EHRServer v0.6 guide

The open source, service-oriented, openEHR data repository

Index

Introduction

EHRServer is a generic, minimal, open source, standard-based, service-oriented, openEHR clinical data storage. It provides an administration Web GUI and a REST API for committing and querying openEHR clinical documents and data values.

EHRServer was designed and developed by Eng. Pablo Pazos Gutiérrez[1] at CaboLabs Healthcare Informatics.

I'm interested… Can I try openEHR?

Yes. You have two options:

1. Test it on our staging server online
2. Install it and test it in your machine

[1] https://www.linkedin.com/in/pablopazosgutierrez

How to use EHRServer from our staging server (TL;DR guide)

You are busy, we know it. This is the shortest explanation of what you can do by using the EHRServer Web GUI:

1) You need an account

Open the Web App: https://cabolabs-ehrserver.rhcloud.com/ehr

Go to "Create an Account". You will receive an email with an organization number and a link to reset your password. After this you are all set to use EHRServer.

2) Login and create some patients

When you created your account, you also created an organization. The organization can be a clinic, hospital, etc. You need to create some patients, and those patients will be associated to your organization.

Login into the EHRServer, using your username, password and organization number.

Go to People > New Person, fill the form and click on Create.

Go to People, and on the patient's row, click on Create EHR. Now your patient has an empty EHR.

So far so good, now you need to add some data to your patient's EHR.

3) Commit data to an EHR

The easiest way of committing data to an EHR in EHRServer, is to use our testing app: EHRCommitter.

Go to: http://committer-ehrserver.rhcloud.com/committer

Login using your EHRServer credentials: username, password and organization number.

Select a clinical document from the list.

Select a patient from the list (your patient should be there!).

Fill the form or keep it as it is (with autogenerated testing data), and click on Save. This will send the data to EHRServer (invokes the commit service from the REST API). You can send as many documents you want.

4) I want to query data!

Yes, once you get your data in, you don’t want it to be stuck in the EHR. You want to query data and use it in different ways.

You can query for clinical documents or for data values.

Go to Queries > New Query.

Assign a name and select a type: “composition” means you want to query documents, “datavalue” means you want to query data.

4.1) Querying documents

1) Assign a name and select type “composition”.

2) Select a concept to define the criteria e.g. “Blood Pressure”

3) Select a data point to define the criteria e.g. Diastolic

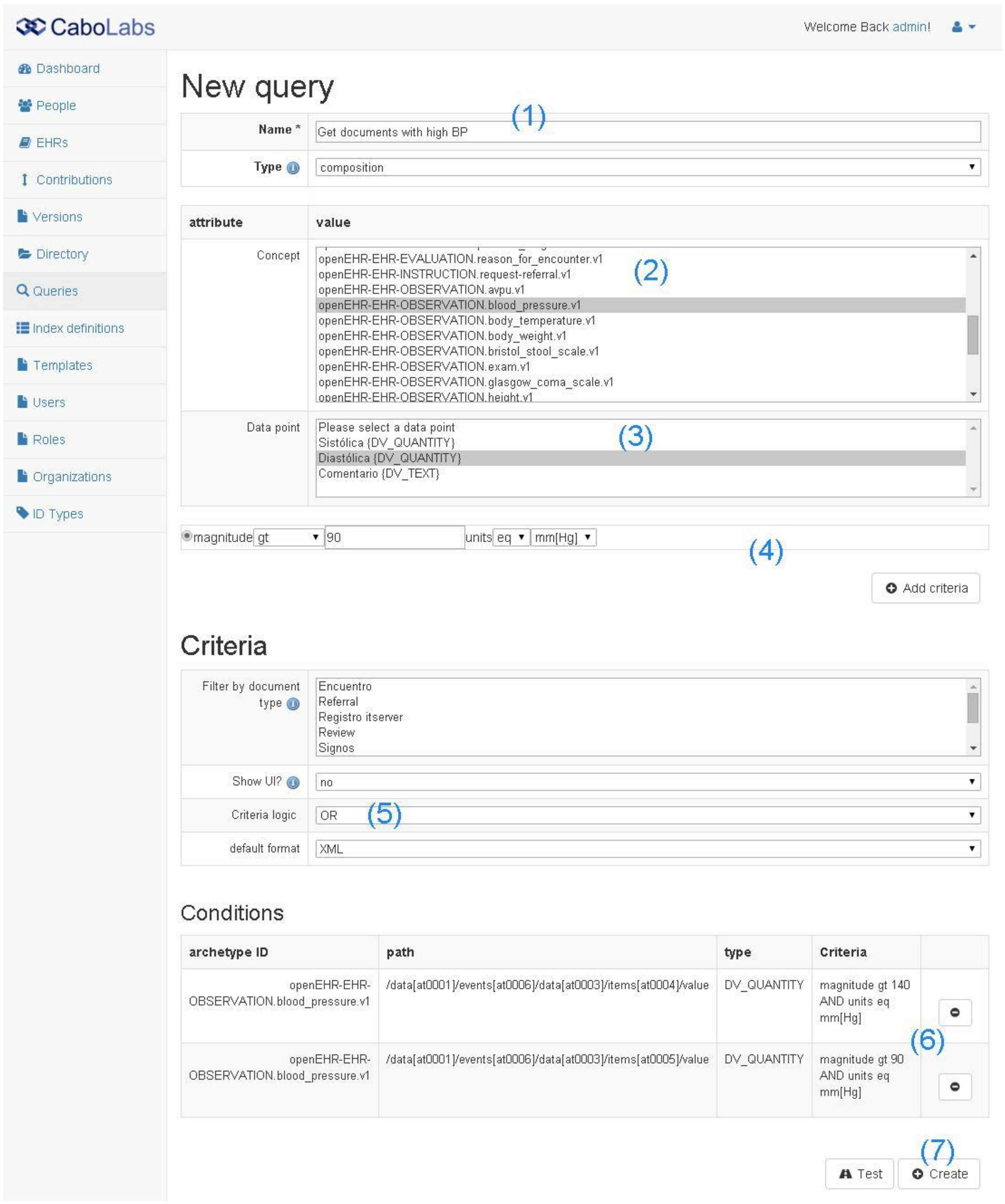
4) Define the criteria e.g. magnitude > 90 and units = mm[Hg]

You can define as more conditions as part of your query criteria. We also defined a condition over Systolic to have magnitude > 140 and units = mm[Hg]

5) Select the criteria logic “and” or “or”, this will define how to interpret all your conditions. We selected “or”.

6) Our final criteria match all the clinical documents that have a record of high blood pressure. You can create other queries to match documents based on your requirements.

7) Click on “Create” to save your query.


CaboLabs

Welcome Back admin!

Dashboard
People
EHRs
Contributions
Versions
Directory
Queries
Index definitions
Templates
Users
Roles
Organizations
ID Types

New query

Name *	Get documents with high BP (1)
Type	composition

attribute	value
Concept	openEHR-EHR-EVALUATION.reason_for_encounter.v1 openEHR-EHR-INSTRUCTION.request-referral.v1 openEHR-EHR-OBSERVATION.avpu.v1 openEHR-EHR-OBSERVATION.blood_pressure.v1 openEHR-EHR-OBSERVATION.body_temperature.v1 openEHR-EHR-OBSERVATION.body_weight.v1 openEHR-EHR-OBSERVATION.bristol_stool_scale.v1 openEHR-EHR-OBSERVATION.exam.v1 openEHR-EHR-OBSERVATION.glasgow_coma_scale.v1 openEHR-EHR-OBSERVATION.height.v1 (2)

Data point	Please select a data point
	Sistólica {DV_QUANTITY}
	Diastólica {DV_QUANTITY}
	Comentario {DV_TEXT} (3)

magnitude gt 90 units eq mm[Hg] (4)

Add criteria

Criteria

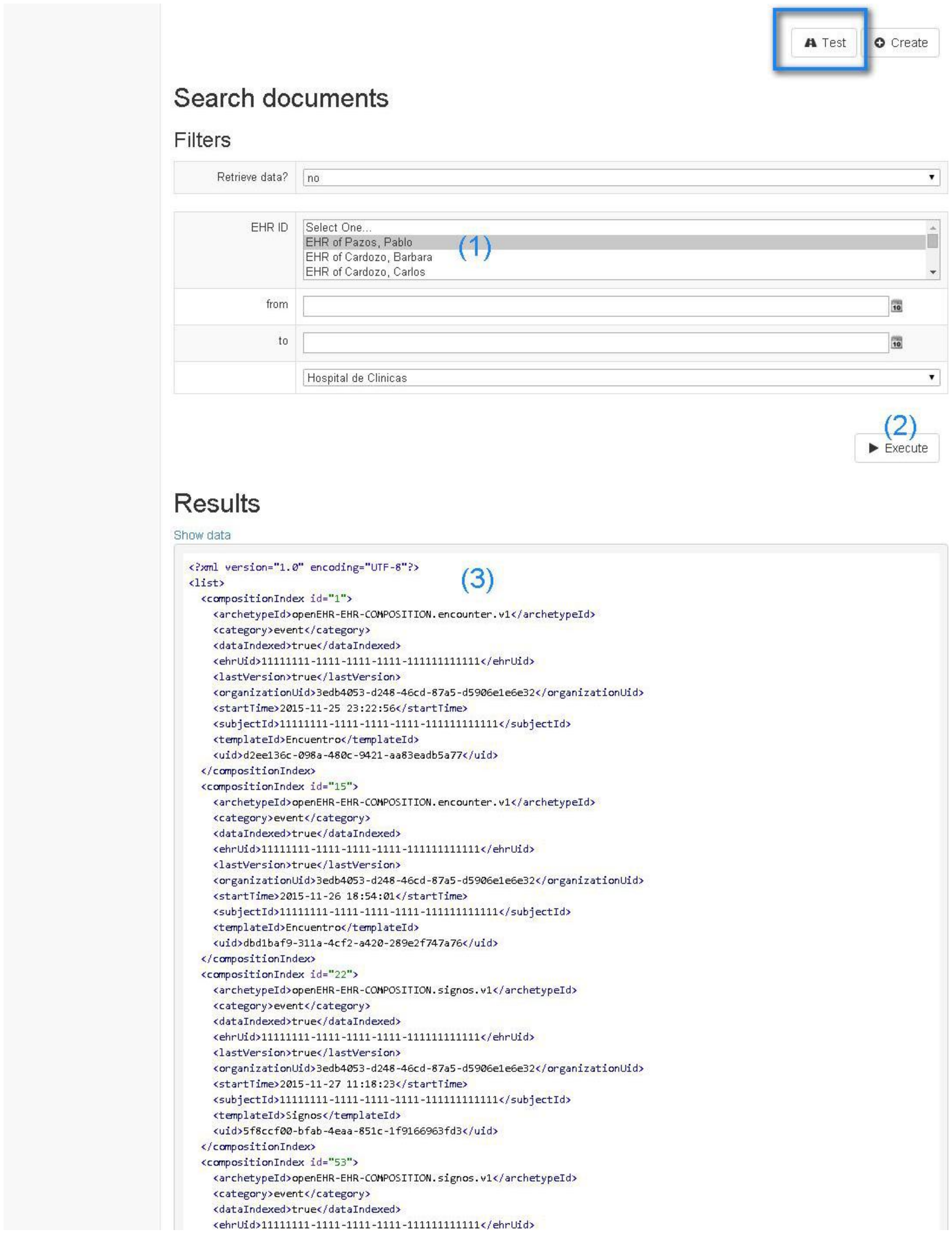
Filter by document type	Encuentro Referral Registro itserver Review Signos
Show UI?	no
Criteria logic	OR (5)
default format	XML

Conditions

archetype ID	path	type	Criteria	
openEHR-EHR-OBSERVATION.blood_pressure.v1	/data[at0001]/events[at0006]/data[at0003]/items[at0004]/value	DV_QUANTITY	magnitude gt 140 AND units eq mm[Hg]	
openEHR-EHR-OBSERVATION.blood_pressure.v1	/data[at0001]/events[at0006]/data[at0003]/items[at0005]/value	DV_QUANTITY	magnitude gt 90 AND units eq mm[Hg]	(6)

Test (7) Create

From the query creation screen you can also test your query to see if it was correctly defined and you get the data you expect. Click on “Test” and 1) Select an EHR of a patient (if no EHR is selected, the query will be executed over all the available EHRs, 2) then click on “Execute” and 3) Review the results (results are indexes, i.e. pointers to clinical docs, if you want the data, select “Retrieve data”).



Search documents

Filters

Retrieve data? no

EHR ID
Select One...
EHR of Pazos, Pablo
EHR of Cardozo, Barbara
EHR of Cardozo, Carlos

from

to

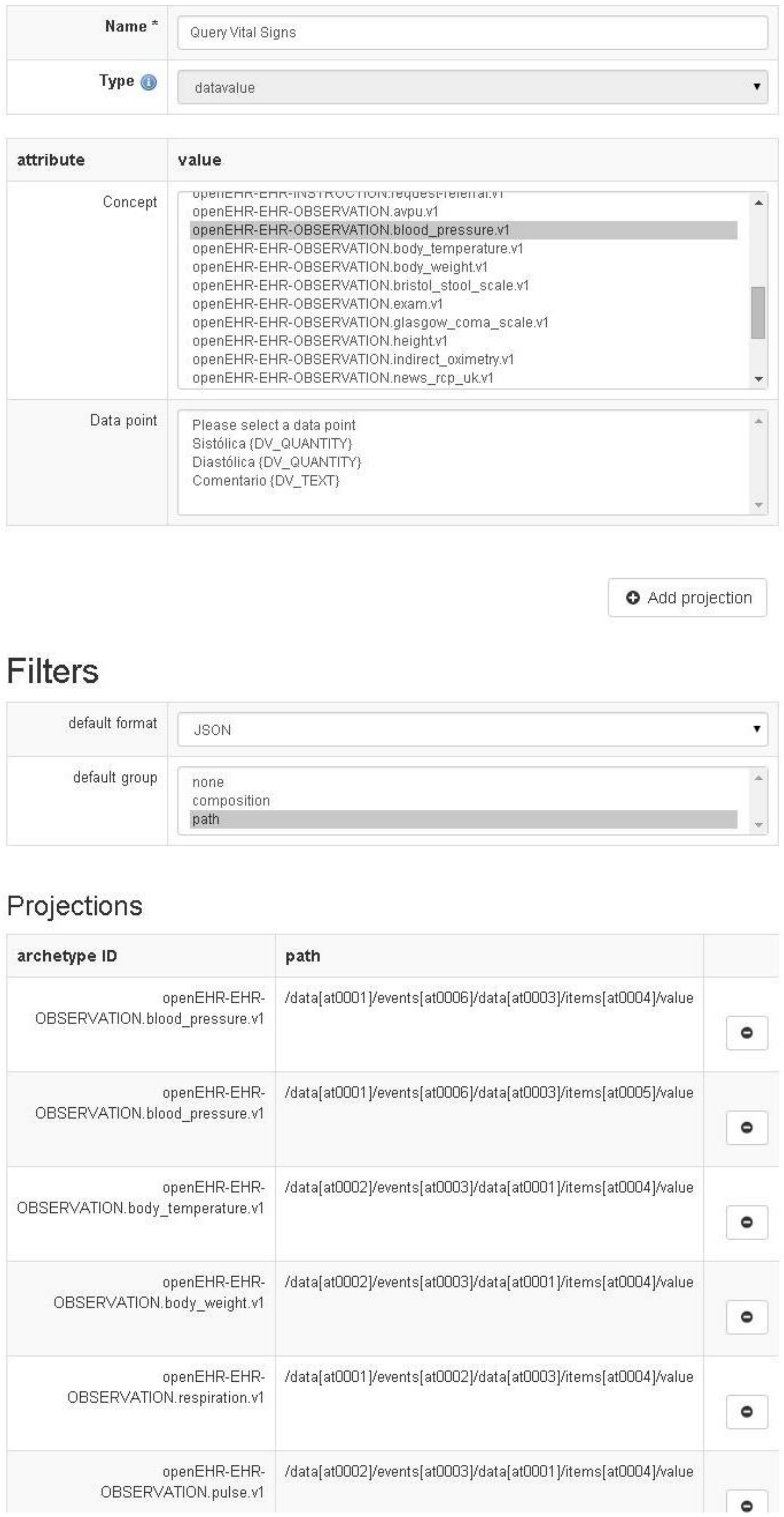
Hospital de Clinicas

Results

```
<?xml version="1.0" encoding="UTF-8"?>
<list>
  <compositionIndex id="1">
    <archetypeId>openEHR-EHR-COMPOSITION.encounter.v1</archetypeId>
    <category>event</category>
    <dataIndexed>true</dataIndexed>
    <ehrUid>11111111-1111-1111-1111-111111111111</ehrUid>
    <lastVersion>true</lastVersion>
    <organizationUid>3edb4053-d248-46cd-87a5-d5906e1e6e32</organizationUid>
    <startTime>2015-11-25 23:22:56</startTime>
    <subjectId>11111111-1111-1111-1111-111111111111</subjectId>
    <templateId>Encuentro</templateId>
    <uid>d2ee136c-098a-480c-9421-aa83eadb5a77</uid>
  </compositionIndex>
  <compositionIndex id="15">
    <archetypeId>openEHR-EHR-COMPOSITION.encounter.v1</archetypeId>
    <category>event</category>
    <dataIndexed>true</dataIndexed>
    <ehrUid>11111111-1111-1111-1111-111111111111</ehrUid>
    <lastVersion>true</lastVersion>
    <organizationUid>3edb4053-d248-46cd-87a5-d5906e1e6e32</organizationUid>
    <startTime>2015-11-26 18:54:01</startTime>
    <subjectId>11111111-1111-1111-1111-111111111111</subjectId>
    <templateId>Encuentro</templateId>
    <uid>dbd1baf9-311a-4cf2-a420-289e2f747a76</uid>
  </compositionIndex>
  <compositionIndex id="22">
    <archetypeId>openEHR-EHR-COMPOSITION.signos.v1</archetypeId>
    <category>event</category>
    <dataIndexed>true</dataIndexed>
    <ehrUid>11111111-1111-1111-1111-111111111111</ehrUid>
    <lastVersion>true</lastVersion>
    <organizationUid>3edb4053-d248-46cd-87a5-d5906e1e6e32</organizationUid>
    <startTime>2015-11-27 11:18:23</startTime>
    <subjectId>11111111-1111-1111-1111-111111111111</subjectId>
    <templateId>Signos</templateId>
    <uid>5f8ccf00-bfab-4eaa-851c-1f9166963fd3</uid>
  </compositionIndex>
  <compositionIndex id="53">
    <archetypeId>openEHR-EHR-COMPOSITION.signos.v1</archetypeId>
    <category>event</category>
    <dataIndexed>true</dataIndexed>
    <ehrUid>11111111-1111-1111-1111-111111111111</ehrUid>
```

4.2) Querying data

Querying data is pretty easy, just select Concept and Data points and click on “Add projection”, that means that you want that data in the results. In the sample below we have selected Systolic and Diastolic Blood Pressure, Body Temperature, Body Weight, Respiration Rate, and Heart Rate (Pulse), so this is a pretty complete vital sign query.



Name *	Query Vital Signs
Type	datavalue

attribute	value
Concept	openEHR-EHR-INSTRUCTION.request-referral.v1 openEHR-EHR-OBSERVATION.avpu.v1 openEHR-EHR-OBSERVATION.blood_pressure.v1 openEHR-EHR-OBSERVATION.body_temperature.v1 openEHR-EHR-OBSERVATION.body_weight.v1 openEHR-EHR-OBSERVATION.bristol_stool_scale.v1 openEHR-EHR-OBSERVATION.exam.v1 openEHR-EHR-OBSERVATION.glasgow_coma_scale.v1 openEHR-EHR-OBSERVATION.height.v1 openEHR-EHR-OBSERVATION.indirect_oximetry.v1 openEHR-EHR-OBSERVATION.news_rcp_uk.v1
Data point	Please select a data point Sistólica (DV_QUANTITY) Diastólica (DV_QUANTITY) Comentario (DV_TEXT)

Filters

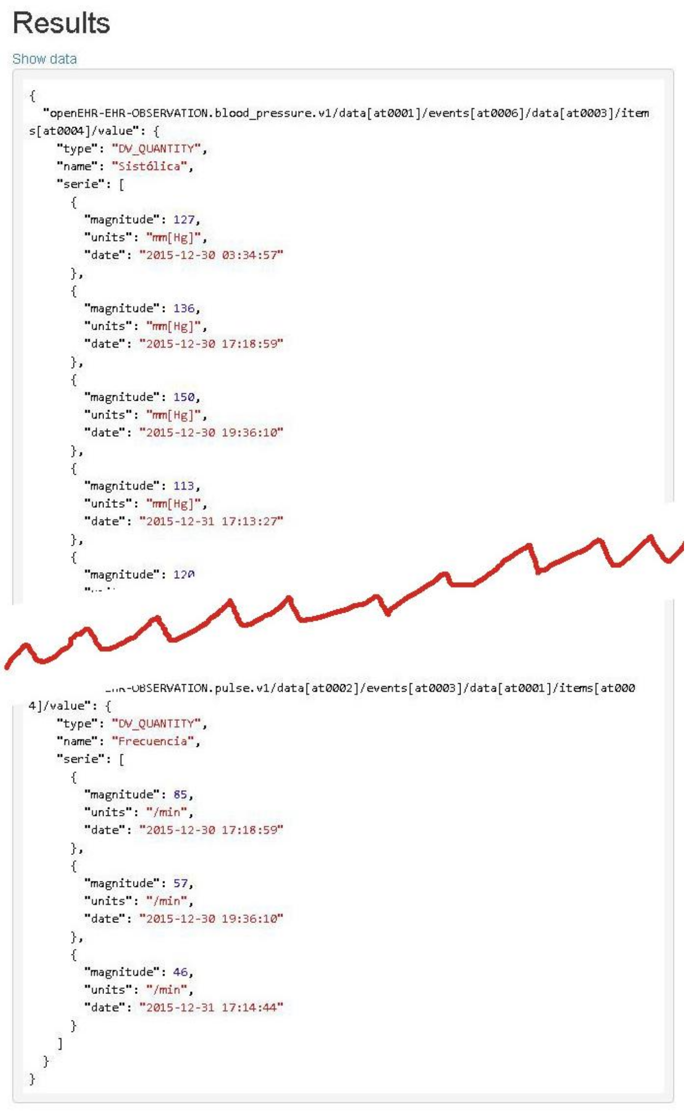
default format	JSON
default group	none composition path

Projections

archetype ID	path
openEHR-EHR-OBSERVATION.blood_pressure.v1	/data[at0001]/events[at0006]/data[at0003]/items[at0004]/value
openEHR-EHR-OBSERVATION.blood_pressure.v1	/data[at0001]/events[at0006]/data[at0003]/items[at0005]/value
openEHR-EHR-OBSERVATION.body_temperature.v1	/data[at0002]/events[at0003]/data[at0001]/items[at0004]/value
openEHR-EHR-OBSERVATION.body_weight.v1	/data[at0002]/events[at0003]/data[at0001]/items[at0004]/value
openEHR-EHR-OBSERVATION.respiration.v1	/data[at0001]/events[at0002]/data[at0003]/items[at0004]/value
openEHR-EHR-OBSERVATION.pulse.v1	/data[at0002]/events[at0003]/data[at0001]/items[at0004]/value

There are some output options like the output format (JSON or XML) and the default group (no grouping, group by composition or group by path).

Group by composition: data will be grouped by the clinical doc that contains the data.
Group by path: data will be grouped by the type of data, e.g. all Heart Rates will be contained on the same series (easy to chart). You can also test this query, and the process is the same as the composition query test.

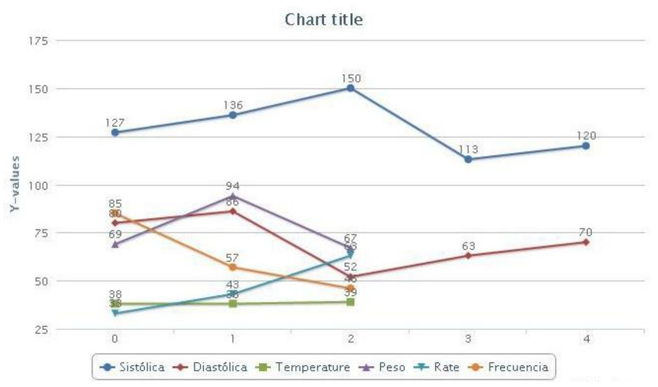


Results

Show data

```
{
  "openEHR-EHR-OBSERVATION.blood_pressure.v1/data[at0001]/events[at0006]/data[at0003]/item
s[at0004]/value": {
    "type": "DV_QUANTITY",
    "name": "Sistólica",
    "serie": [
      {
        "magnitude": 127,
        "units": "mm[Hg]",
        "date": "2015-12-30 03:34:57"
      },
      {
        "magnitude": 136,
        "units": "mm[Hg]",
        "date": "2015-12-30 17:18:59"
      },
      {
        "magnitude": 150,
        "units": "mm[Hg]",
        "date": "2015-12-30 19:36:10"
      },
      {
        "magnitude": 113,
        "units": "mm[Hg]",
        "date": "2015-12-31 17:13:27"
      },
      {
        "magnitude": 120
```

```
    -OBSERVATION.pulse.v1/data[at0002]/events[at0003]/data[at0001]/items[at000
4]/value": {
    "type": "DV_QUANTITY",
    "name": "Frecuencia",
    "serie": [
      {
        "magnitude": 85,
        "units": "/min",
        "date": "2015-12-30 17:18:59"
      },
      {
        "magnitude": 57,
        "units": "/min",
        "date": "2015-12-30 19:36:10"
      },
      {
        "magnitude": 46,
        "units": "/min",
        "date": "2015-12-31 17:14:44"
      }
    ]
  }
}
```



If you selected JSON as the result format, grouped the results by path, and the result includes numeric data, EHRServer will generate a small chart to show you the results in a graphical way (easy to verify that the results are the expected).

Installing EHRServer locally

Prerequisites

1) Download and Install MySQL Server
https://dev.mysql.com/downloads/mysql/

2) Download and Install Grails 2.5.3
http://www.grails.org/download.html

Installing

3) Download EHRServer

You can download latest development version of EHRServer from here:
https://github.com/ppazos/cabolabs-ehrserver/archive/master.zip

You can download the latest release from here:
https://github.com/ppazos/cabolabs-ehrserver/releases

4) Configure the database

Edit the DaraSource, under the "development" environment, see:

https://github.com/ppazos/cabolabs-ehrserver/blob/master/grails-app/conf/DataSource.groovy

If the database you configured doesn't exist, you need to create it in your DBMS (e.g. MySQL).

5) Create working folders and configure paths

opts

The project includes a folder called "opts" where the default Operational Templates (definitions of openEHR clinical documents) are located. You can move that folder to any location, but you need to update the entry app.opt_repo to reflect that change on the Config script.

xsd

The project includes a folder called "xsd" where the needed XML Schemas are located. You can move that folder to any location, but you need to update these entries on the Config script:

- app.version_xsd

- app.xslt
- app.opt_xsd

versions

You need to create a working folder to store the committed versions. That folder should have permissions to read and write. After you create that folder, you need to update the entry app.version_repo on the Config script.

By default, that folder is ehrserver/versions, where “ehrserver” is the folder in which the EHRServer code is.

6) Run the EHRServer

Run:

Execute this command line from the project folder:

```
ehrserver/ grails -Dserver.port=8090 run-app
```

This will run the server locally, on the port 8090, so you will be able to access it through: http://localhost:8090/ehr

Login:

Use admin / admin / 1234 (username, password, organization) to login, and you are ready to go. That is the administration user, so it has special access to all the functionalities of the EHRServer.

For a more constrained user, you can use this login: orgman / orgman / 1234 (username, password, organization). That user is an organization manager, and can only manage it’s organizations, so some items on the menu are hidden from this user as only the admin has rights to access them.

7) Create environment variables if you will use the “create account” feature locally

When an account is created, it needs to send an email with some basic account information, and a link to reset the password. The email service needs to be configured to do that. We use these environment variables to do that configuration:

- EHRSERVER_EMAIL_HOST: URL / IP of your SMTP server
- EHRSERVER_EMAIL_PORT: port number of your SMTP server
- EHRSERVER_EMAIL_USER: valid user on your SMTP server (probably an email address)
- EHRSERVER_EMAIL_PASS: password corresponding to the user
- EHRSERVER_EMAIL_FROM: the email address that will appear to the receiver as “from”

You can see where this configuration is used at:

https://github.com/ppazos/cabolabs-ehrserver/blob/master/grails-app/conf/Config.groovy#L218-L224

Note: If you want to deploy EHRServer on the cloud, for example for our staging server we use OpenShift, there are some email server solutions you can use:

- SendGrid: https://developers.openshift.com/en/external-services-sendgrid.html
- RoundCube: https://blog.openshift.com/free-paas-email-server-with-roundcube/

EHRServer Management

Supporting more clinical documents

Before committing any data, you need an Operational Template (OPT) that specifies the structure, semantics, constraints and terminology of your clinical documents. To upload new OPTs, you need to login and go to the Templates section > Upload Templates.

You can create your own OPTs by creating archetypes or using archetypes from the openEHR CKM (http://ckm.openehr.org/ckm/), and aggregating those archetypes in a Template using the Template Designer (http://www.openehr.org/downloads/modellingtools). You can find some OPT samples in our GitHub repo[2].

OPTs should comply with this XSD to be accepted by EHRServer:

https://github.com/ppazos/cabolabs-ehrserver/blob/master/xsd/OperationalTemplate.xsd

After you have an OPT loaded in the EHRServer, you can start committing compositions that follow the OPT definition. This way EHRServer can be extended indefinitely to support more and more clinical document structures, to be stored and queried through the EHRServer REST API.

[2] https://github.com/ppazos/cabolabs-ehrserver/tree/master/opts

EHRServer REST API

POST /login

Get an authorization token to be used on all the other endpoints.

Parameters:

- username: username associated with your account
- password: password associated with your account
- organization: organization number associated with your account

Result sample: (Content-Type: application/json)

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Im9yZ21hbiIsIm..."
}
```

That token should be used to send requests to ALL the endpoints described below, by adding this header to the request:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6Im9yZ21hbiIsIm...
```

GET /profile/$username

Get data about the user with username = $username.

Parameters:

- format: output format, valid values are "xml" or "json".

Result sample: (Content-Type: application/json)

```
{
    "username": "orgman",
    "email": "pablo.swp+orgman@gmail.com",
    "organizations": [
        {
            "name": "Hospital de Clinicas",
            "number": "1234",
            "uid": "3edb4053-d248-46cd-87a5-d5906e1e6e32"
        }
    ]
}
```

GET /patients

Get the patients associated with the organization used on /login

Parameters:

- format: output format, valid values are “xml” or “json”.

Result sample: (Content-Type: application/json)

```
{
    "patients": [
        {
            "uid": "44444444-1111-1111-1111-111111111111",
            "firstName": "Mario",
            "lastName": "Gomez",
            "dob": "19640919",
            "sex": "M",
            "idCode": "5677565-0",
            "idType": "CI",
            "organizationUid": "3edb4053-d248-46cd-87a5-d5906e1e6e32"
        },
        {
            "uid": "407bd5b9-076b-48d7-b95b-6f3f6183ddc3",
            "firstName": "José",
            "lastName": "Gomez",
            "dob": "19541221",
            "sex": "M",
            "idCode": "4567897",
            "idType": "Passport",
            "organizationUid": "3edb4053-d248-46cd-87a5-d5906e1e6e32"
        },
        ...
    ],
    "pagination": {
        "max": 15,
        "offset": 0,
        "nextOffset": 15,
        "prevOffset": 0
    }
}
```

GET /ehrs

Get the EHRs associated with the organization used on /login

Parameters:

- format: output format, valid values are “xml” or “json”.

Result sample: (Content-Type: application/json)

```
{
    "ehrs": [
        {
            "uid": "11111111-1111-1111-1111-111111111111",
            "dateCreated": "20151125T015252,000+0000",
            "subjectUid": "11111111-1111-1111-1111-111111111111",
            "systemId": "ISIS_EHR_SERVER",
            "organizationUid": "cd69aa7c-0a11-46db-89c8-64435615536f"
        },
        {
            "uid": "22222222-1111-1111-1111-111111111111",
            "dateCreated": "20151125T015252,000+0000",
            "subjectUid": "22222222-1111-1111-1111-111111111111",
            "systemId": "ISIS_EHR_SERVER",
            "organizationUid": "cd69aa7c-0a11-46db-89c8-64435615536f"
        },
        ...
    ],
    "pagination": {
        "max": 15,
        "offset": 0,
        "nextOffset": 15,
        "prevOffset": 0
    }
}
```

GET /ehrs/ehrUid/$ehrUid

Get one EHR which UID match $ehrUid.

Parameters:

- format: output format, valid values are “xml” or “json”.

Result sample: (Content-Type: application/json)

```
{
     "uid": "22222222-1111-1111-1111-111111111111",
     "dateCreated": "20151125T015252,000+0000",
     "subjectUid": "22222222-1111-1111-1111-111111111111",
     "systemId": "ISIS_EHR_SERVER",
     "organizationUid": "cd69aa7c-0a11-46db-89c8-64435615536f"
}
```

GET /ehrs/subjectUid/$subjectUid

Get one EHR which patient’s UID match $subjectUid.

Parameters:

- format: output format, valid values are “xml” or “json”.

Result sample: (Content-Type: application/json)

```
{
     "uid": "22222222-1111-1111-1111-111111111111",
     "dateCreated": "20151125T015252,000+0000",
    "subjectUid": "22222222-1111-1111-1111-111111111111",
    "systemId": "ISIS_EHR_SERVER",
    "organizationUid": "cd69aa7c-0a11-46db-89c8-64435615536f"
}
```

GET /compositions

Get the clinical documents from a patient's EHR.

Parameters:

- format: output format, valid values are "xml" or "json".
- ehrUid: UID of the EHR to get the compositions from.

Result sample: (Content-Type: application/json)

```
[
    {
        "archetypeId": "openEHR-EHR-COMPOSITION.test_all_datatypes.v1",
        "category": "event",
        "ehrUid": "11111111-1111-1111-1111-111111111111",
        "lastVersion": true,
        "organizationUid": "2aeba74a-3d35-4082-aab3-9a05e52c4a53",
        "startTime": "2014-09-02 05:26:00",
        "subjectId": "11111111-1111-1111-1111-111111111111",
        "templateId": "Test all datatypes_en",
        "uid": "e6fc1e00-faec-4726-a156-b702c55cfe88"
    }
    ,
    ...
]
```

GET /compositions/$uid

Get the clinical document with UID = $uid.

Parameters:

- format: output format, valid values are "xml", "json" or "html".

Result sample: composition version object (Content-Type: application/json)

```
{
    "version": {
        "@xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
        "@xmlns": "http://schemas.openehr.org/v1",
        "@xsi:type": "ORIGINAL_VERSION",
        "contribution": {
            "id": {
                "@xsi:type": "HIER_OBJECT_ID",
                "value": "ad6866e1-fb08-4e9b-a93b-5095a2563775"
            },
            "namespace": "EHR::COMMON",
            "type": "CONTRIBUTION"
        },
        "commit_audit": {
            "system_id": "CABOLABS_EHR",
            "committer": {
                "@xsi:type": "PARTY_IDENTIFIED",
                "name": "Dr. Pablo Pazos"
            },
            "time_committed": {
                "value": "20140901T233114,065-0300"
            },
            "change_type": {
                "value": "creation",
                "defining_code": {
                    "terminology_id": {
                        "value": "openehr"
                    },
                    "code_string": 249
                }
            }
        },
        "uid": {
            "value": "91cf9ded-e926-4848-aa3f-3257c1d89554::EMR_APP::1"
        },
        "data": {
            "@archetype_node_id": "openEHR-EHR-COMPOSITION.test_all_datatypes.v1",
            "@xsi:type": "COMPOSITION",
            "name": {
                "value": "Test all datatypes"
            },
            "uid": {
                "@xsi:type": "HIER_OBJECT_ID",
                "value": "d6fa1aa6-cfc7-4c28-ba51-555ee55b0ae1"
            },
     ...
}
```

POST /commit

Commits a set of compositions to the EHR of a patient.

Parameters:

- ehrUid: UID of the EHR to commit the compositions to.
- auditCommitter: name of the person or system that commits the composition, for audit purposes.
- auditSystemId: identifier of the system that commits the composition, for audit purposes.

Request body:

The body should contain a set of versions in XML. Each version should be compliant with this XSD: https://github.com/ppazos/cabolabs-ehrserver/blob/master/xsd/Version.xsd

Sample XML:
https://github.com/ppazos/cabolabs-emrapp/blob/master/committed/versions_4172_.xml

Considerations about the XML to commit:

- versions.version.commit_audit.committer.name is mandatory (required by EHRServer).
- versions.version.contribution should be the same for all the versions.version (a commit represents one contribution).
- versions.version.uid.value should be unique for document creation (see versions.version.commit_audit.change_type.value).
- versions.version.uid.value should already exist in the EHRServer for other change types than creation, like amendment. This will be used to commit a new version of an existing composition.
- If versions.version.data.uid is empty, the EHRServer will assign an UID to the composition.

GET /queries

Get the list of queries created in the EHRServer.

Parameters:

- format: output format, valid values are "xml" or "json".

Result sample: (Content-Type: application/json)

```
{
    "queries": [
        {
            "uid": "7b8762ac-eaf4-435b-9fde-d59730b6641f",
            "name": "documents",
            "format": "xml",
            "type": "datavalue",
            "group": "none",
            "projections": [
                {
                    "archetypeId": "openEHR-EHR-OBSERVATION.respiration.v1",
                    "path": "/data[at0001]/events[at0002]/data[at0003]/items[at0004]/value"
                },
                {
                    "archetypeId": "openEHR-EHR-OBSERVATION.terminology_ref.v1",
                    "path": "/data[at0001]/events[at0002]/data[at0003]/items[at0004]/value"
                }
            ]
        },
        {
            "uid": "1133fa73-4bf8-43eb-95a5-e69c7a91ffc9",
            "name": "data",
            "format": "json",
            "type": "composition",
            "criteria": [
                {
                    "archetypeId": "openEHR-EHR-OBSERVATION.blood_pressure.v1",
                    "path": "/data[at0001]/events[at0006]/data[at0003]/items[at0004]/value",
                    "criteria": "dqi.magnitude > 140.0 AND dqi.units = 'mm[Hg]'"
                },
                {
                    "archetypeId": "openEHR-EHR-OBSERVATION.blood_pressure.v1",
                    "path": "/data[at0001]/events[at0006]/data[at0003]/items[at0005]/value",
                    "criteria": "dqi.magnitude > 90.0 AND dqi.units = 'mm[Hg]'"
                }
            ]
        }
    ],
    "pagination": {
        "max": 15,
        "offset": 0,
        "nextOffset": 15,
        "prevoffset": 0
    }
}
```

GET /queries/$queryUid

Get the query with the UID $queyrUid.

Parameters:

- format: output format, valid values are “xml” or “json”.

Result sample: (Content-Type: application/json)

```
{
    "uid": "7b8762ac-eaf4-435b-9fde-d59730b6641f",
    "name": "documents",
    "format": "xml",
    "type": "datavalue",
    "group": "none",
    "projections": [
        {
            "archetypeId": "openEHR-EHR-OBSERVATION.respiration.v1",
            "path": "/data[at0001]/events[at0002]/data[at0003]/items[at0004]/value"
        },
        {
            "archetypeId": "openEHR-EHR-OBSERVATION.terminology_ref.v1",
            "path": "/data[at0001]/events[at0002]/data[at0003]/items[at0004]/value"
        }
    ]
}
```

GET /queries/$queryUid/execute

Executes the query with the UID $queyrUid.

Parameters:

- format: output format, valid values are “xml” or “json”.
- ehrUid: UID of the EHR we want to query. If no ehrUid is specified, the result will contain results for multiple EHRs.
- organizationUid: UID of the organization that owns the EHRs we want to query.

- retrieveData: this parameter specifies if the composition query should return data if the value is “true”
- group: overrides the grouping of the datavalue query, valid values are: “none”, “composition” or “path”.
- fromDate: filter for the results, to get results after this date. Expected format is: yyyyMMdd.
- toDate: filter for the results, to get results before this date. Expected format is: yyyyMMdd.

Result sample for datavalue query: (Content-Type: application/json)

```
{
    "openEHR-EHR-
OBSERVATION.blood_pressure.v1/data[at0001]/events[at0006]/data[at0003]/items[at0004]/value": {
        "type": "DV_QUANTITY",
        "name": "Sistólica",
        "serie": [
            {
                "magnitude": 106,
                "units": "mm[Hg]",
                "date": "2016-01-14 07:34:59"
            }
        ]
    },
    "openEHR-EHR-
OBSERVATION.blood_pressure.v1/data[at0001]/events[at0006]/data[at0003]/items[at0005]/value": {
        "type": "DV_QUANTITY",
        "name": "Diastólica",
        "serie": [
            {
                "magnitude": 56,
                "units": "mm[Hg]",
                "date": "2016-01-14 07:34:59"
            }
        ]
    },
    "timing": "10 ms"
}
```

Result sample for composition query: (Content-Type: application/json)

```
{
    "results": [
        {
            "archetypeId": "openEHR-EHR-COMPOSITION.signos.v1",
            "category": "event",
            "ehrUid": "11111111-1111-1111-1111-111111111111",
            "lastVersion": true,
            "organizationUid": "e7325a3e-489c-4a90-b360-36fb5ba2bc9b",
            "startTime": "2016-01-14 08:21:36",
            "subjectId": "11111111-1111-1111-1111-111111111111",
            "templateId": "Signos",
            "uid": "95e60f9c-842a-42b0-918c-ffc72bb28475"
        }
    ],
    "timing": "312 ms"
}
```

Notes:

For datavalue queries, the result structure will depend on the selected grouping. In the example, the grouping by path is shown.

For composition queries, the result is a list of indexes of compositions, like the result of the /compositions endpoint.

For composition queries, if retrieveData=true, instead of indexes of compositions, the result will include the complete compositions. We discourage the use of this parameter if the filters are too wide a lots of compositions can be retrieved (can take a while). A better approach would be to query composition indexes and then get the data for specific compositions from /compositions/$uid