CaboLabs

EHRServer v0.9 guide

The open source, service-oriented, openEHR clinical 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 full 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

In 2017 we launched a SaaS option. This is useful if you don't want to manage the server yourself. On it's early stages, the CloudEHRServer will be available through the Beta Partners Program (https://cloudehrserver.com/beta_partners_program).

Useful links & communication channels about the Cloud EHRServer:

1. https://cloudehrserver.com
2. https://twitter.com/CloudEHRServer
3. https://www.linkedin.com/pulse/cloud-ehrserver-from-proof-concept-software-service-pazos-gutiérrez

[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 Console:

1) You need an account

Open the Web App: https://ehrserver-cabolabs2.rhcloud.com/

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 EHRs

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

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

Go to EHRs > New EHR, fill the form and click on Create.

Note that the EHR’s patient is referenced through a Subject UID, and no demographic data is stored in the EHRServer. This is a requirement of modern EHRs and the openEHR standard. So all the information in the EHRServer is anonymous, and the Subject UID allows you to reference a patient record that is stored in an external system, like a Master Patient Index.

Go back to EHRs, and you will have a new and empty EHR.

So far so good, now you need to add some data an 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 an EHR from the list (you will see your organization’s EHRs 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

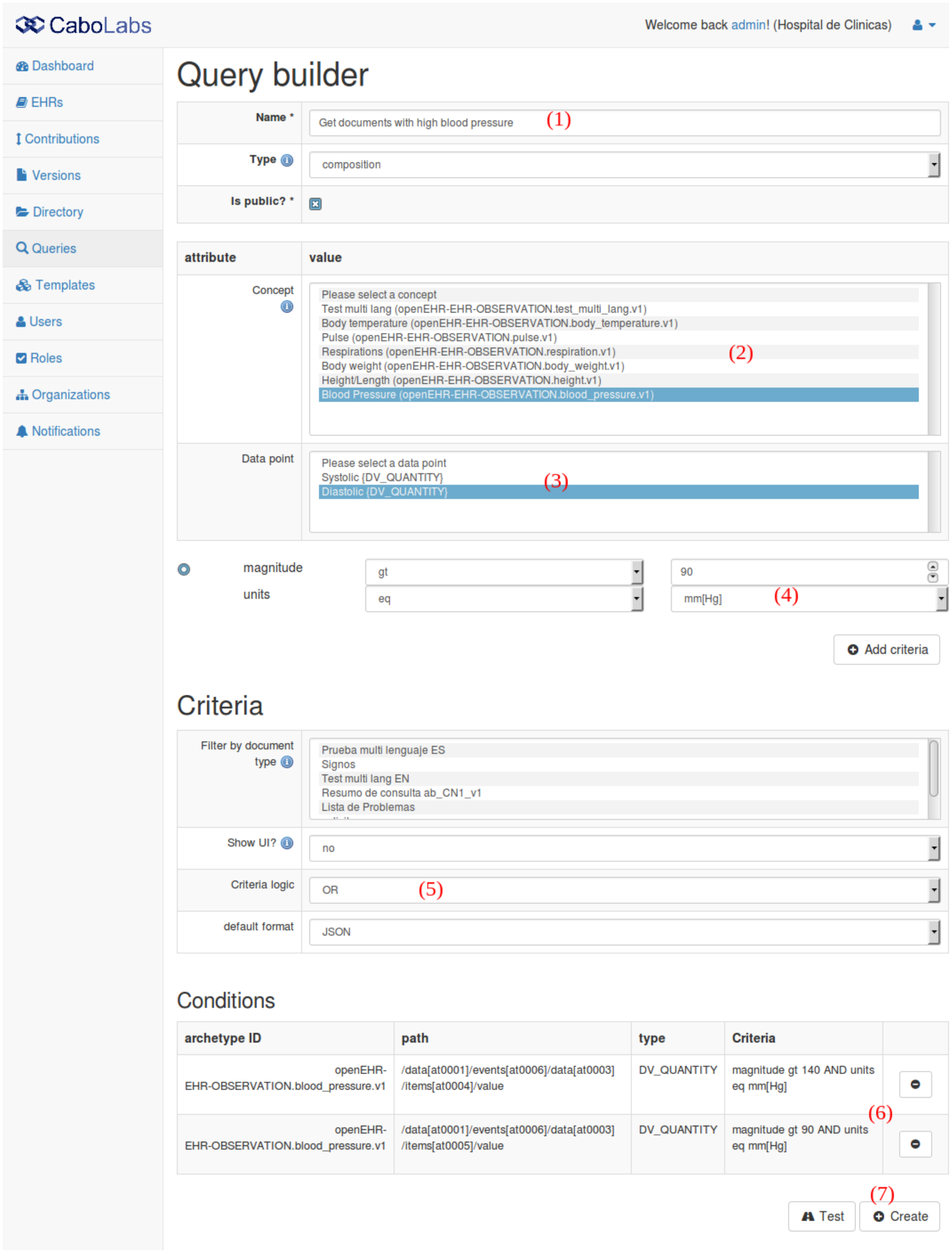
4) Define the criteria e.g. magnitude > 90 and units = mm[Hg]

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! (Hospital de Clinicas)

Query builder

Name *	Get documents with high blood pressure (1)
Type	composition
Is public? *	☒

attribute	value
Concept	Please select a concept Test multi lang (openEHR-EHR-OBSERVATION.test_multi_lang.v1) Body temperature (openEHR-EHR-OBSERVATION.body_temperature.v1) Pulse (openEHR-EHR-OBSERVATION.pulse.v1) Respirations (openEHR-EHR-OBSERVATION.respiration.v1) (2) Body weight (openEHR-EHR-OBSERVATION.body_weight.v1) Height/Length (openEHR-EHR-OBSERVATION.height.v1) Blood Pressure (openEHR-EHR-OBSERVATION.blood_pressure.v1)
Data point	Please select a data point Systolic {DV_QUANTITY} (3) Diastolic {DV_QUANTITY}

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

Add criteria

Criteria

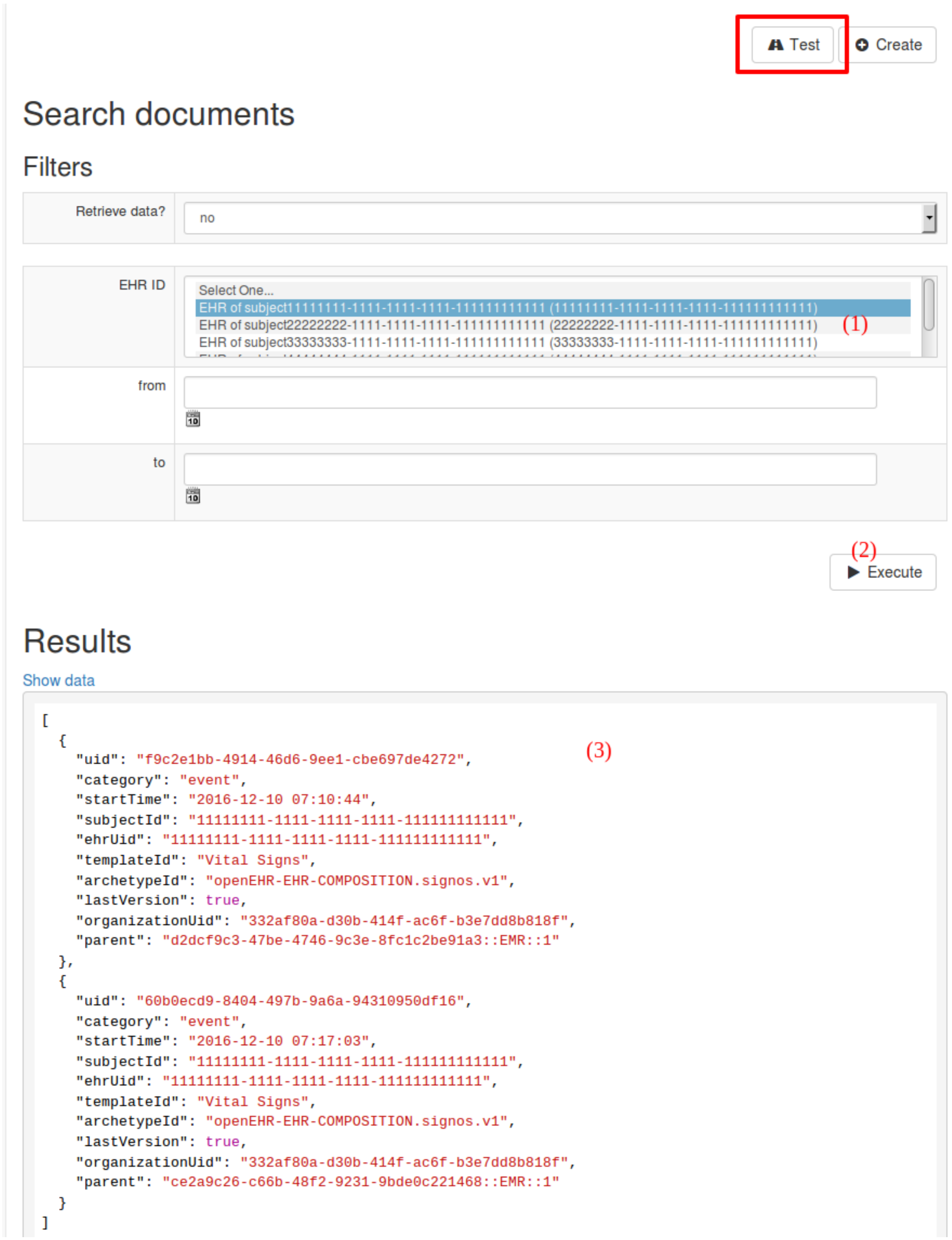
Filter by document type	Prueba multi lenguaje ES Signos Test multi lang EN Resumo de consulta ab_CN1_v1 Lista de Problemas
Show UI?	no
Criteria logic	OR (5)
default format	JSON

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)

Figure 1: Query Builder – clinical document query creation

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 (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”).

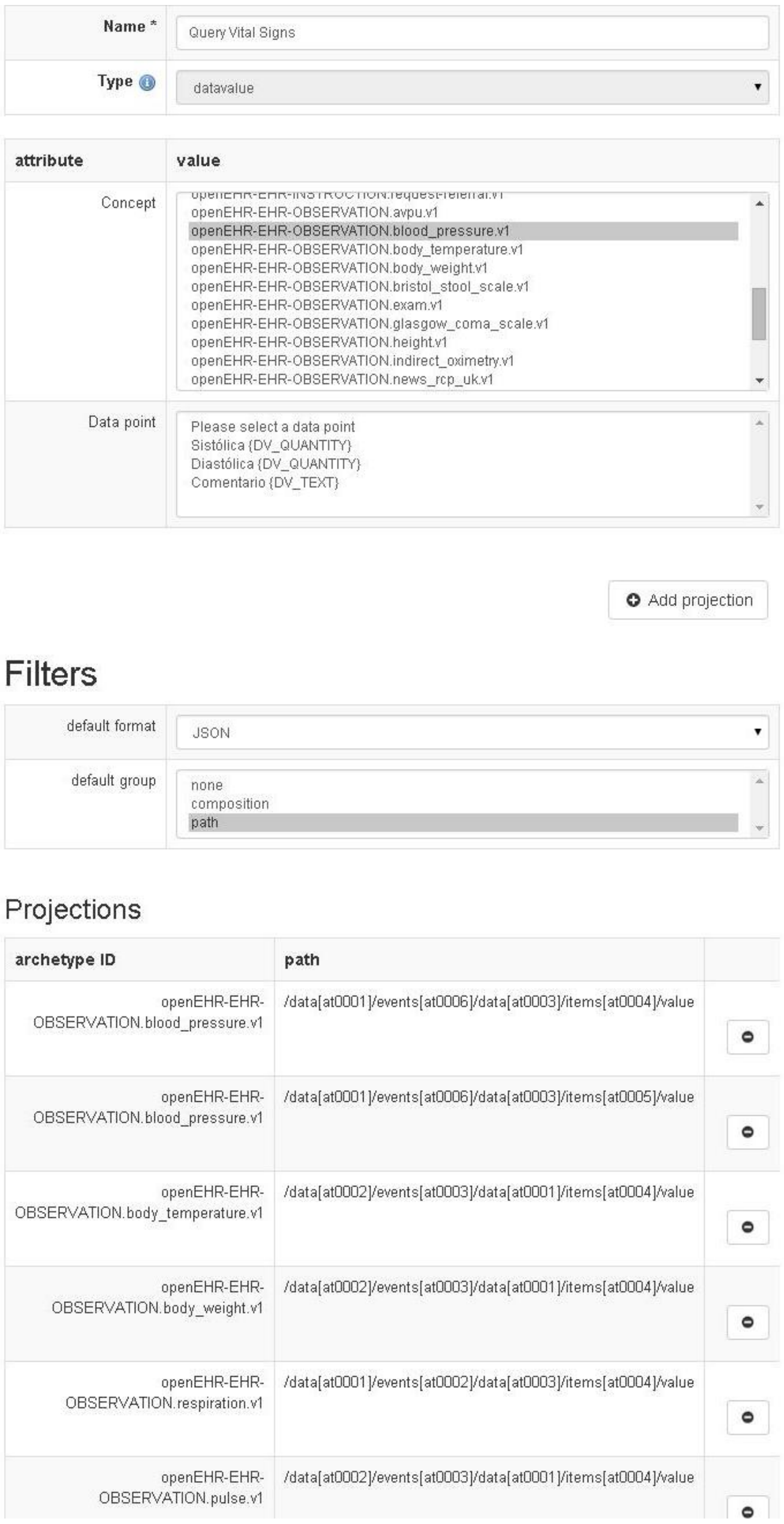

Test
Create
Search documents
Filters
Retrieve data?
no
EHR ID
Select One...
EHR of subject11111111-1111-1111-1111-111111111111 (11111111-1111-1111-1111-111111111111)
EHR of subject22222222-1111-1111-1111-111111111111 (22222222-1111-1111-1111-111111111111)
EHR of subject33333333-1111-1111-1111-111111111111 (33333333-1111-1111-1111-111111111111)
(1)
from
to
(2)
Execute
Results
Show data
(3)

```
[
  {
    "uid": "f9c2e1bb-4914-46d6-9ee1-cbe697de4272",
    "category": "event",
    "startTime": "2016-12-10 07:10:44",
    "subjectId": "11111111-1111-1111-1111-111111111111",
    "ehrUid": "11111111-1111-1111-1111-111111111111",
    "templateId": "Vital Signs",
    "archetypeId": "openEHR-EHR-COMPOSITION.signos.v1",
    "lastVersion": true,
    "organizationUid": "332af80a-d30b-414f-ac6f-b3e7dd8b818f",
    "parent": "d2dcf9c3-47be-4746-9c3e-8fc1c2be91a3::EMR::1"
  },
  {
    "uid": "60b0ecd9-8404-497b-9a6a-94310950df16",
    "category": "event",
    "startTime": "2016-12-10 07:17:03",
    "subjectId": "11111111-1111-1111-1111-111111111111",
    "ehrUid": "11111111-1111-1111-1111-111111111111",
    "templateId": "Vital Signs",
    "archetypeId": "openEHR-EHR-COMPOSITION.signos.v1",
    "lastVersion": true,
    "organizationUid": "332af80a-d30b-414f-ac6f-b3e7dd8b818f",
    "parent": "ce2a9c26-c66b-48f2-9231-9bde0c221468::EMR::1"
  }
]
```

Figure 2: Query builder – clinical document query testing

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

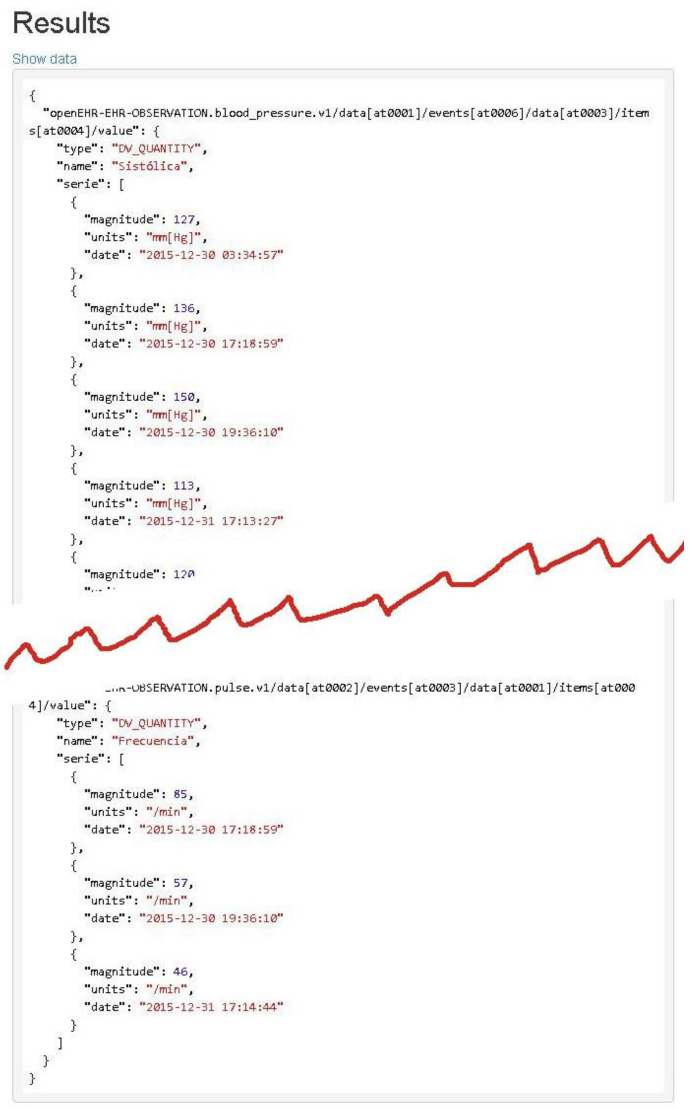
default format	JSON
default group	none composition path

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.

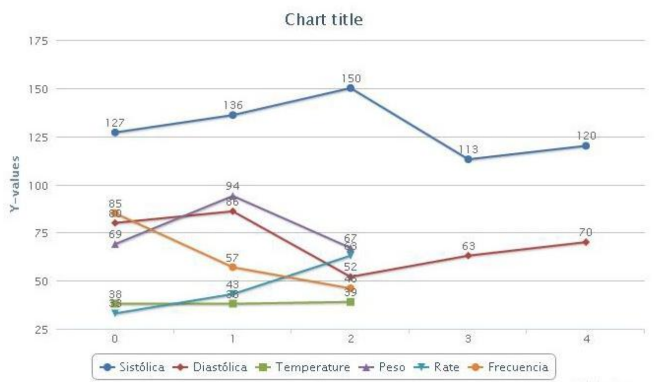


Results

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
```

```
     -HR-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 2.5.5
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 & opts/base_opts

The project includes a folder called "opts". Inside there is a "base_opts" folder, where the default Operational Templates (definitions of openEHR clinical documents) are located. When the EHRServer is started, the OPTs from "base_opts" are copied (and renamed) to "opts", only those definitions will be used by the EHRServer. You can move the "opts" folder to any location, but you need to update the entry "app.opt_repo" in the Config script to reflect the new location of the folder.

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

Note: if you run the EHRServer from the WAR in a Web Server like Tomcat, the xsd folder is not needed because it is packaged with the app in the WAR file.

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/

8) Environment variables needed for production deployment

In order to deploy EHRServer in production, a set of environment variables should be set. The names of this variables are OpenShift dependant (see not below), but you can change them with any names you like. Here we describe each variable:

OPENSHIFT_APP_DNS

Is the URL of the application without the protocol (https/http).

Default value: cabolabs-ehrserver.rhcloud.com

Used from: Config.groovy

OPENSHIFT_MYSQL_DB_HOST

Is the IP of the MySQL database that will be used in production.

Default value: depends on your environment.

Used from: DataSource.groovy

OPENSHIFT_MYSQL_DB_PORT

Is the port of the MySQL database that will be used in production.

Default value: 3306

Used from: DataSource.groovy

OPENSHIFT_MYSQL_DB_USERNAME

Is the username used to connect to the MySQL database that will be used in production.

Default value: depends on your environment.

Used from: DataSource.groovy

OPENSHIFT_MYSQL_DB_PASSWORD

Is the password used to connect to the MySQL database that will be used in production.

Default value: depends on your environment.

Used from: DataSource.groovy

OPENSHIFT_APP_NAME

Is the application name, that is used as the database name. Just change this for the database name.

Default value: cabolabs (name given by OpenShift).

Used from: DataSource.groovy

EHRSERVER_ALLOW_WEB_USER_REGISTER

Set this variable to true to allow users to register themselves in the EHRServer from the Web. If it's set to false, users will be created only from the Web Console or from the API.

Note: you can use any cloud service you like to deploy EHRServer on the cloud, because OpenShift has limitations by country for paid accounts. Some good alternatives are:

1. AWS: https://aws.amazon.com/
2. Linode: https://www.linode.com/

All you need is: Java 7, MySQL and Tomcat installed. Java 8 should also work.

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.

Updating existing document definitions

TBD

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

EHRServer REST API

The URLs of the endpoints documented on this section are relative to a base URL. The base URL depends on a domain or an IP address. If you have a domain like **ehrserver.cabolabs.com** (It's just an example), the base URL for the API endpoints will be:

https://ehrserver.cabolabs.com/ehr/api/v1

Where "v1" is the version of the REST API. This is to allow the evolution of the API while being able to use older versions of it.

If you run the server locally, let's say on port 8080, then the base URL would be:

http://127.0.0.1:8080/ehr/api/v1

So, if you want to invoke the GET /ehrs endpoint from a local deployment, the complete URL will look like this:

http://127.0.0.1:8080/ehr/api/v1/ehrs

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 /users/$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

This endpoint was removed since v0.9 because the EHRServer is not longer storing demographic data.

GET /patients/$uid

This endpoint was removed since v0.9 because the EHRServer is not longer storing demographic data.

POST /person

This endpoint was removed since v0.9 because the EHRServer is not longer storing demographic data.

GET /ehrs

Get the EHRs associated with the organization used on /login

Parameters:

- format: output format, valid values are “xml” or “json”.
- max: maximum number of ehrs to be retrieved from the offset.
- offset: results will be retrieved from the offset, default is 0 (with offset 0, ehrs will be retrieved from the first one, to the “max” one, with offset “max”, ehrs will be retrieved from “max” to “2*max”).

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

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

GET /ehrs/ehrUid/$uid

Get one EHR which UID match $uid.

Parameters:

- format: output format, valid values are “xml” or “json”.
- uid: can be passed as parameter or in the URL.

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

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

POST /ehrs

Creates a new EHR for an externally managed patient.

Parameters:

- format: output format, valid values are "xml" or "json".
- uid: optional unique identifier for the EHR, if not set the EHRServer will generate it.
- subjectUid: external identifier/reference to the patient's demographic record maintained in an external system like a Master Patient Index.

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

```
{
    "uid": "42d7477c-6d01-42cf-ac35-9560f25d6ff1",
    "dateCreated": "2016-11-26 22:58:53",
    "subjectUid": "327e3a5d-dd5c-4158-88e9-a77072a5d3ce",
    "systemId": "CABOLABS_EHR_SERVER",
    "organizationUid": "77cfcb26-b9b2-4fdd-9413-14bdbaab0218"
}
```

GET /contributions

Get the audit logs for an EHR.

Parameters:

- format: output format, valid values are "xml" or "json".
- ehrUid: mandatory UID of the EHR to get the compositions from.
- from: date filter "from", with format yyyyMMdd
- to: date filter "to", with format yyyyMMdd
- max: maximum number of contributions to be retrieved from the offset.
- offset: results will be retrieved from the offset, default is 0 (with offset 0, contributions will be retrieved from the first one, to the "max" one, with offset "max", contributions will be retrieved from "max" to "2*max").

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

```
{
    "contributions": [
        {
            "uid": "30de11b0-7ff8-440e-83e5-dec2a1206709",
            "organizationUid": "a82201e7-f197-4fe3-8d37-8e1fe6b33dc4",
            "ehrUid": "11111111-1111-1111-1111-111111111111",
            "versions": [
                "6f06e7f5-eccf-4e40-b7a3-9018ccaf0199::EMR::1"
            ],
            "audit": {
                "timeCommitted": "2016-06-25 06:47:37",
                "systemId": "EMR",
                "committer": {
```

```
                "namespace": "local",
                "type": "PERSON",
                "value": "1324566",
                "name": "Dr. House"
            }
        }
    },
    {
        "uid": "96d7cbb5-60b7-4714-ad36-0bbff989412b",
        "organizationUid": "a82201e7-f197-4fe3-8d37-8e1fe6b33dc4",
        "ehrUid": "11111111-1111-1111-1111-111111111111",
        "versions": [
            "53105d43-d849-463b-8a6d-f96150bc32cc::EMR::1"
        ],
        "audit": {
            "timeCommitted": "2016-06-25 06:47:47",
            "systemId": "EMR",
            "committer": {
                "namespace": "local",
                "type": "PERSON",
                "value": "1324566",
                "name": "Dr. House"
            }
        }
    }
],
"pagination": {
    "max": 20,
    "offset": 0,
    "nextOffset": 20,
    "prevOffset": 0
}
}
```

GET /compositions

Get the clinical documents from an EHR.

Parameters:

- format: output format, valid values are “xml” or “json”.
- ehrUid: UID of the EHR to get the compositions from.
- max: maximum number of compositions to be retrieved from the offset.
- offset: results will be retrieved from the offset, default is 0 (with offset 0, compositions will be retrieved from the first one, to the “max” one, with offset “max”, compositions will be retrieved from “max” to “2*max”).

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

```
{
    "result": [
        {
            "uid": "0f78e043-aa09-4212-9669-fcef0adaf470",
            "category": "event",
            "startTime": "2016-06-25 07:29:28",
            "subjectId": "11111111-1111-1111-1111-111111111111",
            "ehrUid": "11111111-1111-1111-1111-111111111111",
            "templateId": "Signos",
            "archetypeId": "openEHR-EHR-COMPOSITION.signos.v1",
            "lastVersion": true,
            "organizationUid": "d04809ca-08dc-454a-8390-96a0b125abf1",
            "parent": "90120202-e7a6-4032-a935-fe91f6e7fd28::EMR::1"
        },
        ...
    ],
    "pagination": {
        "max": 20,
        "offset": 0,
        "nextOffset": 20,
        "prevOffset": 0
    }
}
```

Note: parent has the UID of the VERSION that contains the COMPOSITION referenced by the index.

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 an EHR.

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 or JSON format. Each XML version should be compliant with this XSD: https://github.com/ppazos/cabolabs-ehrserver/blob/master/xsd/Version.xsd JSON versions are transformed internally to XML and validated against the same XML Schema.

Rules (derived from the openEHR specs)

1. Format of version.uid.value

version.uid.value should have this format: versioned_object_id::creating_system_id::version_tree_id

Where:

- versioned_object_id: is an UUID, is set by the client if the committed document is a new one.
- creating_system_id: a precoordinated code that identifies the client system.
- version_tree_id: should be 1 for new documents, or the value given by the EHRServer from the checkout service.

2. Versioned documents

The commit of a new document, will generate a versioned object in the EHRServer, and will be a container for all the versions of the same document. The uid of that object will be the versioned_object_id portion of the version uid, all the versions of the same document will reference the same versioned_object_id.

3. Create a new version of an existing document

In order to create a new version of an existing document, client apps should checkout the specific version, using the checkout service. Then make some changes to the document, and commit the modified document, using the same version uid provided on the checkout.

And the commit_audit.change_type information should be amendment or modification. Check the codes in the openEHR terminology (https://github.com/ppazos/openEHR-OPT/blob/master/resources/terminology/openehr_terminology_en.xml#L26-L34). The new commit will generate a new version, associate it with the versioned object and increase the version_tree_id.

4. Encoding

Client applications should encode clinical documents using UTF-8. Other encoding support can be added in the future, for now we need to keep things simple using just UTF-8.

5. Root archetype id

Remember to send the root openEHR archetype id in the data.@archetype_node_id attribute, and all the other nodes that correspond to resolved archetype slots.

6. Commit time

The version.commit_audit.time_committed that is set by client apps will be overriden by the server to be compliant with this rule from the openEHR specs:

“The time_committed attribute in both the Contribution and Version audits should reflect the time of committal to an EHR server, i.e. the time of availability to other users in the same system. It should therefore be computed on the server in implementations where the data are created in a separate client context.”

7. Contributions

The parameters auditSystemId and auditCommitter are used to create the CONTRIBUTION for each commit. To be compliant with the openEHR specs, the client system should use that data to create the VERSION.commit_audit structure. So this rule is met:

"CONTRIBUTION.audit captures to the time, place and committer of the committal act; these three attributes (system_id, committer, time_committed of AUDIT_DETAILS) should be copied into the corresponding attributes of the commit_audit of each VERSION included in the CONTRIBUTION...".

Sample XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<versions xmlns="http://schemas.openehr.org/v1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <version xsi:type="ORIGINAL_VERSION">
     <contribution>
       <id xsi:type="HIER_OBJECT_ID">
         <value>ad6866e1-fb08-4e9b-a93b-5095a2563779</value>
       </id>
       <namespace>EHR::COMMON</namespace>
       <type>CONTRIBUTION</type>
     </contribution>
     <commit_audit>
       <system_id>CABOLABS_EHR</system_id>
       <committer xsi:type="PARTY_IDENTIFIED">
         <name>Dr. Pablo Pazos</name>
       </committer>
       <time_committed>
         <value>20140901T233114,065-0300</value>
       </time_committed>
       <change_type>
         <value>creation</value>
         <defining_code>
           <terminology_id>
             <value>openehr</value>
           </terminology_id>
           <code_string>249</code_string>
         </defining_code>
       </change_type>
     </commit_audit>
     <uid>
       <value>91cf9ded-e926-4848-aa3f-3257c1d89e37::EMR_APP::1</value>
     </uid>
     <data xsi:type="COMPOSITION" archetype_node_id="openEHR-EHR-COMPOSITION.test_all_datatypes.v1">
       <name>
         <value>Test all datatypes</value>
       </name>
       <archetype_details>
         <archetype_id>
           <value>openEHR-EHR-COMPOSITION.test_all_datatypes.v1</value>
         </archetype_id>
         <template_id>
           <value>Test all datatypes</value>
         </template_id>
         <rm_version>1.0.2</rm_version>
       </archetype_details>
       <language>
         <terminology_id>
           <value>ISO_639-1</value>
         </terminology_id>
         <code_string>es</code_string>
       </language>
       <territory>
         <terminology_id>
           <value>ISO_3166-1</value>
         </terminology_id>
         <code_string>UY</code_string>
```

```
        </territory>
        <category>
          <value>event</value>
          <defining_code>
            <terminology_id>
              <value>openehr</value>
            </terminology_id>
            <code_string>443</code_string>
          </defining_code>
        </category>
        <composer xsi:type="PARTY_IDENTIFIED">
          <name>Dr. Pablo Pazos</name>
        </composer>
        <context>
          <start_time>
            <value>20140901T232600,304-0300</value>
          </start_time>
          <setting>
            <value>Hospital Montevideo</value>
            <defining_code>
              <terminology_id>
                <value>openehr</value>
              </terminology_id>
              <code_string>229</code_string>
            </defining_code>
          </setting>
        </context>
        <content xsi:type="OBSERVATION" archetype_node_id="openEHR-EHR-
OBSERVATION.test_all_datatypes.v1">
          <name>
            <value>Blood Pressure</value>
          </name>
          <language>
             <terminology_id>
               <value>ISO_639-1</value>
             </terminology_id>
             <code_string>es</code_string>
           </language>
           <encoding>
             <terminology_id>
               <value>UNICODE</value>
             </terminology_id>
             <code_string>UTF-8</code_string>
          </encoding>
          <subject xsi:type="PARTY_IDENTIFIED">
            <external_ref>
              <id xsi:type="HIER_OBJECT_ID">
                <value>[PATIENT_UID]</value>
              </id>
              <namespace>DEMOGRAPHIC</namespace>
              <type>PERSON</type>
            </external_ref>
          </subject>
          <data xsi:type="HISTORY" archetype_node_id="at0001">
            <name>
              <value>history</value>
            </name>
            <origin>
              <value>20140101</value>
            </origin>
            <events xsi:type="POINT_EVENT" archetype_node_id="at0002">
              <name>
                <value>any event</value>
              </name>
              <time><value>20140101</value></time>
              <data xsi:type="ITEM_TREE" archetype_node_id="at0003">
                <name>
                  <value>Arbol</value>
                </name>
                <items xsi:type="ELEMENT" archetype_node_id="at0011">
                  <name>
                    <value>Count</value>
                  </name>
                  <value xsi:type="DV_COUNT">
                    <magnitude>3</magnitude>
                  </value>
                </items>
              </data>
            </events>
          </data>
        </content>
      </data>
      <lifecycle_state>
        <value>completed</value>
        <defining_code>
          <terminology_id>
            <value>openehr</value>
          </terminology_id>
          <code_string>532</code_string>
        </defining_code>
      </lifecycle_state>
    </version>
</versions>
```

You can find this XML at: https://github.com/ppazos/cabolabs-ehrserver/blob/master/test/resources/commit/test_commit_1.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.

Sample JSON

```
{
     "versions" : {
             "@xmlns:xsi" : "http://www.w3.org/2001/XMLSchema-instance",
             "@xmlns" : "http://schemas.openehr.org/v1",
             "version" : {
                     "@xsi:type" : "ORIGINAL_VERSION",
                     "contribution" : {
                             "id" : {
                                     "@xsi:type" : "HIER_OBJECT_ID",
                                     "value" : "f65421a5-6698-4603-976e-aa3dc5413534"
                             },
                             "namespace" : "EHR::COMMON",
        "type" : "CONTRIBUTION"
    },
    "commit_audit" : {
        "system_id" : "CABOLABS_EHR",
        "committer" : {
            "@xsi:type" : "PARTY_IDENTIFIED",
            "external_ref" : {
                "id" : {
                    "@xsi:type" : "HIER_OBJECT_ID",
                    "value" : "cc193f71-f5fe-438a-87f9-81e74302eede"
                },
                "namespace" : "DEMOGRAPHIC",
                "type" : "PERSON"
            },
            "name" : "Dr. House"
        },
        "time_committed" : {
            "value" : "20161005T022250,000-0300"
        },
        "change_type" : {
            "value" : "creation",
            "defining_code" : {
                "terminology_id" : {
                    "value" : "openehr"
                },
                "code_string" : 249
            }
        }
    },
    "uid" : {
        "value" : "0c708601-558a-4d34-9694-fcd764713f13::EMR::1"
    },
    "data" : {
        "@archetype_node_id" : "openEHR-EHR-COMPOSITION.signos.v1",
        "@xsi:type" : "COMPOSITION",
        "name" : {
            "value" : "Signos vitales"
        },
        "uid" : {
            "@xsi:type" : "HIER_OBJECT_ID",
            "value" : "2442ad59-f5e9-4a73-9ee5-7488015074d4"
        },
        "archetype_details" : {
            "archetype_id" : {
                "value" : "openEHR-EHR-COMPOSITION.signos.v1"
            },
            "template_id" : {
                "value" : "Signos"
            },
            "rm_version" : "1.0.2"
        },
        "language" : {
            "terminology_id" : {
                "value" : "ISO_639-1"
            },
            "code_string" : "es"
        },
        "territory" : {
            "terminology_id" : {
                "value" : "ISO_3166-1"
            },
            "code_string" : "UY"
        },
        "category" : {
            "value" : "event",
            "defining_code" : {
                "terminology_id" : {
                    "value" : "openehr"
                },
                "code_string" : 433
            }
        },
        "composer" : {
```

```
        "@xsi:type" : "PARTY_IDENTIFIED",
        "external_ref" : {
                "id" : {
                        "@xsi:type" : "HIER_OBJECT_ID",
                        "value" : "cc193f71-f5fe-438a-87f9-81ecb302eede"
                },
                "namespace" : "DEMOGRAPHIC",
                "type" : "PERSON"
        },
        "name" : "Dr. House"
},
"context" : {
        "start_time" : {
                "value" : "20161005T022250,000-0300"
        },
        "setting" : {
                "value" : "Hospital Montevideo",
                "defining_code" : {
                        "terminology_id" : {
                                "value" : "openehr"
                        },
                        "code_string" : 229
                }
        }
},
"content" : [ {
        "@archetype_node_id" : "openEHR-EHR-OBSERVATION.blood_pressure.v1",
        "@xsi:type" : "OBSERVATION",
        "name" : {
                "value" : "Blood Pressure"
        },
        "language" : {
                "terminology_id" : {
                        "value" : "ISO_639-1"
                },
                "code_string" : "es"
        },
        "encoding" : {
                "terminology_id" : {
                        "value" : "Unicode"
                },
                "code_string" : "UTF-8"
        },
        "subject" : {
                "@xsi:type" : "PARTY_SELF"
        },
        "protocol" : {
                "@archetype_node_id" : "at0011",
                "@xsi:type" : "ITEM_TREE",
                "name" : {
                        "value" : "Tree"
                }
        },
        "data" : {
                "@archetype_node_id" : "at0001",
                "@xsi:type" : "HISTORY",
                "name" : {
                        "value" : "history"
                },
                "origin" : {
                        "@xsi:type" : "DV_DATE_TIME",
                        "value" : "20161005T022250,000-0300"
                },
                "events" : {
                        "@archetype_node_id" : "at0006",
                        "@xsi:type" : "POINT_EVENT",
                        "name" : {
                                "value" : "any event"
                        },
                        "time" : {
                                "@xsi:type" : "DV_DATE_TIME",
                                "value" : "20161005T022250,000-0300"
                        },
			"data" : {
				"@archetype_node_id" : "at0003",
				"@xsi:type" : "ITEM_TREE",
				"name" : {
					"value" : "blood pressure"
				},
				"items" : [ {
					"@archetype_node_id" : "at0005",
					"@xsi:type" : "ELEMENT",
					"name" : {
						"value" : "Diastolic"
					},
					"value" : {
						"@xsi:type" : "DV_QUANTITY",
						"magnitude" : 76,
						"units" : "mm[Hg]"
					}
				}, {
					"@archetype_node_id" : "at0004",
					"@xsi:type" : "ELEMENT",
					"name" : {
						"value" : "Systolic"
					},
					"value" : {
						"@xsi:type" : "DV_QUANTITY",
						"magnitude" : 126,
						"units" : "mm[Hg]"
					}
				} ]
			},
			"state" : {
				"@archetype_node_id" : "at0007",
				"@xsi:type" : "ITEM_TREE",
				"name" : {
					"value" : "state structure"
				}
			}
		}
	}
}, {
	"@archetype_node_id" : "openEHR-EHR-OBSERVATION.body_temperature.v1",
	"@xsi:type" : "OBSERVATION",
	"name" : {
		"value" : "Body temperature"
	},
	"language" : {
		"terminology_id" : {
			"value" : "ISO_639-1"
		},
		"code_string" : "es"
	},
	"encoding" : {
		"terminology_id" : {
			"value" : "Unicode"
		},
		"code_string" : "UTF-8"
	},
	"subject" : {
		"@xsi:type" : "PARTY_SELF"
	},
	"protocol" : {
		"@archetype_node_id" : "at0020",
		"@xsi:type" : "ITEM_TREE",
		"name" : {
			"value" : "Protocol"
		}
	},
	"data" : {
		"@archetype_node_id" : "at0002",
		"@xsi:type" : "HISTORY",
		"name" : {
			"value" : "History"
		},
		"origin" : {
			"@xsi:type" : "DV_DATE_TIME",
			"value" : "20161005T022250,000-0300"
		},
		"events" : {
			"@archetype_node_id" : "at0003",
			"@xsi:type" : "POINT_EVENT",
			"name" : {
				"value" : "Any event"
			},
			"time" : {
				"@xsi:type" : "DV_DATE_TIME",
				"value" : "20161005T022250,000-0300"
			},
			"data" : {
				"@archetype_node_id" : "at0001",
				"@xsi:type" : "ITEM_TREE",
				"name" : {
					"value" : "Tree"
				},
				"items" : {
					"@archetype_node_id" : "at0004",
					"@xsi:type" : "ELEMENT",
					"name" : {
						"value" : "Temperature"
					},
					"value" : {
						"@xsi:type" : "DV_QUANTITY",
						"magnitude" : 36,
						"units" : "C"
					}
				}
			},
			"state" : {
				"@archetype_node_id" : "at0029",
				"@xsi:type" : "ITEM_TREE",
				"name" : {
					"value" : "State"
				}
			}
		}
	}
}, {
	"@archetype_node_id" : "openEHR-EHR-OBSERVATION.pulse.v1",
	"@xsi:type" : "OBSERVATION",
	"name" : {
		"value" : "Pulso"
	},
	"language" : {
		"terminology_id" : {
			"value" : "ISO_639-1"
		},
		"code_string" : "es"
	},
	"encoding" : {
		"terminology_id" : {
			"value" : "Unicode"
		},
		"code_string" : "UTF-8"
	},
	"subject" : {
		"@xsi:type" : "PARTY_SELF"
	},
	"protocol" : {
		"@archetype_node_id" : "at0010",
		"@xsi:type" : "ITEM_TREE",
		"name" : {
			"value" : "*List(en)"
		}
	},
	"data" : {
		"@archetype_node_id" : "at0002",
		"@xsi:type" : "HISTORY",
        "name" : {
            "value" : "*history(en)"
        },
        "origin" : {
            "@xsi:type" : "DV_DATE_TIME",
            "value" : "20161005T022250,000-0300"
        },
        "events" : {
            "@archetype_node_id" : "at0003",
            "@xsi:type" : "POINT_EVENT",
            "name" : {
                "value" : "*Any event(en)"
            },
            "time" : {
                "@xsi:type" : "DV_DATE_TIME",
                "value" : "20161005T022250,000-0300"
            },
            "data" : {
                "@archetype_node_id" : "at0001",
                "@xsi:type" : "ITEM_TREE",
                "name" : {
                    "value" : "*structure(en)"
                },
                "items" : {
                    "@archetype_node_id" : "at0004",
                    "@xsi:type" : "ELEMENT",
                    "name" : {
                        "@xsi:type" : "DV_CODED_TEXT",
                        "value" : "Frecuencia cardiaca",
                        "defining_code" : {
                            "terminology_id" : {
                                "value" : "local"
                            },
                            "code_string" : "at1027"
                        }
                    },
                    "value" : {
                        "@xsi:type" : "DV_QUANTITY",
                        "magnitude" : 82,
                        "units" : "/min"
                    }
                }
            },
            "state" : {
                "@archetype_node_id" : "at0012",
                "@xsi:type" : "ITEM_TREE",
                "name" : {
                    "value" : "*List(en)"
                }
            }
        }
    }
}, {
    "@archetype_node_id" : "openEHR-EHR-OBSERVATION.respiration.v1",
    "@xsi:type" : "OBSERVATION",
    "name" : {
        "value" : "Respirations"
    },
    "language" : {
        "terminology_id" : {
            "value" : "ISO_639-1"
        },
        "code_string" : "es"
    },
    "encoding" : {
        "terminology_id" : {
            "value" : "Unicode"
        },
        "code_string" : "UTF-8"
    },
    "subject" : {
        "@xsi:type" : "PARTY_SELF"
    },
        "data" : {
            "@archetype_node_id" : "at0001",
            "@xsi:type" : "HISTORY",
            "name" : {
                "value" : "history"
            },
            "origin" : {
                "@xsi:type" : "DV_DATE_TIME",
                "value" : "20161005T022250,000-0300"
            },
            "events" : {
                "@archetype_node_id" : "at0002",
                "@xsi:type" : "POINT_EVENT",
                "name" : {
                    "value" : "Any event"
                },
                "time" : {
                    "@xsi:type" : "DV_DATE_TIME",
                    "value" : "20161005T022250,000-0300"
                },
                "data" : {
                    "@archetype_node_id" : "at0003",
                    "@xsi:type" : "ITEM_TREE",
                    "name" : {
                        "value" : "List"
                    },
                    "items" : {
                        "@archetype_node_id" : "at0004",
                        "@xsi:type" : "ELEMENT",
                        "name" : {
                            "value" : "Rate"
                        },
                        "value" : {
                            "@xsi:type" : "DV_QUANTITY",
                            "magnitude" : 26,
                            "units" : "/min"
                        }
                    }
                },
                "state" : {
                    "@archetype_node_id" : "at0022",
                    "@xsi:type" : "ITEM_TREE",
                    "name" : {
                        "value" : "List"
                    }
                }
            }
        }
    }, {
        "@archetype_node_id" : "openEHR-EHR-OBSERVATION.body_weight.v1",
        "@xsi:type" : "OBSERVATION",
        "name" : {
            "value" : "Peso corporal"
        },
        "language" : {
            "terminology_id" : {
                "value" : "ISO_639-1"
            },
            "code_string" : "es"
        },
        "encoding" : {
            "terminology_id" : {
                "value" : "Unicode"
            },
            "code_string" : "UTF-8"
        },
        "subject" : {
            "@xsi:type" : "PARTY_SELF"
        },
        "protocol" : {
            "@archetype_node_id" : "at0015",
            "@xsi:type" : "ITEM_TREE",
            "name" : {
```

```
            "value" : "*protocol structure(en)"
        }
    },
    "data" : {
        "@archetype_node_id" : "at0002",
        "@xsi:type" : "HISTORY",
        "name" : {
            "value" : "*history(en)"
        },
        "origin" : {
            "@xsi:type" : "DV_DATE_TIME",
            "value" : "20161005T022250,000-0300"
        },
        "events" : {
            "@archetype_node_id" : "at0003",
            "@xsi:type" : "POINT_EVENT",
            "name" : {
                "value" : "Cualquier evento."
            },
            "time" : {
                "@xsi:type" : "DV_DATE_TIME",
                "value" : "20161005T022250,000-0300"
            },
            "data" : {
                "@archetype_node_id" : "at0001",
                "@xsi:type" : "ITEM_TREE",
                "name" : {
                    "value" : "*Simple(en)"
                },
                "items" : {
                    "@archetype_node_id" : "at0004",
                    "@xsi:type" : "ELEMENT",
                    "name" : {
                        "value" : "Peso"
                    },
                    "value" : {
                        "@xsi:type" : "DV_QUANTITY",
                        "magnitude" : 81,
                        "units" : "kg"
                    }
                }
            },
            "state" : {
                "@archetype_node_id" : "at0008",
                "@xsi:type" : "ITEM_TREE",
                "name" : {
                    "value" : "*state structure(en)"
                }
            }
        }
    }
}, {
    "@archetype_node_id" : "openEHR-EHR-OBSERVATION.height.v1",
    "@xsi:type" : "OBSERVATION",
    "name" : {
        "value" : "Height/Length"
    },
    "language" : {
        "terminology_id" : {
            "value" : "ISO_639-1"
        },
        "code_string" : "es"
    },
    "encoding" : {
        "terminology_id" : {
            "value" : "Unicode"
        },
        "code_string" : "UTF-8"
    },
    "subject" : {
        "@xsi:type" : "PARTY_SELF"
    },
    "protocol" : {
```

```
                        "@archetype_node_id" : "at0007",
                        "@xsi:type" : "ITEM_TREE",
                        "name" : {
                            "value" : "List"
                        }
                    },
                    "data" : {
                        "@archetype_node_id" : "at0001",
                        "@xsi:type" : "HISTORY",
                        "name" : {
                            "value" : "history"
                        },
                        "origin" : {
                            "@xsi:type" : "DV_DATE_TIME",
                            "value" : "20161005T022250,000-0300"
                        },
                        "events" : {
                            "@archetype_node_id" : "at0002",
                            "@xsi:type" : "POINT_EVENT",
                            "name" : {
                                "value" : "Any event"
                            },
                            "time" : {
                                "@xsi:type" : "DV_DATE_TIME",
                                "value" : "20161005T022250,000-0300"
                            },
                            "data" : {
                                "@archetype_node_id" : "at0003",
                                "@xsi:type" : "ITEM_TREE",
                                "name" : {
                                    "value" : "Simple"
                                },
                                "items" : {
                                    "@archetype_node_id" : "at0004",
                                    "@xsi:type" : "ELEMENT",
                                    "name" : {
                                        "value" : "Height/Length"
                                    },
                                    "value" : {
                                        "@xsi:type" : "DV_QUANTITY",
                                        "magnitude" : 181,
                                        "units" : "cm"
                                    }
                                }
                            },
                            "state" : {
                                "@archetype_node_id" : "at0013",
                                "@xsi:type" : "ITEM_TREE",
                                "name" : {
                                    "value" : "Tree"
                                }
                            }
                        }
                    }
                } ]
            },
            "lifecycle_state" : {
                "value" : "completed",
                "defining_code" : {
                    "terminology_id" : {
                        "value" : "openehr"
                    },
                    "code_string" : 532
                }
            }
        }
    }
}
```

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",
                    "conditions": {
                        "magnitude": {
                            "gt": [
                                140.0
                            ]
                        },
                        "units": {
                            "eq": "mm[Hg]"
                        }
                    }
                },
                {
                    "archetypeId": "openEHR-EHR-OBSERVATION.blood_pressure.v1",
                    "path": "/data[at0001]/events[at0006]/data[at0003]/items[at0005]/value",
                    "conditions": {
                        "magnitude": {
                            "gt": [
                                90.0
                            ]
                        },
                        "units": {
                            "eq": "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": [
        {
            "uid": "0f78e043-aa09-4212-9669-fcef0adaf470",
            "category": "event",
            "startTime": "2016-06-25 07:29:28",
            "subjectId": "11111111-1111-1111-1111-111111111111",
            "ehrUid": "11111111-1111-1111-1111-111111111111",
            "templateId": "Signos",
            "archetypeId": "openEHR-EHR-COMPOSITION.signos.v1",
            "lastVersion": true,
            "organizationUid": "d04809ca-08dc-454a-8390-96a0b125abf1",
            "parent": "90120202-e7a6-4032-a935-fe91f6e7fd28::EMR::1"
        },
        ...
    ],
    "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

GET /checkout

Gets the latest version of a clinical document (composition) with the aim of creating a new version of it (due a correction or an amendment of the information contained in it).

Parameters:

- ehrUid: identifier of the EHR that contains the composition.
- compositionUid: identifier of the composition to be modified, should be the latest version of the document (use GET /compositions to get the UIDs of the last versions of existing documents)

Sample Result: (Content-Type: text/xml)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<versions xmlns="http://schemas.openehr.org/v1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
   <version xsi:type="ORIGINAL_VERSION">
     <contribution>
       <id xsi:type="HIER_OBJECT_ID">
         <value>ad6866e1-fb08-4e9b-a93b-5095a2563779</value>
       </id>
       <namespace>EHR::COMMON</namespace>
       <type>CONTRIBUTION</type>
     </contribution>
     <commit_audit>
       <system_id>CABOLABS_EHR</system_id>
       <committer xsi:type="PARTY_IDENTIFIED">
         <name>Dr. Pablo Pazos</name>
       </committer>
       <time_committed>
         <value>20140901T233114,065-0300</value>
       </time_committed>
       <change_type>
         <value>creation</value>
         <defining_code>
           <terminology_id>
             <value>openehr</value>
           </terminology_id>
           <code_string>249</code_string>
         </defining_code>
       </change_type>
     </commit_audit>
     <uid>
       <value>91cf9ded-e926-4848-aa3f-3257c1d89e37::EMR_APP::1</value>
     </uid>
     ...
```

Notes:

The returned XML will have the same structure as the documents sent in the request to POST /commit. For now /checkout supports XML only results, in the future we will add JSON support to this endpoint.

When the document is modified on a client application, it should be committed as it is (the version.uid should not be changed by the client app), and the EHRServer will generate the new version for the document, and associate the new version with the previous one, in the POST /commit call.

GET /organizations

Get the information of the organizations associated with the authenticated user.

Parameters:
- format: result format, xml or json

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

```
[
    {
        "uid": "4f9cfbe4-8cdd-43fc-b0de-5544305fcdd5",
        "name": "Hospital de Clinicas"
    },
    {
        "uid": "4ea72e8c-9707-4e1e-8acc-9e10e61b2037",
        "name": "Clinica del Tratamiento del Dolor"
    }
]
```

POST /users

Register a new user for the current organization, through the API. For security reasons, the passwords can't be set though the API and should be set by each user. The organization is taken from the Authorization Token.

Parameters:
- username: username for the new user, spaces are not allowed
- email: email of the new user, a notification to reset the password will be sent to this email
- format: result format, xml or json

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

```
{
    "username": "pablo.pazos",
    "email": "pablo.pazos@cabolabs.com",
    "organizations": [
        {
            "uid": "67559293-738c-4597-b1b0-68b6da0d8f32",
            "name": "Hospital Better Health",
            "number": "456789"
```

```
          }
      ]
}
```