Exercises

Track: **FHIR Profiling**
Speaker: Michel Rutten

This document contains a set of exercises for the ***FHIR for Specifiers*** track at the ***HL7 FHIR DevDays 2017***. All the exercises are based on the current FHIR STU3 release.

Requirements

In order to perform these exercises, participants should have:

- Basic knowledge of FHIR, Resources, REST API
- A Windows (virtual) machine to run Forge, the FHIR profile editor
- An internet connection to access Simplifier, the FHIR profile registry

Tutorials

Participants are also advised to attend the relevant profiling tutorials:

- FHIR Profiling Overview & Introduction (Michel Rutten)
- Profiling: tools overview (Vadim Peretokin)

Documentation

Useful links	
HL7 FHIR Specification (STU3)	https://hl7.org/fhir
Simplifier	https://simplifier.net/
Forge	https://simplifier.net/forge/

Preparations

Simplifier

If you have not done so already, visit https://simplifier.net and create a personal account.

Forge

If you have not done so already, download and install Forge, the FHIR profile editor. Forge is a Windows desktop application. Forge does not support MacOS nor Linux. However, you can install Forge in a virtual machine.

- Visit https://simplifier.net/forge/ and log on
- Download Forge ST3
- Run the ClickOnce installer
  Note: the installer does not require local administrator permissions

Exercise 1 - Dutch Patient

A Dutch hospital needs to exchange information with general practitioners. The organization has decided to implement the information exchange using FHIR. Initially the exchange will focus on patient administrative information and associated general practitioners. After deliberation, the involved parties have drafted the following list of requirements:

A. Patient identification
   a. MUST specify BSN (Dutch social security number)
   b. May also specify other identifiers (e.g. internal EHR identifier)
B. Patient contact information
   a. Accept any number/type of contact details
   b. Empty contact details are not allowed
   c. Clearly identify patient phone number(s) and e-mail address(es)
C. Patient address
   a. Accept any number of addresses
   b. Clearly identify any official address(es)
D. Patient contact address
   a. Same rules apply as for Patient address
E. Marital status
   a. Must use codes as defined by “BurgerlijkeStaatCodelijst”
F. Nationality
   a. Allow to specify patient nationality
   b. Must use codes as defined by “NationaliteitCodelijst”
G. Mappings
   a. Map elements to the national specification: https://zibs.nl/wiki/Patient(NL)
H. The final profile version should be published to Simplifier, so vendors can implement the interface.

Exercise 1.1 - Create new Patient profile

1. Start Forge, the FHIR profile editor
2. Create a new Patient profile
3. Verify that Forge immediately displays a warning message about missing/invalid canonical url.
   The canonical url is a required property. However, Forge cannot automatically generate a value for you and displays a validation warning instead.
4. Specify a unique canonical url to identify the profile
   Note (a): double-click on the warning message in the bottom panel to focus the url property
   Note (b): by convention, use http://example.org for the domain
   Note (c): ensure the url is unique, e.g. include your initials
5. Verify that the validation warning disappears
6. Save the initial version of the patient profile to local disk
   Note: select or create an empty working folder.

***Warning**: When opening or saving a profile from/to disk, Forge scans the containing directory structure for any existing FHIR artifacts. Therefore, it is strongly recommended NOT to store profiles in large folders such as My Documents, Desktop, Temp or drive root (C:\), as this may trigger excessive I/O. Instead, always select or create a separate working folder.*

Exercise 1.2 – Specify profile meta data

Consult the FHIR specification for an explanation of all the available StructureDefinition meta properties: https://www.hl7.org/fhir/structuredefinition.html

1. Specify a name and a title for the profile
2. Initialize profile version = "0.1"
3. Verify status = "Draft" (default)
4. Specify copyright, publisher information and contact details
5. Verify Type = "Patient"
   Note: this value indicates that the profile defines a set of constraints on the Patient resource
6. Verify the profile FHIR version = "3.0.0" (STU3)
   Note: Forge automatically initializes this property to a fixed value, depending on the FHIR version that is supported by the current application release.
7. Verify Base Definition
   Note: this property specifies a reference to the associated base profile.
   Verify that the value equals the canonical url of the core Patient resource.

Exercise 1.3 – Patient Identifier

Specify constraints for requirement (A)

1. Visit https://simplifier.net
2. FHIR defines *naming systems* to recognize different types of identifiers.
   Find a naming system that uniquely identifies Dutch "BSN" identifiers.
   Note: download relevant results to your local working folder.
3. Add constraints to indicate the BSN identifier.
   Reference the naming system you found in step (2)
4. Ensure that the BSN identifier is mandatory
5. Also allow any other identifiers to be specified

Exercise 1.4 – Patient Contact Details

Specify constraints for requirement (B).

1. Add constraints to ensure that any specified phone numbers and e-mail addresses are always clearly identifiable as such.
2. Add constraints to ensure that the actual contact detail value is not empty.

Exercise 1.5 – Patient Address

Specify constraints for requirement (C).

We need a way to indicate/discern official addresses. However, this requirement is not covered by the core Patient resource. Specifically, we need to define a custom extension element for address components to indicate if the component represents an official address.

1. Visit https://simplifier.net
2. Search for relevant extensions to indicate official addresses.
   Note: download relevant results to your local working folder.
3. Open and inspect the downloaded extension definition in Forge
4. Add a new extension element to the Patient address component.
5. Map the new extension element to the definition found in step (2).
   Note: reload the profile to expand the referenced extension definition.

Exercise 1.6 – MaritalStatus

Specify constraints for requirement (E).

1. Verify the default terminology bindings on maritalstatus.
2. Visit https://simplifier.net
3. Find a suitable *ValueSet* resource called "BurgerlijkeStaatCodelijst".
   Note: download relevant results to your local working folder.
4. Bind the maritalstatus element to the ValueSet resource found in step(2)
5. Make sure that derived profiles cannot replace or override the terminology bindings

Exercise 1.7 – Nationality

Specify constraints for requirement (F).

We need a way to specify patient nationality. However, this requirement is also not covered by the core Patient resource. Specifically, we need to define a custom extension element on the Patient profile to indicate the nationality.

1. Visit https://simplifier.net
2. Find a suitable extension definition that represent "patient nationality"
   Note: download relevant results to your local working folder
3. Open and inspect the downloaded extension definition in Forge
4. Add a new extension element to the Patient profile (as a child of the root element)
5. Map the new extension element to the definition found in step (2).
   Note: reload the profile to expand the referenced extension definition.
6. Verify the default terminology constraints on the nationality extension element
7. Search Simplifier for a *ValueSet* resource called "NationaliteitCodelijst"
   Note: download relevant results to your local working folder
8. Bind the nationality extension element to the ValueSet found in step (6).
9. Make sure that derived profiles are allowed to extend the custom terminology bindings

Exercise 1.8 – Contact Address

Specify constraints for requirement (D).

According to the requirements, the same constraints that apply to patient address information also apply to contact address information. The patient profile could simply repeat the address constraints on both elements. However, the client prefers to refer to an external set of common, national Dutch address constraints.

1. Visit https://simplifier.net
2. Search for a suitable Dutch Address profile
   Note (1): include "NL" keyword to find relevant Dutch results.
   Note (2): need to support official address indicator
   Note (3): download relevant results to your local working folder
3. Open and inspect the external Address profile in Forge
4. Update the constraints on Patient address
   Remove all of the previously specified constraints.
   Constraint the Patient address type profile to reference the external Address profile found in step (2).
5. Also apply the Address type profile to the Patient contact address element.
   Note: Forge should automatically expand the external Address profile. However sometimes this fails due to aggressive caching. As a workaround, you can restart the application and reload the patient profile.
6. Verify that the Patient profile inherits the official address extension from the external address profile

Exercise 1.9 – Publish to Simplifier

1. Publish the final patient profile to your Simplifier project
2. Verify the profile rendering and compare to Forge
3. Verify the displayed profile attributes (type, FHIR, status, version) and canonical url
4. Verify hyperlinks to external artifacts
   (such as NamingSystems, ValueSets, type profiles, extension definitions etc.)
5. Visit the XML and JSON tabs and inspect the serialization.
   Note: fortunately you don't need to write this manually…!

Exercise 1.10 – Patient example

In this exercise, you need to create an example Patient resource in order to verify the profile. Unfortunately, you cannot create example resources in Forge, as the application only supports editing StructureDefinitions. Therefore, we need create the example by manually authoring XML or JSON.

As a starting point, you can use the following example XML:

```xml
<Patient>
    <meta>
        <profile
         value="http://example.org/fhir/StructureDefinition/DutchPatient" />
    </meta>
    <active value="true" />
    <name>
        <use value="official" />
        <family value="Jansen" />
        <given value="Jan"/>
    </name>
</Patient>
```

1. Create an example Dutch Patient instance using an (XML) editor
2. Ensure that the example instance claims conformance to your Dutch Patient profile.
   Note: include a Patient.meta.profile element and reference the target profile.
3. Save the example
4. Publish the example patient to your Simplifier project
5. Verify the rendering, should display the correct patient name etc.
6. Navigate to your Dutch Patient profile on Simplifier
   Select the *Example* tab
   Note that Simplifier now displays a link to your Patient example.
7. Click the link to navigate back to the Patient example
   Note that the top-right toolbar displays a "Validate" button
   The validation command is available for resource instances with a profile conformance claim
8. Validate the example using the toolbar button
   Verify the validation results.
   Note: we expect validation to fail because the instance does not specify a mandatory BSN identifier
9. Edit the example and add a (mandatory) BSN identifier value
   Ensure that the BSN example value is properly identified using the correct naming system.
10. Update the previously published example by uploading the new version
    Note: navigate to the example page on Simplifier and use the "Update" toolbar button
11. Validate the updated Patient example
    Note: we expect validation to succeed.
12. Select the *History* tab.
    Verify that Simplifier displays both the original and the updated example versions.
    Click the checkboxes in order to compare specific versions.

Exercise 2 - Dutch Practitioner

The same Dutch hospital also needs to exchange information about general practitioners, with the following requirements:

A. Practitioners identification
   Allow all of the Dutch national identification schemes for medical professionals:
   a. UZI number (optional)
   b. AGB number (optional)
   c. BIG number (optional)
   A practitioner record should specify at least one of these identifiers.
B. Practitioner address
   a. Accept any number of addresses
   b. Clearly identify any official address(es)

Exercise 2.1 – Create a new Practitioner profile

1. Use Forge to create a new Practitioner profile
2. Specify a unique canonical url to identify the profile
   Note (a): double-click on the warning message in the bottom panel to focus the url property
   Note (b): by convention, use http://example.org for the domain
   Note (c): ensure the url is unique, e.g. include your initials
3. Verify that the validation warning disappears
4. Save the initial version of the practitioner profile to local disk
   Note: select or create an empty working folder.

Exercise 2.2 – Specify profile meta data

1. Specify a name and a title for the profile
2. Initialize profile version = “0.1”
3. Verify status = “Draft” (default)
4. Specify copyright, publisher information and contact details
5. Verify Type = “Practitioner”
6. Verify the profile FHIR version = “3.0.0” (STU3)
7. Verify Base Definition
   Verify that the value equals the canonical url of the core Practitioner resource.

Exercise 2.3 – Practitioner Identifier

Specify constraints for requirement (A)

1. Visit https://simplifier.net
2. Find naming systems for the required Dutch national practitioner identification schemes (UZI, AGB, BIG)
3. *Note: download relevant results to your local working folder.*
4. Add constraints for all of the required identifier schemes
   Reference the naming systems you found in step (2)
5. Ensure that the identifier values are mandatory
6. Also allow any other practitioner identifiers to be specified

Exercise 2.4 – Practitioner Address

Specify constraints for requirement (B).
Reuse the external Dutch address profile from exercise (1.8).

Exercise 2.5 – Practitioner Example

Create an example Practitioner resource in order to verify the Practitioner profile.

As a starting point, you can use the following example XML:

```xml
<Practitioner>
    <meta>
        <profile
         value="http://example.org/fhir/StructureDefinition/DutchPractitioner" />
    </meta>
    <identifier>
        <system value="http://fhir.nl/fhir/NamingSystem/dummy" />
        <value value="129854656" />
    </identifier>
    <name>
        <use value="official" />
        <family value="Nicolaes" />
        <given value="Tulp"/>
    </name>
</Practitioner>
```

1. Create an example Dutch Practitioner instance using an (XML) editor
2. Ensure that the example instance claims conformance to your Dutch Practitioner profile.
   Note: include a Practitioner.meta.profile element and reference the target profile.
3. Save the example
4. Publish the example practitioner to your Simplifier project
5. Verify the rendering, should display the correct practitioner name etc.
6. Navigate to your Dutch Practitioner profile on Simplifier
   Select the *Example* tab
   Note that Simplifier now displays a link to your Practitioner example.
7. Click the link to navigate back to the Practitioner example
   Note that the top-right toolbar displays a "Validate" button
   The validation command is available for resource instances with a profile conformance claim
8. Validate the example using the toolbar button
   Verify the validation results.
   Note: we expect validation to fail because the instance does not specify one of the required identifier schemes

13. Edit the example and add a (mandatory) UZI identifier value
    Ensure that the UZI example value is properly identified using the correct naming system.
14. Update the previously published example by uploading the new version
    Note: navigate to the example page on Simplifier and use the "Update" toolbar button
15. Validate the updated Practitioner example
    Note: we expect validation to succeed.
    Select the *History* tab.
    Verify that Simplifier displays both the original and the updated example versions.
    Click the checkboxes in order to compare specific versions.

Exercise 3 –Profile relationships

The client requires that Dutch Patient record should specify references to only Dutch Practitioner records (and not e.g. to German practitioner records).

Exercise 3.1 – Patient practitioner

Constraint the Dutch patient profile such that the practitioner element must be a reference to a Dutch practitioner.

1. Open the previously created Dutch Patient in Forge
2. Make sure that Patient.practitioner values must reference dutch practitioners
   Note: use the type.targetProfile *property to constrain a ResourceReference*
3. Save the new profile version to local disk
4. Publish the new Patient profile version on Simplifier
5. Inspect the rendering of the updated profile
   Note: Simplifier renders the cross-profile relationships as hyperlinks

Exercise 3.2 – Patient example

1. Edit the previously created Patient example instance
2. Specify a reference to the Practitioner example created earlier
3. Publish the updated Patient example to Simplifier
4. Validate the example

Exercise 4 – Implementation Guide

The client requires you to publish an implementation guide on Simplifier to document the Dutch Patient and Practitioner scenarios. The Implementation Guide design should be similar to this example: http://implementationguidetemplate.azurewebsites.net/

1. Log on to Simplifier and visit your personal project page
2. Click on "Guides" tab
3. Use the "Create" button to start designing a new Implementation Guide
4. Add pages for the individual profiles
   Include rendered profile structures in the page
5. Add pages for the example resources
   Render the examples as both XML and JSON
6. Publish the IG

Congratulations! You have finished all of the FHIR profiling exercises. Now go grab yourself a (root) beer, you definitely deserved it.