API REQUIREMENTS

PROOF OF CONCEPT CHUV

Step	Technical Conception
Date	12/01/2015
Author	Florent PERINEL
Version	1.3
Status	Final

Summary

Reference documents

Document	Version / Date	Description

1. INTRODUCTION

1.1. PURPOSE OF THE DOCUMENT

This document is part of the “Proof of concept” of web interface for Human Brain Project.

The purpose of this document is to define contraints and requirements for “CHUV API”.

Without specifications provided by the CHUV, this document will describe how the API will be requested by the middleware and how data must be returned to the frontend to generate charts and tables.

1.2. DEVELOPMENT PRIORITIES

Priority	API method
1	Request without asynchronous processing (chapter: 3.2.4.1)
2	Retrieve a list of all variables (chapter: 3.2.2.1)
3	Other variables method (chapters: 3.2.2.1, 3.2.2.1)
4	Group method (chapter: 3.2.1.1)
5	Values methods (chapters: 3.2.3.1, 3.2.3.2)
6	Request with asynchronous processing (chapter: 3.2.4.2)

2. TECHNICAL REQUIREMENTS

2.1. COMMUNICATION

The API must be a REST API with a JSON exchange format.

2.2. DATA FORMAT

The data format is defined for both requests and results. A wrong data format will cause an error.

Data type	Format	Example
Text	All characters are allowed. The charset must be UTF-8.	-
Integer	Values 0 to 9 and "-" can be used. No decimal part will be accepted, no thousand separator. The value must be prefixed by "-" if negative.	1234 or -1234
Numeric	Values 0 to 9, "." and "-" can be used. A decimal part is accepted, no thousand separator. The value must be prefixed by "-" if negative.	1234.567 or -1234.567
Date	Date must be formatted in ISO8601	2015-08-28T11:13:00Z
Boolean	The boolean value must be represented by "0" for "false" value or "1" for "true" value.	"0" or "1"

2.1. REQUEST METHODS

Action	Resource specified	No resource specified
GET	Return the resource	Return several resources
POST	Error (405)	Create a new resource
PUT	Update one resource	Update several resources
PATCH	Partialy update one resource (only sent data)	Partialy update several resources (only sent data)
DELETE	Delete one resource	Delete several resources
HEAD	Return http header only	Return http header only

2.2. HTTP RESPONSE STATUS CODE

The HTTP response status codes are used to codify response type.

Code	GET	POST	PUT/PATCH	DELETE
SUCCES				
200	Success	Successfully created	Successfully updated	Successfully deleted
201	-	Successfully created	Successfully updated	-
202	-	Request will be executed by an asynchronous process		
204	No resources found	-	-	Successfully deleted
CLIENT ERRORS				
400	Illegal parameter			
401	Authentication required			
403	Action unauthorized			
404	Resource not found	-	Resource to update not found	Resource to delete not found
405	Method not allowed			
SERVER ERRORS				
500	Generic error on the server side processing			
501	Functionality not implemented			
503	Temporaly unavailable			

2.3. RESPONSE

The API response can take many forms.

- A business data (described in the chapter: 3.1).
- An error message (mandatory with a 4xx or 5xx HTTP code).
- An asynchronous token.

2.3.1. ERROR MESSAGE

Attribute	Type/Format	Description
errorCode	Text	Unique error code
errorType	Text	Error type
time	Datetime	Hours of error
message	Text	Message
detail	Text	Detail
request	Text	URL called at the outbreak of the error

Example:

```
{
  "errorCode": "404.123",
  "errorType": "entityNotFoundException",
  "time": "2014-05-27T10:16:00,4Z",
  "message": "User not found",
  "detail": "User 'test' not found in database with given arguments",
  "request": "GET BASE/api/users/users.json/test"
}
```

This message is displayed to the end user. The attribute “message” is the title, “detail” will be the content.

2.3.2. ASYNCHRONOUS TOKEN

Attribute	Type/Format	Description
token	Text	Unique token
Progress	Integer	A value from 0 to 100 indicating current progression
asyncUrl	Text	URL to get asynchronous operation progression
resultUrl	Text	URL to get result after asynchronous operation (returned if asynchronous process is finished).

Example:

```
{
  "token": "123456798123456798",
  "progress": 0,
  "asyncUrl": "<BASE>/requests/DS123456",
  "resultUrl": "<BASE>/datasets/DS123456"
}
```

When a query is by an asynchronous process, an “asynchronous token” will be returned. This token can be used to query the API, get the current progression and the final results when the pocess is done.

The token must be unique, “asyncURL” represents API URL to use for request about progression or to get the result.

Cinematic:

```
> POST: <BASE>/requests
> CONTENT: {object_query}

< HTTP: 202
< CONTENT:
< {
<   "token": "DS123456",
<   "progress": 0,
<   "asyncUrl": "<BASE>/requests/DS123456"
< }

> GET: <BASE>/requests/DS123456

< HTTP: 202
< CONTENT:
< {
<   "token": "DS123456",
<   "progress": 40,
<   "asyncUrl": "<BASE>/requests/DS123456"
< }

> GET: <BASE>/requests/DS123456

< HTTP: 202
< CONTENT:
< {
<   "token": "DS123456",
<   "progress": 100,
<   "asyncUrl": "<BASE>/requests/DS123456",
<   "resultUrl": "<BASE>/datasets/DS123456"
< }

> GET: <BASE>/datasets/DS123456

< HTTP: 200
< CONTENT:
< {object_dataset}
```

2.4. SECURITY

For the POC, the security rules will not be implemented.

API will be consumed without authentication. A limitation can be applied by IP.

3. FUNCTIONNAL REQUIREMENT

3.1. OBJECTS

In objects, a “null” attribute can be omitted in the response.

3.1.1. GROUP

A “group” object represents a variable scope. Each variable is associated to a group. Each group can be contained in other group. The group chaining can be interpreted like a hierarchy.

Attribute	Type	Description
code	Text	Unique group code
label	Text	Group label
groups	group[]	Sub groups

Example:

```
{
  "code": "brain_anatomy",
  "label": "Brain",
  "groups": [
    {
      "code": "grey_matter",
      "label": "Grey matter"
    }, {
      "code": "white_matter",
      "label": "White matter"
    }
  ]
}
```

3.1.2. VARIABLES

A “variable” object represents a business variable. All variable information should be stored in this object.

By default, variable’s values aren’t returned in a “variable” object.

Attribute	Type	Description
code	Text	Unique variable code, used to request
label	Text	Variable label, used to display
group	group	Variable group (only the variable path)
type	Text	I: Integer, T: Text, N: Decimal, D: Date, B: Boolean.
length	Integer	For text, number of characters of value
minValue	Numeric	Minimum allowed value (for integer or numeric)
maxValue	Numeric	Maximum allowed value (for integer or numeric)
units	Text	Variable unit
isVariable	Boolean	Can the variable can be used as a “variable”
isGrouping	Boolean	Can the variable can be used as a “group”
isCovariable	Boolean	Can the variable can be used as a “covariable”
isFilter	Boolean	Can the variable can be used as a “filter”
values	value[]	List of variable values (if is an enumeration variable).

Sub-object “Value”:

A “value” object is a business variable value. All value information sould be stored in this object.

Attribute	Type	Description
code	Text	Unique code of value (for variable), used to request
label	Text	Label of value, used to display

Example:

```
{
  "code": "LeftPallidum",
  "label": "Left pallidum",
  "group": {
    "code": "brain_anatomy",
    "label": "Brain",
    "groups": [ { "code": "grey_matter", "label": "Grey matter" } ]
  },
  "type": "N",
  "length": null,  // can be omitted
  "minValue": -5,
  "maxValue": 100,
  "units": "cm3",
  "isVariable": 1,
  "isGrouping": 0,
  "isCovariable": 1,
  "isFilter": 1
}
```

Example with enumerator:

```
{
  "code": "COLPROT",
  "label": "Protocol",
  "group": {
    "code": "provenance ",
    "label": "Provenance",
    "groups": [ { "code": "protocol", "label": "Protocol" } ]
  },
  "subgroup": "protocol",
  "type": "T",
  "isVariable": 0,
  "isGrouping": 1,
  "isCovariable": 0,
  "isFilter": 0,
  "values": [
    { "code": "ADNI1", "label": "ADNI1" },
    { "code": "ADNI2", "label": "ADNI2" },
    { "code": "ADNIGO", "label": "ADNIGO" }
  ]
}
```

3.1.3. DATASET

A “dataset” object contains all the request data.

Each data row must contain the same number of value. An empty value must be set to “null”.

3.1.3.1. Common usage (no boxplot)

Attribute	Type	Description
code	Text	Unique code of dataset
date	Datetime	Date of dataset generation
header	Text array	List of variable code used in array header
data	data[]	List of data rows

Example:

```
{
  "code": "DS123456",
  "date": "2015-08-28T11:13:00Z",
  "header": [
    "PTGENDER",
    "RightPOparietaloperculum",
    "LeftPOparietaloperculum",
    "RightPoGpostcentralgyrus",
    "LeftPoGpostcentralgyrus"
  ],
  "data": {
    "PTGENDER": ["Male", "Female"],
    "RightPOparietaloperculum": [2.2697785, 1.8678768],
    "LeftPOparietaloperculum": [null, 1.9764309],
    "RightPoGpostcentralgyrus": [10.07747, null],
    "LeftPoGpostcentralgyrus": [11.172205, 10.501977]
  }
}
```

3.1.3.2. Boxplot dataset

Attribute	Type	Description
code	Text	Unique code of dataset
date	Datetime	Date of dataset generation
header	Text array	List of variable code used in array header
data	data{}	List of data rows : - data[header[i=0]]: List of values for the first header. - data[header[i > 0]]: Two dimension array contains list of bloxplot values (q1,q2,min,max,etc..)

```
{
  "code":"hfyui345tjc4scckg80gcwc8k00ocww",
"date":"2015-11-30T13:58:54+0100",
"header":[
  "DX_bl",
  "SUV Frontal"
],
"data":{
  "DX_bl":[
    "AD.0",
    "CN.0",
    "EMCI.0",
    "LMCI.0",
    "AD.1",
    "CN.1",
    "EMCI.1",
    "LMCI.1"
  ],
  "SUV Frontal":[
    [
      0.92081,
      1.1093,
      1.2928,
      1.5984,
      1.8254
    ],
    [
      0.9052,
      1.10335,
      1.1927,
      1.3523,
      1.7213
    ],
    [
      0.9579,
  1.10385,
  1.19735,
  1.3147,
  1.6147
],
[
  0.84847,
  1.05045,
  1.2031,
  1.45215,
  2.0273
],
[
  1.288,
  1.403,
  1.628,
  1.7161,
  1.923
],
[
  0.92851,
  1.21755,
  1.4059,
  1.69775,
  2.093
],
[
  0.9038,
  1.22315,
  1.4284,
  1.62935,
  2.1281
],
      [
        1.0255,
        1.42315,
        1.59675,
        1.76125,
        2.1394
      ]
    ]
  }
}
```

3.1.4. REQUEST

A “request” object contains the plot type (for poc usage only).

Each data row must contain the same number of value. An empty value must be set to “null”.

Attribute	Type	Description
plot	Text	Plot type. Possibles values are: - column - line - scatter - pie - boxplot

Example:

```
{
  "plot": "boxplot"
}
```

3.1.5. QUERY

A “query” object represents a request to the CHUV API.

This object contains all information required by the API to compute a result (dataset) and return it.

Attribute	Type	Description
variables	variable[]	List of variables used by the request, only “code” values are sent.
covariables	variable[]	List of covariables used by the request, only “code” values are sent. These variables are returned in dataset object header.
grouping	variable[]	List of grouping variables used by the request, only “code” values are sent.
filters	filter[]	List of filters objects used by the request
request	request{}	~~Request in “select” clause (reserved for future usage)~~ For the poc usage, this attribute will contain the plot type.

Sub-object “filter”:

Attribute	Type	Description
variable	variable	Variable used to filter, only “code” value is sent.
operator	text	Filter operator : “eq”, “lt”, “gt”, “gte”, “lte”, “neq”, “in”, “notin”, “between”.
values	text[]	List of values used to filter With operators “eq”, “lt”, “gt”, “gte”, “lte”, “neq”, the filter mode “OR” is used. With operator “between”, only two values are sent, they represents the range limits.

Operators:

- “eq”: Equal to
- “lt”: Less than
- “gt”: Greater than
- “gte”: Greater than or equal to
- “lte”: Less than or equal to
- “neq”: Not equal to
- “in”: Contained in a set of values
- “notin”: Not contained in a set of values
- “between”: In a value range

Example:

```
{
  "variables": [
    {"code": "Hippocampus"}
  ],
  "covariables": [
    {"code": "Hippocampus"}, {"code": "AGE"}, {"code": "PTEDUCAT"}
  ],
  "grouping": [
    {"code": "SITE"}, {"code": "DX"}
  ],
  "filters": [
    {
      "variable": {"code": "PTEDUCAT"},
      "operator": "in",
      "values": [18, 16, 12]
    },{
      "variable": {"code": "AGE"},
      "operator": "between",
      "values": [50, 80]
    },{
      "variable": {"code": "PTRACCAT"},
      "operator": "eq",
      "values": ["Black"]
    }
  ],
  "request": "3rdVentricle+4thVentricle as 3rdAnd4thVentricle"
}
```

3.2. METHODS

3.2.1. RETRIEVE VARIABLE GROUPS

3.2.1.1. All groups

This method is used to get all group available with their sub groups.

Method: GET

URL: <BASE>/groups

Parameters: None

Response:

Attribute	Type	Description
Groups	Group	Main group (root) and his sub groups.

Example:

```
GET: <BASE>/groups

RESPONSE: 200
{object_group#1}
```

{object_group}: See chapter 3.1.1

3.2.2. RETRIEVE A LIST OF VARIABLES

3.2.2.1. All variables

This method is used to get all variables available with their values.

Method: GET

URL: <BASE>/variables

Parameters: None

Response:

Attribute	Type	Description
Variables	variable[]	List of variable objects

Example:

```
GET: <BASE>/variables

RESPONSE: 200
[
  {object_variable#1},
  {object_variable#2},
  {object_variable#3},
  {object_variable#4},
  {object_variable#5}
]
```

{object_variable}: See chapter 3.1.2

3.2.2.1. Specific variables

This method can be used to get a variable by its code. Multiple values can be separated by “,”. The method returns the variables values.

Method: GET

URL: <BASE>/variables/<code>[,<code>]

Parameters:

- <code> : Value of attribute “code” of object “variable” (multiple values is allowed, separated by “,”).

Response:

Attribute	Type	Description
Variables	variable[]	List of variable objects

Example:

```
GET: <BASE>/variables/PTETHCAT,PTRACCAT,PTMARRY

RESPONSE: 200
[
  {object_variable#1},
  {object_variable#2},
  {object_variable#3}
]
```

{object_variable}: See chapter 3.1.2

3.2.2.1. Specific variables by attribute

This method can be used to get a list of variables by their attribute value (“group”, “isVariable”, “isGrouping”, “isCovariable”, “isFilter”).

It’s possible to separate multiple values by “,”. The method returns the variables values.

Method: GET

URL: <BASE>/variables/?<attribute>=(<value>[,<value>])

- <attribute>: Attribute name of “variable” object
- <value>: Value of “attribute” of object “variable” to use to filter (multiple values is allowed, separated by “,”).

Response:

Attribute	Type	Description
Variables	variable[]	List of variable objects

Example:

```
GET: <BASE>/variables/?group=("provenance")&subgroup=("protocol","source")&isGrouping=("1")

RESPONSE: 200
[
  {object_variable#1},
  {object_variable#2}
]
```

{object_variable}: See chapter 3.1.2

3.2.3. RETRIEVE VALUES OF VARIABLES

3.2.3.1. All values

This method is used to get all values of a variable.

Method: GET

URL: <BASE>/variables/<code>/values

Parameters:

- <code> : Value of attribute “code” of object “variable”.

Response:

Attribute	Type	Description
Values	value[]	Values of variable

Example:

```
GET: <BASE>/variables/PTMARRY

RESPONSE: 200
[
  {object_value#1},
  {object_value#2},
  {object_value#3},
  {object_value#4}
]
```

{object_value}: See chapter 3.1.2

3.2.3.2. Filtered values

This method is used by autocomplete fields to get values of a variable.

Method: GET

URL: <BASE>/variables/<code>/values/?q=<term>

Parameters:

- <code> : Value of attribute “code” of object “variable”.
- <term> : Value of attribute “label” of object “value”.

Response:

Attribute	Type	Description
Values	value[]	Values of variable

Example:

```
GET: <BASE>/variables/PTMARRY/values/?q="Mar"

RESPONSE: 200
[
  {object_value#1},
  {object_value#2}
]
```

{object_value}: See chapter 3.1.2

3.2.4. REQUEST

3.2.4.1. Without asynchronous processing

This method is used to get a dataset from a request.

Method: POST

URL: <BASE>/requests

Parameters:

- <query> : Query to compute (passed in http request body).

Response:

Attribute	Type	Description
Values	value[]	Values of variable

Example:

```
POST: <BASE>/requests
BODY:
{object_query}

RESPONSE: 200
{object_dataset}
```

{object_query}: See chapter 3.1.1

{object_dataset}: See chapter 3.1.3

3.2.4.2. With asynchronous processing

This method is used to get a dataset from a request.

Method: POST

URL: <BASE>/requests

Parameters:

- <query> : Query to compute (passed in http request body).

Response:

Attribute	Type	Description
Values	value[]	Values of variable

Example:

```
POST: <BASE>/requests
BODY:
{object_query}

RESPONSE: 202
{object_asynchronous_token}
```

{object_query}: See chapter 3.1.1

{object_dataset}: See chapter 3.1.3