Scheduling Workflow

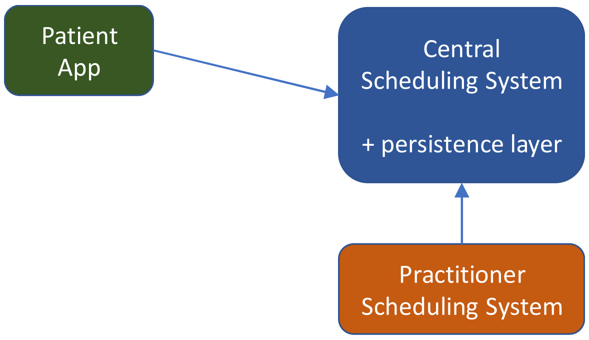
This exercise aims to allow you to familiarize yourself with an example workflow which is supported by the exchange of FHIR resources.

The FHIR version used in this exercise will be FHIR R4. The **[base]** FHIR server (unless the trainer informs you otherwise) used for this exercise will be https://vonk.fire.ly/R4

Context

For this exercise we'll assume that there are three applications:

1. A Patient App (a RESTful FHIR Client) which has the capability of creating, retrieving and updating appointments for the user of the App;
2. A Central Scheduling System (a RESTful FHIR server) which has the capability to manage schedules and deal with appointment requests. This solution is used to support the scheduling of most (human) resources, however some practitioners use their own scheduling system. All scheduling information (regardless of which application manages a particular schedule) are persisted on the Central Scheduling System.
3. A Practitioner Scheduling System (a RESTful FHIR Client) which has the capability of processing appointment requests which involve specific practitioners. This system also manages the schedules (and slots) of those practitioners.



Exercises

All exercises are related to the schedule of a psychologist, Dr. Maria Velasquez. Her schedule (with both free as well as busy slots) for Tuesday next week, and the Tuesday thereafter, has been made available on Central Scheduling System (the [base] FHIR Server).

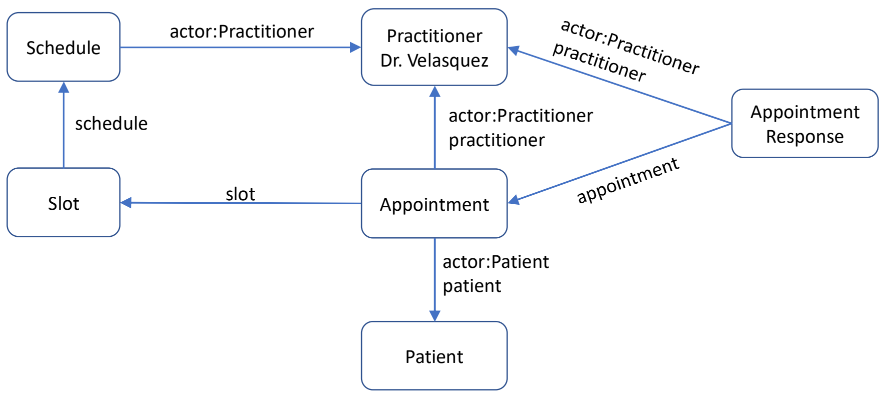
1. (Gezamenlijk) What appointments does the patient with the family name Tuttleworth have ?
   - The focus of your search will be the Appointment resource
   - Alternatively you could search by Tuttleworth's citizen identifier, which is 1036475869

2. Let's assume you (as a patient) would like to make a new appointment with Dr. Velasquez. What are the free slots in her schedule for Tuesday next week ?
   - The focus of your search will be the Slot resource
   - The server will also have Slots related to other clinicians. Your search should ensure that only the Slots of Dr. Velasquez (identifier 2304123) will be returned. Hint: use parameter chaining and a parameter on the Schedule resource type.
   - FYI: this type of search is also covered by the **$find** operation as defined in various Scheduling-oriented implementation guides. That operation is however not available on our public test server.
3. Let's assume you're a new patient, and as such you don't have a preference for any specific psychologist at this point in time. What are the free slots for a psychology session ?
   - The focus of your search will be the Slot resource
   - The code for the Psychology specialty is 230
   - FYI: this type of search is also covered by the **$find** operation as defined in various Scheduling-oriented implementation guides. That operation is however not available on our public test server.
4. As a patient, create a new appointment request.
   - Select a random Patient resource already known on the [base] server (or: create your own, if you know how to do so), and copy the logical Id (the URL) of the selected Patient resource to a text editor.
   - Search for the logical Id (the URL) of Dr. Velasquez. Please copy the logical Id (the URL) of the Practitioner resource to a text editor.
   - Select a free slot in the schedule of Dr. Velasquez. Please copy the logical Id (the URL) of the selected Slot resource to a text editor.
     - Note: another participant in this exercise may have selected the very same free slot. As such your appointment request may be refused if they happen to book an appointment before you do.
   - Edit the example_appointment_request_R4.xml or example_appointment_request_R4.json file (which can be found in your electronic handout). Make sure to update (at least) the following items: (1) the logical id (URL) of Dr. Velasquez (one of the actors involved in the appointment), (2) the logical id (URL) of the Patient (another actor involved in the appointment), and (3) the logical Id (URL) of the free Slot.
   - POST the newly created appointment (-request) to the **[base]** server.
   - (A simulator will process your request within 60 seconds. Note that the simulator won't be up and running should you attempt do this exercise (again) at some future point in time)
   - GET the Appointment and verify that its status has been set to 'booked' by the Practitioner Scheduling System. GET the Slot and verify that its status has been set to 'busy'. GET all AppointmentResponse resources associated with the Appointment resource.
     - If your request failed due to a busy slot, update your Appointment with the logical Id (URL) of a free slot, set the participant.status of the Practitioner to 'needs-action', and PUT the Appointment.
   - FYI: the booking of appointments is also covered by the **$book** operation as defined in various Scheduling-oriented implementation guides. That operation is however not available on our public test server.

Solution

Overview of resources used

The figure below depicts the resource types used during this exercise, as well as their references. The ‘names’ on the references represent the name of the search parameter used for parameter chaining.



Exercise 1. Appointments of patient Tuttleworth

The focus of the search is the Appointment resource. However, Tuttleworth is the name of a patient, and as such we’ll need parameter chaining (patient is a search parameter defined for Appointment, name is a search parameter defined for Patient).

```
GET [base]/Appointment?patient.name=Tuttleworth
```

The status code of the response should be 200 if this patient exists on the chosen server.

Querying by name has its problems, as this may well lead to multiple matching patient resources. Therefore, it would be better to query on the national citizen identifier of Tuttleworth. The focus of the search is the Appointment resource. However, the patient identifier is an attribute of a patient, and as such we’ll need parameter chaining (patient is a search parameter defined for Appointment, identifier is a search parameter defined for Patient).

```
GET [base]/Appointment?patient.identifier= 1036475869
```

If we look at the appointment resource in the response bundle (there should be only one matching appointment) we can see that Tuttleworth has an appointment on Tuesday next week from 09:00 to 09:30, with Dr. Velasquez.

Exercise 2 – Free slots, next Tuesday, for Dr. Velasquez

The focus of the search is the Slot resource. However, Velasquez is the name of a practitioner, and as such we’ll need parameter chaining (schedule is a search parameter defined for Slot, actor is a search parameter defined for Schedule, name is a search parameter defined for Practitioner). We’re only interested in actors of (resource type) Practitioner, hence the use of the **actor:Practitioner** search parameter modifier.

```
GET [base]/Slot?schedule.actor:Practitioner.name=Velasquez
```

Or, using the identifier of Dr. Velasquez:

```
GET [base]/Slot?schedule.actor:Practitioner.identifier= 2304123
```

The ".. Tuesday next week" part of the query is expressed as follows (notes: (1) start is a search parameter for the Slot resource, (2) you will have to include the actual date of 'Tuesday next week', not the date mentioned in our example below, (3) eq = equal to):

```
GET [base]/Slot? …(above).. &start=eq2020-10-07
```

The ".. FREE slots" part of the query is expressed as follows (note: status is a search parameter for the Slot resource):

```
GET [base]/Slot? …(above).. &status=free
```

Exercise 3 – Free slots, any psychologist

The focus of the search will be the Slot resource. However, specialty (in our case: Psychology) is a property of a Schedule resource, so we'll have to use parameter chaining.

```
GET [base]/Slot?status=free&schedule.speciality=230
```

This returns 'free' slots related to multiple different psychologists.

Exercise 4 – New appointment with Dr. Velasquez

After updating the example resource with the references (to the Patient resource, to Dr. Velasquez, to the Slot), post the resource to the server:

```
POST [base]/Appointment
```

The status code of the response should be `201 Created`. Please copy the logical Id (the URL) of the newly created Appointment resource.

(it will take up to 60 seconds for the appointment request to be processed). GET the Appointment resource using the logical Id / the URL of the Appointment resource created during the previous step. Once processed, the status of the appointment will be 'booked'.

If you somehow constructed an erroneous Appointment request, or associated it with a 'busy' slot, the various status codes in both Appointment as well as AppointmentResponse will identify a 'declined appointment'. The reason for the declining the request can be found in the <text> part of AppointmentResponse. Correct your Appointment, and PUT it to update your request.