FHIR Course, Unit 3:

FHIR Advanced

Reading Material

Language: English ©2016-9 HL7 Argentina & HL7 International V4.0

Course Overview

Table of Contents

Unit Content and Learning Objectives

This Unit discusses some advanced topics: transactions, messages and documents using FHIR, the relationship between FHIR and CDA R2, and FHIR architectural approaches: what to use, when?

The power of FHIR, a power you need to unleash, is that for your specific use case and/or scenario, you can exchange a set of resources with one or more servers. FHIR does not constrain the resource type and/or quantity or timing, or even the transport used for the exchange.

You are free to use any resource, combined with any other resource, and use any of the paradigms we will explore in this unit to exchange them.

This freedom comes with a burden: you need to define which resources to exchange, how to combine them, how to relate them, and how/when are they supposed to be exchanged, and is limited by your partners in exchange: what kind of server can they deploy, and which features can they include.

In this unit we will explore the options, and some advanced techniques and good practices, and will review at least an example for each paradigm - REST (including transactions), messages, documents and operations.

FHIR Paradigms

FHIR supports four paradigms of interoperability

REST
Documents
Messages
Services

We will review and compare in this unit the four approaches, giving you tools to decide which is more suited for each project you confront. There is no ‘better’. It’s more ‘appropriate’ for each scenario, technical capabilities, and even non-technical issues, like policy or budget.

The four paradigms are REST (+ Transactions), Documents, Messages and Services (or Operations).

Two more paradigms are being explored, and they are called ‘Storage’ and ‘Bulk data’

FHIR REST + Transactions

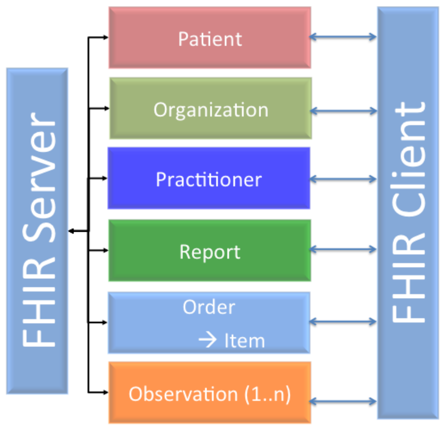
The problem

When exchanging FHIR resources with a server in real time, you are often faced with the problem of having to send a group of resources (related or not) – as opposed to only one resource.

You can choose (as a client) to use the regular REST interactions for each resource, or use FHIR Transactions.

One example of this is a lab report. A lab report in FHIR can be represented as a set of related resources (patient, requesting physician, observations, report in PDF format, order). If you need to post these resources to a server (let's assume it´s the same server), you can send the resources one at a time, as seen in Figure 1.

Figure 1. Multiple resources meaning multiple round trips to the FHIR server



The bidirectional arrows in Figure 1 mean, in the pure RESTful paradigm, assuming we don´t know the server logical id for each resource, that we need to perform a search, and then do a post (create the resource) or put (modify the resource) as needed.

Note: the actual FHIR R4 resource name for our conceptual 'Order' is 'RequestGroup', and each item is a 'ServiceRequest'.

Something like this will be going on over the wire: (questions are 'searches'/ GET to the appropriate resource) and then PUT/POST if/as needed.

Client: Do you know patient with SSN 40928383?

Server: Yes! It's my ID 338

Client: And do you know Organization 'Clinical Lab', National Org Id 333321

Server: Nope

Client: Then add it, this is the information: 'Clinical Lab, 2222 One Clinical Road, Cambridge, CA 90210'

…

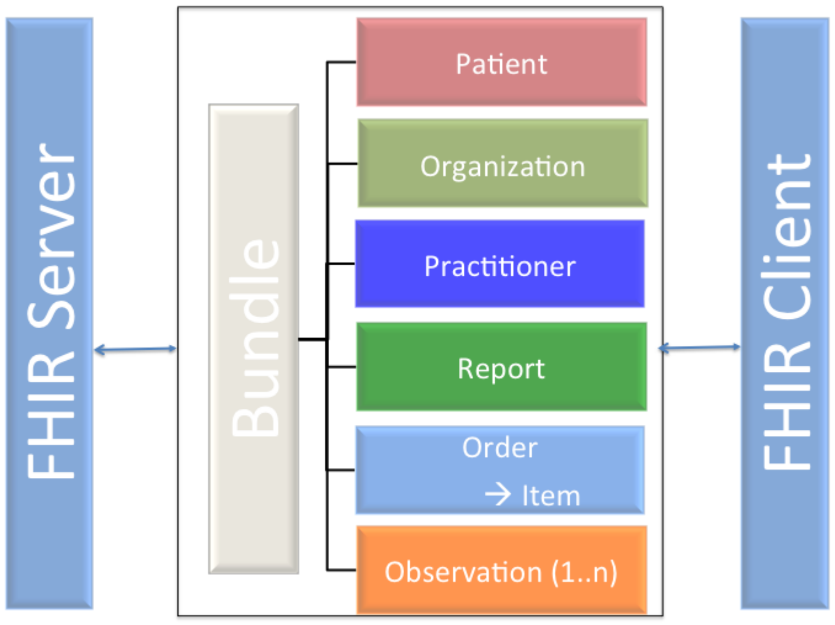
And so on for all the resources…

Imagine a Lab Report with 1 patient, 1 Organization, 1 Practitioner, 1 Order with 10 items, 1 Report with 40 observations and a PDF attached file…this means around a hundred interactions with the server (search…update or create), just to send one report.

Multiple interactions require multiple round-trips to the server and also introduce the risk of a loss of referential integrity if a later interaction fails (example: a document index entry and its related document).

The other option is to send all the resources **bundled** (*remember this magic word*) into one **transaction**: a transaction submits a set of actions to perform on a server in single HTTP request/response, as shown in Figure 2.

Figure 2. Multiple resources bundled for a Lab Report into a transaction



The responsibility for all processing is now on the server - including resolving identifiers (such as the SSN mentioned above) to local ID's which is indicated using 'Conditional Create' or 'Conditional Update'.

The client will indicate whether it wishes the processing to be either as a Batch or a Transaction.

The only difference in FHIR between a batch and a transaction is that a transaction will be successful only if ALL the operations were successful, and a batch can be partially successful: some of the operations will succeed and some will not.

You can mix and match interactions in a transaction or batch, including multiple interactions on different type of resources. Servers can even define their own operations (and we will see this in this unit)

Transactions are useful when we need to avoid multiple interactions.

To perform a transaction in a server, you just need to post to the FHIR [base] of the server, in either of the supported formats (JSON/XML)

POST [base] {?_format=[mime-type]}

Remember the “magic word”? The magic word is ‘BUNDLED’, because a transaction, for FHIR is just a specific kind of BUNDLE, with its own rules.

The content of your POST to the server is a **Bundle** resource.

For transactions, the **Bundle.type** should be valued **“transaction”**, and for batches **“batch”**. And Bundles are composed by entries. Each **entry** in your transaction bundle should contain a **resource**, and a **request**.

The resource is the **data (**the actual information about the resource you are sending to the server), the request is the **action (**what to do with the data). Each action will be processed separately, and a response will be generated for each action (Was the resource created? Which was the assigned id?)

Recipe for Bundle Transactions

So remember,

1. The transaction should be a **Bundle** of type **value="transaction"**, with one or more entries.

2. Each **entry** shall contain a **resource** and a **request** element.

3. The **request** element has two mandatory elements:

method = "POST" / "PUT" / "DELETE" / "GET"

url= The address for the method (resource), also including a search parameter if needed (GET, DELETE, Conditional inserts or updates)

In XML:

```
<Bundle>
  <type value="transaction"/>
  <entry>
        <resource>
            <Patient> <-- or any other ->
            ... Patient Resource contents
            </Patient>
        </resource>
        <request>
            <method value="POST"></method>
            <url value="patient"></url>
        </request>
    </entry>
   ... (n Entries)
</Bundle>
```

Rules for transaction processing

1. The server will either **accept all actions** and issue a 200 OK response or **reject all actions**, and return an HTTP 400 or 500 response.
2. There **is no required order for the entries** (resources) in your bundle in order to be adequately processed. The processing order is the following: **DELETE, POST, PUT, GET**. So the server will try to delete all resources you are asking it to delete, then create all new resources, then update all the resources you are asking it to update and finally process all the searches or direct GETs.
3. A transaction **may include references from one resource to another in the bundle**. Some of these references have a special format and behavior, we will discuss this under ‘Conditional Interactions’

Transaction Responses

The server will always return a Bundle (the magic word again!) to let the client know the outcomes of the transaction.

The Bundle type will be **“transaction-response”** with **as many entries as actions were requested by the client in the same order**, with the HTTP status code, location (assigned new ids for the entries) and ETag (version information for the specific resource) in the response element.

Let´s try to post a small transaction and see how the server response looks like, and what it means. With this POST, we will understand 1) The format and content of the server’s **answer** and 2) **What the server actually does** with the bundled resources.

The transaction to post is in the file **SmallTransaction.xml**.

You will find this example and others XMLs that we will mention in the reading material in the SmallExamples.zip file in this Unit 3’s material block.

Just cut & paste the contents into your REST client and POST it to our FHIR Fundamentals’ server.

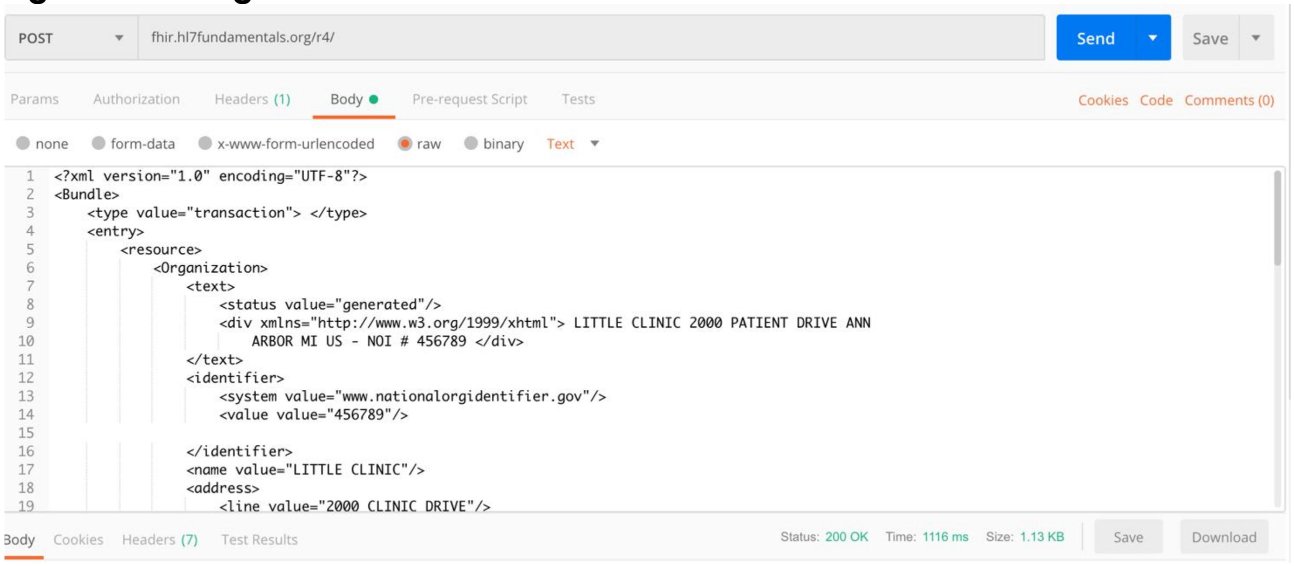
And **transactions are posted to the FHIR base address** (no specific resource path) so you need to post the contents of the file to the base address of our FHIR server.

Remember

- **a-** to set the header to ensure the server understands the content format: **Content-Type** : **application/fhir+xml**
- **b-** The resource ids for the resources you will obtain will be different that those shown in this document.

The screen will look like what we have in Figure 3.

Figure 3. Posting a transaction bundle to a FHIR Server



fhir.hl7fundamentals.org/r4/

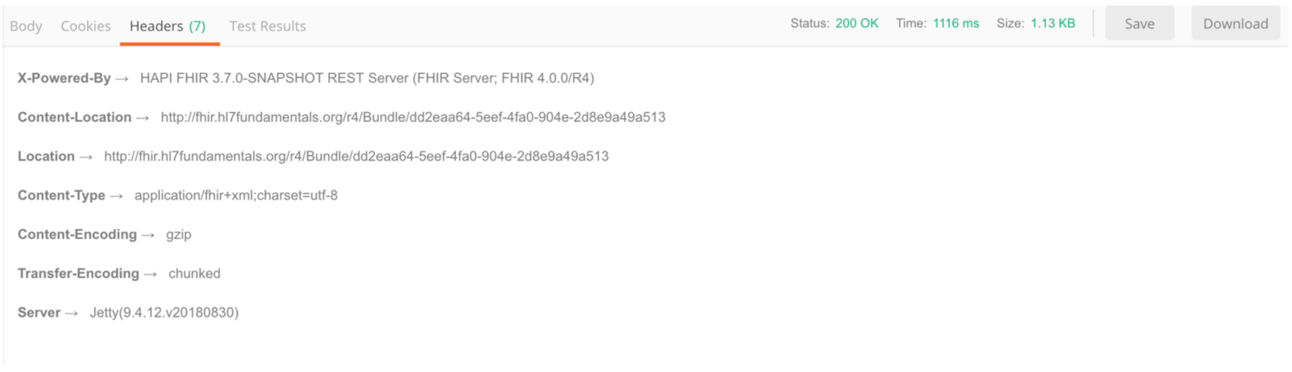
```
<?xml version="1.0" encoding="UTF-8"?>
<Bundle>
    <type value="transaction"> </type>
    <entry>
        <resource>
            <Organization>
                <text>
                    <status value="generated"/>
                    <div xmlns="http://www.w3.org/1999/xhtml"> LITTLE CLINIC 2000 PATIENT DRIVE ANN
                        ARBOR MI US - NOI # 456789 </div>
                </text>
                <identifier>
                    <system value="www.nationalorgidentifier.gov"/>
                    <value value="456789"/>

                </identifier>
                <name value="LITTLE CLINIC"/>
                <address>
                    <line value="2000 CLINIC DRIVE"/>
```

Status: 200 OK Time: 1116 ms Size: 1.13 KB

Let´s review the server's answer (in case you couldn´t try it yourself, a sample server answer is attached as **SmallTransactionAnswer.xml**)

And you can also see it here in Figures 4 and 5.

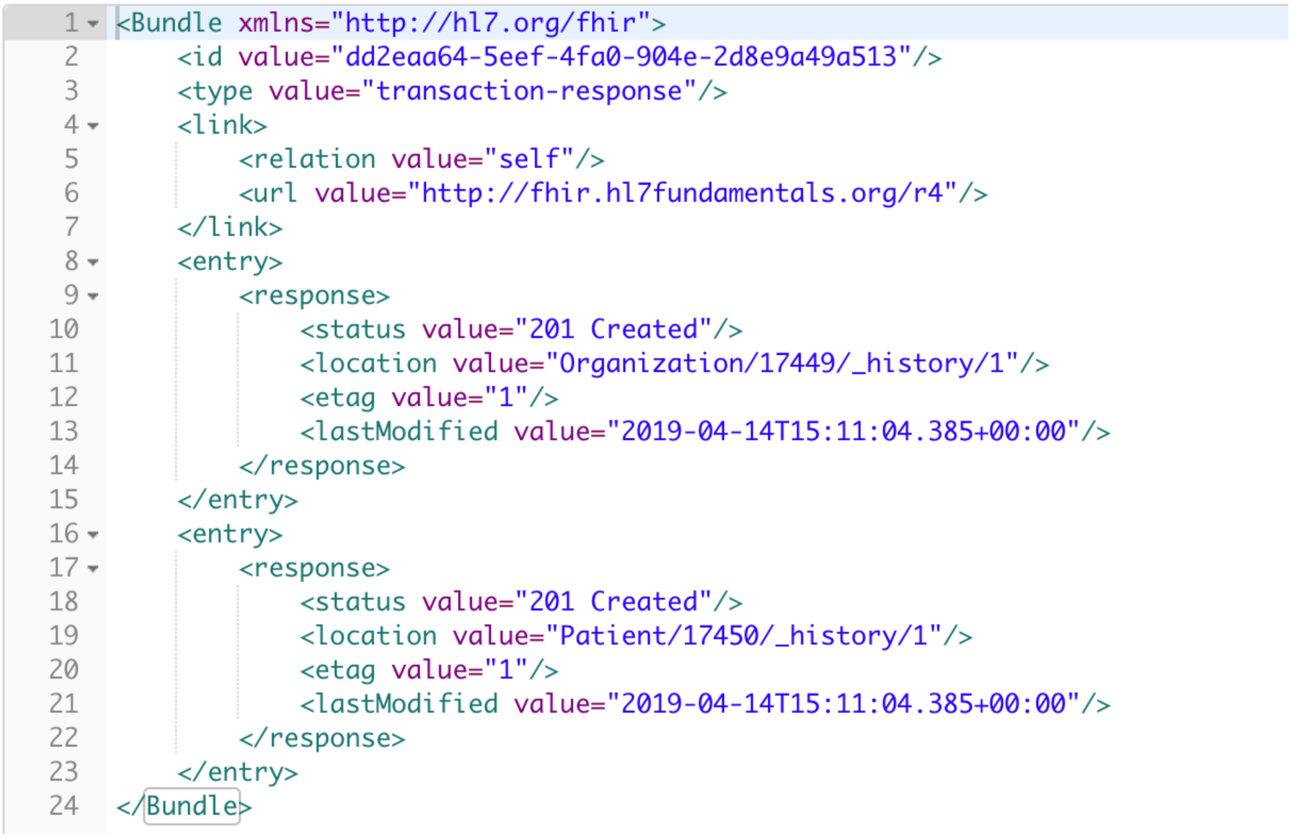
Figure 4. Server's transaction Response Header



Status: 200 OK Time: 1116 ms Size: 1.13 KB

```
X-Powered-By → HAPI FHIR 3.7.0-SNAPSHOT REST Server (FHIR Server; FHIR 4.0.0/R4)
Content-Location → http://fhir.hl7fundamentals.org/r4/Bundle/dd2eaa64-5eef-4fa0-904e-2d8e9a49a513
Location → http://fhir.hl7fundamentals.org/r4/Bundle/dd2eaa64-5eef-4fa0-904e-2d8e9a49a513
Content-Type → application/fhir+xml;charset=utf-8
Content-Encoding → gzip
Transfer-Encoding → chunked
Server → Jetty(9.4.12.v20180830)
```

Figure 5. Server's transaction Response Body



```xml
<Bundle xmlns="http://hl7.org/fhir">
    <id value="dd2eaa64-5eef-4fa0-904e-2d8e9a49a513"/>
    <type value="transaction-response"/>
    <link>
        <relation value="self"/>
        <url value="http://fhir.hl7fundamentals.org/r4"/>
    </link>
    <entry>
        <response>
            <status value="201 Created"/>
            <location value="Organization/17449/_history/1"/>
            <etag value="1"/>
            <lastModified value="2019-04-14T15:11:04.385+00:00"/>
        </response>
    </entry>
    <entry>
        <response>
            <status value="201 Created"/>
            <location value="Patient/17450/_history/1"/>
            <etag value="1"/>
            <lastModified value="2019-04-14T15:11:04.385+00:00"/>
        </response>
    </entry>
</Bundle>
```

Headers: we received an HTTP status code of **200 OK**, so all our resources/actions were processed OK (remember that transactions either succeed or fail completely)

Body: we received a "transaction-response" Bundle, with one entry for each entry in our transaction. The first entry tells us that the server created a new Organization resource ("201 Created"), with the server’s assigned id 17449, and a new Patient resource, with the server’s assigned id 17450.

Now…we want the resources to be RELATED. We need to state that this specific patient is a patient treated by this specific organization.

This can be achieved in the Patient resource by referencing to the ManagingOrganization

So this should be very easy, just add ManagingOrganization to the Patient resource.

The problem is that ManagingOrganization requires a reference, and this reference needs to point to an actual Organization/id in the same (or another server). And we will not know the resource’s id until we POST the transaction and get the results. **So…are we stuck?** No…Temporary Resource Identification to the rescue!

Temporary Resource Identification

This special identification scheme is used in transactions to insert a reference to a server assigned id we do not know at the time.

Temporary Resource Identification’s scope is ONLY inside of the transaction bundle. This means you can reuse temporary resource identifications from one transaction to another, but not in the same transaction.

Suppose you are sending in the same bundle, the PATIENT, and the REPORT, and the REPORT has a reference to the PATIENT resource. How can you make a reference to a resource whose identity you do not still know?

For instance DiagnosticReport contains a ‘Subject’ element relating the specific report to the patient. This Subject element needs a reference to the specific patient resource. Example ‘www.fhirserver.org/Patient/128’

But the transaction creator a) doesn´t know the specific ID that the server will assign to this patient b) doesn´t want to do more than one interaction with the server.

NOTE: Of course this is not an issue when the server allows the clients to create their own IDs for the resources. This is not always the case. Also NOT all FHIR servers implement temporary resource identification. But one of the two options are always available.

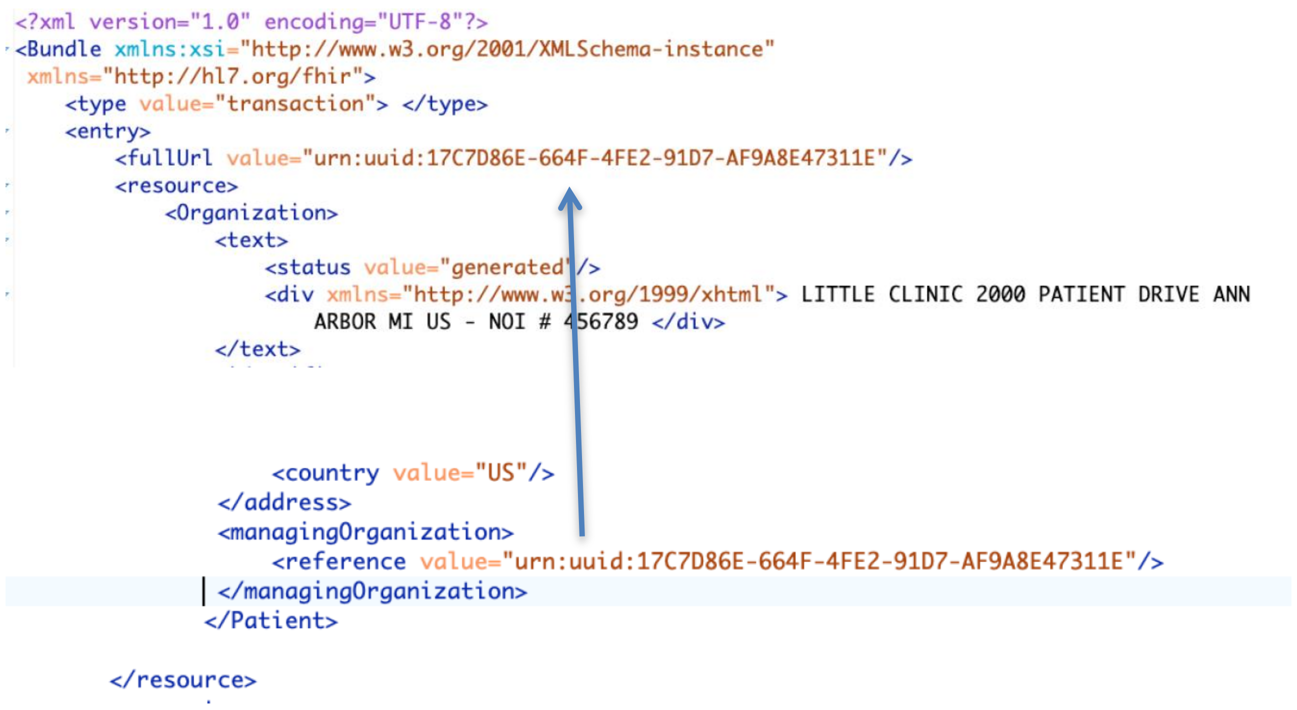
There is a way to include both the request to replace the ID with the one assigned by the server both in the resource holding the ID and the resources referencing the ID: temporary resource identification. The mechanism is simple; just include a GUID in the referenced resource entry, and include this same GUID instead of a usual resource reference, as seen in Figure 6.

```
<Bundle>
  <type value="transaction"/>
  <entry>
         <fullUrl value="urn:uuid:17C7D86E-664F-4FE2-91D7-AF9A8E47311E"/>

         <resource>
            <Patient>

      ... Patient Resource contents
            </Patient>
        </resource>
        <request>
            <method value="POST"></method>
            <url value="patient"></url>
        </request>
    </entry>
    <entry>
        <DiagnosticReport>
          … other diagnostic report information
            <subject>
               <reference value="urn:uuid:17C7D86E-664F-4FE2-91D7-AF9A8E47311E"/>
            </subject>
         … more diagnostic report information
        </DiagnosticReport>
    </entry>
   ... (n Entries)
</Bundle>
```

The server will create new Ids that are appropriate for that server, but will honor the references between resources thus preserving the relationships.

So, going back to our small example, we will just generate a temporary id element to the Organization, and reference it from the Patient resource's ManagingOrganization element, as seen in Figure 6

Figure 6. Temporary Resource Identification and Reference



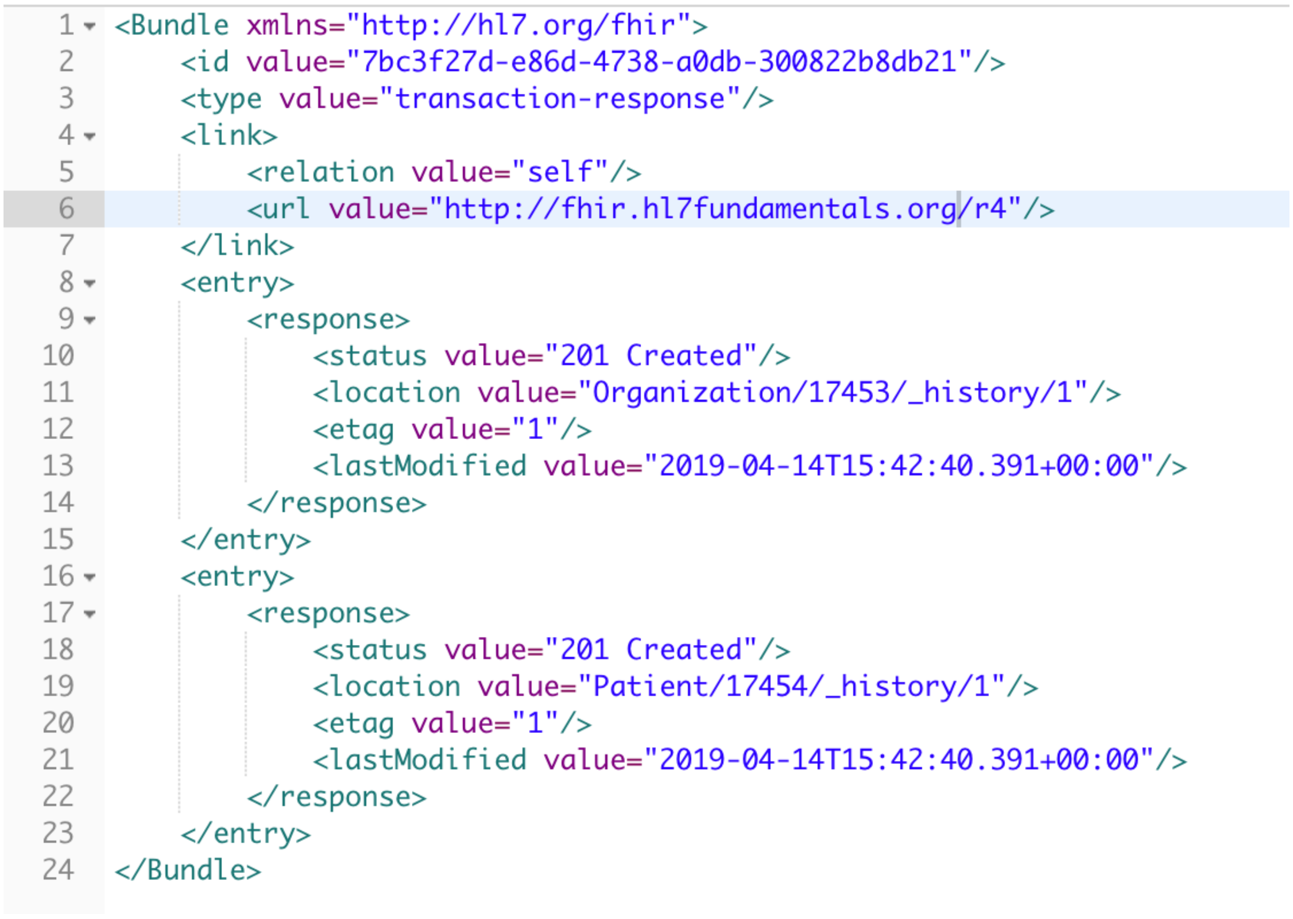
```
<?xml version="1.0" encoding="UTF-8"?>
<Bundle xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://hl7.org/fhir">
    <type value="transaction"> </type>
    <entry>
        <fullUrl value="urn:uuid:17C7D86E-664F-4FE2-91D7-AF9A8E47311E"/>
        <resource>
            <Organization>
                <text>
                    <status value="generated"/>
                    <div xmlns="http://www.w3.org/1999/xhtml"> LITTLE CLINIC 2000 PATIENT DRIVE ANN
                        ARBOR MI US - NOI # 456789 </div>
                </text>

                    <country value="US"/>
                </address>
                <managingOrganization>
                    <reference value="urn:uuid:17C7D86E-664F-4FE2-91D7-AF9A8E47311E"/>
                </managingOrganization>
               </Patient>

        </resource>
```

You can see this in the file **SmallTransactionLinked.XML**

You can even try to post this example to our server too. And closely examine the server response…now that you know how to read it.

Let’s examine the server response in Figure 7: the server didn’t use the GUID as the resource identifier: it replaces it with a new id.

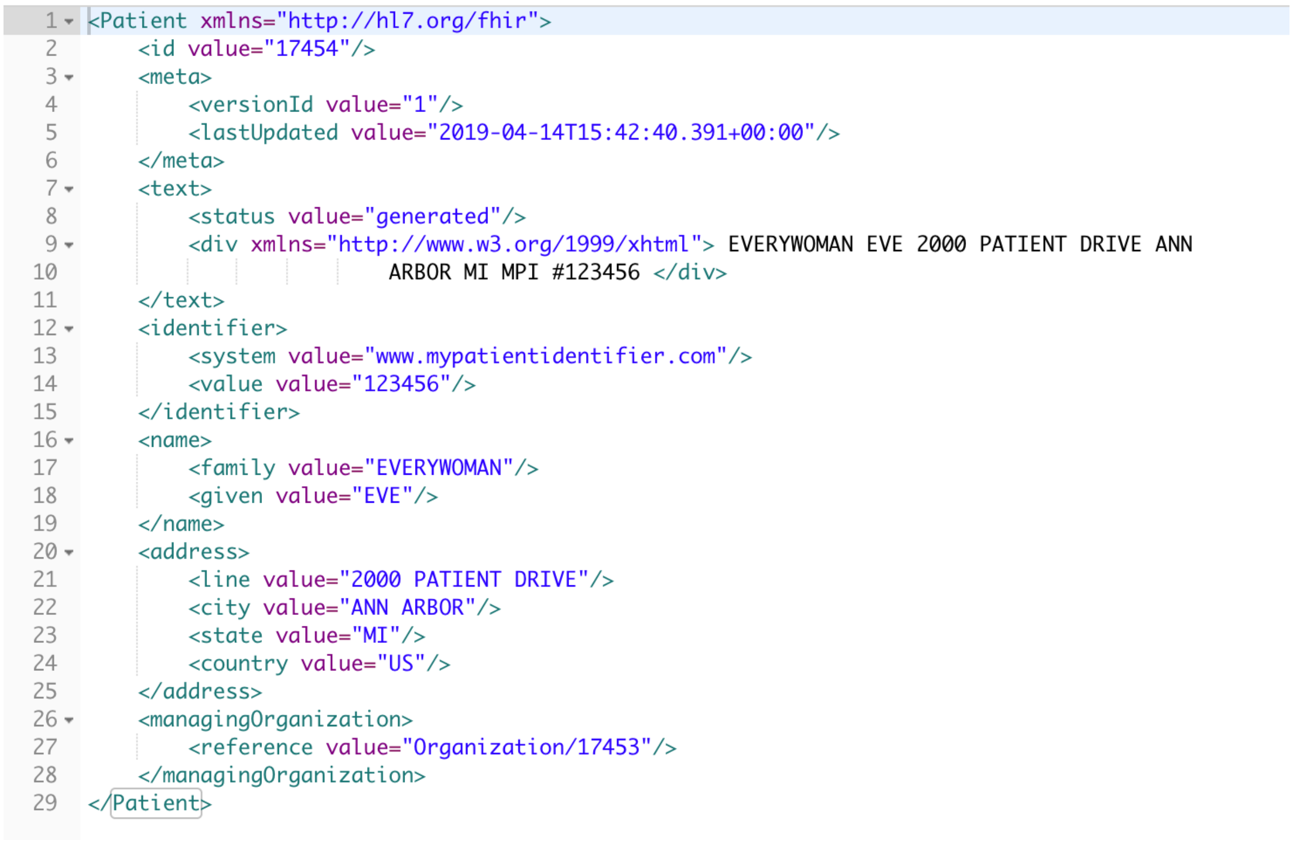
Figure 7. Server Response / Temporary Resource Identification



```xml
<Bundle xmlns="http://hl7.org/fhir">
    <id value="7bc3f27d-e86d-4738-a0db-300822b8db21"/>
    <type value="transaction-response"/>
    <link>
        <relation value="self"/>
        <url value="http://fhir.hl7fundamentals.org/r4"/>
    </link>
    <entry>
        <response>
            <status value="201 Created"/>
            <location value="Organization/17453/_history/1"/>
            <etag value="1"/>
            <lastModified value="2019-04-14T15:42:40.391+00:00"/>
        </response>
    </entry>
    <entry>
        <response>
            <status value="201 Created"/>
            <location value="Patient/17454/_history/1"/>
            <etag value="1"/>
            <lastModified value="2019-04-14T15:42:40.391+00:00"/>
        </response>
    </entry>
</Bundle>
```

It created two new resources, and if we GET the patient resource (please do it yourself), we will get the patient’s resource properly linked to its managing organization, as seen in Figure 8.

Figure 8. Proper Reference from a Temporary Resource Identification



```xml
<Patient xmlns="http://hl7.org/fhir">
    <id value="17454"/>
    <meta>
        <versionId value="1"/>
        <lastUpdated value="2019-04-14T15:42:40.391+00:00"/>
    </meta>
    <text>
        <status value="generated"/>
        <div xmlns="http://www.w3.org/1999/xhtml"> EVERYWOMAN EVE 2000 PATIENT DRIVE ANN
                        ARBOR MI MPI #123456 </div>
    </text>
    <identifier>
        <system value="www.mypatientidentifier.com"/>
        <value value="123456"/>
    </identifier>
    <name>
        <family value="EVERYWOMAN"/>
        <given value="EVE"/>
    </name>
    <address>
        <line value="2000 PATIENT DRIVE"/>
        <city value="ANN ARBOR"/>
        <state value="MI"/>
        <country value="US"/>
    </address>
    <managingOrganization>
        <reference value="Organization/17453"/>
    </managingOrganization>
</Patient>
```

But…there is still an issue. You may have noticed it. Or not.

Every time we send the same resource (same patient, same organization) to the server, it creates the resource again. So if we use transactions for our lab reports, we will have each patient related to its report, but each patient will have a new resource in the server for each report!

The problem still remains…Even using transactions and temporary identifications: "How we can ensure that we don´t end up inserting the same resource twice?" Are we stuck? Or do we need a new super-hero? Conditional interactions to the rescue!

Conditional Interactions

Conditional interactions allow a FHIR client to avoid creating an already existing resource based on some identification criteria, rather than by logical id. To accomplish this, the client issues a PUT as shown: PUT [base]/[type]/?[search parameters]

For inserts, when the server processes the interaction, it performs a search using its standard search facilities for the resource type, with the goal of resolving a single logical id for this request. The action it takes depends on how many matches are found:

No matches: The server performs a create interaction

One Match: The server performs the update against the matching resource

Multiple matches: The server returns a 412 Precondition Failed error indicating the client's criteria were not selective enough

This variant works better yet with PUT so you can conditionally update an existing resource, but searching it for a business identifier instead of the server assigned id, with this bonus feature: you can update the current resource.

For example, a client updating the status of a lab result from "preliminary" to "final" might submit the finalized result using PUT /Observation?identifier=http://my-lab-system|123

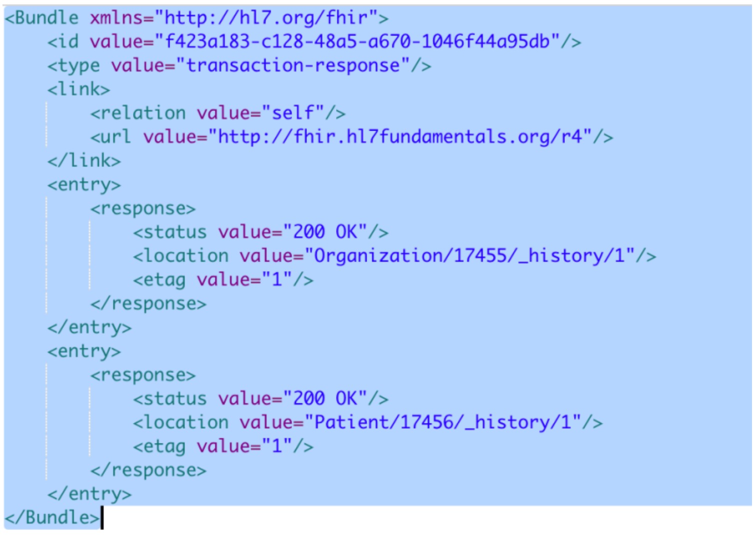
Important Note: transactions and conditional create/update/delete are complex interactions and it is not expected that every server will implement them. Servers that don't support the conditional update should return an HTTP 400 error and an operation outcome.

So…how would this apply to our small transaction…how can we ensure that the server will not create the patient and the organization every time we post the patient and/or organization resource inside of a transaction?

Let's change a little bit our small transaction file. The changes are available in the file **SmallTransactionLinkedConditional.xml**.

Note that we are not using the same identifiers for the organization and patient because we will get several matches and the server will return a 412 Error as described previously (you can try it yourself).

The result of POSTING this bundle is that the resources will be updated if they existed, and created if they don´t. And the temporary identifiers will be replaced with the actual server id for each resource. You will get a 200 OK status for each resource. You can POST this several times, and the server will never add a new resource, and the link will be maintained.



```xml
<Bundle xmlns="http://hl7.org/fhir">
    <id value="f423a183-c128-48a5-a670-1046f44a95db"/>
    <type value="transaction-response"/>
    <link>
        <relation value="self"/>
        <url value="http://fhir.hl7fundamentals.org/r4"/>
    </link>
    <entry>
        <response>
            <status value="200 OK"/>
            <location value="Organization/17455/_history/1"/>
            <etag value="1"/>
        </response>
    </entry>
    <entry>
        <response>
            <status value="200 OK"/>
            <location value="Patient/17456/_history/1"/>
            <etag value="1"/>
        </response>
    </entry>
</Bundle>
```

This concludes our discussion of FHIR transactions, but we will review more Bundle types, related to the Messaging, Document, and Service Interoperability Paradigms

FHIR Messaging

FHIR Messaging Basics

In the messaging paradigm, the interactions between server and client are like the one defined in the HL7 V2.x standard family: there is a **sender** and a **receiver**, there is a **trigger event** (something happening in the real world that triggers the creation and communication of a specific message), a **request message**, and a **response message.**

FHIR **request message** is a FHIR transaction with some special rules:

1. A **FHIR Message** is a Bundle with Bundle.type=**message**, and the first resource is always a special resource called **MessageHeader**
2. The **MessageHeader** holds the **metadata** about the message:
    a. **EventCoding** (code element): the trigger event
    b. **Sender** (the application sending the message)
    c. **Receiver** (the application supposedly receiving the message)
    d. **Message Identifier** (the unique identifier for the message, used for logging and to assign the message answer)
    e. **Date/Time (**message creation date and time)

There is **no need for the transport to be RESTful over HTTP**: you can send and receive messages using File Transfer, Folder Sharing, HTTP, MLLP, MQ Series or any other queuing application. The only requirement for the transport is ensuring the delivery of the messages and the responses.

FHIR messages are classified by its impact:

Consequence: the message represents a change that is supposed to be executed only once. This kind of message will only receive ONE response message.

Notification: The content is not supposed to be current, and it can be reprocessed. This kind of message should only receive ONE response message.

Currency: The message is a response to a query, and supposed to be current. Being a query, this kind of message can receive multiple response messages.

FHIR Event List

The Event List (list of message types and events with a list of some of the bundled resources for each one) is located at: http://www.hl7.org/fhir/valueset-message-events.html

The list right now is pretty small compared to the variety of messages and events defined in HL7 V2.x and V3, mainly due to the fact that most FHIR implementations choose the other paradigms (RESTful, transaction, documents or operations), but it's open to feedback.

These are some examples from the list

diagnosticreport-provide (notification): provide or update a diagnostic report: DiagnosticReport, Patient, Performer, Specimen, Image

MedicationAdministration-Update (consequence): update a medication administration record.

admin-notify (notification): change of an administrative resource (device, location, Patient, Practi-ioner).

Message Header Identifiers

Each message contains TWO identifiers, the Bundle.id (mandatory) and the MessageHeader.id (mandatory). These identifiers should be unique for the sender, and it's better if they are globally unique, by using an OID or UUID. Each time the same message is sent, the Bundle.id should change (the message.id will not change).

When a receiver processes the message, it will answer with a new Bundle, with a new Bundle.id, holding a new message, with a new message.id, and the original MessageHeader.id in the Mes-sageheader.response identifier

Recipe for FHIR Messages

So remember,

1. The transaction should be a **Bundle** of type **value="message"**, with at least one entry: Message-Header and one or more entries for each resource required for the message type.

2. The **Bundle.id** element is mandatory, and has to be (globally) unique.

3. The **MessageHeader.id** element is mandatory, and has to be (globally) unique. It will be used to relate the answer of the message to the message.

4. There is no timestamp element in the MessageHeader so the timestamp for the message should be included in the **Bundle.timestamp** mandatory, and it represents the message date/time

5. The **MessageHeader.source** element is mandatory, and represents the sender's **endpoint** (man-datory) and **name** (optional)

6. The **MessageHeader.destination** element is optional, and represents the receiver's **endpoint** (mandatory) and **name** (optional)

7. The **MessageHeader.focus** element is mandatory, and points to the first entry or focal entry, the first 'payload' resource for the message – remember that the MessageHeader is metadata about the message, not about the actual event that happened.

In XML:

```
<Bundle xmlns="http://hl7.org/fhir">

  <id value="a4b0eb3c-a3f3-4739-aef3-db9f718a0b15"></id>
  <type value="message"> </type>
  <timestamp value="2019-01-04T09:10:14Z"></timestamp>

    <entry>
        <resource>
            <MessageHeader>
                <id value="21448097-009c-4eec-b8d3-6aba818b3a74"></id>
                <eventCoding>
                    <code value="admin-notify"></code>
                </eventCoding>
                <destination>
                    <name>RECEIVING SYSTEM</name>
                    <endpoint value="www.receivingsystem.com"></endpoint>
                </destination>
                <source>
                    <name>SENDING SYSTEM</name>
                    <endpoint value="www.sendingsystem.com"></endpoint>
                </source>
                <focus>
                    <reference value="urn:uuid:3a78d9c4-5d87-4264-b3b4-
f4d0506a373c"></reference>
                </focus>
            </MessageHeader>
        </resource>
    </entry>
    <entry>
        <fullUrl value="urn:uuid:3a78d9c4-5d87-4264-b3b4-f4d0506a373c"></fullUrl>
        <resource>
        …

        </resource>
     </entry>



…any other entries
</Bundle>
```

Message Processing

Messages can be processed using the $process-message operation, only through POST, or directly to the FHIR base address. It accepts a message, processes it according to the definition of the event in the message header, and returns a one or more response messages. The server can store the resources included in the transaction and make them available, or choose whichever business decisions about the information contained in the resources. The server may return a 200 OK status code if the message was successfully processed.

Other error codes:

4xx errors: the error is in the message content. Do not resubmit

5xx errors: the error is in the server side. Please resubmit

Usually there should be a Message Response for each message. The message response will contain the detailed status for the message processing.

The resource used to show the status of the message processing is OperationOutcome

Recipe for FHIR Message Responses

```
<Bundle xmlns="http://hl7.org/fhir">
    <id value="0522aa28-8cf1-4b83-a452-8f7164c76d72"></id>
    <type value="message"> </type>
    <timestamp value="2016-10-04T09:10:14Z"></timestamp>


    <entry>
        <resource>
            <MessageHeader>
                <id value="0522aa28-8cf1-4b83-a452-8f7164c76d72"></id>

    <response>
                  <identifier value="efdd254b-0e09-4164-883e-35cf3871715f"/>
                   <code value="ok"/>
                   <details>
                    <reference value="OperationOutcome/94cee761-74c0-495c-82dd-
5e5686b0218e"/>
                   </details>
                </response>
             <source>
                    <name>RECEIVING SYSTEM</name>
                    <endpoint value="www.receivingsystem.com"></endpoint>
                </source>
                <destination>
                    <name>SENDING SYSTEM</name>
                    <endpoint value="www.sendingsystem.com"></endpoint>
                </destination>
                <focus>
                    <reference value="urn:uuid: e7b49eb3-64d8-40d3-a013-
c60a8df5e897"></reference>
                </data>
            </MessageHeader>
        </resource>
    </entry>
    <entry>
        <fullUrl value="urn:uuid:94cee761-74c0-495c-82dd-5e5686b0218e"></fullUrl>
        <resource>
           <OperationOutcome xmlns="http://hl7.org/fhir">
           <id value="e7b49eb3-64d8-40d3-a013-c60a8df5e897"/>
         <text>
                <status value="generated"/>
                <div xmlns="http://www.w3.org/1999/xhtml">
                <p>…detail of all errors…</p>
                </div>
           </text>
           <issue>
                <severity value="…"/>
                <code value="…"/>
                <diagnostics value="…"/>
          </issue>
          </OperationOutcome>
        </resource>
    </entry>

</Bundle>
```

Mapping from V2 to FHIR Messages

For every resource in the spec there is a mapping section where the spec is compared and mapped to the relevant V2.x segments, messages, trigger events and fields to their FHIR equivalent.

This does not mean that you are able to automatically map V2.x messages, segments, and fields to equivalent FHIR resources or messages. There is no project in HL7 to do so, but some projects had attempted at least to cover basic messaging (ADT, ORU, ORM, etc.)

These are interesting reads/views about the experience:

Dr. David Hay (New Zealand) – blog post

http://fhirblog.com/2014/10/05/mapping-hl7-version-2-to-fhir-messages/

Rene Spronk (The Netherlands) – whitepaper

http://www.ringholm.com/docs/04350_mapping_HL7v2_FHIR.htm

Simonne Heckman (Germany) - video

https://vimeo.com/128126357

FHIR Messaging Example

There is a small FHIR message example at the file **SmallMessage.xml,** holding information about a patient using the **admin-notify** event.

If you are trying this file, please POST to a test FHIR server

The Unloved Paradigm

Only a few FHIR test servers support the messaging paradigm yet, because of market preference for REST & Document.
You can check this out in this video from Rene Spronk, from The Netherlands, in his DevDays 2018 Presentation/Video “Messaging, The Unloved Paradigm”, here:
https://www.youtube.com/watch?v=_ov5bYIRTpg

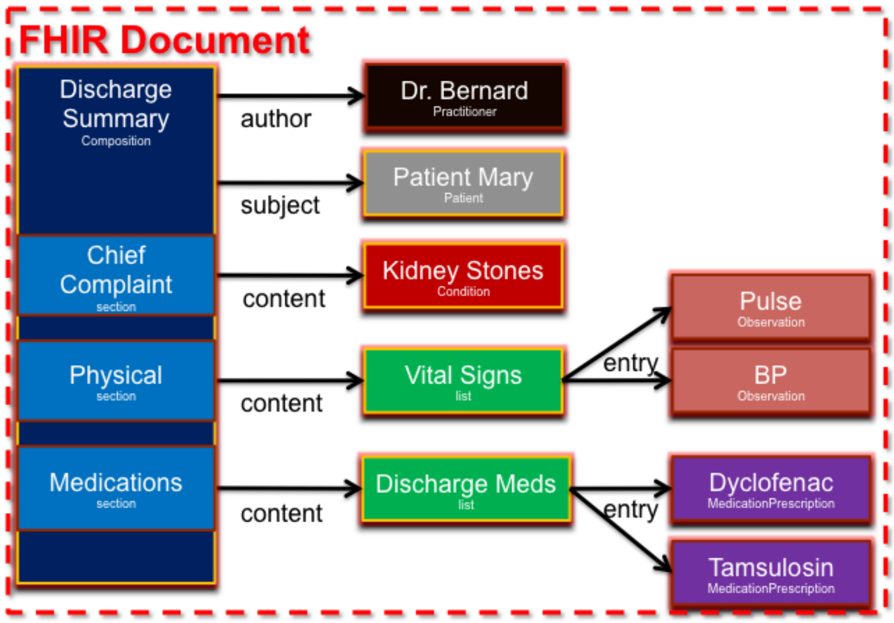
FHIR Documents

Clinical Documents Characteristics

There are six characteristics defined for Clinical Documents by HL7 CDA R2: **Persistence** , **Stewardship**, **Potential for authentication**, **Context**, **Wholeness** and **Human readability**

Persistence: The document should be stored and remain accessible in its original form (as authenticated) for the retention period. FHIR does not require resources to be stored as sent, but this is generally an expectation for documents.
Potential for Authentication: The document can be signed. In FHIR, authentication is of the whole content, and of a specific rendered view of content. And it gives better tools than CDA to specify digital signatures and provenance for the different resources.
Context: In CDA R2, it is given by the document header. In FHIR, context exists independently in each resource, however documents indicate a set of information intended to be consumed together (human context vs. technical context). We will see that when we discuss document structure: this is achieved by the use of the Composition resource.
Wholeness: authentication and context applies to the whole content of the document, not to specific parts.
Human readability: Human readability is optional for resources, but mandatory for documents, including specific rendering rules.
FHIR documents are more flexible than the CDA R2 document structure, they are a generic structure, can be about any subject (CDA R2 documents target is always the patient): trial designs, device reports, public health cases, etc.
Finally FHIR documents can be processed in ways that do not meet the 6 characteristics of clinical documents defined by CDA (for instance, you can just parse the included resources and not store the whole document as sent by the document creator).
This doesn´t preclude any specific project to be based on FHIR documents while preserving the 6 clinical document characteristics.

FIGURE 9 – Conceptual view of a FHIR Document


FHIR Document
Discharge Summary
Composition
Chief Complaint
section
Physical
section
Medications
section
author
subject
content
content
content
Dr. Bernard
Practitioner
Patient Mary
Patient
Kidney Stones
Condition
Vital Signs
list
Discharge Meds
list
entry
entry
Pulse
Observation
BP
Observation
Dyclofenac
MedicationPrescription
Tamsulosin
MedicationPrescription

FHIR Document Structure

The content of your POST to the server for a FHIR document is a **Bundle** resource.

The **Bundle.type** should be valued **"document"**. The document will include a list of entries. For documents, the first entry shall be a Composition, and some other resources are mandatory as well, either included in the Bundle (recommended for wholeness) or externally referenced from the Composition.

A Document, no matter how complex and nested, is flattened to a list of entries, the Document's header or Composition being the first. The document header (and any other the other resources) includes references to each other using normal references to reflect the document's nesting. There may be a digital signature (on the whole Bundle) to attest to the content of the document.

Recipe for FHIR Documents

So remember,
1. The transaction should be a **Bundle** of type **value=**"**document**", with at least one entry: Composition and one or more entries for each internal reference (some of them are required). This is a MINIMUM recipe, so we will list only the MANDATORY elements and references, although for your specific scenario you may need to include other elements as references as well.
Composition Attributes
2. The **Bundle.id** : bundle identification, has to be (globally) unique.
3. The **Composition.id :** has to be (globally) unique. It is the identifier of the document.
4. The **Composition.date :** the date/time when the document was created.
5. The **Composition.title :** human readable title for the document
6. The **Composition.code :** Document Type, from http://www.hl7.org/fhir/valueset-doc-typecodes.html (which refers to all LOINC document class codes)
7. The **Composition.status :** current state of the document. Choices: preliminary, final, amended, entered-in-error
Composition References (and entries if internally referenced)
8. Composition.subject: Who and/or what the composition is about (usually the patient)
9. Composition.author: Practitioner, device, patient or related person authoring the document.
10. Composition.attester.party: Patient or Practitioner attesting the accuracy of the composition.
11. Composition.custodian: Organization, which maintains the composition
12. Composition.event.detail: The event being documented.
13. Composition.encounter: Context of the composition.
14. Composition.section.entry: the actual clinical content
Due to the size of the document, we cannot include it in this module, but you can refer to the example to see the structure.

FHIR Document Example

The file **SmallDocument.xml** is a minimum semantically valid example of a FHIR document.

Again, if you want to test this document against our FHIR server, you will need to POST it to the Bundle endpoint, since not all servers allow processing of the elements of the composition.

http://fhirserver.hl7fundamentals.org/fhir/Bundle

FHIR Document Presentation

The document should be presented in this order: Composition Resource Narrative, Subject Resource Narrative, section.text Narratives, using the basic style for narratives, explained here: http://www.hl7.org/fhir/narrative.html#css

In addition, a document can reference or contain one or more stylesheets that contains additional styles that apply to the collated narrative.

This is done by including the stylesheet reference in the bundle. The uri can be a reference to an existing file in a web server, or a reference to a resource included in the same bundle.

```
<Bundle xmlns="http://hl7.org/fhir">
<link>
 <relation value="stylesheet"/>
<url value="[uri]"/>
…
</Bundle>
```

FHIR Document Transport

There are several options for sending documents to a FHIR server, through different endpoints.

None of this is mandatory. Documents can be exchanged in any way you can devise (File System, FTP, USB sticks, etc.)

***baseurl/*Bundle:** This works like a normal end-point for managing a type of resource, but it works with whole document bundles - i.e. a read operation returns a bundle, an update gets a bundle and a search returns a bundle of bundles

***baseurl/*Binary:** Stores the entire document as a sequence of bytes and return exactly that sequence when requested. Usually this would be associated with a Document Reference so that applications can find and process the document.

***baseurl* (Transaction):** Ignore the fact that the bundle is a document and process all the resources as individual resources. Clients SHOULD not expect that a server will be able to reassemble the document exactly. (Even if the server can reassemble the document, the result cannot be expected to be in the same order, etc. A document signature will very likely be invalid.)

FHIR Document Architecture Considerations

Document Persistence: There is no defined order to entries in a document other than the first, so round-tripping between persistence and XML/ JSON (or RDF) may produce different orders. Even with canonicalization, signatures may not hold if document elements are stored in their constituent parts. So, if you care about signature, store documents as a **binary**

Digital Signature: Any bundle (including Documents) can have an XML digital signature. Signatures aren't required to use FHIR documents – other means of verifying integrity, etc. are fine as well. One or more resources in a document can also be signed by including a Provenance resource.

Tags and Security: Tags (and 'security' metadata) can appear on both resource entries and at the bundle level. If using security tags, must establish precedence. Safest is 'most restrictive applies', but business rules may differ. Tags inside document entries can't be changed without breaking the document signature

FHIR Documents and CDA R2

CDA R2 Related FHIR Projects in HL7 International

These are the two main CDA R2-FHIR related projects in HL7 International

1) **CDA ON FHIR:** Project to define what "clinical documents" should look like in FHIR. Will include: Profile of the Composition resource (we will see Profiles during the next week), Mapping of CDA header to Composition and a high-level mapping of root entries to possible corresponding resources in FHIR.

https://www.hl7.org/fhir/cda-intro.html

2) **C-CDA ON FHIR:** Project to map between CCDA and a FHIR-equivalent, including profiles. Long term vision is a new version of CCDA with exact FHIR equivalents balloted in parallel. Project URL:

http://wiki.hl7.org/index.php?title=C-CDA_on_FHIR

3) For a full comparison of CDA R2 and FHIR, please see: https://www.hl7.org/fhir/comparison-cda.html

FHIR Operations

Extended operations in the RestFUL API

When The RESTful API defined by FHIR as a set of common interactions (Create/Read/Update/Delete) on the set of FHIR identified resources is not enough, FHIR allows for the definition of operations using an RPC-like paradigm:

1. Named operations are performed with inputs and outputs (Execute).
2. Operations are used
    a. when the server needs to play an active role in formulating the content of the response
    b. when the intended purpose is to cause side effects such as the modification of existing resources, or creation of new resources.
3. The FHIR specification describes a lightweight operation framework that seamlessly extends the RESTful API, as follows:
    1. Each operation has a name
    2. Each operation has a list of 'in' and 'out' parameters
    3. Parameters are either resources, data types or search parameters
    4. Operations are subject to the same security constraints and requirements as the RESTful API
    5. The URIs for the operation end-points are based on the existing RESTful API address scheme
    6. Operations may make use of the existing repository of resources in their definitions
    7. Operations may be performed on a specific resource, a resource type, or a whole system

Executing Operations

Operations are a POST to a FHIR endpoint, where the name of the operations is prefixed by a "dollar sign" ('$') character. For example: POST http://fhir.**someserver**.org/fhir/Patient/1/$everything

Operations can be invoked on four types of FHIR endpoints:
1. **The "base" FHIR service endpoint** (e.g. http://fhir.someserver.org/fhir): Operate on the full scale of the server. For example, "return me all extensions known by this server"
2. **A Resource type** (e.g. http://fhir.someserver.org/fhir/Patient): Operate across all instances of a given resource type
3**. A Resource instance** (e.g. http://fhir.someserver.org/fhir/Patient/1): Involve only a single instance of a Resource. Example: the $everything operation.
4. **A specific version of a resource instance** (http://fhir.someserver.org/fhir/Patient/1/_history/4): Only to allow manipulation of profile and tag metadata of past versions

Operation Parameters

The body of the invocation contains a special infrastructure resource called **Parameters,** which represents a collection of named parameters as <key , value> pairs, where the value may be **any primitive or complex datatype or even a full Resource.** It may also include strings formatted as search parameter types.

Operation Response

Upon completion, the operation returns another Parameters resource, containing one or more output parameters. This means that a FHIR operation can take any parameter in and return a set of result parameters out. Both the body of the POST and the returned result are always a Resource. If an operation succeeds, an HTTP Status code of 200 OK is returned. An HTTP status code of 4xx or 5xx indicates an error, and an OperationOutcome may be returned.

FHIR Defined Operations

The FHIR specification defines several operations, which can be found at http://www.hl7.org/fhir/operations.html#defined

Two important operations we will use are $everything (it applies to a specific patient) and $vali-date (it validates a resource against the resource definition and/or a specific profile. We will see profiles in the next unit)

You can try the $everything operation against any of the patient resources you've uploaded to our FHIR server, by performing a get like this, with [id] being the server assigned id for the patient:
http://fhirserver.hl7fundamentals.org/fhir/Patient/[id]/$everything

It will return a Bundle with all the current resources stored in the server pertaining to the patient.

Implementation Defined Operations

Implementations are able to define their own operations in addition to those defined by the FHIR specification. This is part of FHIR profiling and we will explore this in next unit.

When to use...? When to avoid?

When to use RESTful Interactions / Transactions

Simple, **out-of-the-box interoperability**
Leverage HTTP – protocol that drives the web
Use only Pre-defined operations: Create, Read, Update, Delete - Also: History, Read Version, Search, Updates, Validate, Conformance & Batch
Client-server architectures, with **client-driven orchestration** (Mobile, PHR, Registries)

When to avoid RESTful Interactions / Transactions

Complex or server-driven orchestration
Unit of work is not the resource
There is no “natural” server

When to use FHIR Messaging

Request/Response workflow
Need **asynchronous mode**
More complex behaviors than CRUD
CRUD operations **on transports other than HTTP**

When to avoid FHIR Messaging

When there’s another standard way to do it

When to use FHIR Documents

Focus is on **persistence**
No workflow involved, other than post/retrieve document
Need tight rules over **authenticated content**
Want to communicate **multiple resources** with **control over how data is presented**
Data spans multiple resources

When to avoid FHIR Documents

Need for workflow
Request/response, **decision support**
Data is dynamic, i.e.: want view of data now, not at time of authorship
Multiple contributors over time
Resources need to be **accessed independently**

When to use FHIR Operations or Services

Operations **other than CRUD without messaging overhead**

More **complex workflow than request/response**

Mix **documents & behavior**

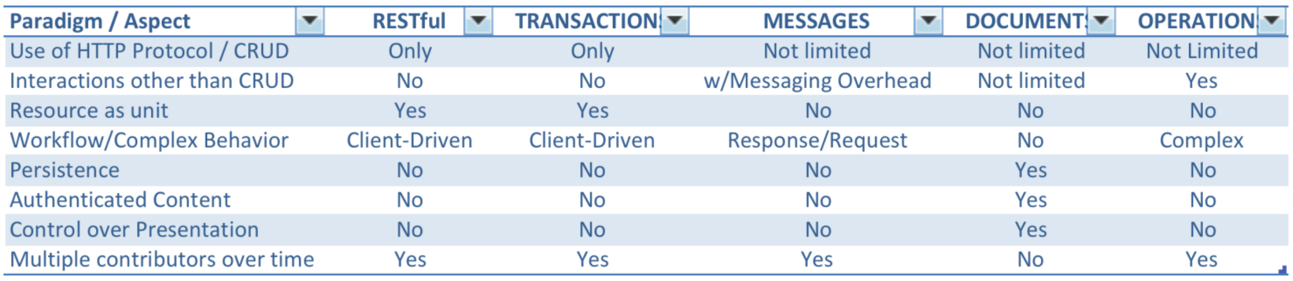
When to avoid FHIR Operations or Services

Need **tight control over data display**

Want to **avoid pre-negotiation of behavior**

Overall Guidance

There are no absolutes. This section is for guidance only. Your choice maybe influenced by preference and legacy, and maybe other non-technical considerations. Do not limit to only one choice: mix and match to meet your scenarios, use cases, and communication partners.



Paradigm / Aspect	RESTful	TRANSACTION	MESSAGES	DOCUMENT	OPERATION
Use of HTTP Protocol / CRUD	Only	Only	Not limited	Not limited	Not Limited
Interactions other than CRUD	No	No	w/Messaging Overhead	Not limited	Yes
Resource as unit	Yes	Yes	No	No	No
Workflow/Complex Behavior	Client-Driven	Client-Driven	Response/Request	No	Complex
Persistence	No	No	No	Yes	No
Authenticated Content	No	No	No	Yes	No
Control over Presentation	No	No	No	Yes	No
Multiple contributors over time	Yes	Yes	Yes	No	Yes

This week’s assignment

We will try to create a transaction for a [fake] wearable device and send/retrieve vital signs information to our FHIR repository. Go, Now! This is the FHIR FUN part!

Unit Summary and Conclusion

FHIR will not do anything for you, but enables construction of solid and up-to-date interoperable systems. This advanced unit gave you some hints on FHIR paradigms, and when / how to use them.

We will try to summarize in a page what you've learned, just in case someone asks you in the elevator what you learned this week.

REST-ful: CRUD Operations only. Exchanging information one resource at a time (get to see if the resource exists in the server, PUT/POST to update if/as needed)

Transactions: Bundle/type=**transaction**. Mix and match any kind of resources. Fail or Succeed as a whole. Needs some help when server ids are not known (temp resource id, conditional interactions).

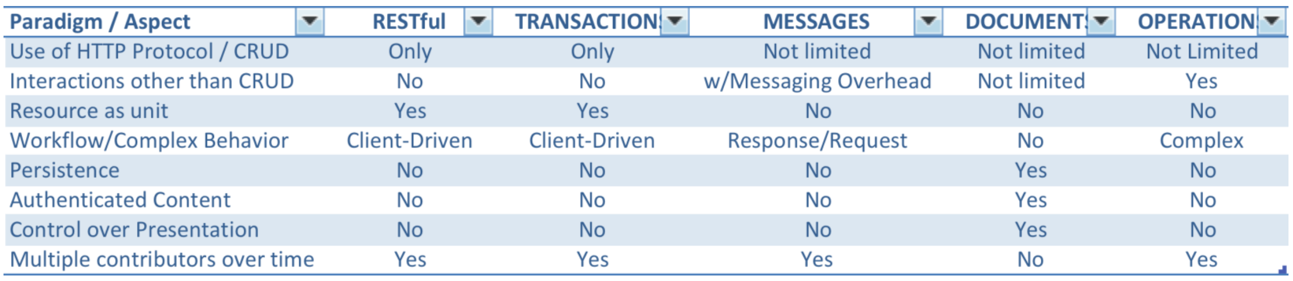
Batch: Bundle/type=**batch**. Same than transactions, but can partially succeed. Resources are not related

Documents: Bundle/type=**document**. First entry is Composition, with metadata about the context of the included clinical entries. Some attributes and references/entries are mandatory.

Message: Bundle/type=**message**. First entry is MessageHeader, with metadata about the message, sender and receiver. Based on the trigger event message paradigm. Events defined by HL7.

Operations: extending behavior of servers for specific, complex operations. Begin with the $ sign. May apply to the whole server, specific resource types, or specific resource instances.

When to use Guidance



Paradigm / Aspect	RESTful	TRANSACTION	MESSAGES	DOCUMENT	OPERATION
Use of HTTP Protocol / CRUD	Only	Only	Not limited	Not limited	Not Limited
Interactions other than CRUD	No	No	w/Messaging Overhead	Not limited	Yes
Resource as unit	Yes	Yes	No	No	No
Workflow/Complex Behavior	Client-Driven	Client-Driven	Response/Request	No	Complex
Persistence	No	No	No	Yes	No
Authenticated Content	No	No	No	Yes	No
Control over Presentation	No	No	No	Yes	No
Multiple contributors over time	Yes	Yes	Yes	No	Yes

Additional Reading Material

Information about FHIR

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

- The specification itself is available on-line at www.hl7.org/FHIR. It is fully hyperlinked & very easy to follow. It is highly recommended that you have access to the specification as you are reading this module, as there are many references to it - particularly for some of the details of the more complex aspects of FHIR.
- All the subjects detailed in this unit are deeply documented in the FHIR specification. If you can´t explain something, this should be the first source-of-truth.
- 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 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 member.