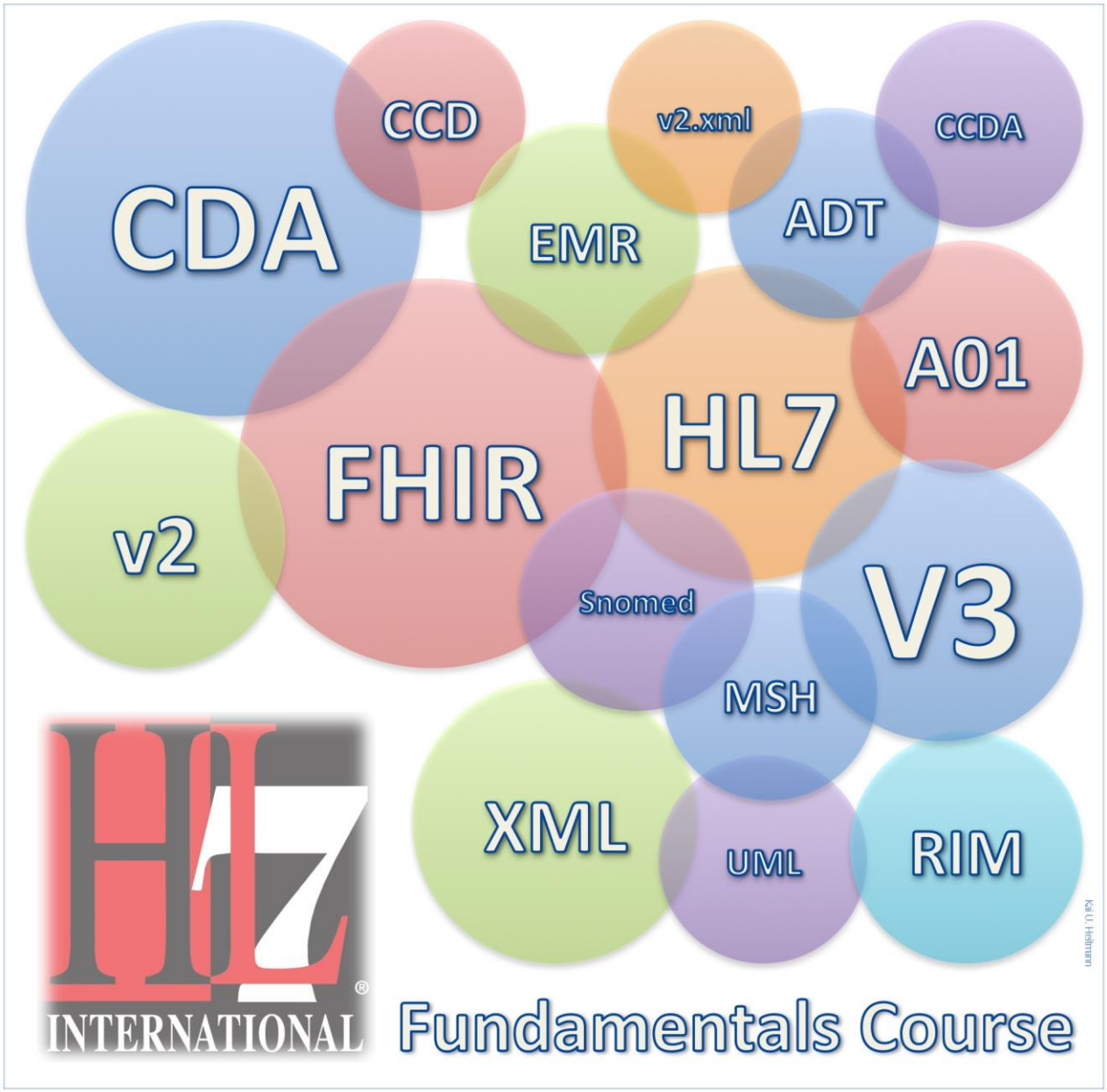


FHIR Module, Unit 1:

Introduction to HL7 FHIR

Reading Material

Course Overview

Module I: Introduction

Unit I.01 Introduction to Healthcare Interoperability
Unit I.02 Introduction to Vocabularies in Healthcare
Unit I.03 Introduction to UML (Unified Modeling Language)
Unit I.04 Introduction to XML (Extended Markup Language)

Module V: HL7 V2.x

Unit V.01 Introduction to HL7 version 2.x, Data Types, Acknowledgments
Unit V.02 HL7 V2.x Patient Administration, Orders and Results
Unit V.03 HL7 V2.x Implementation / Profiles / Z-Segments
Unit V.04 HL7 v2x.xml, the XML Implementation of V2.x Messages

MODULE C: HL7 CDA R2

Unit C.01 Introduction to HL7 CDA Release 2 (CDA R2)
Unit C.02 CDA R2 Architecture: Header, Body and Entries
Unit C.03 CDA R2 Implementation Guides
Unit C.04 CDA R2 Entries: Clinical Statements

MODULE F: HL7 FHIR

Unit F.01 Introduction to FHIR

Table of Contents

Unit Content and Learning Objectives

This Unit discusses the new FHIR standard developed by HL7. In December 2018, FHIR R4 has been approved as normative. We will try to keep this document accurate to the specification, but in the event of any differences always treat the specification as the truth.

1. Why FHIR?

Background

In January 2011, the Board of HL7 International commissioned a small Task Force to answer the question: "*If HL7 were starting afresh today, what would the interoperability standards look like?*" In considering this question, the task force noted that:

- Version 2 was (and is) extremely successful, but the technology is old and not well suited to the newer requirements.
- Version 3, while based on a robust model, has not been widely accepted and is perceived as difficult to implement.
- CDA has been hugely successful, but was designed as a document exchange mechanism and using it elsewhere doesn't really fit well in all scenarios
- Tooling for HL7 standards has always been an issue, as these generally need to be designed - and built - specifically for HL7, and this does not' always occur in a timely fashion.
- There are new Use Cases - especially involving mobile devices - where the current standards were not a good fit.
- Particularly in the online space, the use of a REST-based architecture is widely used in other domains.

In response to these and other issues, Task Force member Grahame Grieve surveyed the industry for best practices in modern interoperability frameworks outside the healthcare space. The most highly regarded was a commercial application named "Highrise". He leveraged a similar approach, applying it to healthcare.

Fast Healthcare Interoperability Resources (FHIR) grew out of this work.

The goal is to produce a standard that:

- Is easy to develop with a low learning curve and minimal custom tooling requirements
- Is easy to implement (or as easy as healthcare interoperability ever can be)
- Is semantically robust. This means that it can be mapped back to the v3 RIM (and often to other specifications such as openEHR archetypes)
- Is 'implementer friendly' - e.g. uses common tools and formats, and web-based technologies for the specification
- The artifacts should make sense to a human looking at them. While not intended for direct human viewing, being directly understandable helps both implementers and support personnel
- The artifacts should be able to be validated electronically - as far as that is possible
- Integrates well with and leverages modern web-based communication technologies (HTTP, XML, JSON, etc.)

It should be noted that FHIR is not 'version 4' of HL7, although it builds on the long history of HL7 messaging standards.

To achieve this, the FHIR team has established:

- A specification that is hosted on the web and is fully hyperlinked. For example, clicking on a datatype in a resource definition will take you to the definition of that data type.
- An easy-to-read format for all resources that includes a 'pseudo-xml' definition, UML diagrams and links for formal definitions. The format is such that clinicians are able to understand what a resource contains and represents (although the target audience remains implementers)

- A number of examples for each resource that show how each resource is intended to be used.
- A standards-development 'build' process that automatically generates all the artifacts from a small number of key definition files in a manner similar to a typical software development project. The build process validates all definitions and the examples to ensure a high quality result.
- Freely available Reference Implementations in Delphi, Java, JavaScript, Swift, Python, C# and Objective C that implementers can download and use - or can use as the basis of their own developments. Links are available to these (and other useful resources) on the front page of the specification (www.hl7.org/implement/standards/fhir/index.htm). Additional implementations may be introduced in the future.
- A number of publicly available FHIR test servers that can be used by implementers to test their developments: http://wiki.hl7.org/index.php?title=Publicly_Available_FHIR_Servers_for_testing
- Regular 'Connectathons' (inspired by the IHE Connectathons) where implementers can meet and test their work.
- A number of communications channels (List servers, Wiki, Skype conversations, Stack Overflow, etc.) where implementers can contact the FHIR development team and other implementers directly. The best way to contact fellow FHIR users, implementers and developers is the zulip channel: https://chat.fhir.org/

In December 2018 FHIR was published as a normative standard. It can be accessed at www.hl7.org/FHIR.

2. Scope

The scope of FHIR includes all aspects of healthcare-related interoperability - clinical care, administration, research, etc. Furthermore, FHIR supports interoperability via four common information exchange architectures/paradigms. These are:

- Messaging
- Documents
- Services
- REST (Representational State Transfer - on-line access)
- Persistence/Data bases

All of these paradigms use the same resources to represent the content - they are just wrapped in 'packages' that suit the particular paradigm.

HL7 has considerable experience in the messaging and documents paradigm and some experience in the services paradigm. However, the REST architecture was new to HL7 but it's used by most implementations.

Unlike a messaging paradigm where the messages are used to update repositories (as well as implementing behavior) the REST paradigm means that the information may be accessed from some other server as needed, so it supports more of a distributed model.

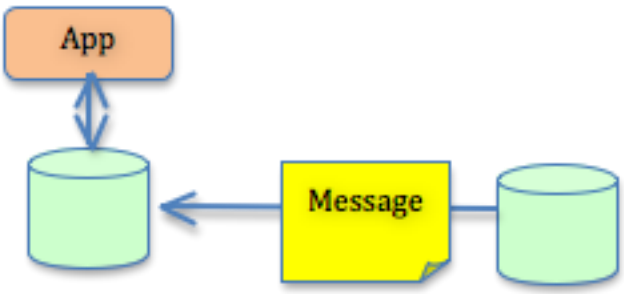


Figure 1: Information is replicated in both sending and receiver data stores using messages

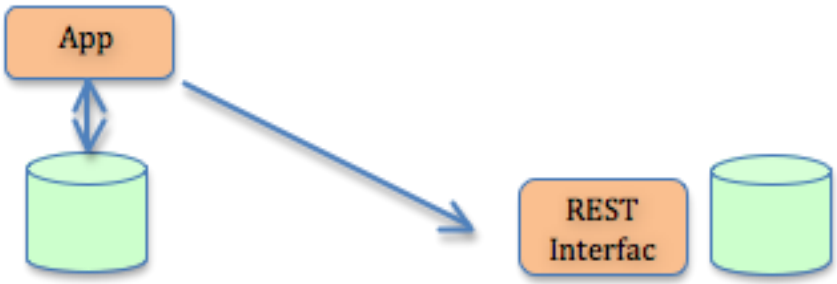


Figure 2: Application directly querying a remote system

Of course, many other variations are possible - for example, the application could query the remote server, and then store a copy of the data locally for subsequent access. In addition, there can be any number of remote servers.

3. Governance & Methodology

The FHIR standard is an open standard. While it is developed by HL7, there is no need to be an HL7 Member to use it (though participation is encouraged).

The Standards Governance Board

The Standards Governance Board oversees FHIR (and all other standard) development and has final say on the methodology and consistency of the specification. It coordinates all the Standard Management Groups
More information at http://www.hl7.org/Special/Committees/sgb/overview.cfm

The FHIR Management Group

The FHIR Management Group provides day-to-day oversight of FHIR-related work group activities including performing quality analysis, monitoring scope and consistency with FHIR principles and aiding in the resolution of FHIR-related intra and inter-work group issues.
Actual development of resources is performed by the HL7 Work Groups that 'owns' the resource. E.g. the Pharmacy Work Group is responsible for all medication related resources. In addition to the primary owner, there may be additional Work Groups that are consulted as part of the development process.
More information http://www.hl7.org/Special/committees/fhirmg/index.cfm

4. Relationship with other SDOs

The FHIR team (and HL7 in general) has established working relationships with other Standards Development Organizations where that is applicable. Examples of these relationships include:

- IHE is cooperating on the development of resources to support RESTful implementation of their XDS, PIX and ATNA profiles which are currently modeled as the DocumentReference, Patient and AuditEvent resources. There is a separate discussion on FHIR support of XDS later in this Unit. More information at https://wiki.ihe.net/index.php/Category:FHIR

- openEHR has done a significant amount of work in modelling the clinical domains. They have taken a slightly different approach to HL7 by creating domain-specific models that are 'maximal data sets' for that domain. More information at www.openehr.org

- DICOM is working with the FHIR team on image-related resources; more information at https://www.dicomstandard.org/dicomweb/dicomweb-and-hl7-fhir/

5. License

FHIR is released under an open license. Implementers do not need to be a member of HL7 International to use FHIR (although there are other significant benefits on being a member)
From the license page of the specification:

- FHIR is © and ® HL7. The right to maintain FHIR remains vested in HL7
- You can redistribute FHIR
- You can create derivative specifications or implementation-related products and services
- Derivative Specifications cannot redefine what conformance to FHIR means
- You can't claim that HL7 or any of its members endorses your derived [thing] because it uses content from this specification
- Neither HL7 nor any of the contributors to this specification accept any liability for your use of FHIR

6. Using FHIR - Example Architectures

There are a number of ways that FHIR could be used, especially as FHIR-capable systems will need to interact with systems using existing standards. Some of these options include:

Message Broker

When using a messaging paradigm, an application such as an integration engine can bi-directionally convert between FHIR resources and other standard instances such as CDA and v2, as seen in Figure 3.

There are no current plans to do this for complete HL7 v2 messages as their use is quite variable - however guidance for doing so will be made available, and in many ways v2 will be simpler to map than CDA - for example, in general terms, a v2 segment maps to a FHIR resource. This post from one of the project leads at www.healthintersections.com.au/?p=972 talks about converting from V2.x messages, and this one at www.healthintersections.com.au/?p=979 is v3 / CDA focused. Each resource definition has a mapping section where the mapping to V2 is included if available.

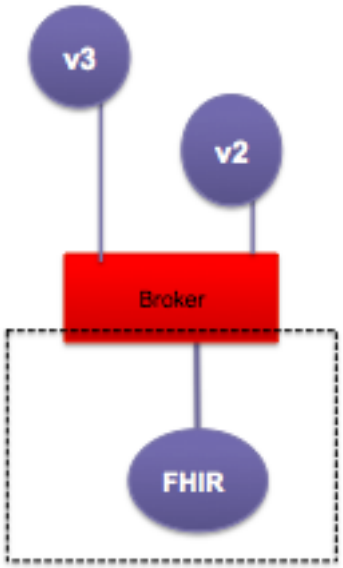


Figure 3: Message Broker + FHIR engine

Native FHIR Server with existing Back End

This is most applicable where there is an existing data source of some type (e.g. an EMR or PHR system) and the users want to put a FHIR interface in front of it -either as a read, an update or both effectively 'FHIR enabling' the system. This is also called a “FHIR Façade”. An application will be needed to perform the conversion (perhaps based on one of the FHIR reference implementations) - e.g. receive a request for a FHIR resource, query the back-end system for the data, and then convert it to another FHIR resource and return it.

Note that the conformance resource and profiles are very useful in indicating what resources and functions are supported.

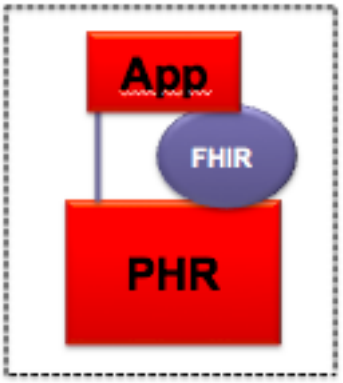


Figure 4: FHIR Server with Existing Back-End

Native FHIR Server with FHIR Back End

In this architecture shown in Figure 5, the FHIR resources are stored directly in the back end data store, and queried as required. A number of systems (including several of the test/reference servers available) have taken this approach - some using 'no-sql' database engines, some others use a relational database with a simple structure.

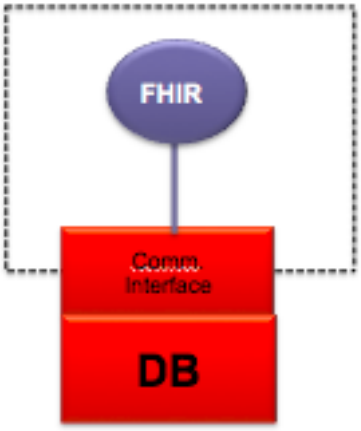


Figure 5: FHIR server and FHIR back-end

7. Using FHIR : Sample Scenarios

Examples of when you might use a FHIR interface include:

- On-line queries to a RESTful server by a mobile or web application
- Using the XDS-compatible resources to store and locate documents (which could be FHIR documents, CDA, PDF etc) through the MHD Profile.
- Internal messaging of events where you would generally use HL7 V2.x Admission, Discharge and Transfer (ADT) messages
- Sending a Discharge Summary to a repository

8. Key Concepts of FHIR

There are a number of key concepts of FHIR including:

- Resources
- Extensions
- Datatypes
- Bundles
- Profiles

Resources

What is a resource?

A resource is the smallest unit of exchange that 'makes sense' in interoperability - such as an observation, a patient or a condition. They are roughly analogous to a segment in a v2 message.
The intention is that there are a small number of fundamental resources (100-150) that form the building blocks of all FHIR artifacts.
A resource has the notion of 'identity' - something that identifies it as a logical 'thing' and will have a location (a URI) where it can be found, that will include both the id and the host where that resource is stored.
A resource is made up of elements, each of which is a particular datatype (like string or CodeableConcept). In many resources, a particular element can be one of several different datatypes, though an element occurrence of a particular instance of a resource will be only of one of those datatypes. For example, in the Observation resource shown below, the *value* can be a Quantity, a CodeableConcept and a number of others. If a specific instance was of type Quantity, then the name of that element would be *valueQuantity*. If, in another instance it was a CodeableConcept, then the name of the element would be *valueCodeableConcept.*

Examples of a resource

To see an example of an Observation, click in the 'Examples' tab in the definition of Observation in the specification: www.hl7.org/implement/standards/fhir/observation-examples.htm. You will see both XML and JSON examples there.

```
<?xml version="1.0" encoding="UTF-8"?>
<Observation xmlns="http://hl7.org/fhir">
 <id value="f001"/> <!--  urn:oid:2.16.840.1.113883.4.642.1.7  --><!--  2.16.840.1.113883.4.642.1.118  --
  ><text> <status value="generated"/> <div xmlns="http://www.w3.org/1999/xhtml"><p> <b> Generated
  Narrative with Details</b> </p> <p> <b> id</b> : f001</p> <p> <b> identifier</b> : 6323 (OFFICIAL)</p>
  <p> <b> status</b> : final</p> <p> <b> code</b> : Glucose [Moles/volume] in Blood  6.3 mmol/l</td> </tr>
  </table> </div>
   </text>
 <identifier>
  <use value="official"/>
  <system value="http://www.bmc.nl/zorgportal/identifiers/observations"/>
  <value value="6323"/>
 </identifier>
 <status value="final"/>
 <code>
  <coding>
   <system value="http://loinc.org"/>
   <code value="15074-8"/>
   <display value="Glucose [Moles/volume] in Blood"/>
  </coding>
 </code>
 <subject>
  <reference value="Patient/f001"/>
  <display value="P. van de Heuvel"/>
 </subject>
 <effectivePeriod>
  <start value="2013-04-02T09:30:10+01:00"/>
 </effectivePeriod>
 <issued value="2013-04-03T15:30:10+01:00"/>
 <performer>
  <reference value="Practitioner/f005"/>
  <display value="A. Langeveld"/>
 </performer>
 <valueQuantity>
  <value value="6.3"/>
  <unit value="mmol/l"/>
  <system value="http://unitsofmeasure.org"/>
  <code value="mmol/L"/>
 </valueQuantity>
 ...
</Observation>
```

Figure 6: Snippet of an XML version of an Observation

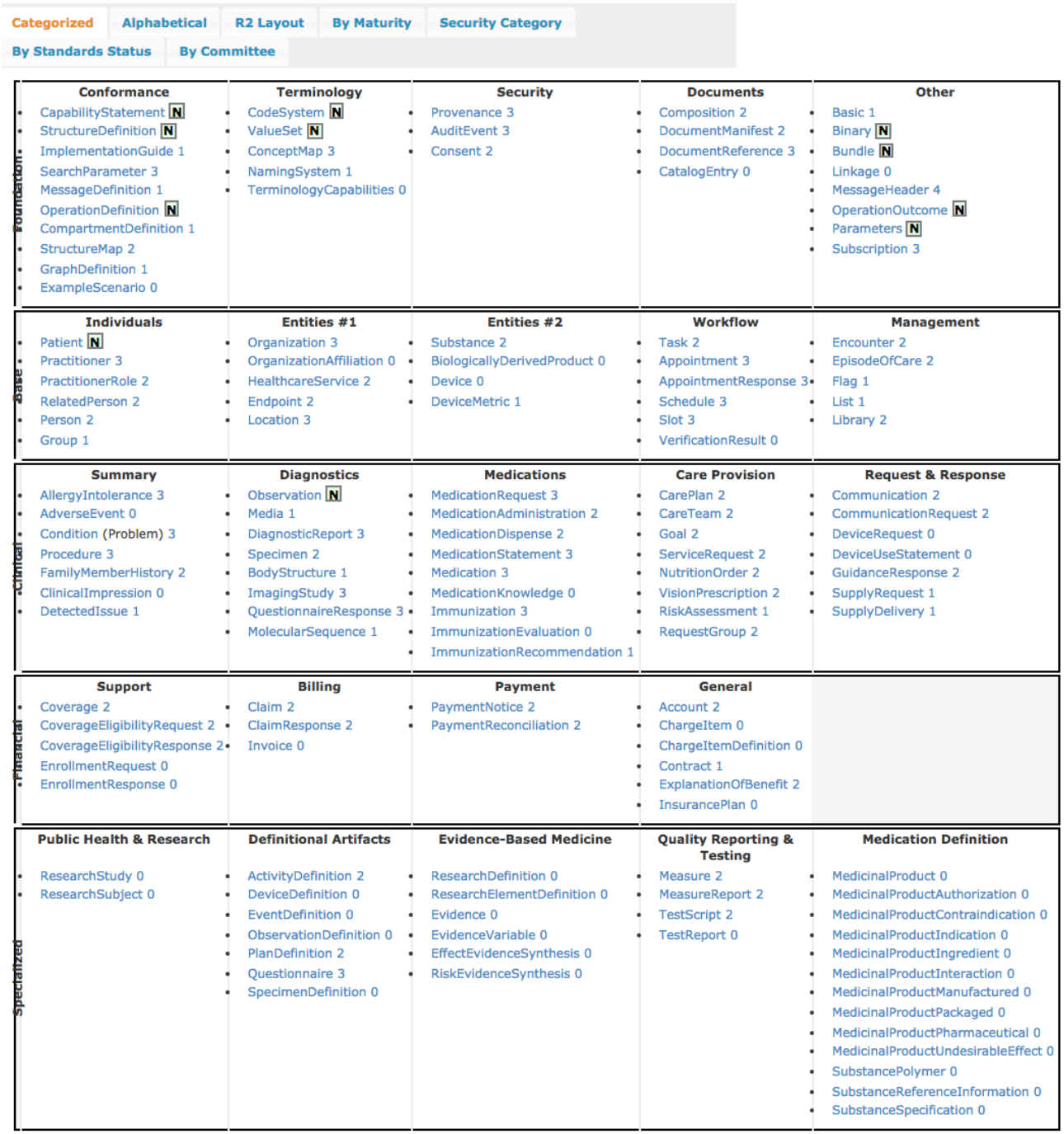
```
{
  "resourceType": "Observation",
  "id": "f001",
  "text": {
    "status": "generated",
    "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">…</div>"
  },
  "identifier": [
    {
      "use": "official",
      "system": "http://www.bmc.nl/zorgportal/identifiers/observations",
      "value": "6323"
    }
  ],
  "status": "final",
  "code": {
    "coding": [
      {
        "system": "http://loinc.org",
        "code": "15074-8",
        "display": "Glucose [Moles/volume] in Blood"
      }
    ]
  },
  "subject": {
    "reference": "Patient/f001",
    "display": "P. van de Heuvel"
  },
  "effectivePeriod": {
    "start": "2013-04-02T09:30:10+01:00"
  },
  "issued": "2013-04-03T15:30:10+01:00",
  "performer": [
    {
      "reference": "Practitioner/f005",
      "display": "A. Langeveld"
    }
  ],
  "valueQuantity": {
    "value": 6.3,
    "unit": "mmol/l",
    "system": "http://unitsofmeasure.org",
    "code": "mmol/L"
  },…
```


-Figure 7: Snippet of a JSON version of an Observation

Types of Resource in FHIR

There are a number of different types of resource that FHIR defines and these are described at www.hl7.org/implement/standards/fhir/resourcelist.html

There are several possible views of the resources: by category, alphabetical, by maturity, by security category, by standard status, and by committee.
Next to each resource you will see a number or an N letter.
This is called the FMM (FHIR Maturity Level) and goes from 0 to 5 and finally N (Normative)
See the description of the levels here:
http://www.hl7.org/implement/standards/fhir/versions.html#maturity



9. FHIR on the Wire

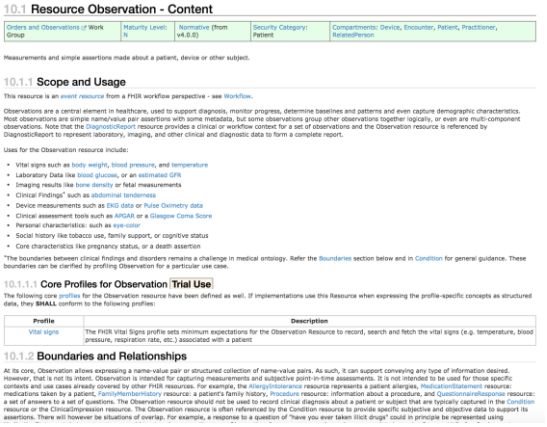
Any FHIR resource can be represented either as an XML document or as a JSON document - indeed all the examples in the specification (on the example tab of each resource) have both representations. Representation as RDF will also be possible, though RDF usage will generally be appropriate for secondary analysis rather than primary exchange, and this will be a secondary rather than primary representation.

The FHIR team has defined a JSON syntax that is very similar to the XML syntax both for ease of conversion between the two and to ensure that the extensibility of FHIR can be expressed in both formats.

This post discusses this representation and the reason why it was chosen;
http://thefhirplace.com/2013/03/16/attributes-versus-elements-in-fhir-xml

1. Definition & Documentation in the Specification

In the FHIR specification, resources are described in a number of different ways (and incidentally this is where the value of building the specification, as if it was a software project really has benefits: all the representations are consistent - they are validated and enforced during the build process from a single source - including all the examples).


10.1 Resource Observation - Content

Orders and Observations of Work Group | Maturity Level: N | Normative (from v4.0.0) | Security Category: Patient | Compartments: Device, Encounter, Patient, Practitioner, RelatedPerson

Measurements and simple assertions made about a patient, device or other subject.

10.1.1 Scope and Usage

This resource is an event resource from a FHIR workflow perspective - see Workflow.

Observations are a central element in healthcare, used to support diagnosis, monitor progress, determine baselines and patterns and even capture demographic characteristics. Most observations are simple name/value pair assertions with some metadata, but some observations group other observations together logically, or even are multi-component observations. Note that the DiagnosticReport resource provides a clinical or workflow context for a set of observations and the Observation resource is referenced by DiagnosticReport to represent laboratory, imaging, and other clinical and diagnostic data to form a complete report.

Uses for the Observation resource include:

- Vital signs such as body weight, blood pressure, and temperature
- Laboratory Data like blood glucose, or an estimated GFR
- Imaging results like bone density or fetal measurements
- Clinical Findings* such as abdominal tenderness
- Device measurements such as EKG data or Pulse Oximetry data
- Clinical assessment tools such as APGAR or a Glasgow Coma Score
- Personal characteristics: such as eye-color
- Social history like tobacco use, family support, or cognitive status
- Core characteristics like pregnancy status, or a death assertion

*The boundaries between clinical findings and disorders remains a challenge in medical ontology. Refer the Boundaries section below and in Condition for general guidance. These boundaries can be clarified by profiling Observation for a particular use case.

10.1.1.1 Core Profiles for Observation Trial Use

The following core profiles for the Observation resource have been defined as well. If implementations use this Resource when expressing the profile-specific concepts as structured data, they SHALL conform to the following profiles:

Profile	Description
Vital signs	The FHIR Vital Signs profile sets minimum expectations for the Observation Resource to record, search and fetch the vital signs (e.g. temperature, blood pressure, respiration rate, etc.) associated with a patient

10.1.2 Boundaries and Relationships

At its core, Observation allows expressing a name-value pair or structured collection of name-value pairs. As such, it can support conveying any type of information desired. However, that is not its intent. Observation is intended for capturing measurements and subjective point-in-time assessments. It is not intended to be used for those specific contexts and use cases already covered by other FHIR resources. For example, the AllergyIntolerance resource represents a patient allergies, MedicationStatement resource: medications taken by a patient, FamilyMemberHistory resource: a patient's family history, Procedure resource: information about a procedure, and QuestionnaireResponse resource: a set of answers to a set of questions. The Observation resource should not be used to record clinical diagnosis about a patient or subject that are typically captured in the Condition resource or the ClinicalImpression resource. The Observation resource is often referenced by the Condition resource to provide specific subjective and objective data to support its assertions. There will however be situations of overlap. For example, a response to a question of "have you ever taken illicit drugs" could in principle be represented using

The header and first three sections for each resource include the following information about the resource:

HL7 Workgroup: in charge of the maintenance of the specification
Maturity level: 0 (Draft) -> N (Normative) and normative status.
Security Category: Sensitivity level (i.e.: Anonymous / Business/ Individual/ Patient /)
Compartment: logical grouping of resources for access control
Scope and Usage: Defining the precise scope and use scenarios
Boundaries and Relationships: When to use it, when to use other resources, which resources are related or referenced to/from this one.

2. Specification of Resource Content

The following are the representations you will see for each resource content in the FHIR standard:

1. UML Diagram

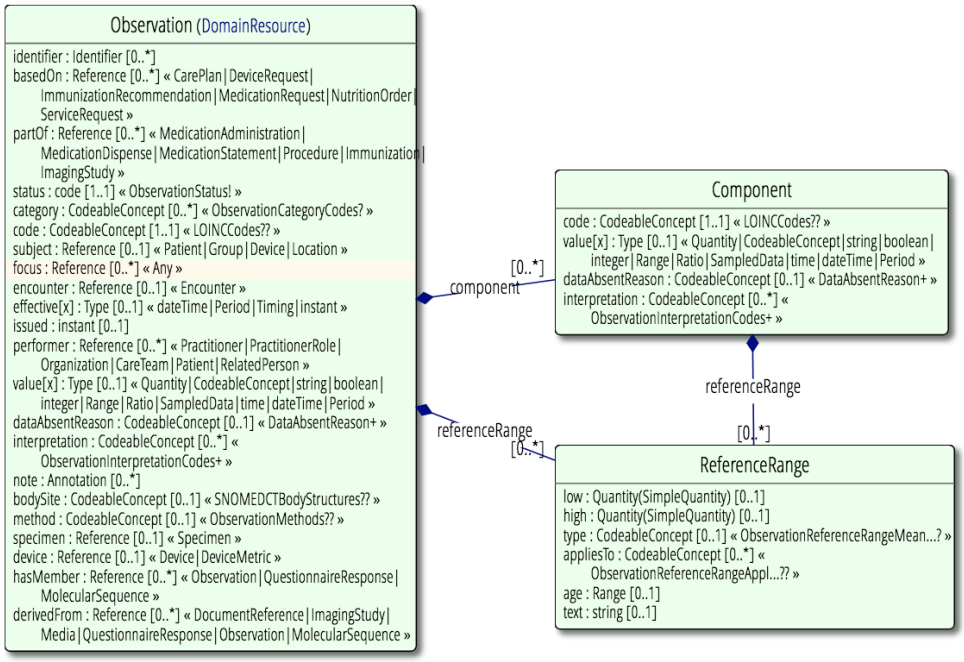


Figure 8: A UML diagram of the Resource

The "root" of the resource is the class indicated with "(Resource)". Each element is represented as an attribute indicating its data type(s) and cardinality. Complex elements (those with nested sub-elements) are represented as associations to other component classes. Coded concepts have a link to their vocabulary.

2. A simple Pseudo-XML and JSON syntax.

The screenshot below shows the definition for an Observation resource (refer to the specification (www.hl7.org/implement/standards/fhir/observation.html) for the full definition (which may be different to this one by the time you read it). This image is the 'pseudo-xml' type of image - which is actually very close to what an actual instance will look like. Copying and pasting this pseudo-XML into an XML editor will give a head start to creating instances. Same for JSON.
The diagram below shows the multiplicity and data types for each resource element and the data types are hyperlinked to their definition in the specification. The simplified 'one-line' description for each element is hyperlinked to the more complete definition:

```
<Observation xmlns="http://hl7.org/fhir">
 <!-- from Resource: id, meta, implicitRules, and language -->
 <!-- from DomainResource: text, contained, extension, and modifierExtension -->
 <identifier><!-- 0..* Identifier Business Identifier for observation --></identifier>
 <basedOn><!-- 0..* Reference(CarePlan|DeviceRequest|ImmunizationRecommendation|
   MedicationRequest|NutritionOrder|ServiceRequest) Fulfills plan, proposal or order --></basedO
n>
 <partOf><!-- 0..* Reference(MedicationAdministration|MedicationDispense|
   MedicationStatement|Procedure|Immunization|ImagingStudy) Part of referenced event --></partOf
>
 <status value="[code]"/><!-- 1..1 registered | preliminary | final | amended + -->
 <category><!-- 0..* CodeableConcept Classification of  type of observation --></category>
 <code><!-- 1..1 CodeableConcept Type of observation (code / type) --></code>
 <subject><!-- 0..1 Reference(Patient|Group|Device|Location) Who and/or what the observation is
 about --></subject>
 <focus><!-- 0..* Reference(Any) What the observation is about, when it is not about the subject
 of record --></focus>
 <encounter><!-- 0..1 Reference(Encounter) Healthcare event during which this observation is mad
e --></encounter>
 <effective[x]><!-- 0..1 dateTime|Period|Timing|instant Clinically relevant time/time-period for
 observation --></effective[x]>
 <issued value="[instant]"/><!-- 0..1 Date/Time this version was made available -->
 <performer><!-- 0..* Reference(Practitioner|PractitionerRole|Organization|
   CareTeam|Patient|RelatedPerson) Who is responsible for the observation --></performer>
 <value[x]><!-- 🔒 0..1 Quantity|CodeableConcept|string|boolean|integer|Range|Ratio|
   SampledData|time|dateTime|Period Actual result --></value[x]>
 <dataAbsentReason><!-- 🔒 0..1 CodeableConcept Why the result is missing --></dataAbsentReason>
 <interpretation><!-- 0..* CodeableConcept High, low, normal, etc. --></interpretation>
 <note><!-- 0..* Annotation Comments about the observation --></note>
```

Figure 9: A Pseudo-XML Definition of a FHIR Resource

3. General notes

These describe the purpose and scope of the resource, and any other pertinent notes.

4. Search Parameters

Each resource defines the search parameters that 'makes sense' for that resource. There is no requirement that a FHIR server should support all search parameters. Systems can use the conformance statement and profile to indicate what searches they do support.
Note that there is nothing stopping a FHIR server implementing any search parameter it wants to - but if a particular implementation requires a search not defined here, it is worth contacting the FHIR team to see if it is worth including in the main specification.

5. Terminology Bindings

When the datatype of a resource is a Code or a CodeableConcept, then it can optionally be 'bound' to a particular terminology by the resource designers.

10.1.3.1 Terminology Bindings

Path	Definition	Type	Reference
Observation.status	Codes providing the status of an observation.	Required	ObservationStatus
Observation.category	Codes for high level observation categories.	Preferred	ObservationCategoryCodes
Observation.code Observation.component.code	Codes identifying names of simple observations.	Example	LOINCCodes
Observation.dataAbsentReason Observation.component.dataAbsentReason	Codes specifying why the result (`Observation.value[x]`) is missing.	Extensible	DataAbsentReason
Observation.interpretation Observation.component.interpretation	Codes identifying interpretations of observations.	Extensible	ObservationInterpretationCodes
Observation.bodySite	Codes describing anatomical locations. May include laterality.	Example	SNOMEDCTBodyStructures
Observation.method	Methods for simple observations.	Example	ObservationMethods
Observation.referenceRange.type	Code for the meaning of a reference range.	Preferred	ObservationReferenceRangeMeaningCodes
Observation.referenceRange.appliesTo	Codes identifying the population the reference range applies to.	Example	ObservationReferenceRangeAppliesToCodes

Figure 10: A list of Terminology Bindings for a FHIR Resource

The **type** of the binding has a couple of options, depicted here from least to most flexible (the binding type is also called 'Binding Strength':

- **Example:** Example bindings are used when an element has a very broad meaning, or there is no consensus over the correct codes to be used. The system/code values MAY be one of the codes in the value set, or some other coded value MAY be used, or (for a CodeableConcept), a text alternative MAY be provided. Systems are not supposed to actually use the referenced CodeSystem.
- **Preferred:** Coded values for actual instances SHOULD be drawn from the referenced value sets, but If they are not, they will be considered conformant to the FHIR spec anyway.
- **Extensible:** This is a tricky type, what the spec says is that IF there is a code in the referenced value set that represents the concept to be transmitted, then it MUST be used. So the only case when you can use a local code is when the referenced value set cannot represent what you want to transmit, or, if you want to use your local code system, you need to do it additionally to the specified code system

6. Example Instances

The spec authors aim for having several examples for each resource class - in both XML, JSON and RDF (Turtle) formats. These examples are validated during the build process to make sure that they match the specification.
The intention is that eventually there will be examples that show all the possible variations on a resource - or at least the common ones.

7. Mappings

Where possible, mappings to the RIM of HL7 v3 and to HL7 v2 have been included. Other specifications are mapped to (when appropriate).

8. Schema + Schematron

Each resource has an XML schema that is specific to that resource (it is built automatically also). The Schematron assertions are manually entered and displayed in the spec as constraints to cover the rules that cannot be expressed in XML schema. In addition, the schemas and Schematrons can be downloaded at http://hl7.org/implement/standards/fhir/downloads.html.

9. Key Parts of a Resource

A resource has four main parts:

1. Metadata: logical identification, versioning, tags
2. The narrative or text (human readable) section. Described below, this allows a human to safely view a resource.
3. Structured, defined data, otherwise known as the 'core dataset' or ‘standard data’. This is the list of elements that appears in the specification, and which all FHIR implementers need to understand. To be included in this dataset, the rule of thumb is that 80% of systems currently support that property.
4. Any number of extensions. Extensions are described below. They allow implementers to add an element needed by their implementation but is not included in the core data set. This is similar to the HL7 V2.x 'Z segment', but there is a defined extension mechanism to avoid the issues that have occurred with Z segments in version 2.
5. (There is actually a 4th part that a resource can have - other 'contained' resources - but this is an advanced topic and is not discussed further in this Unit)

An example of this is:

```
<Patient xmlns="http://hl7.org/fhir">
  <id value="glossy"/>
  <meta>
    <lastUpdated value="2014-11-13T11:41:00+11:00"/>
  </meta>
  <text>
    <status value="generated"/>
    <div xmlns="http://www.w3.org/1999/xhtml">
      <p>Henry Levin the 7th</p>
      <p>MRN: 123456. Male, 24-Sept 1932</p>
    </div>
  </text>
  <extension url="http://example.org/StructureDefinition/trials">
    <valueCode value="renal"/>
  </extension>
  <identifier>
    <use value="usual"/>
    <type>
      <coding>
        <system value="http://hl7.org/fhir/v2/0203"/>
        <code value="MR"/>
      </coding>
    </type>
    <system value="http://www.goodhealth.org/identifiers/mrn"/>
    <value value="123456"/>
  </identifier>
  <active value="true"/>
  <name>
    <family value="Levin"/>
    <given value="Henry"/>
    <suffix value="The 7th"/>
  </name>
  <gender value="male"/>
  <birthDate value="1932-09-24"/>
  <careProvider>
    <reference value="Organization/2"/>
    <display value="Good Health Clinic"/>
  </careProvider>
</Patient>
```

Resource Identity & Metadata

Human Readable Summary

Extension with URL to definition

Standard Data:
- MRN
- Name
- Gender
- Birth Date
- Provider

Figure 11: Key parts of a FHIR resource

Narrative

The FHIR specification states:

> *Each resource may include a human-readable narrative that contains a summary of the resource, and may be used to represent the content of the resource to a human. If narrative is present, it SHALL reflect all content needed for a human to understand the essential clinical and business information otherwise encoded within the resource. Resource definitions may define what content should be represented in the narrative to ensure clinical safety.*

Core Content or Standard Data

The Core content of the resource are those common elements that the responsible committee have determined are currently used by the majority systems that hold data matching the resource – this is known as the "80% rule", as this is a rule-of-thumb used to help with the process -.

The purpose of this is to focus the resource design **only on the things that are widely agreed** as opposed to the v3 (and openEHR) concept of a 'maximal dataset' – resulting in very bloated resources and endless discussions and decades of development time, due the wide variability in process and design.
This is one of the most hotly contested aspects of resource design. However, in practice, it works because FHIR enables formal definition of extensions to cover elements needed by a particular project but not in the common core specification.

Extensions

The extension mechanism is what sets FHIR apart from HL7 version 3 and - oddly enough - closer to version 2. FHIR has a philosophy of the '80%' - a particular property of a resource is only included in the 'core' resource if it is currently being used by 80% of existing systems. Not surprisingly, what is in the 80% is probably the most contentious aspect of FHIR!
Taking this approach has the advantage that the resources themselves are kept to a manageable size, and much easier to implement that if every requirement from every realm needed to be in every resource.
However, doing so creates the need for a mechanism to allow a particular realm to add the properties that it needs to record within a resource. For example, in New Zealand there are the concepts for a patient of 'iwi' and 'hapu' - the tribe and sub-tribe of the native Maori population. These concepts are of no interest to a North/South American or European audience.
It is to accommodate these types of requirement that the FHIR extension mechanism has been defined. Extensions can be quite sophisticated and it is worth reviewing the relevant section of the FHIR specification for details: www.hl7.org/implement/standards/fhir/extensibility.html
Extensions are defined within a profile.

Simple Extensions: adding an element

In its most basic form, an extension can be used to hold the value of one element that is not in the Core dataset.
Each extra property that needs to be recorded has its own extension, and there can be any number of extensions in a resource. Extensions can also be nested if required. In the example above there would be 2 extensions - one for "iwi" and one for "hapu".
Each extension has the following properties:

Name	Description
url	This is a reference to the profile within which the extension is defined. This means that anyone who receives a resource with an extension that they are unfamiliar with can download the definition of that extension. Refer to the discussion of the profile for further information.
value[x]	The actual value of the resource. As in the Observation example above, the '[x]' signifies the datatype of the element in exactly the same way as it does for core elements.

In order to avoid the issue that arose with V2.x Z segments where each jurisdiction defined its own segments without regard for what other have done, the FHIR team intends to establish a hierarchy of profile repositories that an implementer can query both to determine what a particular extension means, and also to see if there is already an extension defined for their particular need. This is made even easier because a profile is itself a resource, and so can be stored in (and queried from) a FHIR server. The hierarchy that is envisaged includes:

- An 'official' HL7 registry that has extensions for data elements that were not' common enough to make the 80%, but which are nevertheless often required. For example: a patient's religious affiliation. This registry is published along the core specification, here: http://hl7.org/fhir/extensibility-registry.html
- FHIR Global Community registry: Extensions defined by the FHIR Community: https://registry.fhir.org
- Realm (or country) registries that specify extensions that are specific to that realm (and the iwi/hapu example above would fit here). Several examples can be found these sites:
    - Simplifier: https://simplifier.net/
    - Trifolia on FHIR: https://trifolia-fhir.lantanagroup.com/home

There is no difference technically between the profiles stored in these registries - simply the governance that is around them.
It is also important that once an extension is in use it should not be changed - if you need to make a breaking change, then create a new extension.

Extending a core element

Extensions are also used when it is necessary to modify or extend a core element. For example, the *Patient.name* element (which is a *HumanName* datatype) contains elements for family name, given name, prefix and suffix - what if you wanted to identify a particular given name as a person's middle name due to the special requirements of the local environment?
The page at www.hl7.org/implement/standards/fhir/extensibility-examples.html describes in detail how this is done - first defining the extension in a profile, and then referring to the extension in the resource instance as follows:

```
<name>
 <use value="official" />
 <given value="Ian">
   <extension url="http://hl7.org/fhir/Profile/iso-21090#name-qualifier" >
     <valueCode value="MID" />
   </extension>
 </given>
</name>--
```

Figure 12: This example shows how to define a middle name of 'Ian'.

Once an extension has been defined, it can be referred to by any resource that needs it. In the example above, any resource author that wishes to specify a middle name could use the same extension definition.
Because it is possible for an extension to change the meaning of an element within a resource, e.g. to quantify or negate it's primary meaning, there is a specific element - *modifierExtension* - you can add to the profile of the extension to make sure that the recipient understands what it means. If a FHIR client doesn't understand an extension which has modifierExtension set to true, then it should either alert the user or not process the resource. These kind of extensions should be avoided by implementers if possible.
For further discussion on extensions, please see:
www.hl7.org/implement/standards/fhir/extensibility.html

Resource Metadata

There are a number of aspects to a resource that are more 'about' the instance of the resource itself than about the actual contents of the instance. They are not included as elements within a resource, but a FHIR server should maintain them.

Identity

All resources have the concept of identity. The identity is fixed over the lifetime of the resource - a change to the resource does not change the identity. The identity can be set either by the client that creates the resource, or (usually) by the server that receives a new one.

Version

The version of the resource instance changes each time the content of a specific resource instance changes. Combining the identity and the version leads to two important concepts:

- **Logical id.** The fixed identity on the server that hosts the resource. When you ask for a resource from a server based on the logical id, then you will get the most recent version of the resource (This is discussed further in the REST section below). Note that the id can be absolute (specifying

the server) or relative (to the containing structure - like a bundle or a resource or a web page). An example of this could be:

```
<server>/FHIR/Patient/100
```

- **Version specific id.** This is the identity of a particular version of a resource - it may or may not be the most recent version. An example of this could be:

```
<server>/FHIR/Patient/100/_history/2.
```

Refer to the REST section below for more details and examples.

Last Updated

This is the date / time that the resource was last updated, and is set by the server as it updates a resource. It is possible to retrieve resources based on this date/time, which might be useful if you are wanting to synchronize resources between servers, or have a feed that informs you when resources change.
Refer to the specification at https://www.hl7.org/fhir/resource.html for more information about the resource metadata (tags, security, etc.)

How to get all the versions of a resource

This can be achieved using the history of a resource, at different levels of granularity. The following examples are using the REST interface:

```
GET /FHIR/Patient/2/_history?_since=2012-12-01
```

Return all versions of the patient whose id is 2 since December 1st 2012

```
GET /FHIR/Patient/_history?_since=2012-12-01
```

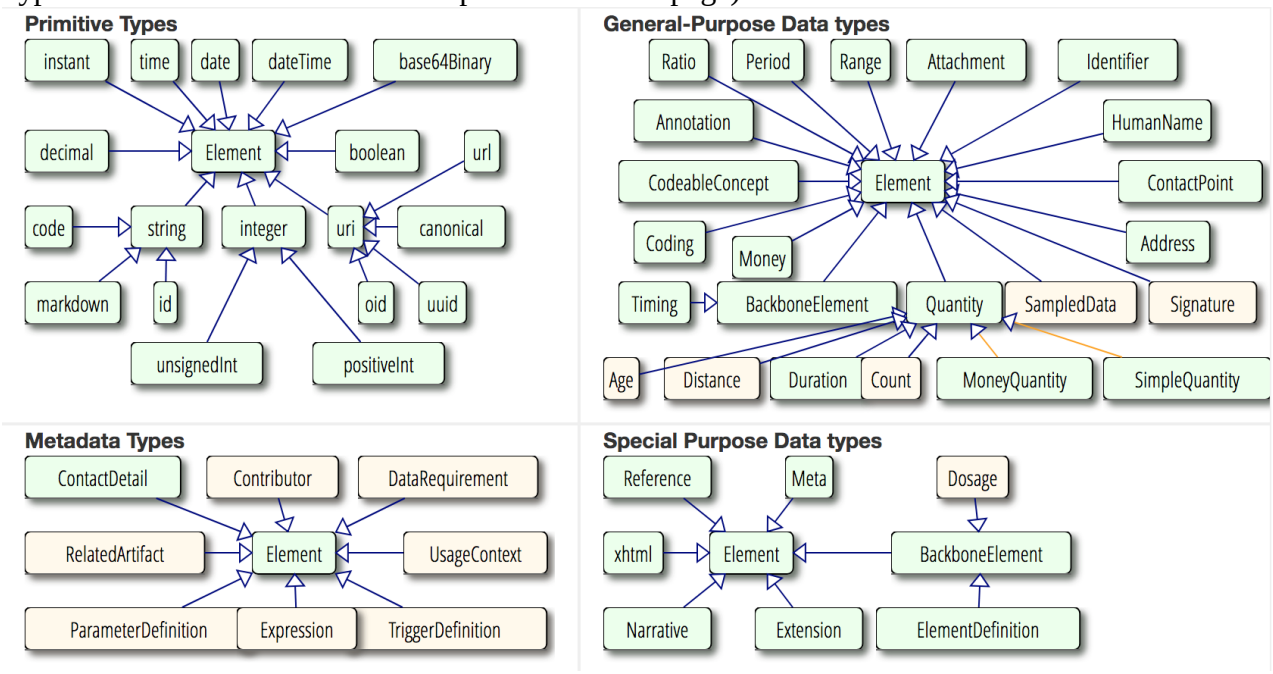
Return all versions of all patients since December 1st 2012

```
GET /FHIR/_history?_since=2012-12-01
```

Return all versions of all resources on this server modified since December 1st 2012
The resulting bundle can be very large!
Sorting and filtering on date can be tricky, particularly concerning time zones, so refer to the specification for more details if performing these types of query.

10. Data Types

Each element within a resource is of a particular datatype (This is the same for all HL7 standards of course). The FHIR datatypes are a simplified version of the V3 datatypes - and are also based on the W3C schema) and the following diagram gives an overview (in the specification, each image hyperlinks to a more detailed description within the page):



Primitive types

The specification describes these datatypes. Some that are more commonly used include:

Code

A code is used when it is expected that a computer system will need to make a 'choice' based on the code - for example, there is a workflow associated with the resource. Example: the 'status' of a Condition Resource might influence whether to display a condition in the patients list of conditions or not.

The possible values of the code will usually be set in the definition of the resource itself, unless it is a reference to an external internet specification, like mime/type.

CodeableConcept

The CodeableConcept is the most commonly used datatype for coded data, as it is the most flexible. It is analogous to the CE datatype of V3 or CWE/CNE from V2 and is defined as follows:

```
<[name] xmlns="http://hl7.org/fhir">
 <!-- from Element: extension -->
 <coding><!-- 0..* Coding Code defined by a terminology system --></coding>
 <text value="[string]"/><!-- 0..1 Plain text representation of the concept -->
--</[name]>
```

The key features include:

- It supports multiple codes (or “translations”) which allows common resource instances to be used in situations where different recipients might require different codes.
- It also allows for migration between code systems over time.
- It supports conveying just a textual representation of the concept, which is important for situations where no appropriate code exists or where the recipient does not recognize the code system or wishes to see the full detail of the original concept, not what was expressible by the selected code.

Resource reference

The resource reference is a 'special' 'datatype', as it allows one resource to refer to another. A very common example of this is a resource representing a clinical concept like a problem or an observation, which needs to refer to the patient resource.
A resource reference has the following format:

```
<[name] xmlns="http://hl7.org/fhir">
 <!-- from Element: extension -->
---- <reference value="[string]"/><!--0..1 relative, internal or absolute URL reference -->
 <display value="[string]"/><!-- 0..1 Text alternative for the resource -->
</[name]>
```

The contents are:

- [name] : the name of the reference within the resource. For example, the Condition resource has a 'subject' reference to a Patient resource.
- -reference: the URL reference to the resource. This can be a relative, an internal or an absolute reference and can also be version specific. See the discussion below for further details on this.
- Display: this is to visually identify what is being referenced. For example, if the reference is to a patient then it might contain the patient name. Specifically, **it is not** the same as the text element of the referenced resource.

For example:

```
<subject>
 <reference value="Patient/example"/>
 <display value=""patient example"/>"
</subject>
```

This example would indicate that the subject of this reference is a patient whose name is 'patient example', has the id 'example' and can be found on the same server as the resource containing the reference. "

Contained Resources

A variant on the resource reference, where the resource being referenced is actually contained within the 'parent' resource. An example of this might be in a care plan where the plan is a short term one concerning a condition that is not in the patients main list of conditions.
The following snippet shows a condition 'included' in a care plan:

```
<contained>
  <Condition id="p1">
    <subject>
     <reference value="Patient/example"/>
     <display value="Peter James Chalmers"/>
    </subject>
    <code>
     <text value="Obesity"/>
    </code>
  </Condition>
</contained>
```

Note that the Condition element has an id - this is used as an internal (to the instance) reference (it is not like the id that identifies a resource) and in this example is the target of a reference from the concern element as shown in this snippet:

```
<concern>
  <reference value="#p1"/>
  <display value="obesity"/>
</concern>
```

An important aspect to contained resources is that the resource being contained does not have its own identity - i.e. it is not stored on a server somewhere independent of the resource that contains it. If it is, then the reference should point to the resource in the usual way.
In the words of the spec: "*This should never be done when the content can be identified properly, as once identification is lost, it is extremely difficult (and context dependent) to restore it again.*"

11. Bundles

There are many situations where a collection of resources is required. These include:

- The results from a search
- The collection of versions of a particular resource
- A FHIR document
- A FHIR message
- A batch of resources to be processed.

In all these scenarios, FHIR uses the Bundle resource to represent the collection of resources.
A bundle has some header information, and then any number of resources (entries).
The header contains:

- The identification for the bundle
- The time (as an instant) when it was assembled
- An identifier for the bundle. This is used if the bundle itself is saved.
- A code identifying the bundle **type** (document, message, transaction, batch, search set, collection, response)

Each entry contains a resource and, for some bundle types – transaction, batch- a request (which states what to do with the resource.
Here you can find an example of a FHIR bundle: https://www.hl7.org/fhir/bundle-transaction.xml.html

12. Profiles

The resources that are described by FHIR can be used in many different healthcare contexts, so it is often necessary for a particular project to be more specific on exactly how it needs to use these resources. This includes:

- Explain how a set of resources is used in a particular context
- Describe restrictions on the use of the elements defined as part of the resource(s)
- Define the extensions that are used with the resources
- Define the searches that apply in this context.
- Describe how resources are bound to terminology in a particular context

All these things are described using a resource called **StructureDefinition** that describes how resources are used in a particular context. Profiles have a metadata section that describes who published the profile, and why, as well as optional lists of resources, constraints, extension definitions, and vocabulary bindings.

Profiles are extremely important in FHIR, but can be complex to develop and use.

There are several tools to assist in developing FHIR profiles, among them:

Trifolia On FHIR (Web Based): https://trifolia-fhir.lantanagroup.com/home

Forge (Windows Based): https://simplifier.net/forge

Here you can see a profile for a ClinicalDocument: https://www.hl7.org/fhir/clinicaldocument.html

Interestingly, the base FHIR resources are themselves described using StructureDefinition resources.

Profiles are an extremely important aspect to FHIR, but you do not need to fully understand them to get started.

Here is a list of all HL7 defined, 'official' profiles: https://www.hl7.org/fhir/profilelist.html

13. Security & Audit

The security aspects of FHIR are still in active development, and IHE is contributing significantly to this work. FHIR itself does not define security functionality, but does depend on other services to provide that security - of which OAuth (http://oauth.net/) is one of the more important.
See www.hl7.org/implement/standards/fhir/security.html for details.
The specifics of how security concerns are addressed will vary according to the particular paradigm being used - REST, Message, Document or Service.
From an audit perspective, there are a number of resources that can be used to record activities required for auditing. These include:

- **Provenance**. This resource is used to indicate where a particular resource came from. Note that the provenance resource points to the resource that it describes and not the other way around.
- **SecurityEvent** resources are equivalent to the IHE ATNA Audit Record.

14. Interoperability Paradigms

As described above, it is intended that the same FHIR resources should be able to be used in all the interoperability paradigms required in healthcare. For example, a patient resource is the same no matter how it is exchanged.
These include:

REST

This stands for Representational State Transfer, and is used for on-line, real-time access to information using HTTP protocols - like a web browser. It can be used for updates as well as querying for information. REST has become very popular and is used by many other applications - such as Twitter or Facebook - due to its simplicity and ease of use.
FHIR fully supports REST - in fact, it is the most developed so far, but is not confined to that.
Most of the examples in this module use the REST paradigm.
There is a good tutorial on REST at http://rest.elkstein.org

Messages

A message is used when you want to send information from one system to another, and you expect the recipient system to update itself as required and then delete the message (other than any audit records of course).
HL7 V2.x is all about messaging in this way. More on FHIR messaging here:
https://www.hl7.org/fhir/messaging.html

Documents

A document is all about recording a set of information that applies at a 'point in time' about a patient - such as a Discharge Summary or a Referral. CDA is all about documents, so refer to the module about CDA for further information. The FHIR take on documents is described below (p38).

Service

A Service is also intended to be used in an on-line real-time way (usually), but the difference from REST is that a service can incorporate more complex workflow than the simple REST interface can provide. For example you might use a Service in an ordering application if you wanted the service to apply basic decision support to the order, possibly modifying or rejecting it.
Use of Services is not further described in this module, but you can read more here:
https://www.hl7.org/fhir/services.html

FHIR REST Basics

The REST part of FHIR is the one that has received the most attention thus far.
REST is an architectural style rather than a standard and seeks to fully utilize the HTTP protocol (http://en.wikipedia.org/wiki/Http) in proving access to resources - it is not simply XML over HTTP - and thus uses the verbs, headers and status codes defined by HTTP.
The FHIR spec utilizes HTTP as faithfully as possible, and the page at http://hl7.org/implement/standards/fhir/http.html describes this usage in detail.
So let's start playing with FHIR…

15. Playing with FHIR

This is the section where you can start to do some practical things with FHIR - and you are encouraged to do so! We will use one of the test servers that have been established - see the list at:
http://wiki.hl7.org/index.php?title=Publicly_Available_FHIR_Servers_for_testing

1. Getting started

The commands in this section can be directly entered into a browser as we are only going to use GET functions. However, the response will be displayed in different ways by different browsers and you will not be able to see (and set) the HTTP headers, which are an important part of REST. Most of the common browsers have 'plugins' that give you more control over this process. In case we could not find a proper plugin for a browser, we recommend using a platform specific application.

Note: We have searched and tested all these plugins / apps with our examples, which is no guarantee that they will exist forever or will continue working with our examples. ples. We do not endorse the use of any of these tools. They are just shortcuts to your Google search (but we also tested them for basic REST use).

Postman (recommended for all platforms) at:
https://www.getpostman.com/
There is a small video on how to use Postman here:
https://www.youtube.com/watch?v=jBjXVrS8nXs
Firefox: There is a handy plugin called RESTClient at:
https://addons.mozilla.org/en-US/firefox/addon/restclient/
Internet Explorer: We could not find a plugin, but you can use Fiddler if you are running Windows. You can get Fiddler here https://www.telerik.com/fiddler
MacOS: Cocoa Rest Client at:
https://mmattozzi.github.io/cocoa-rest-client/
iOS (iPhone, iPod, iPad): HTTP bot:
https://itunes.apple.com/us/app/httpbot/id1232603544?mt=8
If you cannot use none of these because of security/policy problems, you can also try this online:
https://apitester.com/

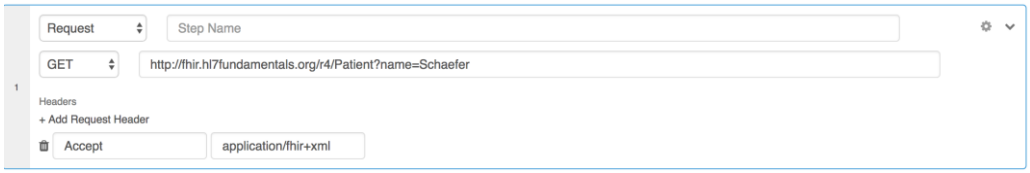
2. A simple search

Let's find all patients whose name contains the letters 'Schaefer'. Enter the following url:
http://fhir.hl7fundamentals.org/r4/Patient?name=Schaefer
What you will get back is a Bundle with the matching patient resources (two of them), e.g.:

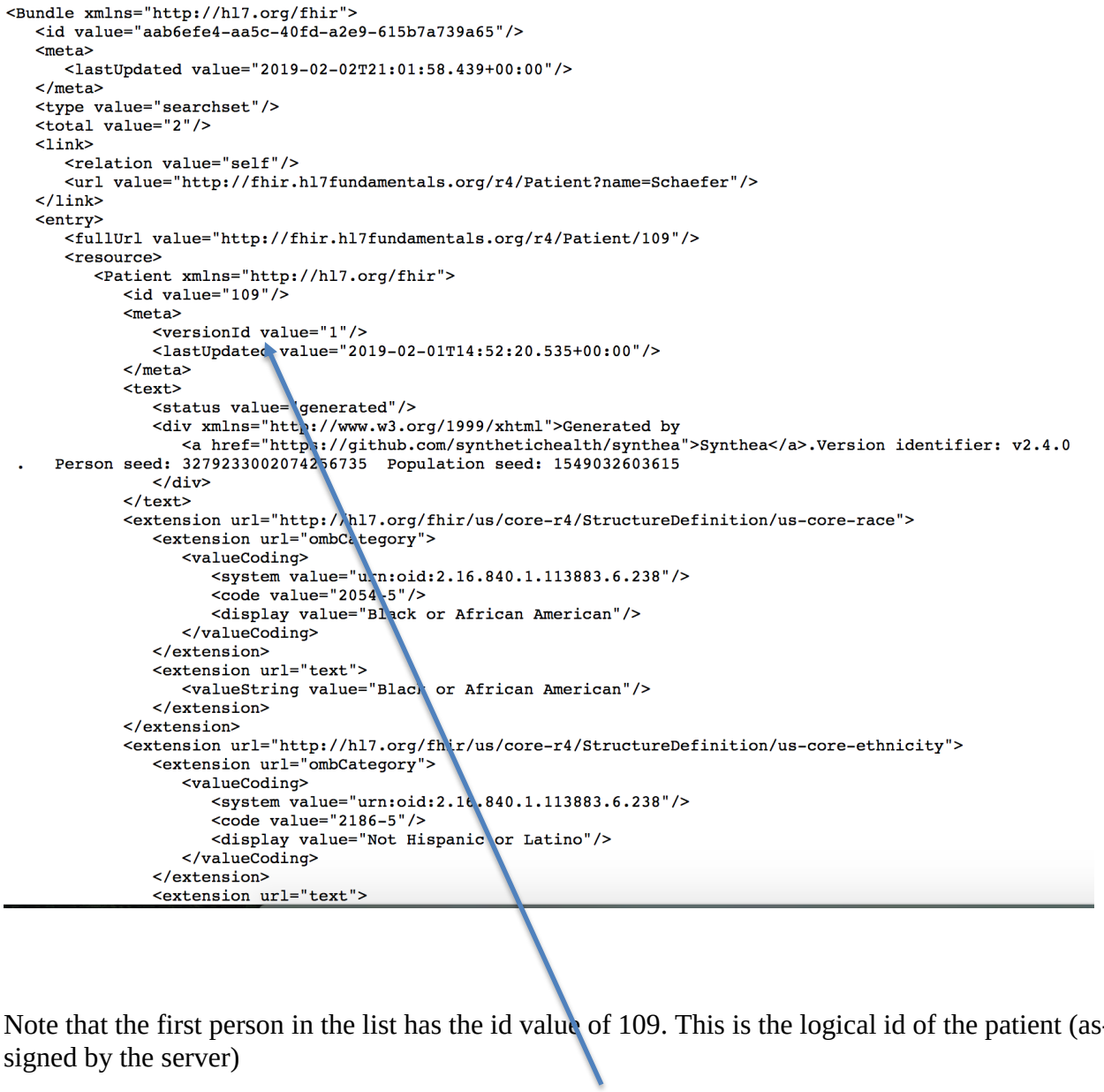
```
{
  "resourceType": "Bundle",
  "id": "f7a8d050-dd5b-4327-9d24-08da862820ae",
  "meta": {
    "lastUpdated": "2019-02-02T20:52:27.061+00:00"
  },
  "type": "searchset",
  "total": 2,
  "link": [
    {
      "relation": "self",
      "url": "http://fhir.hl7fundamentals.org/r4/Patient?name=Schaefer"
    }
  ],
  "entry": [
    {
      "fullUrl": "http://fhir.hl7fundamentals.org/r4/Patient/109",
      "resource": {
        "resourceType": "Patient",
        "id": "109",
        "meta": {
          "versionId": "1",
          "lastUpdated": "2019-02-01T14:52:20.535+00:00"
        },
        "text": {
          "status": "generated",
          "div": "<div xmlns=\"http://www.w3.org/1999/xhtml\">Generated by <a href=\"https://github.com/syntheticheа
seed: 3279233002074256735  Population seed: 1549032603615</div>"
        },
        "extension": [
          {
            "url": "http://hl7.org/fhir/us/core-r4/StructureDefinition/us-core-race",
            "extension": [
              {
                "url": "ombCategory",
                "valueCoding": {
                  "system": "urn:oid:2.16.840.1.113883.6.238",
                  "code": "2054-5",
                  "display": "Black or African American"
                }
              },
              {
                "url": "text",
                "valueString": "Black or African American"
              }
            ]
          },
```


The server will answer in its preferred format (in this case, JSON).
To get the list as JSON, you have two options:

- create a Request Header called 'Accept' with a value of 'application/FHIR+xml' (You will need an HTTP client that allows you to specify this).



- or add another parameter to the url - _format=xml
    - http://fhir.hl7fundamentals.org/r4/Patient?name=Schaefer&_format=xml
- You should get a response like this (in XML):



```
<Bundle xmlns="http://hl7.org/fhir">
   <id value="aab6efe4-aa5c-40fd-a2e9-615b7a739a65"/>
   <meta>
      <lastUpdated value="2019-02-02T21:01:58.439+00:00"/>
   </meta>
   <type value="searchset"/>
   <total value="2"/>
   <link>
      <relation value="self"/>
      <url value="http://fhir.hl7fundamentals.org/r4/Patient?name=Schaefer"/>
   </link>
   <entry>
      <fullUrl value="http://fhir.hl7fundamentals.org/r4/Patient/109"/>
      <resource>
         <Patient xmlns="http://hl7.org/fhir">
            <id value="109"/>
            <meta>
               <versionId value="1"/>
               <lastUpdated value="2019-02-01T14:52:20.535+00:00"/>
            </meta>
            <text>
               <status value="generated"/>
               <div xmlns="http://www.w3.org/1999/xhtml">Generated by
                  <a href="https://github.com/synthetichealth/synthea">Synthea</a>.Version identifier: v2.4.0
   .   Person seed: 3279233002074256735  Population seed: 1549032603615
               </div>
            </text>
            <extension url="http://hl7.org/fhir/us/core-r4/StructureDefinition/us-core-race">
               <extension url="ombCategory">
                  <valueCoding>
                     <system value="urn:oid:2.16.840.1.113883.6.238"/>
                     <code value="2054-5"/>
                     <display value="Black or African American"/>
                  </valueCoding>
               </extension>
               <extension url="text">
                  <valueString value="Black or African American"/>
               </extension>
            </extension>
            <extension url="http://hl7.org/fhir/us/core-r4/StructureDefinition/us-core-ethnicity">
               <extension url="ombCategory">
                  <valueCoding>
                     <system value="urn:oid:2.16.840.1.113883.6.238"/>
                     <code value="2186-5"/>
                     <display value="Not Hispanic or Latino"/>
                  </valueCoding>
               </extension>
               <extension url="text">
```

Note that the first person in the list has the id value of 109. This is the logical id of the patient (assigned by the server)

Note that the first person in the list has the id value of 109

3. Read a single patient

Since the first patient in the list has the id value of 109, we will just directly **GET** that specific resource. Enter the URL: http://fhir.hl7fundamentals.org/r4/Patient/109

and you'll get:

REST in Detail

The following sections discuss how to make RESTful queries and updates against a server, but they are by no means comprehensive. You should have the specification open and be prepared to experiment. You will need a client that allows you to specify HTTP methods and headers. Examples of these tools were discussed before in this unit.
During this section, we will refer with <endpoint> to the server FHIR address.
Our course server's endpoint for FHIR R4 is http://fhir.hl7fundamentals.org/r4

Add a new resource

There are two distinct ways to create a new resource - and the one you choose depends on whether the server allows the client to specify the resource id while creating it.
In both of these examples, it is assumed that you have created a resource (in either XML or JSON) and can refer to it using your tool of choice.

Tip: You might find it easier to GET a resource from the test server as described below and then modify that in a suitable editor!

PUT Method + Client Assigns the ID (less common, not all servers allow this)

If the client is assigning the resource Id then use the PUT verb, specifying the 'location' of the resource (which is the server name plus the id):

```
PUT <endpoint>/<resourceType>/<id>
```

You can specify the format of the resource being saved (XML or JSON) either by setting the Content-Type header, or adding an _format parameter to the request – e.g.

```
PUT <endpoint>/<resource>/<id>?_format=json to specify json.
```

If there is already a resource at that location, then it will be replaced by the resource that you supply in the body of the request. Ideally, the server will create a new version, thus maintaining a history of changes for that resource.
If the request is successful, then:

- If there was an existing resource at that location then the status code is 200, otherwise it is 201.
- The **Location** and **Content-Location** headers will be set to the full url (including version)
- The **Last-Modified** header will be the date of update.

The server is able to reject the update for business reasons; in which case the status code is 422.
There are a number of other possible failure codes described in the specification.
The PUT method is also used when what is needed is to update an existing resource.
Note that it is only appropriate to use client assigned Ids in a trusted environment, e.g. where you trust the clients not to overwrite each other's resources.

POST Method - Server Assigns the ID

If the server has the responsibility for creating the Id, then the client should simply POST the resource to the resourcetype root:

```
POST <host>/<resourceType>
```

The server will create a new resource of this type, returning a status code of 201 and setting the response headers as above.

Idempotency

The concept of idempotency is also important when considering POST & PUT.
A method that is idempotent will produce the same results if executed once or multiple times.
Thus, a PUT is idempotent as repeating it simply results in a new version of the resource with the same content, but a POST is not as each call results in a new resource being created.

Retrieve a Resource

To retrieve a resource based on its id, issue a GET request based on the following pattern

```
GET <endpoint>/<resourceType>/<id>
```

Where:

- <endpoint> is the address of the FHIR endpoint of the server (example: http://fhir.hl7fundamentals.org/r4)
- <resource> is the name of the resource class (example: 'Patient' or 'Observation')
- <id> is the id of the resource you are after.

Thus:
GET http://fhir.hl7fundamentals.org/r4/Patient/109
will return the patient resource with the id of 109 from our course's server.
The response status is important:

- 200 indicates that the resource was found
- 410 means there was a resource with this id, but it has been deleted
- 404 indicates 'not found'.

The server will also return a number of response headers. These include:

- **Content-Location** will be the full location to the resource, including any version - i.e. a version specific URL
- **Content-Type** will be the format of the resource - XML or JSON

But there will be others.
You can specify the format that the server should return the resource in two ways:

- In the request, set the request header '**Accept'** to *application/json+fhir* or *application/xml+fhir* to specify JSON or XML respectively. This is the preferred approach.
- In the request URL, use the "_format" parameter to specify the format: http://fhir.hl7fundamentals.org/r4/Patient/109?_format=json

The latter option is provided as a convenience for those clients that cannot (or will not) set request headers.
You will generally find out the server assigned id of a resource as a result of a search operation (see below).xw
A GET is considered 'idempotent' - you can issue it any number of times without affecting the server.

Search for a Resource

In most cases, a client will not know the resource id, but rather will perform a search of some sort to find the resource or resources it wants, and can then get the resource id from the returned bundle.

Searching is one of the most important and complicated parts of FHIR. The reader is advised to study the specification www.hl7.org/implement/standards/fhir/http.htm#search as well as reading this section. We will cover only the basics in this introductory Unit.

Also, note that the searches that can be applied to a particular resource can also be specified in the Profile resource.
In the examples below we will omit the server name to save space, thus:

```
/Patient?name=klein
```

should really be

```
http://fhir.hl7fundamentals.org/r4/Patient?name=klein
```

The general format of a search request is:

```
GET <endpoint>/<resourceType>?param1=&param2= …
```

Where the parameters are the filters to apply to the search. They will generally match an element in the resource - though they don't have to. The parameters have a parameter type as described in the specification. Most of these are straightforward - the tricky one being the token that specifies a namespace as well as a value and is used for elements with datatypes Coding and CodeableConcept.
When multiple parameters of the same name are given, the result can return resources that contain any of the parameters (union) or only resources that contain all of the parameters (intersection). To indicate a union is wanted, use ',' in the search, for an intersection use "&".
For example:

- When searching for Patients who speak either Dutch or French, use /Patient?language=NL, FR
- When searching for Patients who speak both Dutch and French, use /Patient?language=NL&language=FR

'The server should also honor the format request in the Accept header or the _format parameter.
The result of a search request is always a bundle containing the matching resources (even if there is only a single result).
There are a number of 'standard' searches that apply to all resources, and in addition, each resource can define other search parameters that are specific to that resource. We will focus on the Patient resource.

Simple Search

For simple searches - ie for a single resource - simply create a GET request with the required parameters. E.g.

/Patient?name=eve
/Patient?gender=M

Searching across Resources

Cross resource searching is a bit complex at present.
A common scenario is 'de-reference' a resourceReference datatype - for example suppose you want all the conditions for a particular patient.
Examining the Condition resource shows that the patient for a condition is given by the 'subject' element, which is a resource reference - i.e. it refers to the Patient resource with a given id. So, if the patient id is 'example', then:

As we saw before, **/Patient/109** would return the Patient resource with id 109

But **/Condition?subject._id=109** would return all the problems where the subject (patient) was the resource whose id was '109'

Note that this will only work if all of the resources are on the same server. The situation gets a lot more complicated if there are Condition resources on different servers, or where the Condition resource on one server has a resource reference links to a patient on another server.

Retrieve the History of a specific Resource

As described above, FHIR has the concept of resource versions. A server is not obliged to provide versioning functionality, but if it does, then there is a specified way of using it.
The format for this request is:

```
GET <endpoint>/<resourceType>/<id>/_history
```

This will return a bundle containing all the previous versions for the resource.
(As an aside, it is also possible for a server to return history at a higher level - either all the changes for a specific resource type, or all resources. This is intended for use in server synchronization scenarios).

Retrieve a specific Version of a Resource

You can also return just the specific version (assuming you have both the resource id and the version id).
The format for this request is:

```
GET <endpoint>/<resourceType>/<id>/_history/<versionId>
```

The result will be the specified version of the resource (unless it was deleted, in which case the server will return a 410 status code).

Delete a Resource

To remove a resource, use the DELETE verb, e.g.:

```
DELETE <endpoint>/<resourceType>/<id>
```

Assuming the delete was successful (i.e.: there was no business process that prevented it occurring), then the server will return a status code of 204 (no content). If the delete cannot occur, then the returned status code is 405 (method not allowed), 409 (conflict) or 404 (not found).
A successful delete instruction means that the server will no longer return a resource to a simple GET request for that resource. If such a request is made, then the server will return a status code of 410 (gone) - note that this is different to requesting a resource that has never existed, which would return a 404.
The previous versions of the resource are not removed, and can be retrieved using a version specific request.

Transaction Batch Updates

The transaction update facility allows a client to submit a number of different resources in a single bundle. There are two main purposes:

- To support the notion of a transaction (as the transaction processing must succeed or fail as a whole)
- To support push-based publication-subscription - e.g. where different servers are being synchronized.

From a server perspective, transaction processing is hard - especially with regard to id management, but from a client perspective it is easy - just create a bundle with the given resources and POST it to the server root.

The server processes the transaction as if each resource had been submitted separately (though does need to treat the transaction as a transaction and be able to roll back a failed update).

After processing, the updated transaction is returned to the client (and there may be some differences between the submitted transaction and the processed transaction - especially if Id's were assigned by the server).

Note that although the transaction processing uses a bundle to contain the resources to be processed, there are a couple of important differences:

- The order of resources in the transaction must not be significant.
- Any one resource can only occur once in the transaction.

If you do intend to use transaction processing, then read the specification carefully at https://www.hl7.org/fhir/http.html#transaction

16. FHIR-Specific REST Features

The Binary Resource

The binary resource http://www.hl7.org/implement/standards/fhir/binary.html allows a FHIR server to store arbitrary binary data, but reference and manipulate it in the same way as any other resource with a couple of exceptions:

- There is no provision for search (as the contents are opaque)
- The format is fixed to that of the document (and Content-Type HTTP headers are used to indicate this)

The actual endpoint for binary resources is [host]/Binary
A common use of the binary resource is in the XDS support (see next section) where the DocumentReference resource has the location element, which is a url that can point to the document stored in the binary "endpoint".
The following snippet illustrates this.

```
...
<location value="http://example.org/xds/mhd/Binary/07a6483f-732b-461e-86b6-edb665c45510"/>
...
```

Another scenario could be images references by the ImagingStudy resource:

```
<image>
    <number value="1"/>
    <uid value="urn:oid:2.16.124.113543.6003.189642796.63084.16748.2599092903"/>
   <dicomClass value="urn:oid:1.2.840.10008.5.1.4.1.1.2"/>
    <url
 value="http://localhost/fhir/Binary/1.2.840.113619075792384034087 00.3.0.14.19970327150033"/>
  </image>
```

You can read more details about treatment of Binary resources here:
http://www.hl7.org/implement/standards/fhir/binary.html#rest

Specific REST-based Use Cases

XDS

The IHE XDS profile is designed to support the sharing of documents between systems. More detail is available at this link (http://wiki.ihe.net/index.php?title=Cross-Enterprise_Document_Sharing), but basically it defines an 'affinity domain' of co-operating systems that contains a single registry holding 'index' information about documents that are stored in multiple repositories within the domain. (It gets a lot more complex than this in practice, but this will do for now).

There are then a number of specific transactions that are defined, including:

1. A **document source** saves a document (along with the required metadata) to a **document repository**, which in turn registers the document/s with the **document registry.**
2. A **document consumer** queries the **document registry** for a patient and other query parameters, and receives a list of documents that match that query
3. From the list, the **document consumer** then queries the indicated **document repository** directly for the document/s that it wishes to view.

The XDS profile supports many different types of document - PDF, CDA, text - but has a fixed set of metadata about those documents.

In FHIR the registry entry is represented by the **DocumentReference** resource which contains all the required XDS metadata to enable a document consumer to query the registry. (in exactly the same way as querying the document registry would do). Thus a **document consumer** in FHIR would do a normal FHIR query against the **DocumentReference** resource to return a list of matching resources (in an atom bundle of course), and can then query the repository directly to retrieve the document.

The location of the document is given by the value of the **location** property of the resource - which is a standard HTTP URI. This could be a FHIR document (stored at the **binary** endpoint of the server) or any other valid URI. It is up to the repository to enforce any security that is required.

Expressing the above transactions in FHIR-speak, and assuming that you are saving a PDF document to a FHIR Sever, then one way you can achieve this is as follows:

1. A **document source** saving a document. This actually has two parts:
   a - The **document source** saves the PDF to the **binary** endpoint of the FHIR server. This can be either a POST (where the server will assign an id and return it in the location header) or a PUT where the document source assigns the id, and the server will replace any existing document at that location.
   b - The **document source** creates a **DocumentReference** resource, and saves it in the FHIR server.
2. Querying the registry for matching documents is a standard FHIR search against the **DocumentReference** endpoint.
3. Retrieving the document is a standard HTTP GET against the URI of the document from the search results bundle.

Although the first transaction - adding a new document - is more complex than the IHE equivalent, the big advantage is that the whole process is 'normal' FHIR transactions - nothing special.

There are a number of alternatives to the '2 phase approach' of storing a document.

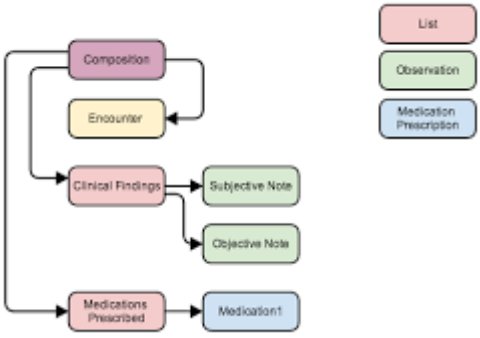
- You could place the DocumentReference and the content (as a base-64 encoded Binary resource) into a bundle, and POST it to the server root and have the server do all the work

- Place the Binary resource inside the DocumentReference resource as a contained resource. This does have the side effect of storing the content with the DocumentReference which may or may not be required - or desirable.
- Create a custom Service that accepts both binary and DocumentReference and performs the required actions.

You can read more about the FHIR R4 version of the IHE XDS profile (called MHD) here: https://www.ihe.net/resources/public_comment/#IT

FHIR Documents

A document is all about recording a set of information that applies at a 'point in time' about a patient - such as a Discharge Summary or a Referral. It is very similar to a CDA document - intentionally so - and HL7 is working on how to easily convert between CDA and FHIR documents. Note that this will probably only work well where the CDA document is fully "templated", as there many ways to represent the same thing in CDA. Of course, this is mainly a problem for level 3 CDA - level 2 is likely to be easier.
The module in this on-line training program on CDA has more details on the overall nature of documents, but briefly a document has a header that establishes context (patient, author, date, document type) and a body containing the data in one of more sections (e.g.: medications, lab results, problems).
In FHIR, a document is constructed using a bundle that contains resources just like any other bundle, with the addition of a specific resource - the **Composition** resource - that contains information equivalent to the CDA header, plus any number of section elements that point to other resources. The Composition resource must appear first in the bundle.



FHIR Clinical Document (Credit for the Picture: Dr. David Hay, https://fhirblog.com/tag/cda/)
You can find a FHIR example of a clinical document here: https://www.hl7.org/fhir/document-example-dischargesummary.xml.html
(Too big to be included in-line in this unit)

Just like any other bundle, you can either physically include the resource inside the bundle, or just include a reference to it.

Key Resources for Documents

Some resources are particularly useful for documents

The Composition Resource

The Composition resource is equivalent to the CDA header. The following table represents the mapping from a CDA header to a FHIR document. Note that this table provides a high level mapping only - the details within each property will be different between CDA & FHIR, and may require the use of extensions. A complete mapping from CDA to FHIR (and back) is provided by this Project: https://goo.gl/gbKnTk

CDA Header Property	FHIR equivalent	Comment
realmCode		
typeId		Not required for FHIR - simply identifies the CDA as a document
templateId	Class/type	
Id	versionIdentifier	The usual FHIR Version specific id
Code	type	
Title	title	
effectiveTime	created	
confidentialityCode	confidentiality	
languageCode		
setId	identifier	
versionNumber		Not required for FHIR - implicit in the versionIdentifier
copyTime		
Recordtarget	subject	
Author	author	
dataEnterer		
Informant		
Custodian	custodian	
informationRecipient		
legalAuthenticator	attester	
Authenticator	attester	
Participant		
inFullfillmentOf		
DocumentationOf	event	
relatedDocument	replaces	
Authorization		
componentOf	encounter	
component	section	

Note that for any CDA items that are not currently present in FHIR documents, you can use extensions to represent that data if required.

The List Resource

A general resource that summarizes other resources and provides a reference to them. For example, if you wanted to represent a list of medications, then you would include the medication resources in the bundle, and then have the document section point to the list resource, which in turn points to all the medications in that list (or section). You also have the option of having the list point to a resource that is held separately - i.e. not physically inside the bundle.
This is likely to be a common way of assembling documents as it supports the structured body of CDA level 2 & 3.

Document Layout Options

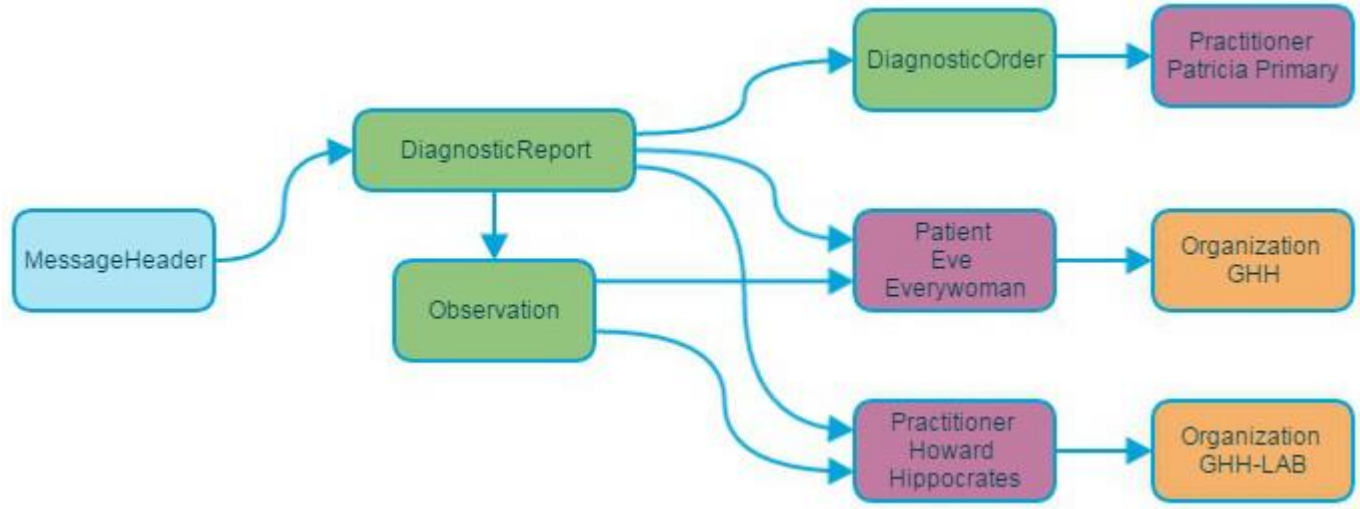
The following skeleton shows how the sections within a document could be laid out.

Composition
- Section
  - List Resource (code = problem)
    - Detail Resources (eg problem)
    - Detail Resources (eg problem)
- Section
  - List resource (code = medication)
    - Detail Resources (eg medication)
- Section
  - Detail resource (code=procedure)

FHIR Messages

FHIR messages are directly analogous to v2 messages. They can be considered to be a collection of resources sent as a result of some real-world event intended to accomplish a particular purpose, and there are a number of event codes that are defined for particular purposes.
As with FHIR documents, a FHIR message is essentially a bundle of resources with a first 'MessageHeader' resource containing the message specific information.

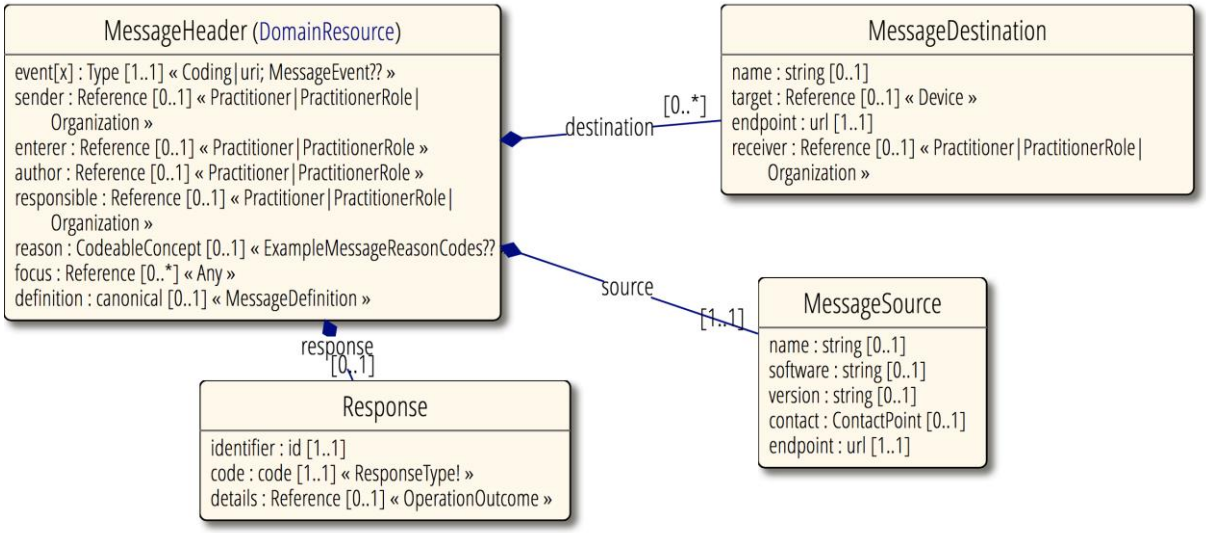
The Message Package (Bundle)



FHIR MessageHeader and other resources - Image Credit:
http://www.ringholm.com/docs/04350_mapping_HL7v2_FHIR.htm

The MessageHeader Resource

The following diagram shows the UML diagram for the message resource.



Services

The 'standard' REST based services support direct access to those resources, including different search options. This works well when the access requirements are straight forward, but there are times when more complex server processing is required.

In this scenario, more complex services (that could present a REST or a SOAP based interface) can be defined that still pass FHIR resources back & forth, but which define more complex server processing.

For example, when collecting data from a device, the device itself may emit data in a non-FHIR format, but the server may create and store FHIR resources - for example, the service may connect the patient to the observation resource based on the device id.

At the time of writing, the use of Services in FHIR has yet to be fully defined, but there is an interesting concept called ‘Operations’: you can define specific operations for each resource, with its own parameters. To know more about FHIR Operations, please read:
http://hl7.org/fhir/operationslist.html

Conformance Statement

As will be evident from the description above, while FHIR is simple to consume, it can be complex to provide access to FHIR resources - particularly for searching and transaction processing. Further, not all servers will support all resources - in fact, it is very likely that most servers will support only a subset of FHIR resources, and only a subset of the defined searches.

A server therefore has a mechanism to describe the resources, and the actions on those resources that it provides in a Conformance Statement

A conformance statement is actually a standard FHIR resource that is available at the 'conformance' endpoint. This endpoint has a special name: ‘**metadata**’. It returns a CapabilityStatement resource. This resource’s definition can be read here: https://www.hl7.org/fhir/capabilitystatement.html

For our own server, you can READ the ConformanceStatement issuing this operation:
GET http://fhir.hl7fundamentals.org/r4/metadata
This operation will a return a FHIR conformance resource that describes the capabilities of the server.
The ConformanceStatement resource describes:

- General information about the server and the software
- The REST resources that are supported
- The Messaging events supported
- The Document types supported

It is envisaged that they will be useful in a number of scenarios:

- Describing an existing implementation
- Describing what a solution can do
- Describing what it is desired for a solution to do

To describe what search facilities a server supports, the server uses the Profile resource. The profile facility in FHIR is described previously, and allows a system or implementation to take the base FHIR standard and describe how it should be used in a specific context - which includes Search capabilities.

Unit Summary and Conclusion

FHIR is the first HL7 specification fully created in and for the 21st century. HL7 V2.x, V3 and CDA will be with us for the next 5-10 years, but this new HL7 standard is the future, now. We hope this brief introduction brings you to know it a little more.
Please keep connected to HL7 FHIR to see how it grows, and if possible, participate!

Additional Reading Material

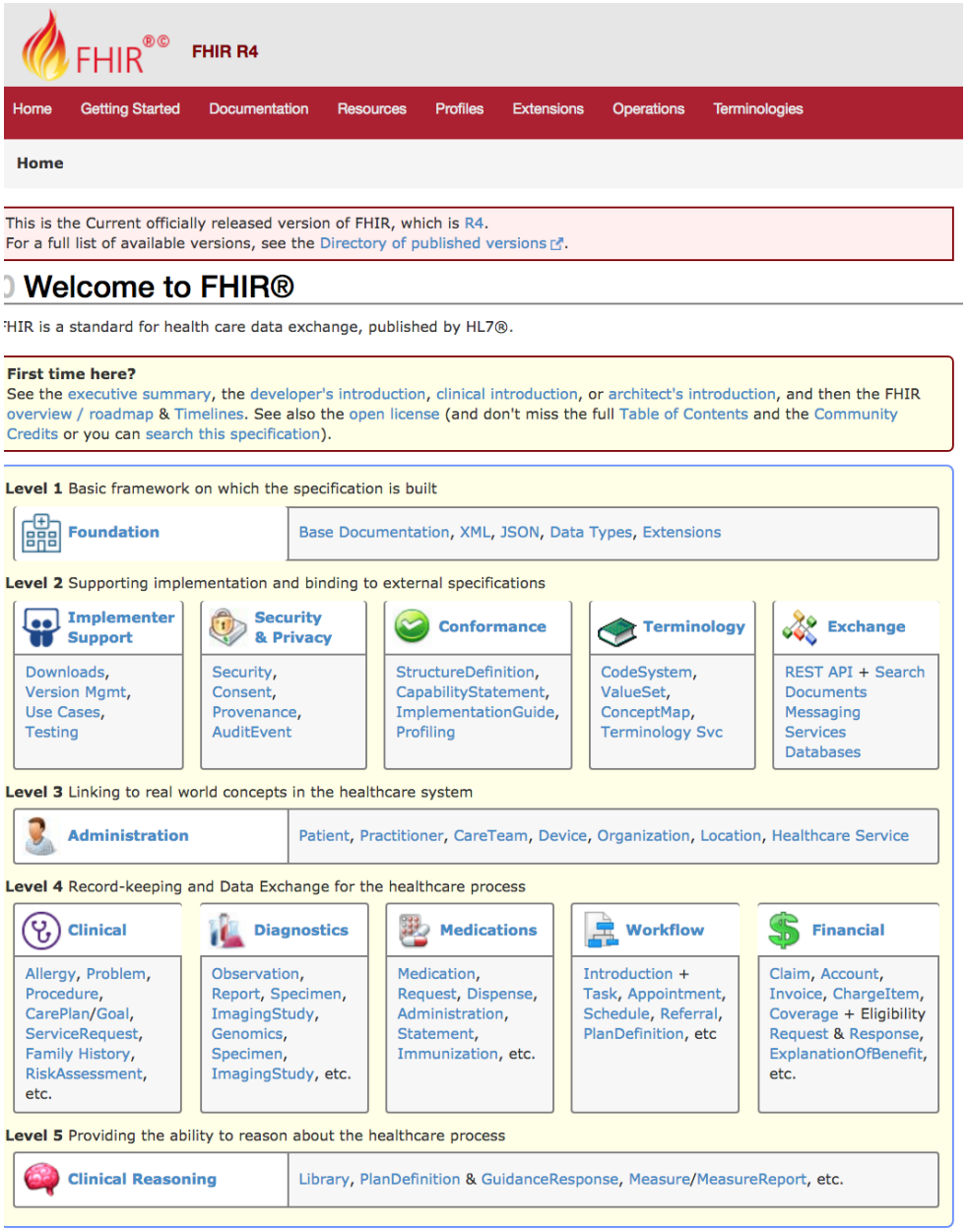
Information about FHIR

There are a number of places where you can get information about FHIR.

- The FHIR R4 specification itself is available on-line at www.hl7.org/FHIR. It is fully hyperlinked & very easy to follow. It is highly recommended that you have access to the specification as you are reading this module, as there are many references to it - particularly for some of the details of the more complex aspects of FHIR.
- The root HL7 wiki page for FHIR can be found at http://wiki.hl7.org/index.php?title=FHIR . The information here is more for those developing resources, but still very interesting. Some wiki information is more historical and may not reflect the most recent version of the specification.
- There are other resources on-line available, in particular some blogs by the FHIR core team. These can be found at:
  Grahame Grieve : http://www.healthintersections.com.au/
  Ewout Kramer : http://thefhirplace.com/
  David Hay: http://www.fhirblog.com
- As stated previously, the best resource for asking FHIR related questions and get acquainted with the FHIR community is https://chat.fhir.org also known as the 'FHIR zulip chat'.
- The FHIR team also uses the 'stack overflow' site (http://stackoverflow.com/questions/tagged/hl7-fhir) as a place to answer implementation-related questions - and therefore have both question and answer available for reference.

Navigating the FHIR Spec

The following image shows the front page of the FHIR specification as loaded from **http://hl7.org/fhir/**



This web page is the beginning of your FHIR journey. Enjoy it.