Argonaut Scheduling

04/26/2017

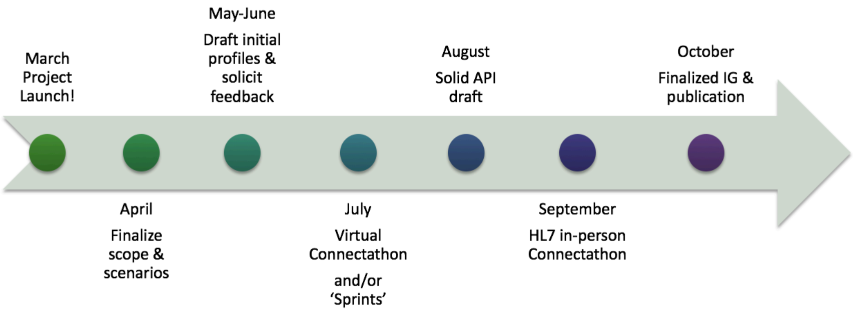
Eric Haas

Brett Marquard

Overview

- Welcome and Introductions
- Review of last call
- Walkthrough simple use case

Timelines



March
Project
Launch!

April
Finalize
scope &
scenarios

May-June
Draft initial
profiles &
solicit
feedback

July
Virtual
Connectathon
and/or
'Sprints'

August
Solid API
draft

September
HL7 in-person
Connectathon

October
Finalized IG &
publication

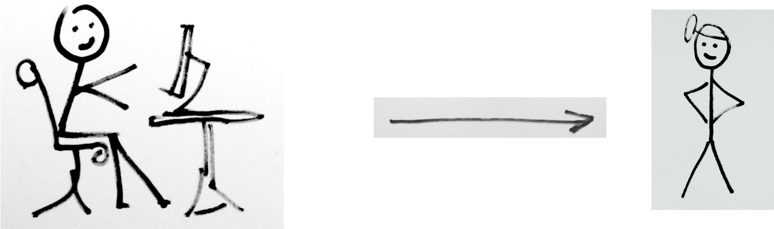
Summary from last call

- *DECISION to Work in R3 and create an R2 version after publishing if demand and budget is there*
- **Agreed to following Phases**
  - Base patient access
  - Base provider access
  - Advanced provider/patient/both

Note - meeting agenda and notes are archived at: https://github.com/argonautproject/scheduling/tree/master/meeting-notes

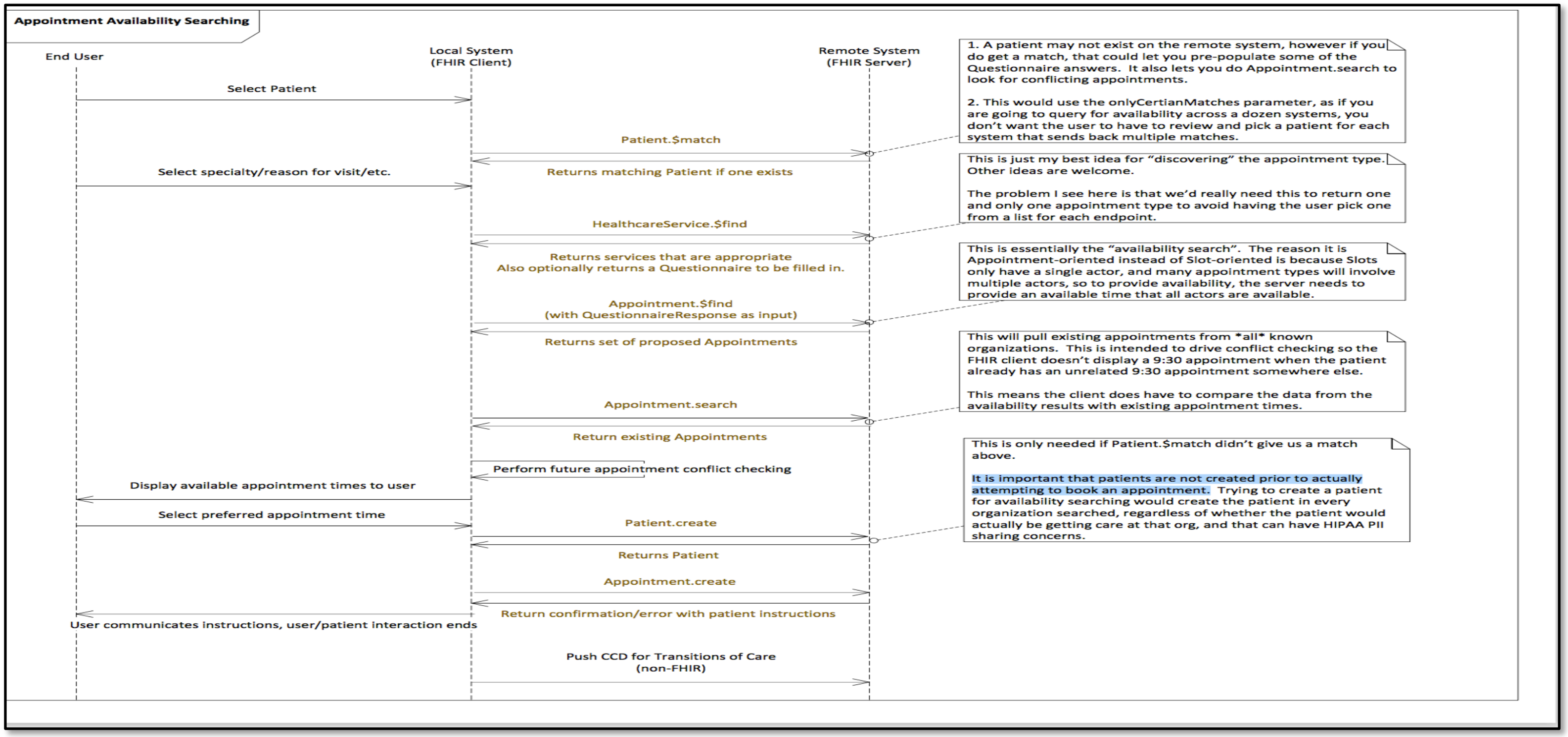
Scheduling Walk-Through

Simple Scenario: Patient Scheduling for single provider



- Prerequisites:
  - Single Provider (Single Specialty, Location, Service, Endpoint, etc)
  - Single System
  - Schedule
    - Assume system has available Schedule resource
  - Patient
    - New or Existing
  - Other scenarios: Update/Cancel/Change Appointments – out of scope for this scenario

Interaction Diagram by Cooper


Appointment Availability Searching
End User
Local System
(FHIR Client)
Remote System
(FHIR Server)
Select Patient
Patient.$match
Returns matching Patient if one exists
Select specialty/reason for visit/etc.
HealthcareService.$find
Returns services that are appropriate
Also optionally returns a Questionnaire to be filled in.
Appointment.$find
(with QuestionnaireResponse as input)
Returns set of proposed Appointments
Appointment.search
Return existing Appointments
Perform future appointment conflict checking
Display available appointment times to user
Select preferred appointment time
Patient.create
Returns Patient
Appointment.create
Return confirmation/error with patient instructions
User communicates instructions, user/patient interaction ends
Push CCD for Transitions of Care
(non-FHIR)

1. A patient may not exist on the remote system, however if you do get a match, that could let you pre-populate some of the Questionnaire answers. It also lets you do Appointment.search to look for conflicting appointments.

2. This would use the onlyCertianMatches parameter, as if you are going to query for availability across a dozen systems, you don't want the user to have to review and pick a patient for each system that sends back multiple matches.

This is just my best idea for "discovering" the appointment type. Other ideas are welcome.

The problem I see here is that we'd really need this to return one and only one appointment type to avoid having the user pick one from a list for each endpoint.

This is essentially the "availability search". The reason it is Appointment-oriented instead of Slot-oriented is because Slots only have a single actor, and many appointment types will involve multiple actors, so to provide availability, the server needs to provide an available time that all actors are available.

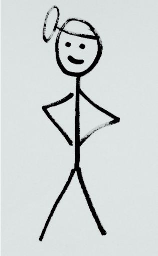
This will pull existing appointments from *all* known organizations. This is intended to drive conflict checking so the FHIR client doesn't display a 9:30 appointment when the patient already has an unrelated 9:30 appointment somewhere else.

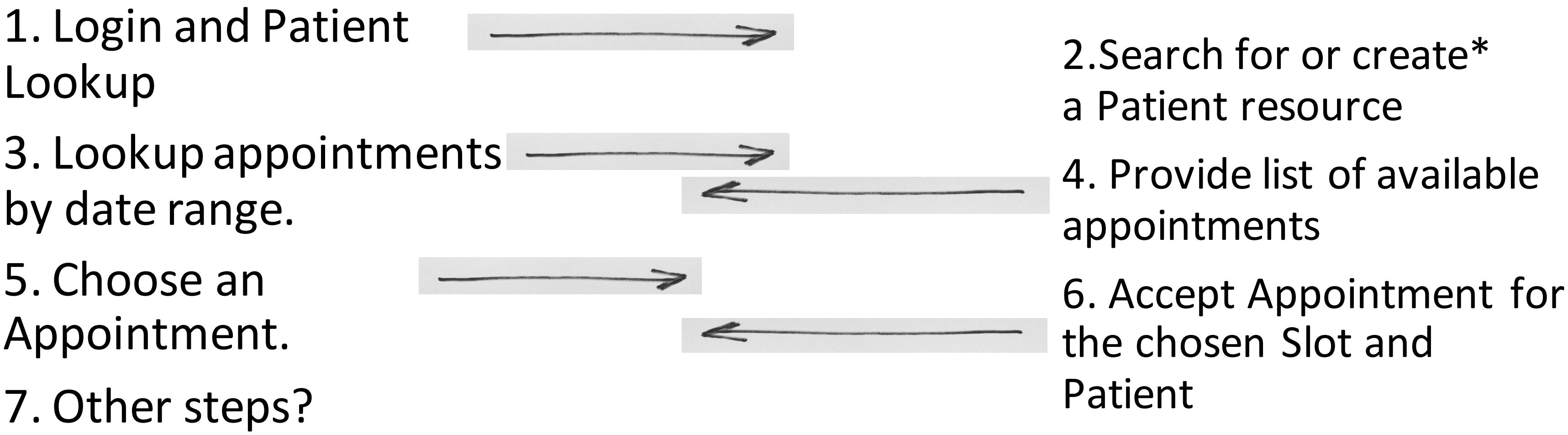
This means the client does have to compare the data from the availability results with existing appointment times.

This is only needed if Patient.$match didn't give us a match above.

It is important that patients are not created prior to actually attempting to book an appointment. Trying to create a patient for availability searching would create the patient in every organization searched, regardless of whether the patient would actually be getting care at that org, and that can have HIPAA PII sharing concerns.

Use Case: Overview of Steps





1. Login and Patient Lookup

2.Search for or create* a Patient resource

3. Lookup appointments by date range.

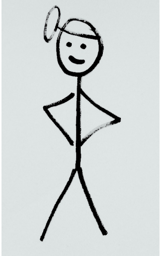
4. Provide list of available appointments

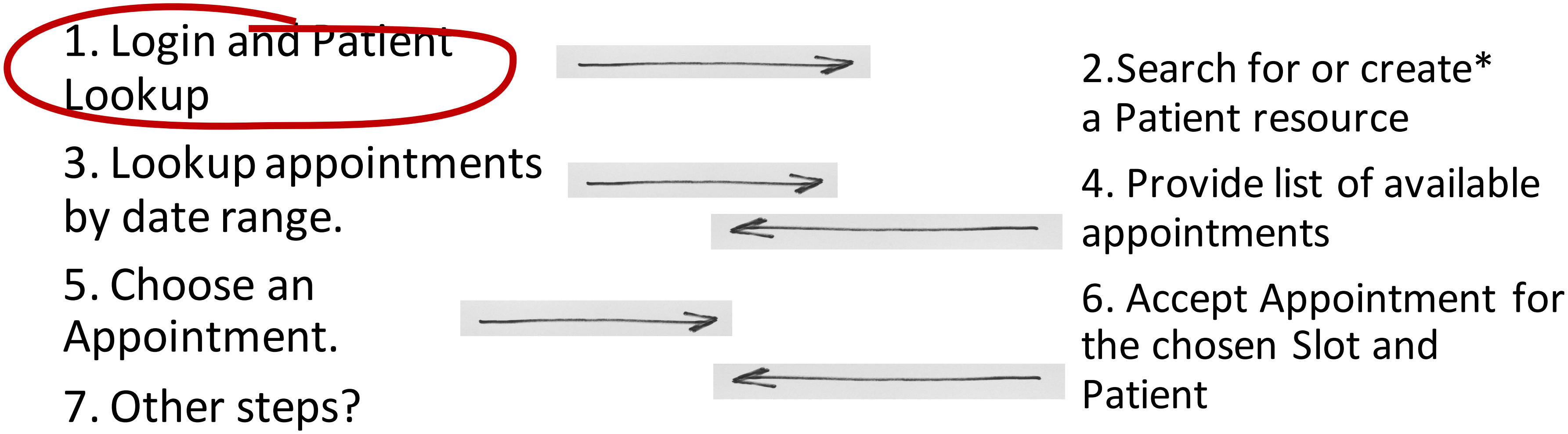
5. Choose an Appointment.

6. Accept Appointment for the chosen Slot and Patient

7. Other steps?

Use Case: Overview of Steps





1. Login and Patient Lookup

2.Search for or create* a Patient resource

3. Lookup appointments by date range.

4. Provide list of available appointments

5. Choose an Appointment.

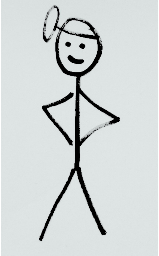
6. Accept Appointment for the chosen Slot and Patient

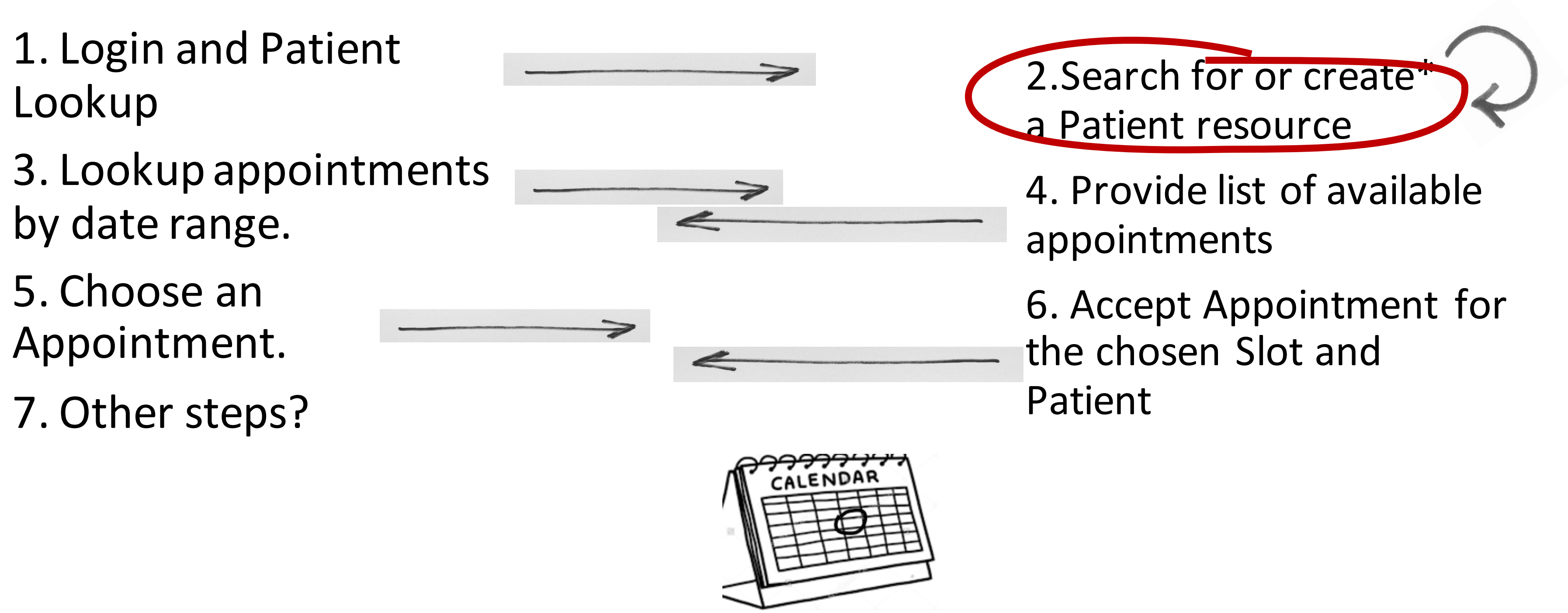
7. Other steps?

Step 1: Login and Patient Lookup

- Login: assume client has access and not going to discuss today
- Patient Lookup
  - Existing Patient
    - need to provide some identifying information
      - Name
      - Password
      - ID?
      - Other
  - New Patient
    - need to provide some identifying information

Use Case: Overview of Steps





1. Login and Patient Lookup

2.Search for or create* a Patient resource

3. Lookup appointments by date range.

4. Provide list of available appointments

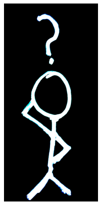
5. Choose an Appointment.

6. Accept Appointment for the chosen Slot and Patient

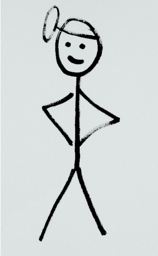
7. Other steps?

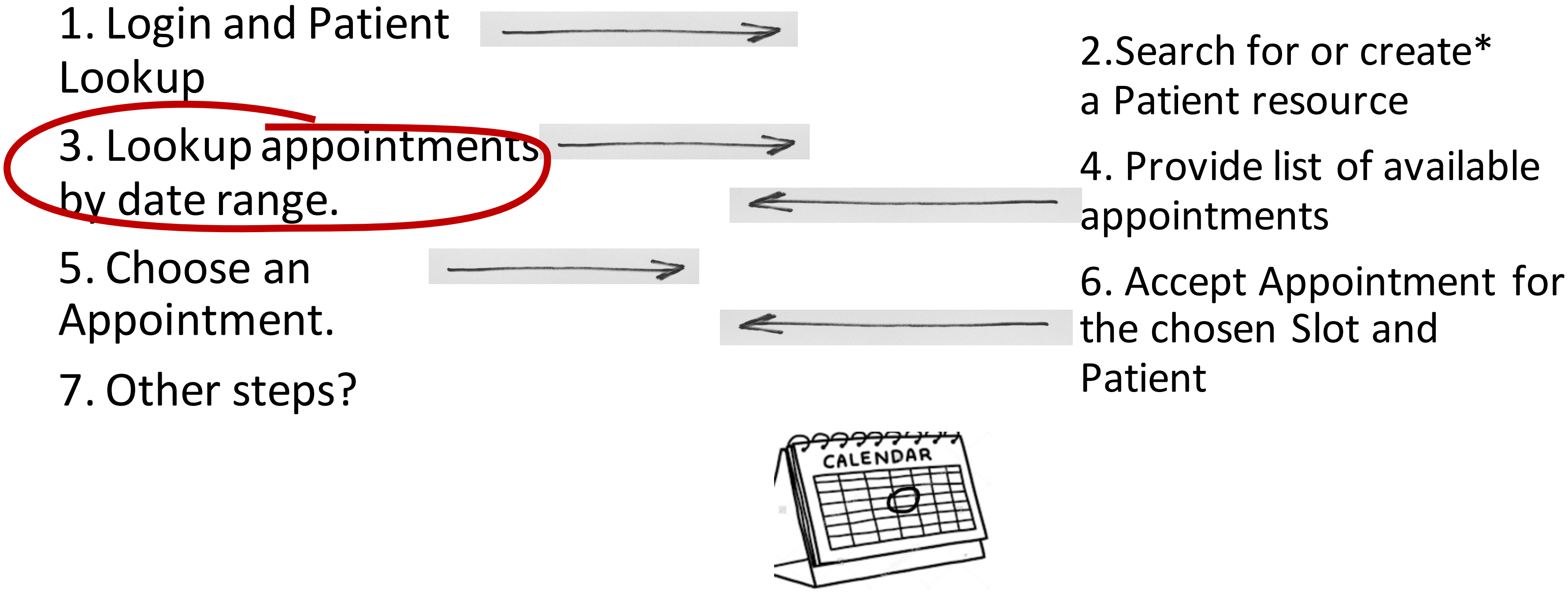
Step 2: (Server) Patient Matching or create new Patient

- Patient is logged in – now what?
  - Does server identify patient based on user login
    - (Patient.$match?)
    - If Not how is the patient identified
  - New Patient: how do to deal with this?
    - ‘It is important that new patients are not created prior to actually attempting to book an appointment.’ (Cooper)
  - What, if anything, needs to be returned back to client in step 2:
      - Patient?



Use Case: Overview of Steps





1. Login and Patient Lookup

2.Search for or create* a Patient resource

3. Lookup appointments by date range.

4. Provide list of available appointments

5. Choose an Appointment.

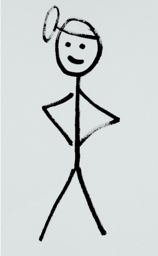
6. Accept Appointment for the chosen Slot and Patient

7. Other steps?

Step 3: (Client) Patient Searches for Availability

- What are the operations or queries we need?
  - See Cooper's OperationDefinition Appointment $find https://simplifier.net/Scheduling/Appointment-find/rendered
    - **Returns proposed 'prefilled' Appointme**nts.
  - see NHS $gpc.getschedule https://nhsconnect.github.io/gpconnect/appointments_use_case_search_for_free_slots.html#request-headers
    - Returns a bundle of slots and other information
- What are basic search requirements for client
  - Available slots for the Provider Schedule
    - Any?
    - Within time range
      - Maximum? e.g NHS limited to 2 week period
    - Other search parameters see NHS design decision: https://nhsconnect.github.io/gpconnect/appointments_design.html#search-parameters
  - Chief Complaint here or in Step 5?

Use Case: Overview of Steps



1. Login and Patient Lookup

2.Search for or create* a Patient resource

3. Lookup appointments by date range.

4. Provide list of available appointments

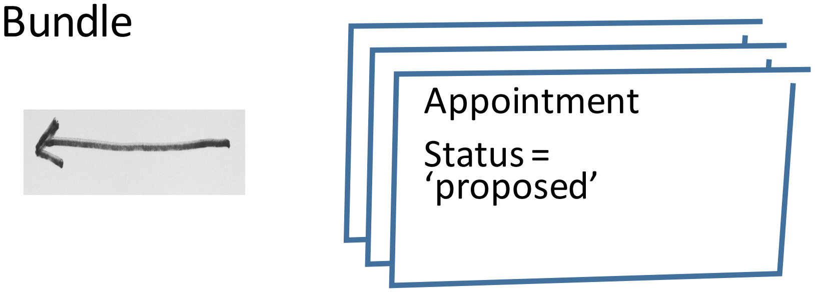
5. Choose an Appointment.

6. Accept Appointment for the chosen Slot and Patient

7. Other steps?

Step 4: (Server) Fetches and Returns Available Appointments Based on Client Supplied Parameters

- Based the operations or queries from step 3
  - Return a Searchset Bundle

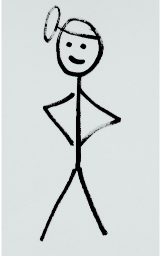


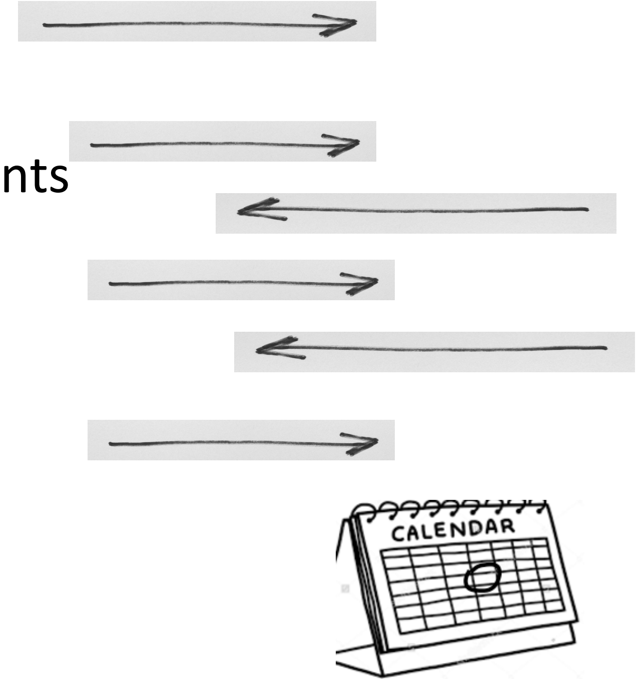
Appointment

Status = ‘proposed’

- Errors
  - e.g. No slots available

Use Case: Overview of Steps





1. Login and Patient Lookup

2.Search for or create* a Patient resource

3. Lookup appointments by date range.

4. Provide list of available appointments

5. Choose an Appointment.

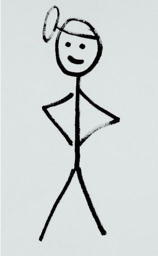
6. Accept Appointment for the chosen Slot and Patient

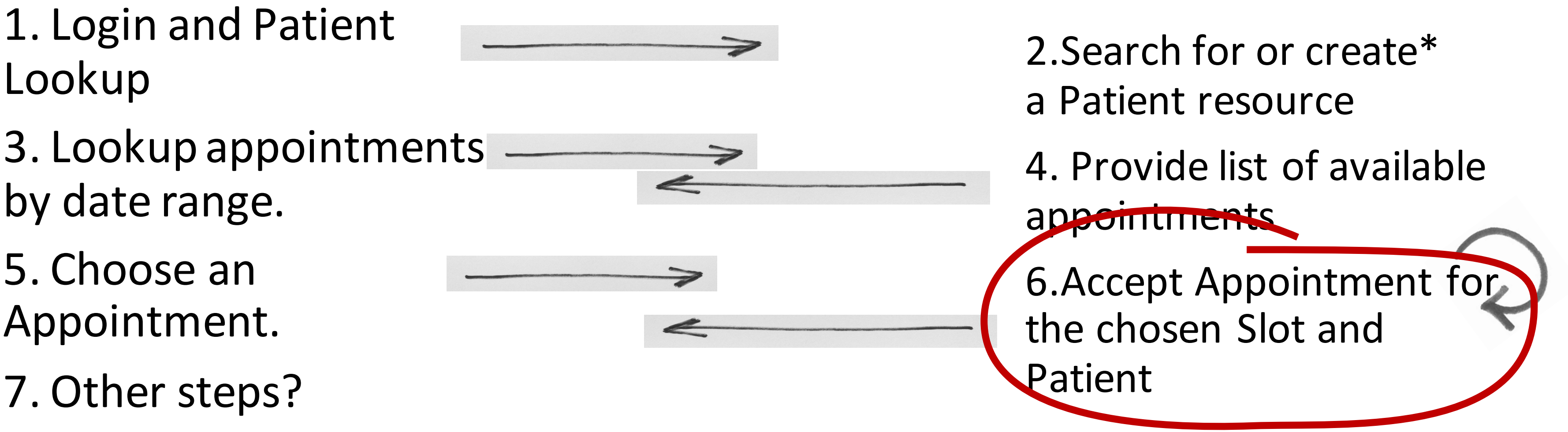
7. Other steps?

Step 5: (Client) Choose an Appointment time

- What are the operations/queries we need to request an appointment?
  - **POST** /Appointment with updated status for participants
    - See NHS example https://nhsconnect.github.io/gpconnect/appointments_use_case_book_an_appointment.html
- What data is required to choose an appointment slot
  - Chief Complaint here or in Step 2 ?
- Alternatively asks for more slots – back to step 3

Use Case: Overview of Steps





1. Login and Patient Lookup

2.Search for or create* a Patient resource

3. Lookup appointments by date range.

4. Provide list of available appointments

5. Choose an Appointment.

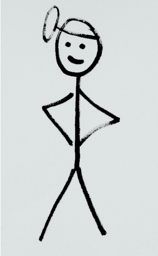
6.Accept Appointment for the chosen Slot and Patient

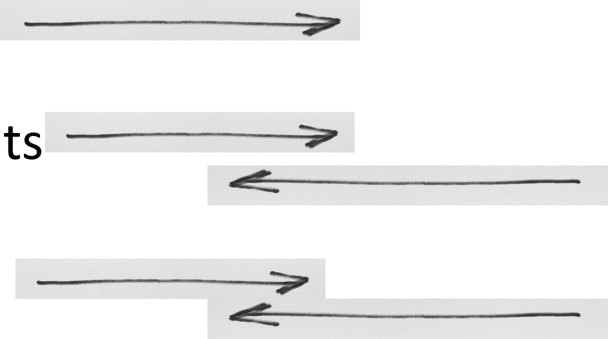
7. Other steps?

Step 6: (Server) Accepts Appointment request from Client

- Verify and Accept Appointment Request
- Stuff to return to Client
  - Appointment resource with updated status Appointment
    - See NHS example https://nhsconnect.github.io/gpconnect/appointments_use_case_book_an_appointment.html#request-operation
  - Do we use AppointmentResponse or some other workflow pattern (Tags?)
    - See NHS decision: https://nhsconnect.github.io/gpconnect/appointments_design.html#do-we-need-to-use-the-appointmentresponse-resource
- Create Patient here?
  - 'It is important that new patients are not created prior to actually attempting to book an appointment.' (Cooper)
- Error Handling
  - Slots busy, bad info etc

Use Case: Overview of Steps





1. Login and Patient Lookup

2.Search for or create* a Patient resource

3. Lookup appointments by date range.

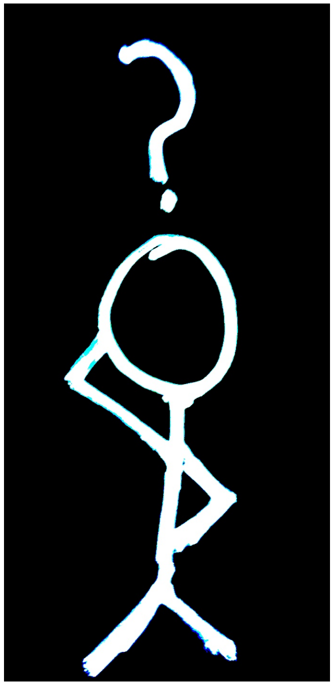
4. Provide list of available appointments

5. Choose an Appointment.

6.Confirm Appointment for the chosen Slot and Patient

7. Other steps?

Other Steps?



Key Points

- Exchange of Appointment resource only
    - References to Schedule, Slots, Provider, Patient, etc
- Creating a new Patient resource after Appointment made

Next Time: Other Use Cases

- Get all appointments for a Patient
  - Patient View
  - Provider View
- Cancelling and Rescheduling Appointments
  - Keep old cancelled and new booked Appointments