This is where the value would go in a ‘simple’ extension (one without children). Note that the multiplicity is 0..0 – an extension can have a value, or child extensions, but not both.

(Actually I’m not sure why the last 2 aren’t in positions 4 & 5 which would seem to make more sense. It may be just that that’s where the build tool put them, or they may need to be at the end).

It pays to have a play around using the tool against the extensions defined by the specification – it does take a little time to get your head around this stuff – at least it did for me! You can also compare what clinFHIR shows for an extension with what the specification does. I have to admit that the spec is prettier – but it doesn’t show this level of detail, which is what I was after.

So that’s how it’s done (at least, I think that’s how it’s done!)

To create your own complex extensions using the extension editor, follow these steps.

1. With Grahames server selected as the conformance server, click on the ‘New Extension’ button at the top of the form.
2. Enter the name and description for the extension. Remember that the url for the extension will be derived from the name – if you choose one that is already in use you will get a warning – best to choose another name. You can also enter any of the other elements that are needed – eg the Resource Type that this extension applies to.
3. Where it says ‘Extension type’, click on the button labeled ‘complex’ (top middle of the form) and a new tab will appear labeled ‘Complex Contents’. Select that tab.
4. For each child extension enter:
   1. The Name (make it a single word – this will be the value of the url in the profile, which is the discriminator). It must be unique as well.
   2. The short description
   3. The Multiplicity
   4. The DataType. If you select a coded value you’ll have the opportunity to select the ValueSet to bind to it.
   5. Make sure to click the plus (+) symbol to the far right to add the contents of the row as a new child extension.
5. You can click the ‘Validate’ button at any stage to check that the profile (StructureDefinition) is valid. This uses the [validate](http://hl7.org/fhir/resource-operations.html#validate) operation – quite handy actually!
6. When finished, click the Save button to save the new profile to the server.

And that it! You can select the profile and see what elements have been created (I hope they are correct – if not, then I hope that someone will correct me!)

Of course, you don’t need to know any of this stuff to use FHIR. Tooling is provided by the project that will create these definitions ([Forge](http://fhir.furore.com/forge) and the specification build tool itself) and also to validate that a particular resource is conformant to one. But it’s good to know what’s happening ‘under the hood’!

And, as always a disclaimer that clinFHIR only supports a fraction of the functionality defined by the specification – and implemented by tooling such as forge. In fact, if you do have a play you’ll find that it won’t update any extension other than ones created by clinFHIR – for that very reason.

Now I need to update the resource builder to cope with complex extensions. Sigh.

2.75 Uncertain dates in FHIR

NOVEMBER 16, 2015 [LEAVE A COMMENT](https://fhirblog.com/2015/11/16/uncertain-dates-in-fhir/#respond)

There are a lot of date properties in FHIR resources, but sometimes you don’t have the exact date. For example AllergyIntolerance has an ‘onset’ property which is a dateTime – but what if all we’re told is that they have been allergic to peanuts since childhood?

Well, assuming that a string is not as an option (as it is for [Condition.onset](http://hl7.org/fhir/condition-definitions.html#Condition.onset_x_)for example) then one way is to use an extension. We’d add an onset property, but leave the value blank and just add the extension.

Here's what is looks like in JSON:

"\_onsetDateTime": {

"extension": [

{

"url": "hl7.org.nz/fhir",

"valueString": "Since early childhood"

}

]

},

And in XML:

<onsetDateTime>

<extension url="hl7.org.nz/fhir">

<valueString value="Since childhood" />

</extension>

</onsetDateTime>

Just for fun, let’s think about how we’d create an extension definition that describes that (remember that extension definitions are actually StructureDefinition resources).

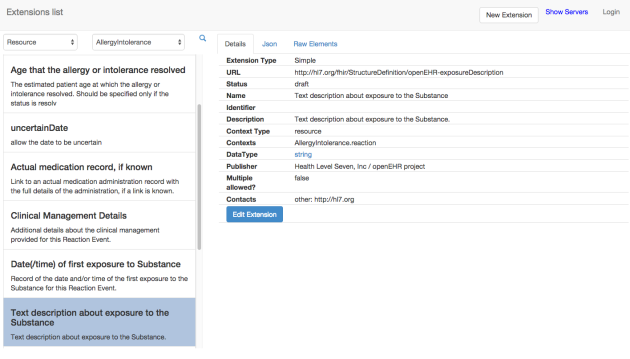
First off, we need to decide exactly what the extension is on. There are a couple of possibilities.

1. We could create an extension against the dateTime datatype itself. That way we could use that extension whenever we have a dateTime property that is uncertain
2. We could create an extension that is specifically used for AllergyIntolerance.onset

There are no hard and fast rules for this – probably the first is more reusable, and indeed it’s one that could be defined as an ‘HL7 level’ one (once we’ve figured out the process for doing that of course). However, for this exercise let’s use one against AllergyIntolerance.onset specifically.  (In truth, because we’re going to use clinFHIR to create the extension, and it doesn’t support extensions on datatypes – yet).

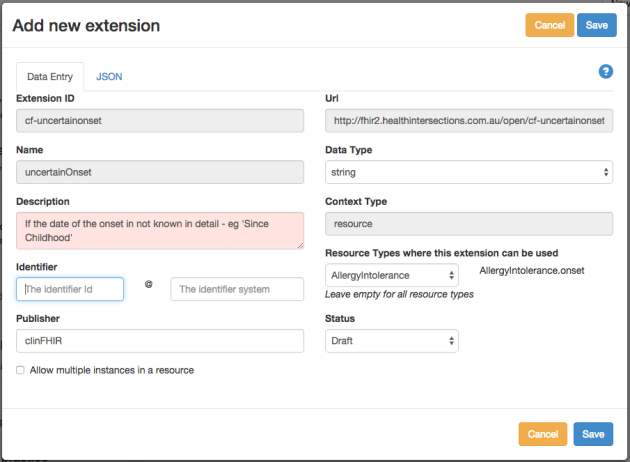
First we need to make sure there isn’t already an extension that we can use. As a community, it’s best if there’s only one extension for any particular purpose to maximize interoperability and we’d prefer not to ‘re-invent the wheel’. The infrastructure isn’t quite there to support that yet – Furore have got an [initial version of the registry](http://foundry.furore.com/) up, but it’s not yet fully populated – but we can see how it could work using [clinFHIR](http://clinfhir.com/). Start the app, then select the ‘[Extensions Editor](http://clinfhir.com/extensions.html)‘. In the search box to the upper left, select ‘Resource’ as the search type, then ‘AllergyIntolerance’ from the drop down of reauce types. Then click the magnifying glass (or ‘search’) icon.

After a short delay’ you’ll get a screen like the one below:

[](https://fhirblog.files.wordpress.com/2015/11/screen-shot-2015-11-16-at-2-37-31-pm.png)

These are all the extensions that have been created to be used for the AllergyIntolerance. You can scroll down the list to see if there’s one that matches – there wasn’t when I wrote this post, but that might change… This is a pretty basic search – and it only works against extension definitions on Grahames server (or whatever conformance server you select in the top NavBar, but you can see how it could work, and where it could be improved (more sophisticated searching – maybe with ‘fuzzy’ logic, checking multiple registries etc.)

So we need to create a new extension. At the top right is a button labelled ‘New Extension’. Click it and you’ll get the following modal dialog. (Note that I’ve filled in the details).

[](https://fhirblog.files.wordpress.com/2015/11/screen-shot-2015-11-16-at-2-50-35-pm.png)

One thing to look at is the resource context (Where it says ‘Resource types where this extension can be used’ at the lower right). Note that it has the complete path of the onset property. This indicates that the extension is to be used on that specific property  not on the resource as a whole. (You are prompted to select the path when selecting the resource type – it’s optional).

So that’s about it – you can save the extension, and if you refresh the list you should see it there, and it’s available to be used in a resource. We can talk about how we could create a profile on AllergyIntolerance that used this extension in another post.

Just a couple of notes:

* The resource builder in clinFHIR doesn’t support extensions on properties – it places them all at the root. (It doesn’t like complex extensions either).
* The Extension Builder won’t create extensions on datatypes. It will display them ok though.

I’m working on these.

By the way, as part of the testing for this post, I noticed that a number of the test servers didn’t like empty properties – I’m not sure if that is a bug or whether there’s an issue in the spec with empty properties. The healthConnex one (<http://sqlonfhir-dstu2.azurewebsites.net/fhir/> ) works just fine though…

2.76 Creating (and finding) FHIR Extension definitions

NOVEMBER 4, 2015 [2 COMMENTS](https://fhirblog.com/2015/11/04/creating-and-finding-fhir-extension-definitions/#comments)

We’ve talked on quite a few occasions about [profiling](https://fhirblog.com/category/profile-2/) in FHIR, and creating extensions when we need to represent elements that aren’t in the core resources. At [Orion Health](https://orionhealth.com/), we’re putting a fair amount of effort into FHIR (as are all forward looking vendors in this space) so we needed a way to more easily manage the extensions we need to create. [Forge](http://fhir.furore.com/forge) is an option – but our extensions are quite simple. clinFHIR has always been able to create extensions, but the process is klunky and we needed a better way.  So I’ve added an ‘[Extension Editor](http://clinfhir.com/extensions.html)’ to make this a bit easier.

I do need to point out that the capabilities of clinFHIR are limited compared to the official tool [Forge](http://fhir.furore.com/forge) from Furore – it manages only a subset of the capabilities FHIR offers and will only edit Extensions that were originally authored through the tool.

First, a bit of background.

Broadly, there are 2 aspects to ‘profiling’ resources in FHIR (we’re just talking about altering resources at the moment – profiling is a large topic).

* Creating extensions, which is used to add new properties to resources. As we’ve discussed before, this process is a fundamental part of FHIR – there is no stigma to using extensions, and they have the same capabilities as ‘native’ elements. Our belief is that most real implementations will use them.
* Constraining resources, which is where we take the core resource and alter the defined elements. Possibilities include:
  + Mark some as required and other as not used (within some limits)
  + ‘Slicing’ to refine elements supporting multiple values
  + Change the terminology binding of coded elements
  + Prescribe systems for Identifiers

Both of these are performed using the [StructureDefinition](http://hl7.org/fhir/structuredefinition.html) resource. For example, we might define an extension for ‘Eye Colour’ to be represented as a string and used in a Patient resource. (We could alternatively use a CodeableConcept as the datatype, which might be a better choice if we wanted to report on the prevalence of eye colour within a population, but I digress…)

Given that another fundamental part of FHIR is discoverability – where a resource instance with an extension must refer to its definition (ie the StructureDefinition resource), it begs the question of where to store these definition in a way that they can be found by any recipient – and in a way that is Internet Scalable.

There are 2 parts to this.

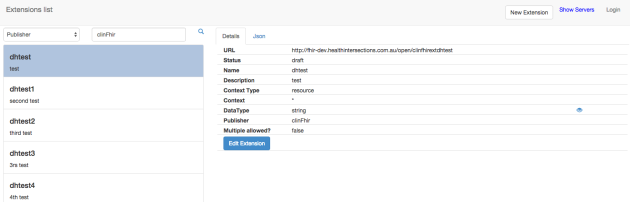
First, all StructureDefinitions (SD) have a globally unique ‘url’ property. Strictly this is a URI, though the spec recommends that it should be ‘resolvable’ – eg entering it into a web browser should download the StructureDefinition. However, this is not required – it only has to be globally unique. There are different ways to create this url – for example an organization could manually assign it based on a web domain it controls – for example [www.clinfhir.com/StructureDefinition/myExtension](http://www.clinfhir.com/StructureDefinition/myExtension) – or it could let the server hosting it create it. It is even possible that the SD is stored as a file in, say, gitHub and referenced from there.

If the url doesn’t HAVE to be resolvable, it follows that there must be some way of storing it somewhere, and searching for it based on its unique url. This is the role of a registry – a FHIR server that is focused on storing and returning SD’s based on the url. Pretty much any FHIR server can do this, but it makes sense that there are some ‘well known’ registries for this purpose – such as the ‘simplifier’ registry hosted by Furore on behalf of HL7 and accessible at <http://foundry.furore.com/>. Using such registries means that a client has a place to go to find SD’s that won’t resolve – or just using it as a matter of course. We can refer to this registry as a **Conformance Server** – as it stores conformance resources , of which there are a half dozen or so (see the [list in the spec](http://hl7.org/fhir/resourcelist.html))

It’s very much up to the community as to how this all works out in practice – as a ‘platform’ spec FHIR doesn’t dictates how an implementation should work – it just establishes the ground rules.

So, with that background, let’s see how to use clinFHIR to create and manage Extensions (or more correctly the StructureDefinitions that define extensions). For clarity we will refer to them as ‘Definitions’ – but they are really just StructureDefinition resources.

First, navigate to clinFHIR (<http://clinfhir.com/>) and select the Extensions Editor from the list of components on the left. This will open the editor in a separate window. Here’s what it looks like (I really need to create some better extension definitions for the front window!)

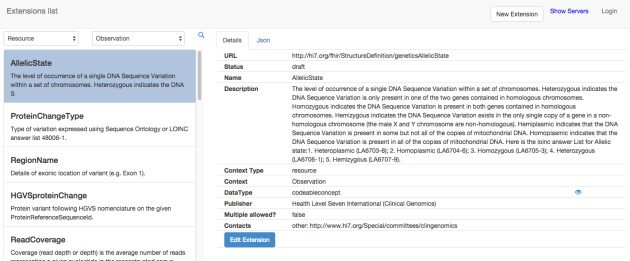
[](https://fhirblog.files.wordpress.com/2015/11/screen-shot-2015-11-04-at-10-13-40-am.png)

The app automatically loads all of the definitions that clinFHIR has created and generates a list from which you can select one to view and potentially edit. To do so, it needs to connect to a registry (Conformance server) as described above. You can configure this server using the ‘Show Servers’ link on the top right – Grahames server tends to have the most up to date list, but HAPI and Furore also have a good selection. (Eventually [simplifier](http://foundry.furore.com/) will be the default – but the REST interface isn’t quite ready)

The tool can show any definition on the server (though not necessarily all the details of that definition at the moment). To do this you can search the server using criteria at the top left of the screen. Search options include:

* Publisher – all definitions where the publisher field contains the search string.
* Name – all definitions where the name field contains the search string.
* Description – all definitions where the description field contains the search string.
* Identifier – if you add an identifier to the SD then you can use this option to search on them. This is a handy way to further group SD’s – say all of those created by a specific team within an organization – or any other purpose for that matter.
* Resource – this displays a list of resource types. Select one of them and clinFHIR will display all the extensions that are designed to work with that resource type. The ‘\*’ option displays those extensions that work with any resource type.

(click the magnifying lens icon to perform the search).

[](https://fhirblog.files.wordpress.com/2015/11/allelicstate.png)

Here’s a query of all definitions that apply to the Observation resource, with the ‘AllelicState’ definition selected. Note that the right hand side shows some of the details of the definition (and an option to view the json representation of the complete resource). These include:

* The URL. This is the unique uri for the definition as described above.
* The Status. This can be draft, active & retired. For definitions authored though clinFHIR this can be altered using the edit extension screen – described below.
* The Name and Description of the definition. (Note how thorough the Genomics folk are!)
* The context type – or where the extension can be applied. Mostly you’ll apply an extension to a resource or resource element, but it’s also possible to create an extension definition for a datatype as well (eg a HumanName datatype may have extensions needed for non-english countries)
* The specific context of use. This is related to the context type, but assuming that this is a resource, then the context is the actual resource type where this extension should be used. There can be multiple types and also path’s within the resource. For an example of the latter, search for definitions on Patient, then the ‘Time of Day of Birth’. The context here is Patient.birthDate – indicating that the extension is specific to the date of birth.
* DataType of the extension. Clocking the ‘eye’ icon will display the datatype definition in the spec. If the datatype is coded, and there is a Binding defined then the Binding strength and URL of the ValueSet is displayed, and another eye icon to explore the ValueSet appears.
* Publisher – a string
* Whether multiple instances of this extension can appear in a single resource instance.

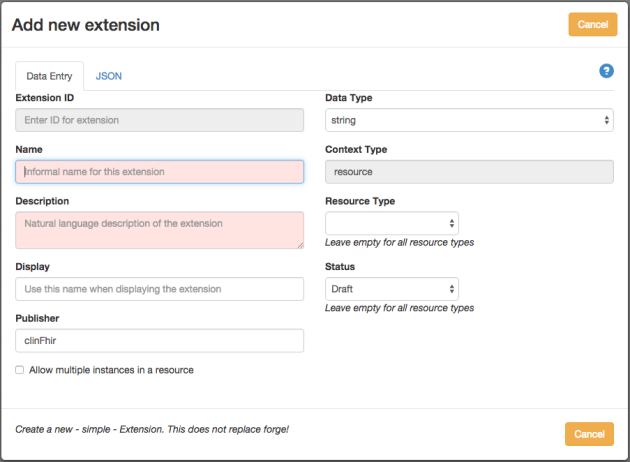
Other items that may appear if defined are

* Identifier, if one has been defined
* Contacts

At the bottom of the screen is an edit button. Clicking this will bring up a modal that displays the existing data – and if authored by clinFHIR, allows changes to be made. (Definitions from other tools are read only, as the capabilities of the clinFHIR editor is fairly basic at present).

The form is self-explanatory – when used in this ‘edit’ mode, the name cannot be changed as it is used by the tool to generate the unique url.

To create a new definition, click the ‘New Extension’ button at the top right. This displays the same modal window, but this time the name field can (and should) be edited. Enter a name for the definition (and note how both the server and the Url update). ClinFHIR creates the url for you based on the server location and the name. When you’ve finished entering the name, move out of the field and clinFHIR will check that the url is unique. If it is not then you’ll get a warning – you can choose to overwrite the definition if you wish – but you should be sure that is what you want to do.

[](https://fhirblog.files.wordpress.com/2015/11/screen-shot-2015-11-04-at-11-11-43-am.png)

So what’s happening under the hood? (Warning: techie stuff ahead!)

Well, when defining resources, FHIR describes a hierarchy of ‘entities’ that ‘inherit’ from a Base entity (very similar in concept to Object Orientated programming). The nature of the change to entities as you move ‘down’ the tree is described by the StructureDefintion resource.

So start with the base [Resource](http://fhir-dev.healthintersections.com.au/open/StructureDefinition/Resource) entity (like the Object in OO). This has an Id, some metadata and a couple of other elements. A [DomainResource](http://fhir-dev.healthintersections.com.au/open/StructureDefinition/DomainResource) extends Resource to add elements like text, a holder for contained resources and extensions. Then a ‘business’ resource – like [Condition](http://fhir-dev.healthintersections.com.au/open/StructureDefinition/Condition) – extends DomainResource and adds the elements specific to that resource. Each of these entities is described by a StructureDefinition which has (amongst other elements) the base (that it is refining), a differential list (which represents the changes from the base – used for tooling) and a snapshot (which is a list of all the elements in that particular entity).

You can see this quite nicely in the spec as well – start with a [Condition](http://hl7.org/fhir/condition.html#resource)  and click on the DomainResource type. You can continue to move ‘up’ the hierarchy in this way.

So what we’re doing when we create an Extension definition is altering the [Extension](http://hl7.org/fhir/extensibility.html#Extension) resource – our definition will be an ExtensionDefinition that has a base that is the Extension object, and uses the snapshot & differential elements to record the specific changes we are making. ClinFhir keeps it simple – the extensions always have a base of Extension (And you can see the resource that is created by clicking on the Json tab in the detail window), but more complex tools such as forge and the specification tooling allow the hierarchy to continue to be extended.

That’s a really high level description of what’s happening – refer to the specification for more details – or read the [Furore blog](http://thefhirplace.com/category/background) for various in-depth articles.

In this post we’ve confined our discussion on how to use the StructureDefinition to create extension definitions. You also use the StructureDefinition to constrain a resource – and to link the constrained resource to extensions definitions. Another post perhaps…

I should also emphasize that ‘best practice’ is to search for existing definitions before creating a new one – as that increases the chances that a recipient will understand it. Ideally too, an extension can be used in multiple resources (assuming that makes sense). Tooling to do this kind of searching is very much a work in progress – this tool is one option, and the furore registry is another. It is very likely that the two will converge over time.

Some limitations:

* Currently the tool only works with ‘simple’ extension definitions. It’s possible to define inherited extensions and extensions that are nested within other ones. Support for this will follow. (If you do select on of these more complex extensions, the Json tab will display the StructureDefinition, but the Details tab and modal window will be incomplete, and editing disabled.