

```
"resourceType" : "Patient"
"text" : {
  "status" : "generated"
  "_status" : {
    "id" : "12344"
  },
  "div" : "<div
},
"identifier"
{
  "use"
  "ty
```

FHIR FUNDAMENTALS COURSE

FAST HEALTHCARE INTEROPERABILITY RESOURCES



FHIR COURSE

Introduction to FHIR - Resources

Reading Material

Course Overview

Module I: Introduction

Introduction to FHIR

Resources

Module II: Working with FHIR

RESTful FHIR

Searching with FHIR

Module III: Advanced FHIR

Transactions

Paradigms

Messaging

Documents and CDA R2

Operations

Module IV: FHIR Conformance

Conformance Resources

Extensions

Profiles

Implementation

Table of Contents

Why FHIR?	5
Scope	7
Governance & Methodology	8
Relationship with Other SDOs	8
License	9
Using FHIR - Example Architectures	9
Using FHIR - Sample Scenarios	10
Key Concepts of FHIR	11
Resources	11
Types of Resource in FHIR.....	13
FHIR on the Wire	14
1. Definition & Documentation in the Specification	15
2. Specification of Resource Content	15
3. General notes	17
4. Search Parameters.....	17
5. Terminology Bindings	17
6. Example Instances	18
7. Mappings	18
8. Schema + Schematron	18
Narrative.....	20
Core Content.....	20
Extensions	20
Extending a Core Element	22
Resource Metadata	23
Identity	23
Last Update.....	24
Datatypes	25
Code	25
Resource reference	26
Contained Resources	27
XML and JSON representations.....	27
Complex Types.....	28
Representation of the Resources	30

XML Schema and Schematron	34
JSON Representation of Resources.....	34
Comparison with XML.....	36
JSON Representation of Resources.....	37
Bundles	38
Unit Summary and Conclusion	39
Additional Reading Material	40

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.

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.
- Clinical Document Architecture (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 shallow 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 they are 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 data type 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, .NET 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 languages may be introduced in the future.
- A number of publicly available FHIR test servers that can be used by implementers to test their developments.
- 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.

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 - online 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 is new to HL7.

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 when needed, so it supports more of a distributed model.

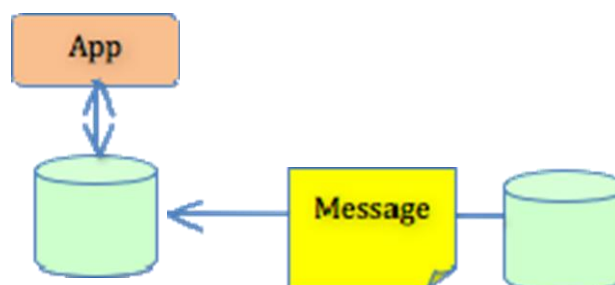


Figure 1: Information is replicated in both sending and receiver data stores using messages

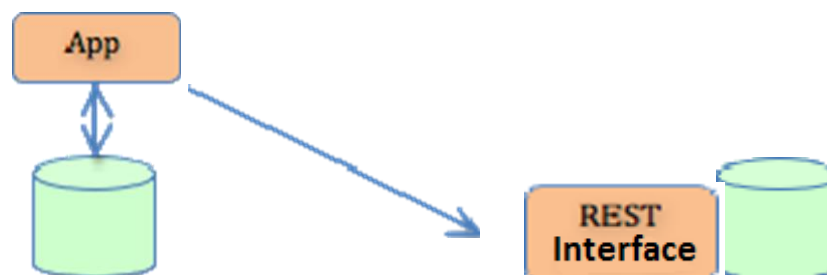


Figure 2: Application directly querying a remote system

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.

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 FHIR Governance Board

The FHIR Governance Board oversees FHIR development and has final say on what resources are defined. It also takes the lead in coordinating resources development with external groups and Standard Development Organizations.

More information at http://wiki.hl7.org/index.php?title=FHIR_Governance_Board

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 group that ‘owns’ the resource. For example, 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. In a few cases - generally due to bandwidth issues - the ‘owner’ is designated as the FHIR Core Team.

More information at http://wiki.hl7.org/index.php?title=FHIR_Management_Group

Relationship with Other SDOs

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

- IHE is cooperating on the development of resources to support RESTful implementation of their XDS and ATNA profiles which are currently modeled as the DocumentReference and SecurityEvent resources. There is a separate discussion on FHIR support of XDS later in this unit. More information at www.ihe.net.
- 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 <http://medical.nema.org>.
- W3C As the initial work has been in the REST paradigm, FHIR attempts to be as faithful as possible in the use of HTTP constructs (verbs, headers, response codes, mime types) and other standard constructs such as the Atom standard. The W3C is also assisting with the representation of FHIR in RDF and using other semantic web tools. More information at www.w3.org.

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 significant benefits in 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 cannot 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.

Using FHIR - Example Architectures

There are a number of ways that FHIR can be used, especially as FHIR-capable systems 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 bidirectionally 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 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.

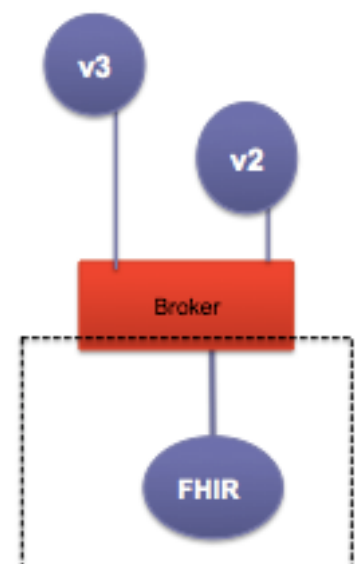


Figure 3: Message broker + FHIR engine

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. There will need to be an application of some sort performing the conversion (perhaps based on one of the reference implementations) - e.g. receive a request for a FHIR resource, query the back end system for the data, then convert to another FHIR resource and return.

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

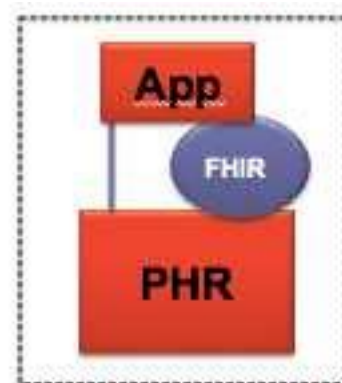


Figure 4: FHIR server with existing back end

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 the early systems (including both of the main test servers available on the net) have taken this approach - one using a ‘NOSQL’ data store, and the other an SQL database with a simple structure.

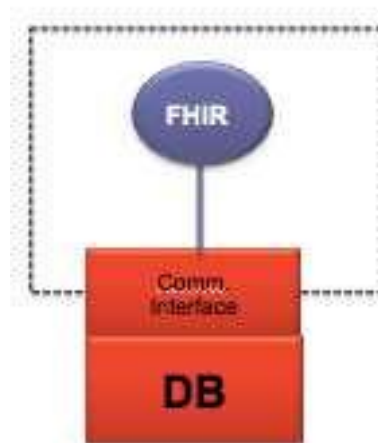


Figure 5: FHIR server and FHIR back end

Using FHIR - Sample Scenarios

Examples of when you might use a FHIR interface include:

- Online 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.

Key Concepts of FHIR

There are a number of FHIR key concepts, including:

- Resources
- Extensions
- Data types
- 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 or a CMET in the v3 world.

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 data type (like string or Codeable-Concept). In many resources, a particular element can be one of several different data types, though an element occurrence of a particular instance of a resource will be of only one of those data types.

For example, in the Observation resource shown below, the value can be a Quantity, a Codeable-Concept 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: <http://www.hl7.org/fhir/observation-examples.html>. 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 <span> (Details : {LOINC code '15074-8' = 'Glucose
  [Moles/volume] in Blood', given as 'Glucose
  [Moles/volume] in Blood'})</span> </p> <p> <b>subject</b> : <a> P. van de Heuvel</a> </p> <p> <b>
  effective</b> : 02/04/2013 9:30:10 AM --&gt; (ongoing)</p> <p> <b>issued</b> : 03/04/2013 3:30:10 PM</p> <p>
  <b>performer</b> : <a> A. Langeveld</a> </p> <p> <b>value</b> : 6.3 mmol/l<span> (Details: UCUM code
```

```

mmol/L = 'mmol/L')</span> </p> <p> <b> interpretation</b> : High <span> (Details :
{http://terminology.hl7.org/CodeSystem/v3-ObservationInterpretation code 'H'
= 'High', given as 'High'})</span> </p> <h3> ReferenceRanges</h3> <table> <tr> <td> -</td> <td> <b>
Low</b> </td> <td> <b> High</b> </td> </tr> <tr> <td> *</td> <td> 3.1 mmol/l<span> (Details: UCUM code
mmol/L = 'mmol/L')</span> </td> <td> 6.2 mmol/l<span> (Details: UCUM code mmol/L = 'mmol/L')</span> </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>
<interpretation>
<coding>
<system value="http://terminology.hl7.org/CodeSystem/v3-ObservationInterpretation"/>
<code value="H"/>
<display value="High"/>
</coding>
</interpretation>
<referenceRange>
<low>
<value value="3.1"/>
<unit value="mmol/l"/>
<system value="http://unitsofmeasure.org"/>
<code value="mmol/L"/>
</low>
<high>
<value value="6.2"/>
<unit value="mmol/l"/>
<system value="http://unitsofmeasure.org"/>
<code value="mmol/L"/>
</high>
</referenceRange>
</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'><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</p><p><b>interpretation</b>: { LOINC code '15074-8' = 'Glucose [Moles/volume] in Blood', given as 'Glucose [Moles/volume] in Blood' }</p><p><b>subject</b>: <a>P. van de Heuvel</a></p><p><b>effective</b>: 02/04/2013 9:30:10 AM --></p></div>"
  }
}

```

```

(ongoing)</p><p><b>issued</b>: 03/04/2013 3:30:10 PM</p><p><b>performer</b>: <a>A. Langeveld</a></p><p><b>value</b>: 6.3 mmol/l<span>
(Details: UCUM code mmol/L = 'mmol/L')</span></p><p><b>interpretation</b>: High <span>(Details : {http://terminology.hl7.org/CodeSystem/v3-
ObservationInterpretation code 'H' = 'High', given as 'High'})</span></p><h3>ReferenceRanges</h3><table><tr><td>-
</td><td><b>Low</b></td><td><b>High</b></td></tr><tr><td>*</td><td>3.1 mmol/l<span> (Details: UCUM code mmol/L =
'mmol/L')</span></td><td>6.2 mmol/l<span> (Details: UCUM code mmol/L = 'mmol/L')</span></td></tr></table></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"
},
"interpretation": [
{
"coding": [
{
"system": "http://terminology.hl7.org/CodeSystem/v3-ObservationInterpretation",
"code": "H",
"display": "High"
}
]
}
],
"referenceRange": [
{
"low": {
"value": 3.1,
"unit": "mmol/l",
"system": "http://unitsofmeasure.org",
"code": "mmol/L"
},
"high": {
"value": 6.2,
"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>

Categorized		Alphabetical	R2 Layout	By Maturity	Security Category
By Standards Status		By Committee			
Foundation	Conformance <ul style="list-style-type: none">CapabilityStatement NStructureDefinition NImplementationGuide 1SearchParameter 3MessageDefinition 1OperationDefinition NCompartmentDefinition 1StructureMap 2GraphDefinition 1ExampleScenario 0	Terminology <ul style="list-style-type: none">CodeSystem NValueSet NConceptMap 3NamingSystem 1TerminologyCapabilities 0	Security <ul style="list-style-type: none">Provenance 3AuditEvent 3Consent 2	Documents <ul style="list-style-type: none">Composition 2DocumentManifest 2DocumentReference 3CatalogEntry 0	Other <ul style="list-style-type: none">Basic 1Binary NBundle NLinkage 0MessageHeader 4OperationOutcome NParameters NSubscription 3
	Individuals <ul style="list-style-type: none">Patient NPractitioner 3PractitionerRole 2RelatedPerson 2Person 2Group 1	Entities #1 <ul style="list-style-type: none">Organization 3OrganizationAffiliation 0HealthcareService 2Endpoint 2Location 3	Entities #2 <ul style="list-style-type: none">Substance 2BiologicallyDerivedProduct 0Device 0DeviceMetric 1	Workflow <ul style="list-style-type: none">Task 2Appointment 3AppointmentResponse 3Schedule 3Slot 3VerificationResult 0	Management <ul style="list-style-type: none">Encounter 2EpisodeOfCare 2Flag 1List 1Library 2
	Summary <ul style="list-style-type: none">AllergyIntolerance 3AdverseEvent 0Condition (Problem) 3Procedure 3FamilyMemberHistory 2ClinicalImpression 0DetectedIssue 1	Diagnostics <ul style="list-style-type: none">Observation NMedia 1DiagnosticReport 3Specimen 2BodyStructure 1ImagingStudy 3QuestionnaireResponse 3MolecularSequence 1	Medications <ul style="list-style-type: none">MedicationRequest 3MedicationAdministration 2MedicationDispense 2MedicationStatement 3Medication 3MedicationKnowledge 0Immunization 3ImmunizationEvaluation 0ImmunizationRecommendation 1	Care Provision <ul style="list-style-type: none">CarePlan 2CareTeam 2Goal 2ServiceRequest 2NutritionOrder 2VisionPrescription 2RiskAssessment 1RequestGroup 2	Request & Response <ul style="list-style-type: none">Communication 2CommunicationRequest 2DeviceRequest 0DeviceUseStatement 0GuidanceResponse 2SupplyRequest 1SupplyDelivery 1
	Support <ul style="list-style-type: none">Coverage 2CoverageEligibilityRequest 2CoverageEligibilityResponse 2EnrollmentRequest 0EnrollmentResponse 0	Billing <ul style="list-style-type: none">Claim 2ClaimResponse 2Invoice 0	Payment <ul style="list-style-type: none">PaymentNotice 2PaymentReconciliation 2	General <ul style="list-style-type: none">Account 2ChargeItem 0ChargeItemDefinition 0Contract 1ExplanationOfBenefit 2InsurancePlan 0	
	Specialized	Public Health & Research <ul style="list-style-type: none">ResearchStudy 0ResearchSubject 0	Definitional Artifacts <ul style="list-style-type: none">ActivityDefinition 2DeviceDefinition 0EventDefinition 0ObservationDefinition 0PlanDefinition 2Questionnaire 3SpecimenDefinition 0	Evidence-Based Medicine <ul style="list-style-type: none">ResearchDefinition 0ResearchElementDefinition 0Evidence 0EvidenceVariable 0EffectEvidenceSynthesis 0RiskEvidenceSynthesis 0	Quality Reporting & Testing <ul style="list-style-type: none">Measure 2MeasureReport 2TestScript 2TestReport 0

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. In the future, representation as RDF will also be possible, though RDF usage will generally be appropriate for secondary analysis rather than primary exchange.

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.

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

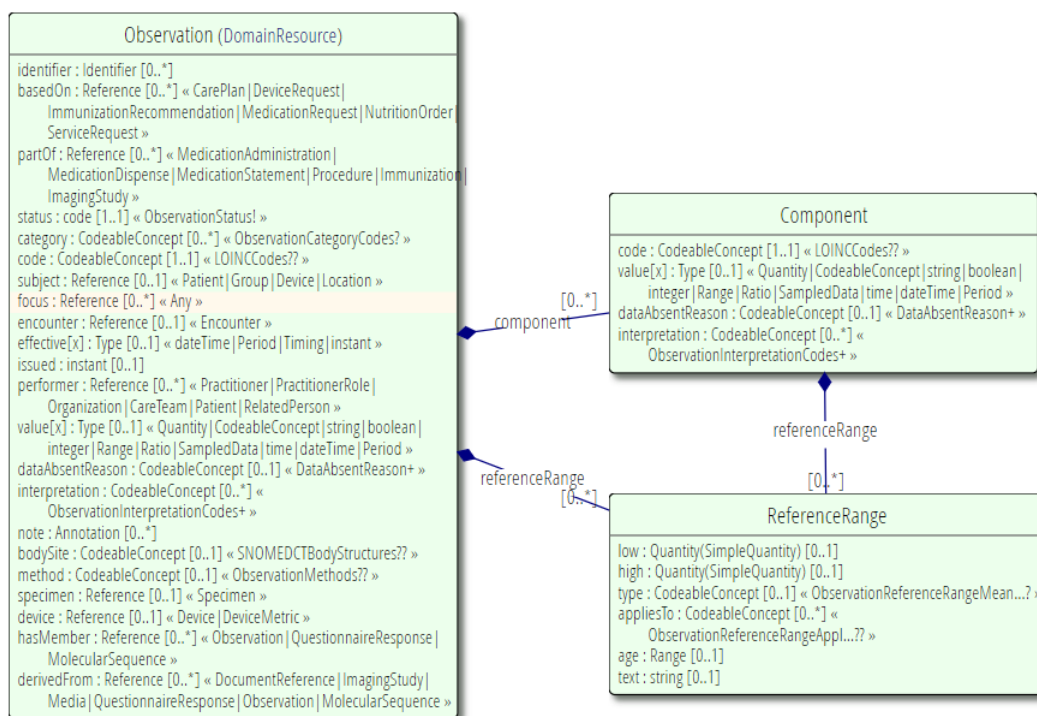


Figure 8: A UML diagram of the resource

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 --></basedOn>
  <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 made --></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 1: 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 2: 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>

Key Parts of a Resource

A resource has four main parts:

1. The identity and metadata.
2. The narrative or text section. This allows a human to safely view a resource.
3. Any number of extensions. Extensions are described below, but allow an implementer to add an element that is 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.
4. Structured, defined data, otherwise known as the 'core dataset'. 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.
5. (There is actually a 5th 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:

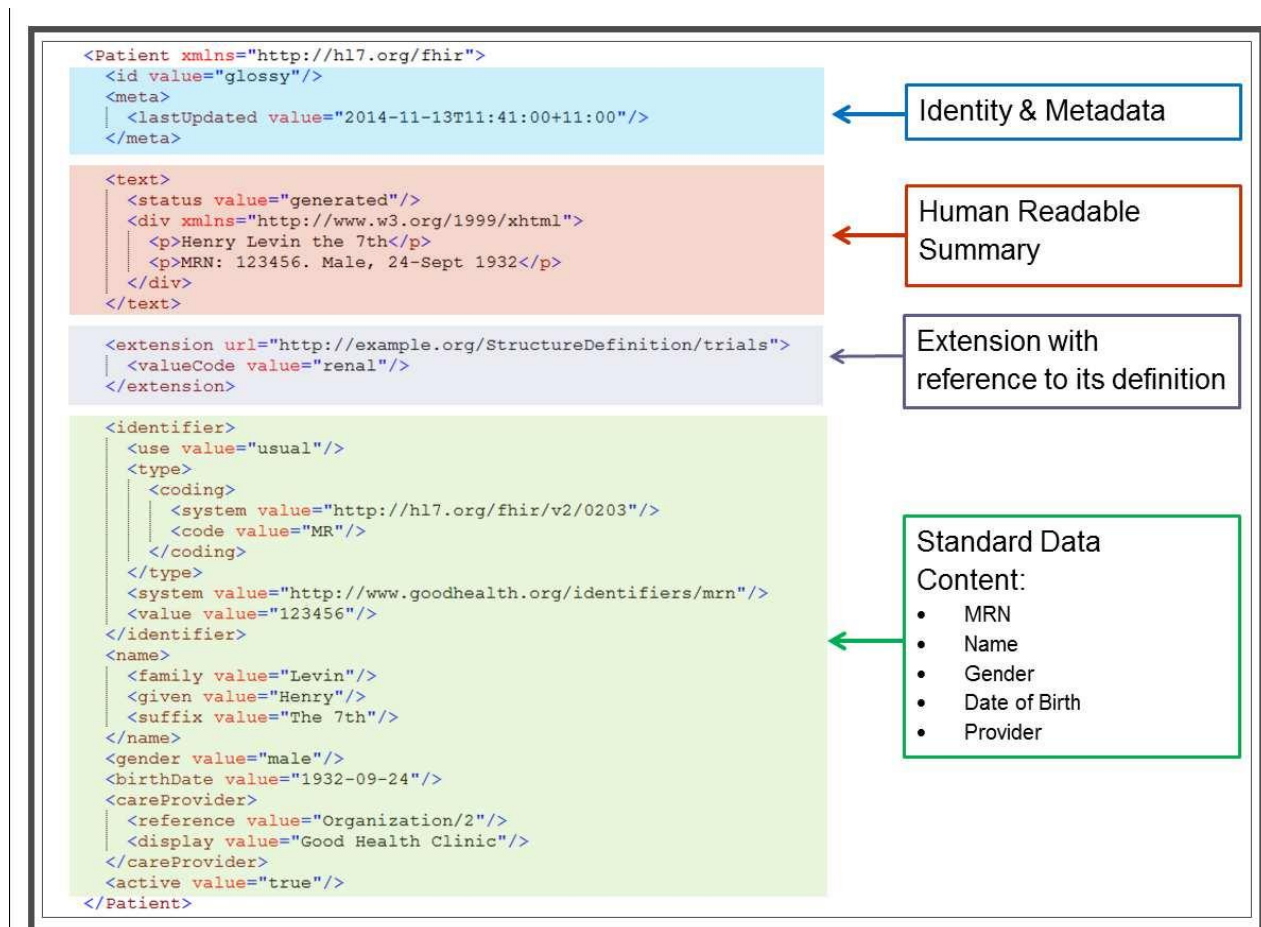


Figure 11: An example of a FHIR resource

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

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 – puts it 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!

One of the core principles with FHIR is that the core specification only includes those structures that will actually be used by the vast majority of implementers. Because healthcare is such a diverse industry, there are numerous data elements that are only used in a limited number of environments. Past experience has shown that attempting to document all of these edge-case elements as part of the core specification quickly overwhelms the implementer community. It becomes hard to find the data elements a given implementer really cares about, and there is no way to differentiate which elements are 'common' vs. unusual or edge-case. Limiting the specification to only common elements makes it easier to read and use for everyone. Non-core elements are handled using FHIR's extensibility mechanism.

A few things to keep in mind:

- There is no formal statistical analysis involved. A rough assessment is made by the work group responsible for the resource based on their knowledge of the business area. In most cases, the answer is obvious. Either very few systems use the element or almost all systems use it. In cases where the answer is unclear, the work groups are instructed to err on the side of exclusion. The behaviour of DSTU implementations will be monitored to make a final

determination.

- The assessment is based on the full scope of the resource, taking into account implementation in all countries, all implementation environments, all types of care, etc. It is possible for an element to appear in most (or even all) systems in a particular environment, but be rare in most environments.
- The assessment is based on what systems support, not necessarily what gets populated. For example, most healthcare systems support tracking a patient's date of death, even though it might not be populated for most patients within those systems.
- In rare cases, exceptions to the 80% rule may be made with the approval of the FHIR Management Group to support key implementation use cases.

Taking this approach has the advantage that the resources themselves are kept to a manageable size, and much easier to implement than 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 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 an 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 two 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 data type of the element in exactly the same way as it does for core

In order to avoid the issue that arose with V2.x Z segments where each jurisdiction defined its own segments without regard for what others had done, the FHIR team intends to establish a

hierarchy of profile repositories that implementers 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.
- Realm (or country) registries that specify extensions specific to that realm (such as the iwi/hapu example above).
- Other registries for anything else.

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 who 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 its primary meaning, there is a specific element - *modifierExtension* - that can be added 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 that has *modifierExtension* set to true, then it should either alert the user or not process the resource.

For further discussion on extensions, go to:

www.hl7.org/implement/standards/fhir/extensibility.html

How Do I Design My System For extensions?

Extensions are a core aspect of FHIR. Most instances will have at least a few extensions, because most instances will need to communicate requirements that are specific to a particular region, discipline or organizational process. Therefore, systems must be prepared to deal with extensions, including extensions they may not recognize.

The first question when receiving an unrecognized extension is whether to retain that piece of data. If possible, data should be retained as it will likely be useful downstream. However, persisting ‘unrecognized’ data means that the persistence layer must have some mechanism of capturing ‘extra’ information, be that in a name/value pair mechanism, a blob mechanism, via specially tagged ‘notes’ or some similar means. Not all systems will have persistence layers that can accommodate this. Such systems will have no choice but to drop unrecognized extension content.

If unrecognized extensions are persisted, care should be taken to remove extensions that may have been invalidated due to the update of related data. For example, if there is an unrecognized extension on Patient.birthDate and the birthDate element is changed, then it may be necessary to remove the unrecognized extension as its value may no longer be valid. On the other hand, a change to the patient’s name would have no impact on the validity of the birth date extension.

An additional consideration with unrecognized extensions is whether to display them. Because the definitions can be looked up along with display names, permitted values, etc., it is possible for a system to adjust its user interface on the fly to accommodate extensions that the system might not actually understand. If this capability is supported, it is probably wise to give users the ability to configure which extensions are exposed. Many extensions will contain data that may not be relevant to most users, so suppressing the information may be needed to avoid having the user interface overwhelmed with unneeded information.

Resource Metadata

There are a number of aspects to a resource that are more about the resource itself than the actual contents of the resource. 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 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, 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 Update

This is the `dateTime` 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 `dateTime`, which might be useful if you want to synchronize resources between servers, or have a feed that informs you when resources change.

This is done using the history of a resource and can be at different levels of granularity. The following examples are using the REST interface (the other paradigms such as messaging queries have yet to be defined).

Examples:

Return all versions of the patient whose id is 2 since Dec. 1, 2012:

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

Return all versions of all patients since Dec. 1, 2012:

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

Return all versions of all resources on this server modified since Dec. 1, 2012:

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

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.

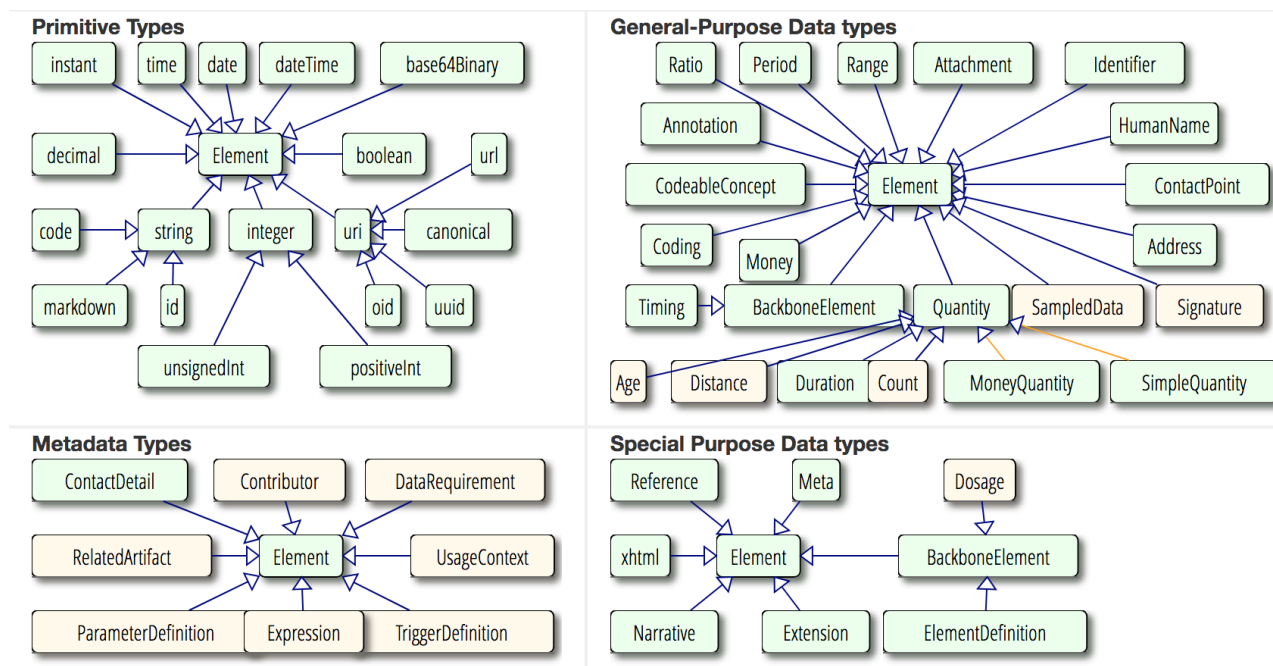
Datatypes

Each element within a resource is of a particular data type (this is the same for all HL7 standards, of course).

The FHIR specification defines a set of datatypes that are used for the resource elements. There are two categories of datatypes: simple/primitive types, which are single elements, and complex types, which are reusable clusters of elements.

The FHIR data types are a simplified version of the V3 data types, and are also based on the W3C schema. 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."*

XML and JSON representations

All elements using these primitive types have some combination of value as described above, an internal identity (e.g. xml:id), and extensions. The value is represented in XML as an attribute named 'value':

```
<count value="2"/>
```

and as the value of the property in JSON:

```
"count": 2
```

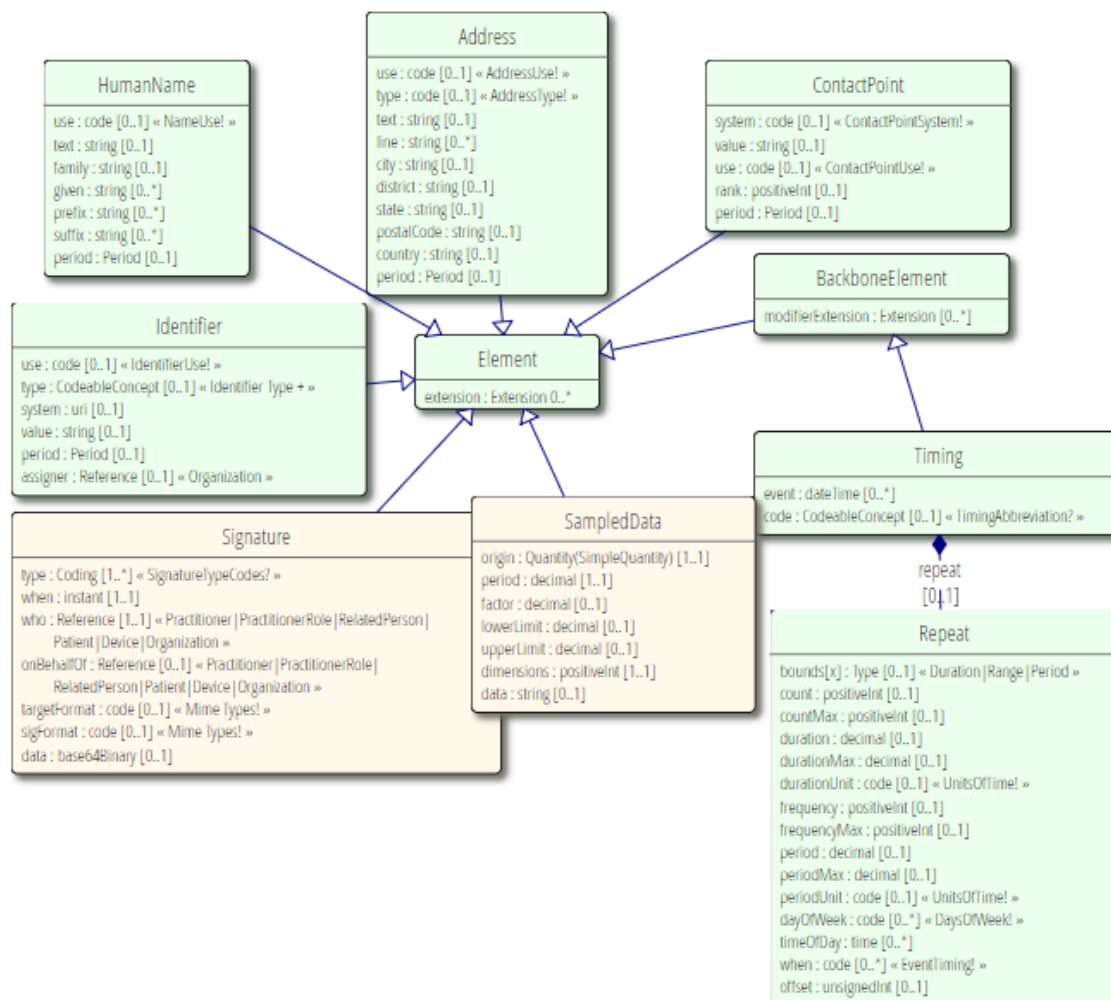
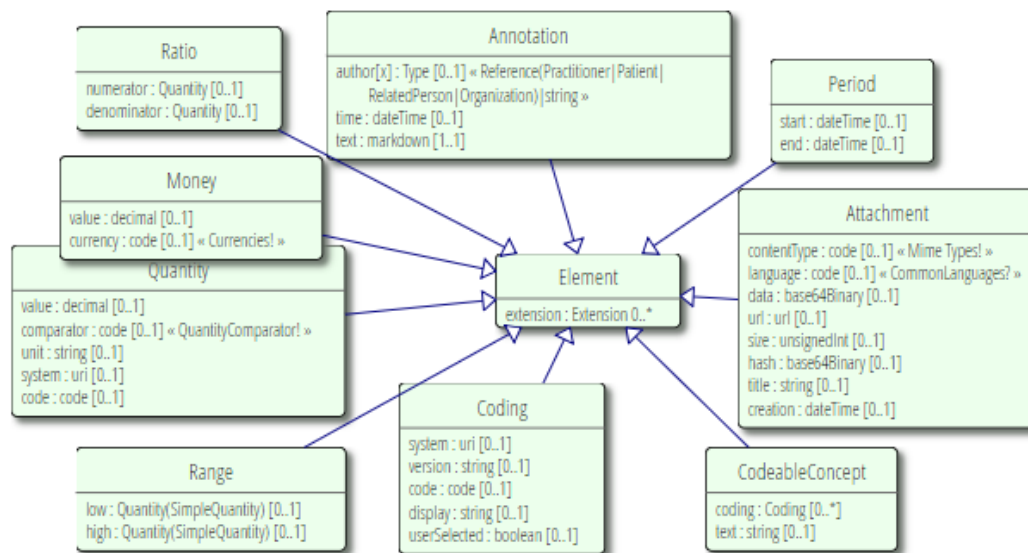
For additional details, including how the internal identity and extensions are represented, see the [XML](#) and [JSON](#) formats. When the value is missing, it is not represented in the instance; the XML value attribute or the JSON property is not represented at all. This means that in XML, attributes are never present with a length of 0 (value=""), and properties are never a 0 length string or null in JSON ('name': "" is not valid). (Note: there is one specific use of the null in the JSON representation.)

According to XML schema, leading and trailing whitespace in the value attribute is ignored for the

types Boolean, integer, decimal, base64Binary, instant, uri, date, dateTime, oid, and uri. Note that this means that the schema-aware XML libraries give different attribute values to non-schema-aware libraries when reading the XML instances. For this reason, the value attribute for these types SHOULD NOT have leading and trailing spaces. String values should only have leading and trailing spaces if they are part of the content of the value. In JSON, whitespace in string values is always significant. Primitive types other than string SHALL NOT have leading or trailing whitespace.

Complex Types

These types are represented as XML elements with child elements with the name of the defined elements of the type. The name of the element is defined where the type is used. Any of the XML elements may have an id attribute. In JSON, the datatype is represented by an object with properties named the same as the XML elements. The JSON representation is almost exactly the same, so only the first example has an additional JSON representation.



Complex data types may be "profiled". A Structure Definition or type "constraint" makes a set of rules about which elements SHALL have values and what the possible values are.

The specification describes these data types, <http://hl7.org/fhir/datatypes.html#complex>.

Representation of the Resources

In actual exchange, resources can be represented in the following formats: XML and JSON. Other representations are allowed, but are not described by the specification.

Resource Definition

The resources are described in several different ways:

- A hierarchical table that presents a logical view of the content.
- A UML diagram that summarizes the content graphically.
- A pseudo-XML syntax that provides a visual sense of what the end resource instances will look like in XML.
- A pseudo-JSON syntax that provides a visual sense of what the end resource instances will look like in JSON.

In addition to this descriptive syntax, other definitional forms are available, including the W3C schema and Schematron, and the StructureDefinition syntax defined internally.








Logical Table

The Logical View shows the resources as a tree structure with the following columns:


















Column	Content
Name	The name of the element in the resource (manifests as XML element name, or JSON property name. Some names end with [x] - the meaning of this is discussed below. In addition, this column contains an icon that denotes the underlying type of the content. The icons are described below.
Flags	A set of information about the element that impacts how implementers handle them. The flags are described below.
Card	The lower and upper bounds on how many times this element is allowed to appear in the resource (Card. is short for "cardinality").

Type	The type of the element (hyperlinked to the definition of the type).
Description & Constraints	A description of the element, and details about constraints that are applied to it. Particularly, for coded elements, information about which codes can be used.

Here is an example:

Name	Flags	Card.	Type	Description & Constraints
 Resource Name			Base Type	Definition
 nameA	Σ	1..1	type	description of content
 nameB[x]	?! Σ	0..1		description SHALL at least have a value
 nameBType1		0..1	type1	
 nameBType2 I		0..1	type2	
 nameC		1..*	Element	Definition
 nameD		1..1	type	Relevant Records

Key to Type Icons and Flags

- : The base element for a resource (see [Resources](#))
- : An element that is part of the resource and has elements within it defined in the same resource or profile
- : An element which can have one of a several different types (see below)
- : A data type which describes an element that has a **value** attribute/property
- : A data type which describes an element that has other elements
- : A element that contains a reference to another resource (see [references](#))
- : This element has the same content as another element defined within this resource or profile
- : Introduction of a set of slices (see [Slicing](#))
- : An extension (see [Extensibility](#))
- : A complex extension - one with nested extensions (see [Extensibility](#))
- : An extension that has a value and no nested extensions (see [Extensibility](#))
- : The root of a logical profile
- : This element is a modifying element - see [Modifier Elements](#)
- : This element is an element that must be supported - see [Must-Support Elements](#)
- : This element is an element that is part of the summary set - see [Summary Searches](#)
- : This element defines or is affected by constraints - see [Constraints](#)
- : This element cannot have extensions (some infrastructural elements only)

XML Representation of Resources


The XML representation for a resource is described using this format:

```
<name xmlns="http://hl7.org/fhir" (attrA="value")>

  <nameA><!-- 1..1 type description of content --><nameA>
  <nameB[x]><!-- 0..1 type1|type1 description --></nameB>
  <nameC> <!-- 1..* -->
    <nameD><!-- 1..1 type>Relevant elements --></nameD>
  </nameC>
</name>
```

Using this format:

- To build a valid XML instance of a resource, simply replace the contents of the elements and attributes with valid content as described by the cardinality, type rules and content description found in the comment in each element.

- Resource and Element names are case-sensitive (though duplicates that differ only in case are never defined).
- Elements must always appear in the order documented.
- When an element is allowed to repeat, the elements are ordered, and implementations must preserve order (note: the meaning of the order may not be known).
- A few properties are represented as attributes: primitive values in the **value** attribute, extension URLs in the **url** attribute on an extension, and the **id** property.
- Any of the XML elements may have an id attribute to serve as the target of an internal reference. The id attribute is not shown in this format.
- FHIR elements are always in the namespace <http://hl7.org/fhir>. This is usually specified as the default namespace on the root element. The only other namespace that occurs in FHIR resources is the XHTML namespace (XHTML is found in most resources).
- Infrastructural elements that are common to all resources are not shown in the XML representation. These must appear prior to any other defined child elements in the following order:
 - First, the elements from the baseresource, in order.
 - Second, the elements from the domainresource, in order.
- FHIR elements are never empty. If an element is present in the resource, it SHALL have a value attribute, child elements as defined for its type, or one or more extensions.
- Attributes can never be empty. Either they are absent, or they are present with at least one character of non-whitespace content.
- Implementers SHOULD trim leading and trailing whitespace before writing and SHOULD trim leading and trailing whitespace when reading attribute values.
- The lock icon () denotes that an element defines or is affected by additional rules that control its presence and/or content.
- XML comments, processing instructions and formatting are not part of the contents of a resource.
- XML resources SHALL be exchanged using UTF-8 encoding. Specifying the character encoding using a `<?xml encoding="UTF-8" ?>` processing instruction is optional but recommended.
- Other processing instructions SHOULD not be included, and SHALL NOT be required in order to properly understand and/or present the data or narrative of the resource. Applications MAY preserve processing instructions when handling resources, but are not required to do so.
- The MIME-type for this format is **application/xml+fhir**.

XML Schema and Schematron

This specification provides schema definitions for all of the content models it describes.

The base schema is called "fhir-base.xsd" and defines all of the data types and base infrastructure types. In addition, there is a schema for each resource and a common schema fhir-all.xsd that includes all the resource schemas. For schema processors that do not like circular includes, there is a single schema that contains everything.

In addition to the W3C schema files, this specification also provides Schematron files that enforce the various constraints defined for the data types and resources. These are packaged as files for each resource.

XML that is exchanged SHALL be valid against the W3C schema and Schematron, though being valid against the schema and Schematron is not sufficient to be a conformant instance: this specification makes several rules that cannot be checked by either mechanism. Operational systems may choose to use schema tools to check validation, but are not required to do so. Exchanged content SHALL NOT specify the schema or even contain the schema instance namespace in the resource itself.

Given the way [extensions](#) work, applications reading XML resources will never encounter unknown elements. However, once an application starts trading with other applications that conform to later versions of this specification, unknown XML elements may be encountered. Applications MAY choose to ignore unknown elements in order to foster forward compatibility in this regard, but may also choose not to - which would be the normal behaviour for schema-generated applications. Applications declare their behavior with regard to unknown elements using `conformance.acceptUnknown`.

JSON Representation of Resources

The JSON representation for a resource is described using this format:

```
{
  "resourceType" : "Element",
  // from Source: element #1
  "property1" : "<[primitive]>", // short description
  "property2" : { [Data Type] }, // short description
  "property3" : { // Short Description
    "propertyA" : { CodeableConcept }, // Short Description (Example)
  },
  "property4" : [{ // Short Description
    "propertyB" : { Reference(ResourceType) } // R! Short Description
  }]
}
```

}

Using this format:

To build a valid JSON instance of a resource, replace the contents of the property values with valid content as described by the type rules and content description found in the property value for each element.

In this example:

property1 has a primitive data type; the value of the property will be as described for the stated type.

property2 has a complex data type; the value of the property is an object that has the content as described for the stated type.

property3 is an object property that contains additional properties (e.g. property A; the allowable properties are listed (but also include extensions as appropriate).

property4 is an array property containing items that are objects themselves. The items may have any of the types already encountered in points 1-3.

propertyA has a binding to a value set - the Short description is a link to the value set. In addition, the binding strength is shown.

propertyB is a reference to a particular kind of resource.

- Property names are case-sensitive (though duplicates that differ only in case are never defined).
- Properties can appear in any order.
- XHTML is represented as an escaped string.
- Objects are never empty. If an element is present in the resource, it SHALL have properties as defined for its type, or 1 or more extensions.
- String property values can never be empty. Either the property is absent, or it is present with at least one character of content.
- The **R!** denotes that an element is mandatory - it must be present (or in an array, at least one item must be present).
- In this format, `//` is used for comments but these can't be in the JSON instances.
- The character encoding is always UTF-8.
- The MIME-type for this format is **application/json+fhir**.

Given the way extensions work, applications reading JSON resources will never encounter unknown properties. However, once an application starts trading with other applications that conform to later versions of this specification, unknown properties may be encountered. Applications MAY choose to ignore unknown properties in order to foster forwards compatibility in this regard, but may also choose not to. Applications declare their behaviour with regard to unknown elements using `Conformance.acceptUnknown`.

Comparison with XML

The JSON format is similar to the XML format. The names for the JSON object members are the same as the names of the elements and attributes in XML, including elements that may repeat.

Property names are case sensitive

Just as in XML, JSON objects and arrays are never empty, and properties never have null values (except for a special case documented). Omit a property if it is empty. JSON whitespace is not part of the contents of a resource. Applications MAY preserve the whitespace when handling resources, but are not required to do so. Note that digital signatures may depend on the whitespace

There are differences from XML:

- There are no namespaces in the JSON representation.
- The type of the resource is represented differently in JSON - instead of being the name of the base object (there is none in JSON), it is carried as the property `resourceType`.
- The order of properties of an object is not significant in the JSON representation, though order within an array SHALL be maintained.
- JSON does not have a notion of attributes versus elements, so attributes (e.g. `id`, `value`) are handled differently (see below).
- JSON has the array notation, which is used to represent repeating elements. Note that arrays are used when the item might repeat, even if it does not repeat in a specific instance.
- The XHTML `<div>` element in the Narrative data type is represented as a single escaped string of XHTML. This is to avoid problems in JSON with mixed content, etc. The XHTML SHALL still conform to the rules described for the Narrative
- There is no inherent support in JSON for a comment syntax. As a convention, content that would be comments in an XML representation is represented in a property with the name `fhir_comments`, which is an array of strings and can appear on any JSON object. This is heavily used in example instances, e.g. in this specification, but not usually used in production systems (and production systems may choose to reject resources with comments in them).
- The JSON format for the resources follows the standard XML format closely to make inter-conversion easy, and so that XPath queries can easily be mapped to query the JSON structures. However, the differences - particularly the repeating element one, which cannot be avoided - mean that generic XML --> JSON converters are not able to perform correctly. The reference platforms provide XML <--> JSON conversion functionality that accommodates these FHIR-specific characteristics.

JSON Representation for Repeating Elements

An element that has a maximum cardinality of >1 (e.g. `x..*` in the definitions) may occur more than once in the instance. In XML, this is simply done by repeating the XML element multiple times. In JSON, this is done by using an array type. Note that:

- The name of the array is singular - the same as the XML element.
- An item that may repeat is represented as an array even in the case that it does not repeat so that the process of parsing the resource is the same either way.

```
<coding>
  <system value="http://snomed.info/sct"/>
  <code value="104934005"/>
</coding>
<coding>
  <system value="http://loinc.org"/>
  <code value="2947-0"/>
</coding>
```

is represented in JSON like this:

```
"coding": [
  {
    "system" : "http://snomed.info/sct",
    "code" : "104934005"
  },
  {
    "system" : "http://loinc.org",
    "code" : "2947-0"
  }
]
```

JSON Representation of Resources

A resource is a JSON object with a property `resourceType` which informs the parser which resource type this is:

```
{
  "resourceType" : "Patient",
  "text" : {
    "status" : "generated" ,
    "div" : "<div xmlns=\"http://www.w3.org/1999/xhtml\"><p>...</p></div>"
  }
}
etc...
```

}

Note that parsers cannot assume that the `resourceType` property will come first. This is a problem for several JSON -> Object serializers that assume that the `resourceType` property does come first, including `json.net`. However, some JSON generators do not give the authoring application control of the order of the property values, and so these implementations cannot inter-operate with implementations that make assumptions about order. Given that JSON says that the property values are an unordered map of name/value pairs, this specification cannot require that properties come in any particular order, though implementers may choose to fix the property order if they are able (and the reference platforms provided with this specification do so).

Bundles

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

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

A bundle has some header information, and then any number of resources.

The header contains:

- The title of the feed.
- The time (as an instant) when it was created.
- An id for the bundle. This is used if the bundle is saved.
- A number of link elements. These are used to describe the application that created the bundle, and URLs that can be used for paging large bundles (if the server supports paging)

Each entry contains:

- A title. This is required by atom, and can be used to give a summary of the entry.
- An Id. This is an absolute URI that points to the `logicalId` of the resource in the entry element (even if the entry in the element is not the most recent version).
- The time (as an instant) when the resource in the entry was last updated.
- (optionally) A link element that does point to the version in the bundle - it is a version-specific reference in the format :

`[server]/[resourceType]/[logicalId]/_history/[versionId]`

We will explore this in greater depth in the coming weeks.

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 has helped you get to know it a little better.

Additional Reading Material


Information about FHIR

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

- The specification itself is available online at www.hl7.org/FHIR. It is fully hyperlinked and very easy to follow. It is highly recommended that you have access to the specification as you are reading this module, since 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.
- The core FHIR team 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. You can use the HL7 Help Desk if you are a member.

Navigating the FHIR Spec

The following image shows the front page of the FHIR specification (as of March 2019) as loaded from <https://www.hl7.org/fhir/index.html>


FHIR R4

[Home](#)
[Getting Started](#)
[Documentation](#)
[Resources](#)
[Profiles](#)
[Extensions](#)
[Operations](#)
[Terminologies](#)

Home


This is the Current officially released version of FHIR, which is R4.
 For a full list of available versions, see the [Directory of published versions](#).

Welcome to FHIR®






FHIR is a standard for health care data exchange, published by HL7®.

First time here?
 See the [executive summary](#), the [developer's introduction](#), [clinical introduction](#), or [architect's introduction](#), and then the [FHIR overview / roadmap & Timelines](#). See also the [open license](#) (and don't miss the full [Table of Contents](#) and the [Community Credits](#) or you can [search this specification](#)).


Level 1 Basic framework on which the specification is built

 Foundation	Base Documentation, XML, JSON, Data Types, Extensions
---	---






Level 2 Supporting implementation and binding to external specifications

 Implementer Support Downloads, Version Mgmt, Use Cases, Testing	 Security & Privacy Security, Consent, Provenance, AuditEvent	 Conformance StructureDefinition, CapabilityStatement, ImplementationGuide, Profiling	 Terminology CodeSystem, ValueSet, ConceptMap, Terminology Svc	 Exchange REST API + Search Documents, Messaging Services, Databases
---	--	--	---	---


Level 3 Linking to real world concepts in the healthcare system

 Administration	Patient, Practitioner, CareTeam, Device, Organization, Location, Healthcare Service
---	---

Level 4 Record-keeping and Data Exchange for the healthcare process

 Clinical Allergy, Problem, Procedure, CarePlan/Goal, ServiceRequest, Family History, RiskAssessment, etc.	 Diagnostics Observation, Report, Specimen, ImagingStudy, Genomics, Specimen, ImagingStudy, etc.	 Medications Medication, Request, Dispense, Administration, Statement, Immunization, etc.	 Workflow Introduction + Task, Appointment, Schedule, Referral, PlanDefinition, etc	 Financial Claim, Account, Invoice, ChargeItem, Coverage + Eligibility Request & Response, ExplanationOfBenefit, etc.
---	---	--	--	--

Level 5 Providing the ability to reason about the healthcare process

 Clinical Reasoning	Library, PlanDefinition & GuidanceResponse, Measure/MeasureReport, etc.
---	---

