## FHIR Extensions (Patient)

Date: 10/March/2017

Author: Kevin Mayfield

Ref: <http://www.interopen.org/candidate-profiles/care-connect/CareConnect-Patient-1.html>

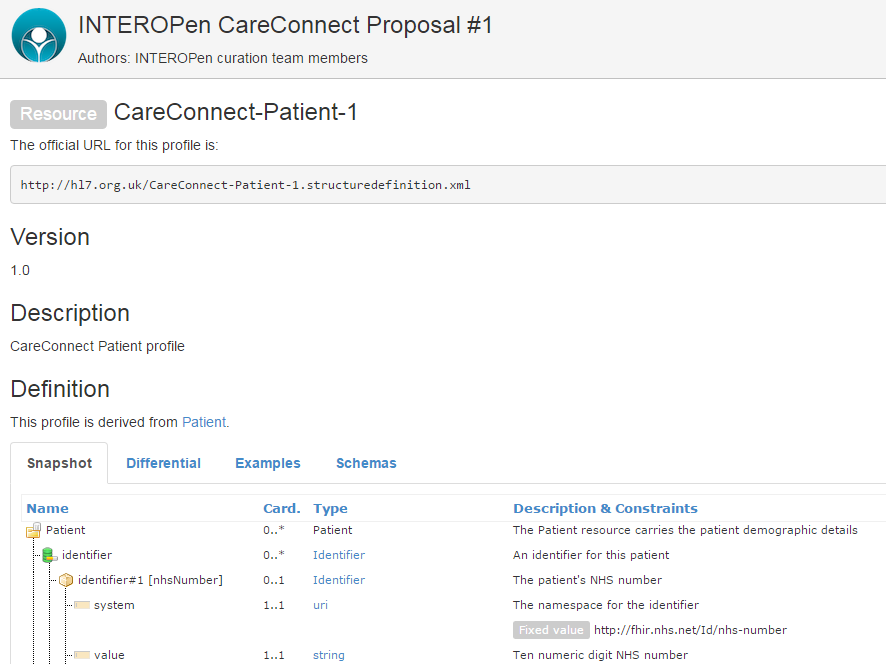
FHIR is designed around common functionality of systems; it is not designed to implement system specific or regional requirements.

|  |
| --- |
| FHIR only includes an element if 80% of systems implement it |

That does not mean FHIR only supports 80% of the requirements, FHIR supports 100% of the functionality people use by using extensions.

Some extensions can be found in the core FHIR specification but normally they are defined in regional/system profiles such as the CareConnect(Interopen) Patient profile

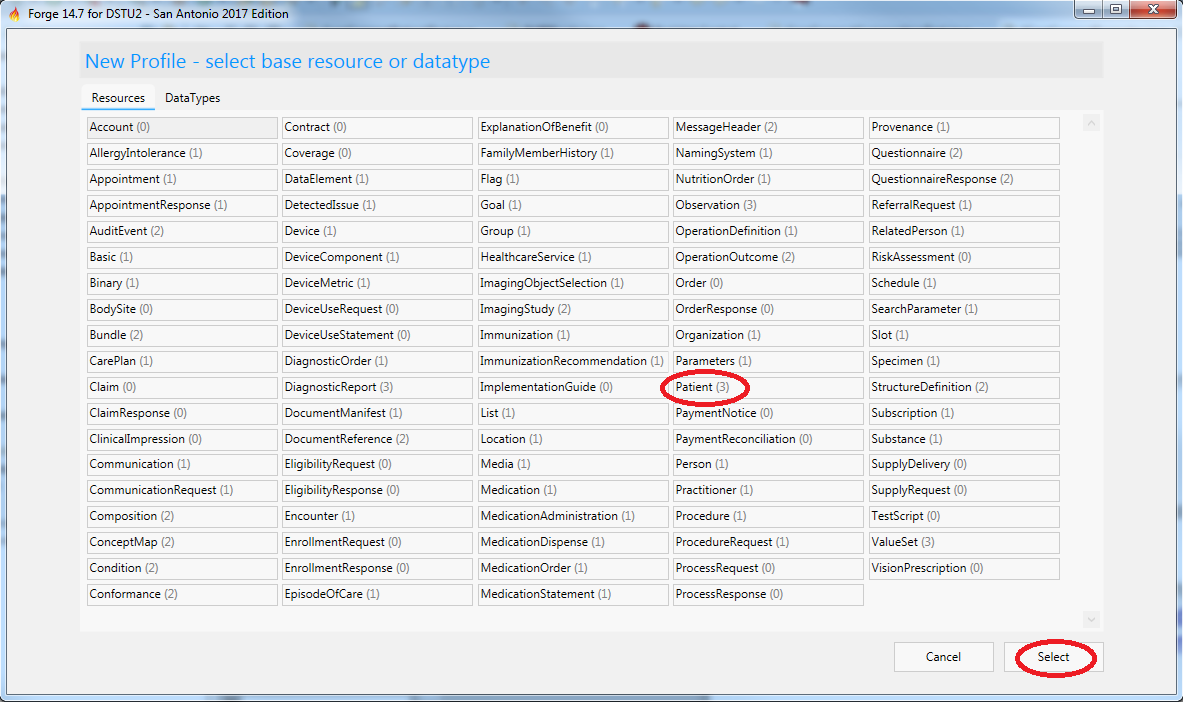
<http://www.interopen.org/candidate-profiles/care-connect/CareConnect-Patient-1.html>



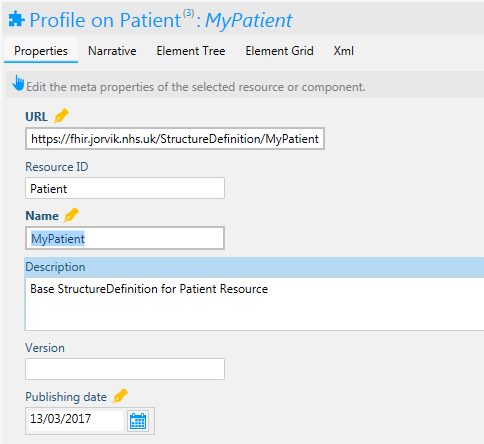
## Profiling

The base resources cover a wide scope and will include elements that aren’t required or too many of them. If we look at the code FHIR Patient resource you will noticed it contains elements for photo and animal (HL7 also covers veterinary).

To begin profiling, install Forge from <https://fhir.furore.com/forge/> and then click on New Profile. In the New Profile screen, select Patient and click on select.

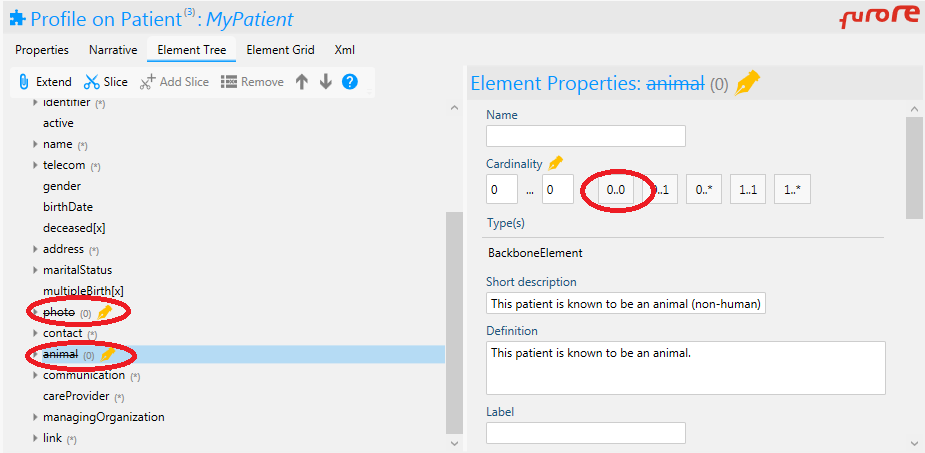


The click on Properties, then change the url to <https://fhir.jorvik.nhs.uk/StructureDefinition/MyPatient>

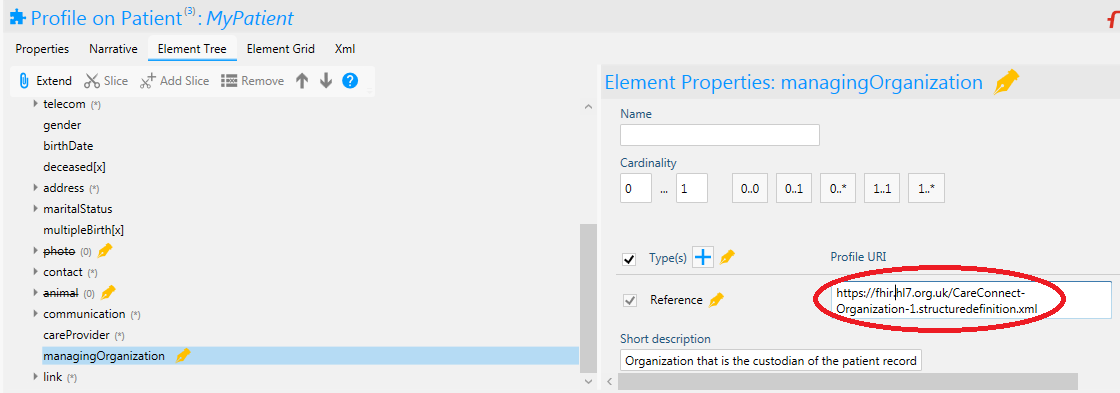


Now click on Element Tree and locate animal. Click on 0..0 to change the cardinality to 0..0, the display for animal will now be struck through as it has been removed from the profile.

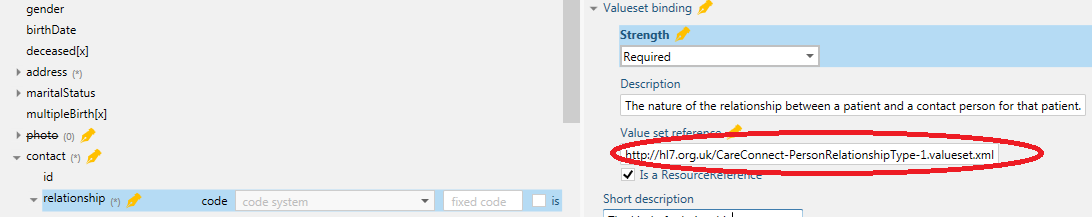
We can do the same with photo. You should see a yellow ink pen nib next to the entries that have just been changed.



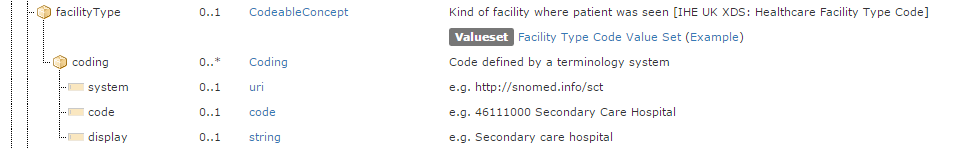
We can also change references to other profiles. E.g. ManagingOrganisation points to base FHIR Organisation but Interopen has defined profile for UK use (<http://www.interopen.org/candidate-profiles/care-connect/CareConnect-Organization-1.html>), so we will use that instead. To do this we simply change the Profile URI for managingOrganization.



Many elements have a list of allowable or recommended values, these are called ValueSets. These can also be changed, e.g. Patients contact (also known as Next of Kin) points to a HL7 valueset. In the UK we have a different list (PERSON RELATIONSHIP) which is defined in CareConnect as <http://www.interopen.org/candidate-profiles/care-connect/CareConnect-PersonRelationshipType-1.valueset.html>



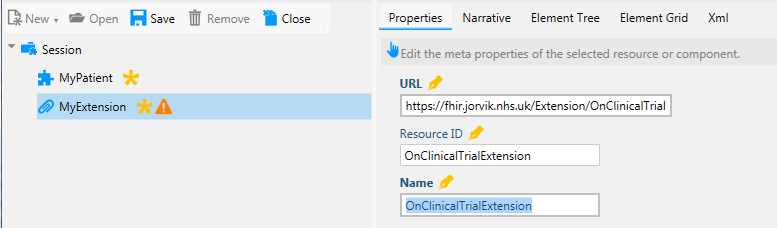
Another use of profiling is to alter the documentation. The example below has amended the DocumentReference profile descriptions to cross reference IHE UK XDS standards and provide example values.

.

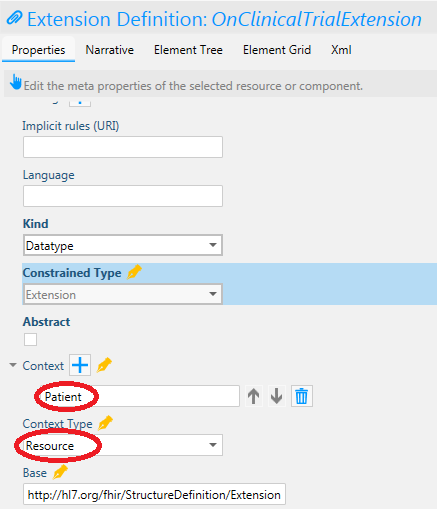
## Extensions

### Simple

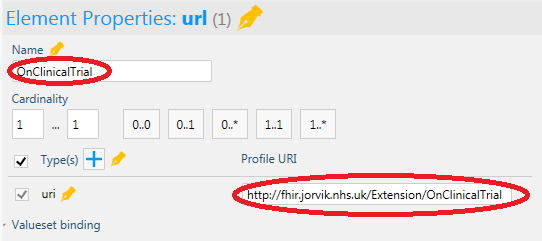
We can easily add elements to the core profiles by using extensions, we will firstly add a simple extension and then a more complex example. In Forge use the menu to create a new Extension, change the url to <https://fhir.jorvik.nhs.uk/Extension/OnClinicalTrial> and change both the resource ID and Name to OnClinicalTrialExtension



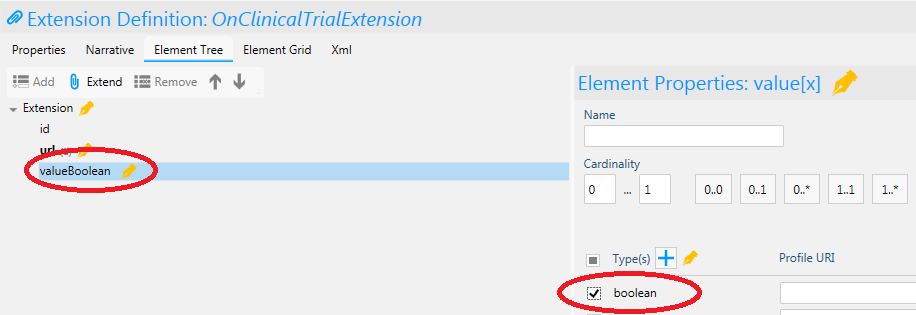
You will have a warning icon next to the extension, to remove this change the ContextType to resource and add a context of Patient. This says the extension is for Patient resource and others can be added if required.



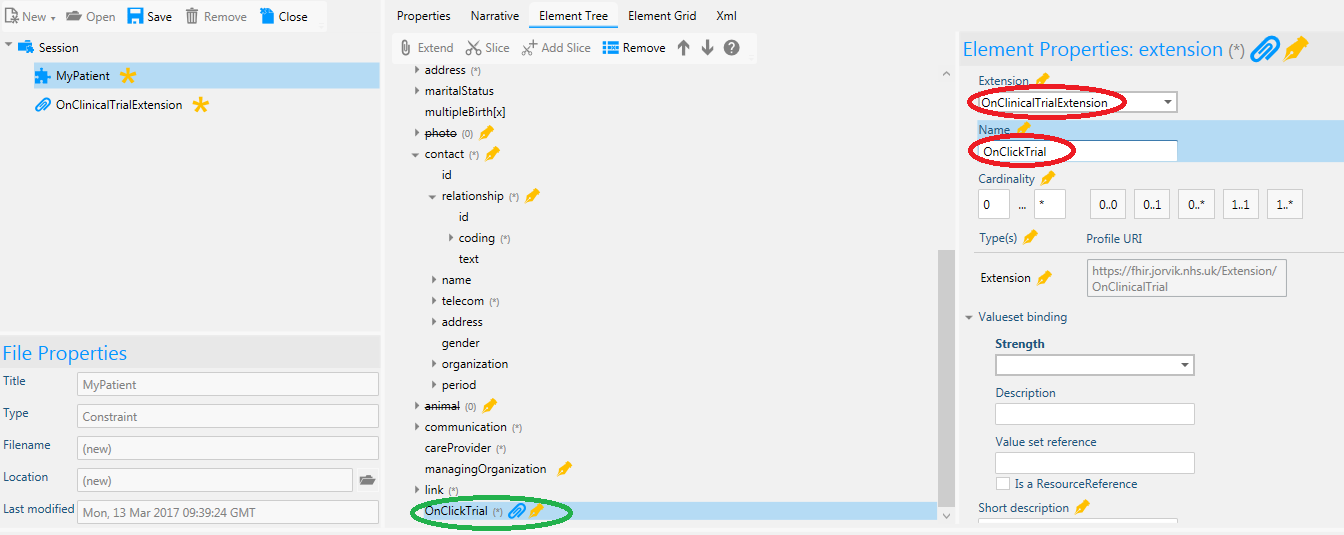
Next we need to identify the extension, this is done by adding a url. A name is also required, note the values used are similar to the values used in the properties tab.



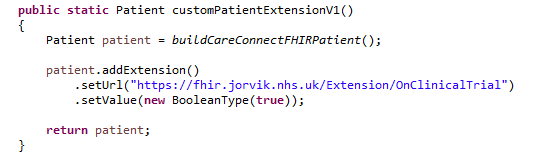
Value currently accepts multiple value types but in our extension we want a simple yes/no Boolean type. Click on the checkbox next to Types to deselect all the values and then select Boolean, value[x] has now changed to valueBoolean.



Lastly we can add the new extension to our Patient Profile. Select MyPatient and then click on Extend (ensure top node, Patient, is selected). In the extension drop down, select the OnClinicalTrialExention and name the extension OnClinicalTrial. Note the extension has been renamed in the Element Tree.



In java (using HAPI libraries) this could look like (the main code for Patient is hidden away in a function):



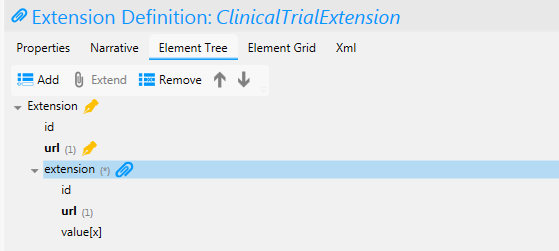
|  |
| --- |
| Extensions can have multiple value types but only one of these can be used at one time. If you need multiple values, use complex extensions. |

### Complex Extension

Create a new extension with URL <https://fhir.jorvik.nhs.uk/Extension/ClinicalTrial>, with resource ID *ClinicalTrialExtension* and named *Clinical Trial Extension*.

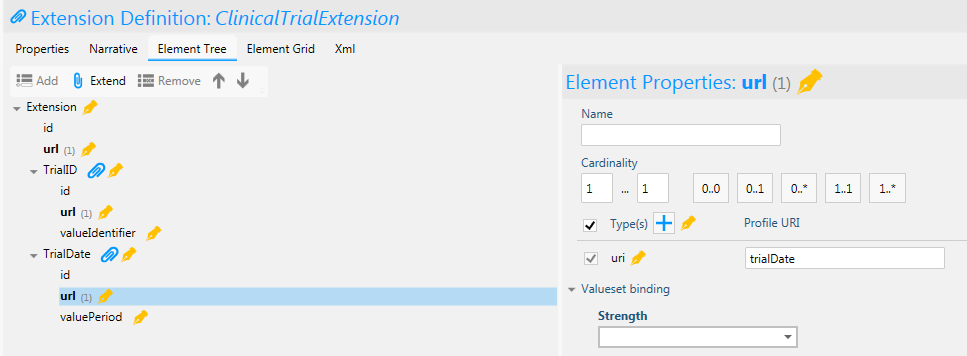
As before set the ContextType to be resource and add a Patient context.

In the Element Tree select Extension and then select Add. This replaces Value[x] with extension - we are extending and extension.



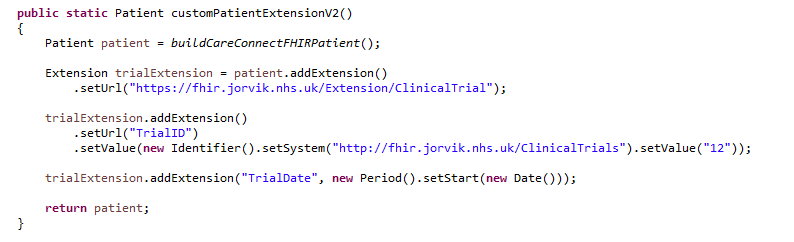
With the new extension highlighted, change the name to TrialID and the cardinality to 0..1 Click on value[x], then uncheck the Types to deselect the types and click on the identifier type to change value[x] to valueIdentifier. In the extension url field enter trialID

Click on Extension again and add another extension, call this TrialDate and set the type to be period. In the url field enter trialDates. Add the new extension to the Patient profile.

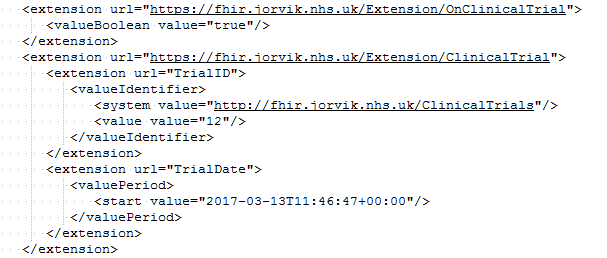


Note: we have not used full uri’s for these extensions.

In java this extension could be coded as



The output in XML would look like this:



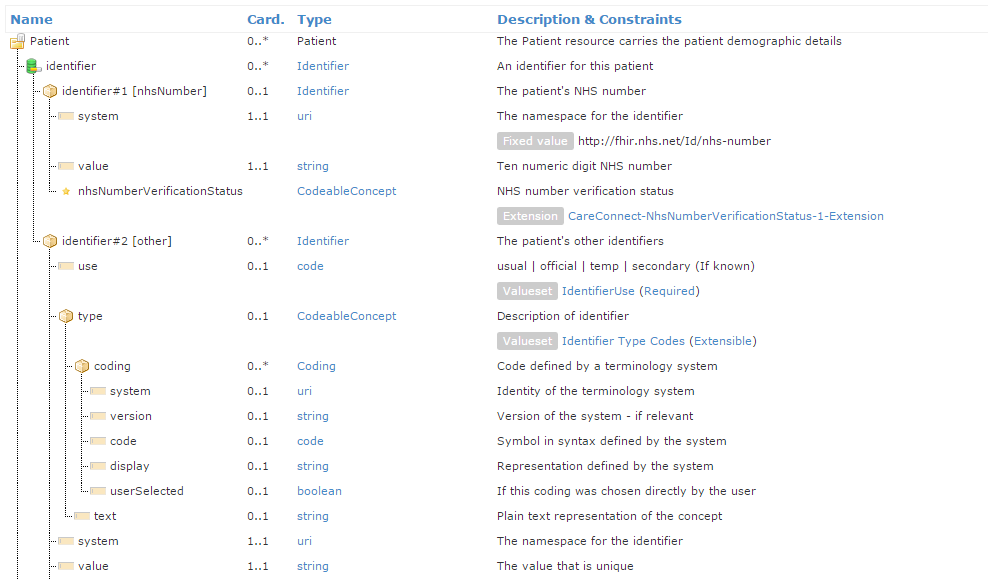
Notice how the extension element is repeated, even though extension can be extended I would not recommend extensions on extensions as this will reduce clarity.

## Derived Profiles

We have shown how to create a profile and add extensions on the Patient resource. In the UK Interopen, HL7 UK and NHS Digital are together working on a set of profiles for use in the UK called CareConnect (<http://www.interopen.org/candidate-profiles/care-connect/>). It is recommended these profiles are used across all methods of using FHIR (api, messaging, documents, etc).

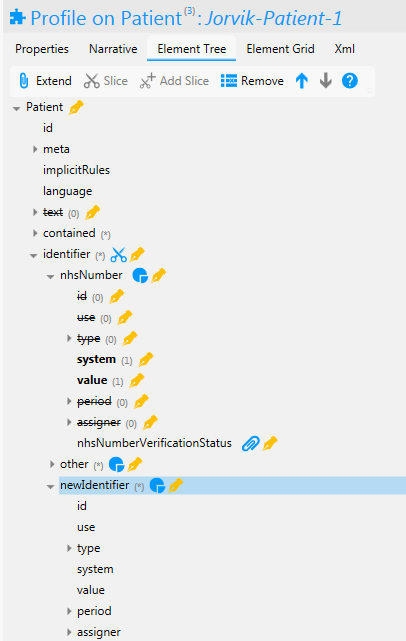
However they will not cover local extensions or restrictions but FHIR does have a way around this called *derived profiles*. *Although the current version 14.7 supports derived profile it has a number of bugs when working with CareConnect profiles and in the examples the CareConnect Patient profile was copied to a new resource.*

The CareConnect Patient profile allows the use local identifiers, to help local users of the profile we wish to specify what these are.



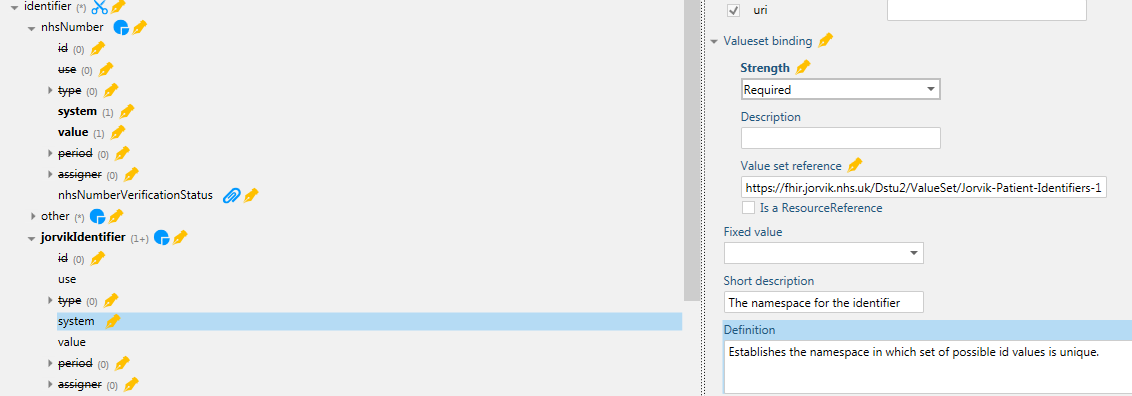
Internally every Patient should have a PAS identifier, we also support PAS identifiers from other trusts in the area. To distinguish the identifiers we have created a ValueSet called <https://fhir.jorvik.nhs.uk/Dstu2/ValueSet/Jorvik-Patient-Identifiers-1>

To add this restriction open up the derived Patient profile and click on identifier. The scissors indicate identifier has been sliced into two parts, one for NHSNumber and one called other. We will add a third called jorvikIdentifiers by clicking on *Add Slice*.

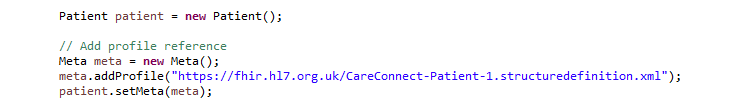


Change the name to jorvik identifier and constraint as required. Click on system, change ValueSet binding strength to required and add [https://fhir.jorvik.nhs.uk/Dstu2/ValueSet/Jorvik-Patient-Identifiers-1](http://jorvik.nhs.uk/Dstu2/ValueSet/Jorvik-Patient-Identifiers-1)

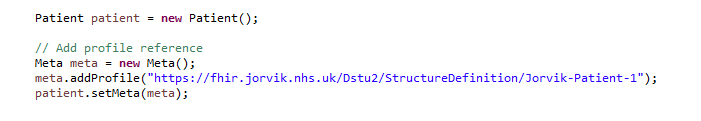
as the Value Set Reference.



The derived profile will output XML/Json that conforms the base profile, so the meta section can point to:



Rather than the derived profile:



Extensions can also be added in the normal manner. These profiles with local extensions should conform to the base profile. FHIR Implementations should ignore extensions they don’t understand but this is not guaranteed. (see Tolerant Reader Pattern <https://martinfowler.com/bliki/TolerantReader.html>)

## Sample Code

Code used in this tutorial can be found on github <https://github.com/KevinMayfield/FHIRTest/tree/master/CareConnectPatient>

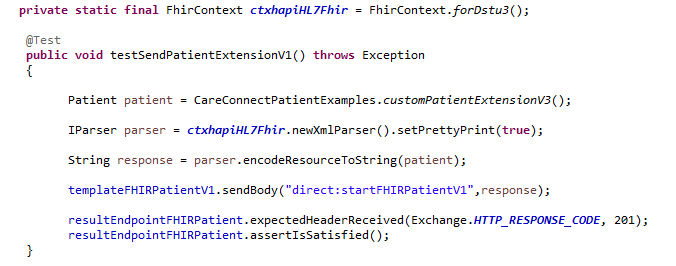
The code includes a basic package to that creates a FHIR Patient resource, sends it to a reference server and then tests the results.

HAPI can communicate directly with a FHIR server but to show the REST/http communications Apache Camel is used. Camel is a popular lightweight integration tool that can use a wide variety of transports and supports testing.

|  |
| --- |
| **Reference Servers**  HAPI FHR has instructions for setting up a Java based reference server (DSTU2 and STU3) <http://hapifhir.io/doc_jpa.html> (see also <https://www.openhealthhub.org/t/howto-build-a-health-database-and-fhir-api-server-in-15-mins-using-open-source/155>)  C# Furore Spark (DSTU2) <https://github.com/furore-fhir/spark>  Intersystems Health Connect (2016) also includes a FHIR Reference Server (DSTU2)  *Note: STU3 will be the current version of FHIR around the 20th March 2017* |

The first part of the code creates a HAPI FHIR context.

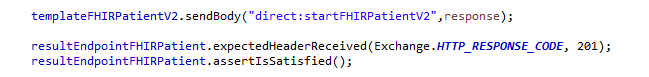
*uk.nhs.jorvik.fhirTest.careConnectPatientTest. CareConnectPatientTest.java*



The buildCareConnectFHIRPatient() builds the resource and the subsequent code serialises this to a string.

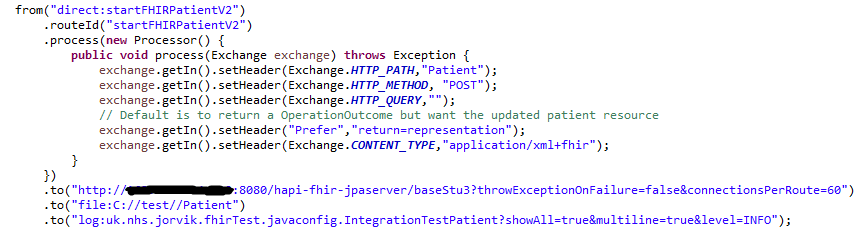
This string is sent directly to the camel route and the http response code is checked to be 201 which means the resource has been created on the server. (*assertIsSatisfied()* is part of camelTest.)

*uk.nhs.jorvik.fhirTest.careConnectPatientTest. CareConnectPatientTest.java*



We will come back to the remainder of the test but first let’s look at the camel route.

*uk.nhs.jorvik.fhirTest.javaconfig.FHIRRoute.java*



The processor section set up the http headers required for the REST call. The response from the server for a Patient POST would normally be FHIR OperationOutcome but we to test the version of the resource the server has stored. To do this we have specified the header **Prefer: return=representation**

The http ip address has been obscured and you should replace with your own reference servers IP address (and if required port number).

### JUnit Testing

The remainder of the code uses a number of junit assertions to confirm the returned data is ok. E.g. Check Religious and ethnic category are present:

