openbiomaps Documentation

Release 5.4

Bán Miklós

July 12, 2023

CONTENTS

Contents:

CHAPTER

ONE

INTRODUCTION

1.1 The objectives of the OpenBioMaps project

- Supporting the management of biodiversity data so that it can be used more effectively to conserve nature
- To maintain a free and openly accessible biological map database service
- Develop open source and freely usable biodiversity data management software tools
- Support biodiversity education and research

1.2 Main properties

- All OpenBioMaps services are free of charge.
- OpenBioMaps is designed to record the occurrence and associated data (biotic data) of living organisms (of conservation importance or useful for biodiversity research).
- OpenBioMaps has the explicit aim of supporting higher education and strengthening the links between research and conservation.
- OpenBioMaps is a decentralised database and framework of sub-databases based on open source applications, with minimal cost and no central control.
- The primary target audience for OpenBioMaps is the natural science and conservation profession, as well as those preparing plans, strategies and decisions affecting the environment.
- Users can create and manage databases with any structure and rules they wish.
- Easy uploading of data from different file formats (ods, xls, xlsx, gpx, shp, csv ...).
- Repeatable and quotable queries.
- Use of persistent identifiers (DOI) for both databases and queries.
- Data can be downloaded in various formats (shp, csv, gpx, json, ...)
- Data can be accessed from remote databases or desktop applications (e.g. R, QGIS).
- Integrable services.
- Data links with other databases.
- Create customizable data upload interfaces.
- Field mobile data collection application.
- Community-edited documentation.

- Interface translatable into any language (currently Hungarian, English, Romanian, Spanish and partly Russian)
- Community feedback based development.

1.3 OpenBioMaps overview

OBM tries to provide support for the entire data lifecycle, from data collection to organisation and use.

OBM Workflow

`Query scheme (pdf)` `Query scheme (odp)`

1.4 Getting started

To create your own database you will need a server. This can be your own server, a rented server, or a server already maintained by someone else to serve OpenBioMaps.

It is easiest to create a new database on an existing OpenBioMaps server. Check the list of known servers to see if you have access to one of them. There are dedicated public servers that host many different databases.

If you need more capacity or you want to control access to the entire server, you can install a dedicated server. It's not that complicated. Here is a tutorial: https://openbiomaps.org/documents/en/server_install.html

If you want to create your own database project on an existing server, you must have access to a database on that server. Once you have this, you can easily set up your own database project there, the steps for which you can see here: https://openbiomaps.org/documents/en/tutorials.html#new-project and here: https://openbiomaps.org/documents/en/new_project.html

1.4.1 OpenBioMaps Consortium

`OpenBioMaps Consortium Document`

CHAPTER

TWO

TUTORIALS

2.1 New server Installation with Docker

New server installation tutorial

2.2 User-generated video educational material

2.2.1 Upload data

Upload interface usage: Video tutorial (Balkanherps, Edvárd Mizsei) in English

Upload interface usage: Video tutorial (Milvus general forms) in Romanian

Upload interface usage: Video tutorial (Milvus atlas form) in Romanian

Upload interface usage: Video tutorial (Milvus nocturnal birds form) in Romanian

Adatfeltöltés fájlból (Duna-Dráva Nemzeti Park, Gáborik Ákos) in Hungarian

2.2.2 Log-in

Első bejelentkezés (Duna-Dráva Nemzeti Park, Gáborik Ákos) in Hungarian

Belépés és adatfeltöltés űrlapról (Duna-Dráva Nemzeti Park, Gáborik Ákos) in Hungarian

2.2.3 Mobile app

Mobilos adatgyűjtő alkalmazás használata (Duna-Dráva Nemzeti Park, Gáborik Ákos) in Hungarian

Kapcsolt listák készítése mobil alkalmazás számára (is) (Balatonfelvidéki Nemzeti Park, Antal Balázs)

Regisztráció és mobil alkalmazás (Balatonfelvidéki Nemzeti Park, Antal Balázs)

2.2.4 Data query and update

Adatlekérdezés, javítás (Duna-Dráva Nemzeti Park, Gáborik Ákos) in Hungarian

2.2.5 Creating/founding a new database project

Video tutorial () in English https:youtu.be/…

2.2.6 Data access using QGIS

OBM adatok használata QGIS-ben (Duna-Dráva Nemzeti Park, Gáborik Ákos) in Hungarian

CHAPTER

THREE

USER INTERFACES

3.1 Log in page

3.1.1 Forgotten password

By entering your registered email address here you can request a temporary login link, which will be sent to you by email.

3.1.2 Registration

Registration requires an invitation, which can be requested via the form available from the registration page. In addition, depending on your project settings, it is also possible to register using Google or another OpenID-based registration.

3.2 Profile page

Settings related to our profile.

3.2.1 Invitations

By default, all registered users can invite additional members.

For more information, visit the invitations page.

3.2.2 Messages

Internal messages are read and sent here. Our messaging system distinguishes between four message types:

- A system message can be sent by a project or a server process to an individual user. Such a message could be, for example, a message from a background validation process.
- A personal message can be sent between users. The message editing interface can be accessed by clicking the Write Message button at the bottom left of the Messages page.
- Comment Notification - This message type is sent when someone has commented on your data, uploads, or users.
- News - when uploading data, creating a new project, or sharing polygons, the system creates a news item for everyone to see.

The messages page is available to all logged-in users. Anyone can send messages to project members.

In addition to individual messages, project administrators can also send messages to groups or notify users of messages by email.

3.2.3 Creation of a new database

Any registered member can create a new database project, which he/she will own and will be completely independent of the project in which it was created.

More information about creating a database

3.3 Project administration

Project level settings. For example, upload forms and map settings, or user administration.

3.4 Map page

If you have map data and valid settings (SQL, Mapserver), you can view and query the map data from this subpage. Some map settings may differ significantly, for example, only the queried data may be displayed. There may be different base maps, e.g. grids, or aerial photos, and sampling locations on a map. You can display point, line, and polygon data (in separate layers). Several base maps can be selected (OSM is the default). In some projects, you can set up a Google base map if the project owners do so by entering some Google account data.

3.4.1 map queries

Spatial queries can be triggered by drawing on the map, tapping on the map (info module), or selecting pre-loaded geometries. When drawing a map, a buffer zone can be specified around the drawing pointer. That is, you can query a point by dropping a point in say a 500m radius circle, or around a line in a 10m zone.

3.4.2 text queries

Arbitrary text query options can be set up in each project to query data, which options can include a number of helper functions such as auto-text completion, list selection, date, time, date interval selection, multiple list item selection, etc…

3.4.3 query storage

The result of a query can be stored on the server, which can be referenced by a persistent keyword. A DOI identifier can be requested for these identifiers. The query can also be stored, which can be used to repeat the query.

3.5 Data upload

Any number of forms can be defined for a data table, with which forms different data can be loaded with different options. For example, some forms may be designed for mobile formatting only or for public data upload, while others may be designed specifically for a certain file type to be imported. At any time during the upload process, it is possible to save and download the upload status in CSV format.

3.5.1 File upload

Supported formats:

- Plain text files: CSV, DSV, TSV, json
- Image files: jpg, tiff (Exif columns are read out)
- Spreadsheet formats: ods (LibreOffice), xls (Excel), xlsx (Excel)
- Spatial formats: esri shape (.shp, .dbf, .cpg, .prj, .shx combined), gpx (GPS data format (xml)), SQLite
- Genetic data files: fasta

Any of the files listed here can be imported by entering a URL (simple GET query)

3.5.2 Web form filling

Data can also be uploaded using a web form. You can add any number of rows to your table when uploading. Data upload can be accelerated using various bulk cell upload functions.

3.5.3 External applications

- Use of API interface (e.g. mobile app, R-package)
- Use SQL connection (e.g. QGIS)

3.5.4 Export data from the upload process

During the data upload process and from the saved state of interrupted uploads, it is possible to export the data to a CSV file.

3.5.5 Abort data upload

The data upload process can be interrupted at any time from the web interface. A backup is automatically created every two minutes, but you can create a backup at any time by clicking on the Save button in the redundant menu bar.

Uploads that have been suspended can be restored by selecting them from the 'Suspended Uploads' list on the profile page.

Completed uploads are automatically deleted from the list.

3.5.6 Data upload history page

The metadata of each data upload is automatically recorded and can be accessed from the user's profile page or datasheet.

3.6 Data sheet

Each data record has its own data sheet, which contains all the associated metadata and data fields for the record. Depending on the settings, the available data content can be restricted in various ways.

3.6.1 Data history page

Each data record has its own data history sheet where you can view the changes to the record. This feature only works if the project host has enabled data change records in the project settings.

3.7 Database summary page

Each database comes with a summary page containing a description of the database and contact details.

3.8 Welcome page

Variable welcome pages can be set for each project.

3.9 Error reporting

The bug submission feature is available from the profile page and the upload page. Clicking on the bug in the bottom right corner of the screen will bring up the bug submission interface.

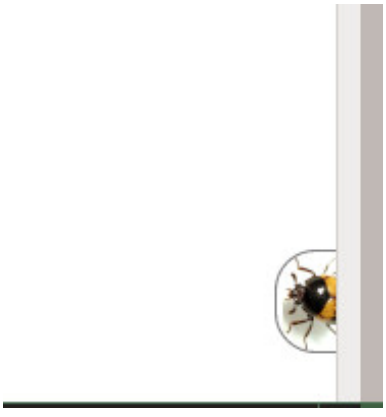

Fig. 1: Bug in the bottom right corner

The interface sends the errors to the OpenBioMaps developer page (https://gitlab.com/groups/openbiomaps/-/issues), from which the user will automatically receive a response from the system for further events.

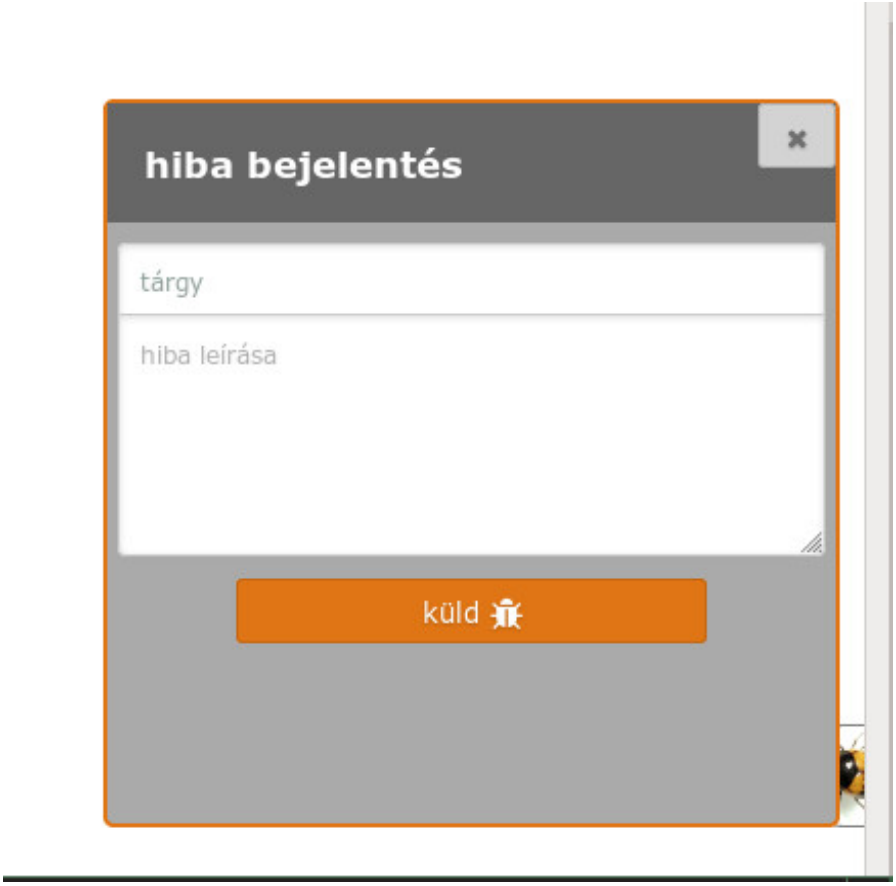

Fig. 2: Simple messaging interface

The error handler can be made available on a server by specifying the AUTO_BUGREPORT_ADDRESS address in the system_vars.php.inc configuration file. More information about the GitLab Issue handler interface can be found here: https://docs.gitlab.com/ee/user/project/issues/

CHAPTER

FOUR

API DOCUMENTATION

HTTP methods: GET, POST, PATCH

API tools: Authentication, Data retrieval, Data push, Settings update

4.1 API handlers:

Authentication handler (oauth):

/oatuh/token.php: Authentication

Authentication based interface (pds):

/projects/*API_VERSION*/*projectname*/pds.php: Data retrieval, Data push, Settings update

Non authenticated requests (web):

/projects/*projectname*/index.php

4.1.1 PDS API version:

Example: http://openbiomaps.org/projects/dead_animals/v2.1/pds.php The default version setting (if the version string missing from the URL) is 1.1., which compatible with the 2.0 and backward compatible with the 1.0.

4.2 OAUTH

An oauth2 implementation based on ttps://bshaffer.github.io/oauth2-server-php-docs/. OAUTH used in the web interface and in the PDS as well.

4.2.1 Variables

grant_type: password username: a rergistered email address password: password string scope: list of requested scope access in the autenticated session access_token: a valid access token

html authentication of clients is nesessary

Available clients are mobile,R,web

4.3 PDS

OBM API interface. Created for R and mobile interface. It uses the OAUTH interface for authentication.

4.3.1 Variables

scope: data methods: see below

value: some scope uses it

header: (put data) JSON list of table columns' names

ignore_warning: (put data) ignore upload warnings

form_id: (put_data) set form id

data: (put data) JSON array of upload data

4.3.2 GET type scopes

get_project_vars: query general project variables (available for non logined users as well).

Additional parameters: project [text]: if not set default is the *template* project

Returns:

- project_url [url string]: web address of the project
- project_description [text string]: short description of project
- game [on/off]: game available for android mobile app
- public_mapserv [url string]: url of publicly accessible map service
- rserver_port [numeric]: numeric port number of R-Shiny server, accessible on project_url

get_project_list: get list and basis info of database projects available on the server.

Additional parameters: only-project [text]: query project parameters only for the selected project, default is to query all accessible projects accessible [text]: all/**accessible**. If *accessible* parameter given and its value is "accessible" (default)

- If user already logined, get list of those projects where user has account and where there are public query/upload interfaces.
- If the user not logined, query public projects only.

Returns:

- project_table [string],
- creation_date [date string],
- Creator [string],
- email [string],
- stage [string] experimental/testing/stable,
- doi [string],
- running_date [date string],
- licence [string],

- rum [string],
- collection_dates [date range string],
- subjects [text],

get_form_list: query the list of available upload forms,

get_form_data: query the fields of the selected form

Additional parameters: value [numeric] numeric id of a form.

Returns: see below.

get_profile: get profile data of a selected user

get_data: get data rows from a selected data table (observation data)

get_specieslist: get species list from a project

get_history: get history of a selected data row

get_report: perform a predefined query and get the result

get_tables: get list of tables in a project

get_trainings: get list of available trainings/forms

Returns: set of traning titles, ids and descriptions,...

get_training_questions: get list of questions for the selected training

Additional parameters: value [numeric] numeric id of a training.

Returns: set of questions,answers and settings

training_results: status list of users' trainings per forms. Status can be -1 (not sent), 0 (not validated yet), 1 (done, ok)

training_toplist: toplist of trainings. Mean, Max, Count values for each forms. Additional parameters: value [text] summary without names (nonames).

get_mydata_rows: json array of uploaded data Additional parameters: value [numeric] limit of array length. If 0, no limit, default is no limit.

4.3.3 POST type scopes

put_data: send/upload data using a selected form

4.3.4 PATCH type scopes

set_rules: update specific settings

4.4 WEB API

There is a web (_GET) api endpoint to retrieve data without authentication. This is the ?query= This is API uses text_filter modules to assemble an SQL query statement.

4.4.1 Variables

query: (API endpoint)

qtable: (data table for data retreive)

report: (data retreive using stored queries)

output: (JSON, XML, CSV, ... file output; If not set, the output is the web interface)

filename: (the file name of the output file)

Get list of active (known) OpenBioMaps servers using query api:

curl http://openbiomaps.org/projects/openbiomaps_network/index.php -G -d 'query={"available":"up"}&output=json&filename=results

Get a filtered table from a non-default table:

curl https://openbiomaps.org/projects/pollimon/index.php -G -d 'query={"q":"2"}&output=json&qtable=pollimon_sample_plots'

LQ API end point:

LQ: (display data from a stored query result)

4.5 Form Data (get_form_data results) explanations

Description: Optional column description

Default value: Fix value for all observation. It can be controlled with the following options:

- '_input' it works as any other field with sticky flag.
- '_list' it works as any other list type field with sticky flag.
- '_geometry' it works as geometry type field
- '_login_name' this value overriden by the user's name if logged in or returns as _input
- '_email' this value overriden by the user's email address if logged in or returns as _input
- '_autocomplete' alias of input
- '_boolean' display as normal boolean list
- '_attachment' display as normal attachments field
- '_datum' display as normal date field
- '_auto_geometry' geometry field without extra options (map, set)
- '_none' not used

Column: The name of the column in the database

Short_name: Visible name of the column for the users

List: json array for menu items of a select menu. Can be {key:value} or [value,value] format

Control: Data checking commands: custom_check, minmax, spatial, nocheck, NULL

Count: (json array) If the control='minmax' this field contains the limit values, e.g 1:100

Type: column's openbiomaps type:

- autocomplete (json array)
- autocomplete_list (json array)
- boolean (two elements list)
- crings (colour rings - text)
- date (YYYY-MM-DD or other clear format)
- datetime (YYYY-MM-DD HH:mm:ss)
- file_id (file names as id by the server)
- line (WKT geometry string)
- list (json array)
- numeric
- point (WKT geometry string)
- polygon (wkt geometry string)
- text
- time (HH:mm)
- timetominutes (numeric value between 0 and 1440)
- tinterval idő intervallum (HH:mm - HH:mm)
- wkt (WKT sting)
- array (json array)

Genlist: json array for menu items of an autocomplete menu. Can be {key:value} or [value,value] format

Obl: 1,2,3 (obligatory, non-obligatory, soft error) Soft error can be handled as non obligatory.

Api_params: json array of control values. Till API v2.0 only 'sticky' as an array element. Above API v2.0: {"sticky":"off","hidden":"off","readonly":"off","list_elements_as_buttons":"off","once":"off"}.

Spatial_limit: WKT polygon string of spatial limit. It is used if the Control type is spatial.

List_definition: JSON array of complex list definition

Custom_function: null

permanent_sample_plots:

API < v.2.1 JSON array: [{"id":"1110","name":"Standard plots","geometry":"POLYGON((16.5625…

API >= v.2.1 in "form_header":{…,"permanent_sample_plots":[{"id":"1110","name":"Standard plots","geometry":"POLYGON((16.5625…

4.6 Training explanations and examples

Examples:

curl -F 'scope=get_trainings' -F 'access_token=9d45…' -F 'project=dinpi' http://localhost/biomaps/pds.php

Result of a successful call: {"status":"success","data":[{"id":"1","form_id":"95","html":" …",,"task_description":" …","e
I.","qorder":"1","project_table":"dinpi"},{

curl -F 'scope=get_training_questions' -F 'access_token=9d45…' -F 'project=dinpi' http://localhost/biomaps/pds.php

Result of a successful call: {"status":"success","data":[{"qid":"1","training_id":"1","caption":"… ?","answers":"[{"Answer":
"…","isRight": "false" },]","qtype":"multiselect"}]}

qtype can be multiselect or singleselect

curl -F 'scope=training_results' -F 'access_token=9bb4…' -F 'project=dinpi' http://localhost/biomaps/pds.php

Result of a successful call: {"status":"success","data":"{"95":1,"96":0,"97":-1,"98":-1}"}

Meaning of values: form-95 done, form-96 done, but not validated yet, form-97,98 not completed yet

curl -F 'scope=training_toplist' -F 'value=nonames' -F 'access_token=5ac3…' -F 'project=dinpi' http://localhost/
biomaps/pds.php

Result of a successful call: {"status":"success","data":{"95":{"mean":"0.5000000000000000000","count":"2","max":"0.7"},"96":{"

curl -F 'scope=training_toplist' -F 'access_token=5ac3…' -F 'project=dinpi' http://localhost/biomaps/pds.php

**{"status":"success","data":{ **

**"95":{"Bán Miki":{"mean":"0.3000000000000000000","count":"1","max":"0.3"}, ** "Dr.
Bán Miklós":{"mean":"0.7000000000000000000","count":"1","max":"0.7"}}, \

"96":{"Dr. Bán Miklós":{"mean":"0.7000000000000000000","count":"1","max":"0.7"}}, \
"97":{"Dr. Bán Miklós":{"mean":"0.7000000000000000000","count":"1","max":"0.7"}}, \
"98":{"Dr. Bán Miklós":{"mean":null,"count":"1","max":null}}}}

4.7 Examples

Authentication: curl \ -u mobile:123 http://openbiomaps.org/oauth/token.php \ -d "grant_type=password&username=foo@foobar.hu&password=abc123&scope=get_form_data+get_form_list+put_data"

Data retrieval (form list): curl \ -v http://openbiomaps.org/projects/checkitout/pds.php \ -d "access_token=d4fba6585303bba8da3e6afc1eb9d2399499ef3e&scope=get_form_list"

Result of a successful get_form_list call: {"status":"success","data":[{"form_id":"93","form_name":"lepke űrlap"},{ …

Data retrieval (form fields):

curl \ -v http://openbiomaps.org/projects/checkitout/pds.php \ -d "access_token=d4fba6585303bba8da3e6afc1eb9d2399499ef3e&scope=get_form_data&value=93"

OR with central pds curl \ -F 'scope=get_form_data' \ -F 'value=93' \ -F 'project=checkitout' \ http://openbiomaps.org/projects/checkitout/pds.php

OR with access token to retrieve data of a restricted form curl \ -F 'access_token=…' \ -F 'scope=get_form_data' \ -F 'value=124' \ -F 'project=checkitout' \ http://openbiomaps.org/projects/checkitout/pds.php

Result of a successful get_form_data call:

API < v.2.1

{“status”:”success”,

“data”:[{“description”:null,”default_value”:null,”column”:”egyedszam”,”short_name”:”egyedszam”,”list”:””,”control”:”minmax

{“description”:”faj neve”,”default_value”:null,”column”:”faj”,”short_name”:”faj”,”list”:””,”control”:”nocheck”,”count”:”{}”,”typ

API >= v.2.1

{“status”:”success”,

“data”:[

“form_header”:{“login_name”:”John Smith”,”login_email”:”jsmith@openbiomaps.org”}, “form_data”:[

{“description”:”faj neve”,”default_value”:null,”column”:”faj”,”short_name”:”faj”,”list”:””,”control”:”nocheck”,”count”:”{}

Data upload: curl \ -i \ -X POST \ -H “Content-Type:application/x-www-form-urlencoded” \ -H “Authorization:Bearer …” \ -d “scope=put_data” \ -d “form_id=128” \ -d “header=[“obm_geometry”,”obm_datum”,”time”,”datum”,”comment”,”longitude”,”latitude”,”observer”]” \ -d “data=[{“obm_geometr y”:”point(48.071187 19.293714)”,”obm_datum”:”2018-04-03 23:05”,”time”:”12”,”datum”:”2018-04-03”,”comment”:”asdad”,”longitude”:”0”,”latitude”:”0”,”observer”:”sdsaada”}]” \ -d “ignore_warning=1” \ ‘http://openbiomaps.org/projects/checkitout/pds.php’

Data upload with multiple attachments (files): curl \ -F “access_token=…” \ -F ‘scope=put_data’ \ -F ‘form_id=58’ \ -F ‘header=[“faj”,”obm_geometry”,”obm_files_id”]’ \ -F ‘batch=[\ {“data”:[{“faj”:”Sylvia curruca”,”obm_geometry”:”POINT(22.0 46.3)”}],”attached_files”:”file1,file2”},\ {“data”:[{“faj”:”Lanius Collurio”,”obm_geometry”:”POINT(21.5 47.1)”}],”attached_files”:”file3”}]’ \ -F ‘file1=@file1’ \ -F ‘file2=@file2’ \ -F ‘file3=@file3’ \ http://localhost/biomaps/projects/template/pds.php

Packed data upload. Data line in ZIP archive. This is the old mobile app’s export format. The ZIP file contains the following file
geometry.wkt PICT01.JPG PICT02.JPG note.txt

The ZIP file name is ‘Sun May 13 08:52:51 CEST 2018.zip’ which created from the observation date-time sting. The note.txt contains the observation comment which can be associated with one column of the form. In this example it is the ‘faj’. The other 3 columns shouldn’t be replaced or neglected. If there are some obligatory column in the form, those can be filled by the default_value parameter. In this example the ‘egyedszam’ column is an obligatory field which will be filled with ‘1’. Packed lines can be super packed. In this case ‘packed_line’ parameter should be changed to ‘multipacked_lines’ and the zip archive should contains zip files detailed above.

curl \ -F ‘scope=put_data’ \ -F ‘table=dinpi’ \ -F ‘form_id=58’ \ -F ‘header=[“obm_geometry”,”obm_files_id”,”faj”,”dt_to”]’ \ -F ‘default_values={“egyedszam”:”1”}’ \ -F ‘packed_line=@Sun May 13 08:52:51 CEST 2018.zip’ \ http://localhost/biomaps/pds.php

Data retrieval (non-authenticated report): wget http://localhost/biomaps/projects/dinpi/?report=2@szamossag&output=csv

Refresh token (from R): curl \ -F ‘grant_type=refresh_token’ \ -F ‘refresh_token=…’ \ -F ‘client_id=R’ \ http://openbiomaps.org/oauth/token.php

Returns: {“access_token”:”…”, “expires_in”:3600, “token_type”:”Bearer”, “scope”:”get_form_data …”, “refresh_token”:”…”}

Project list (using central pds): curl \ -F ‘scope=get_project_list’ \ http://localhost/biomaps/pds.php

Returns: JSON array of those project which has public upload forms, or the user (if logined) member of it.

4.8 General API answers

Based on: https://labs.omniti.com/labs/jsend

JSON: {“status”:”X”,”data”:”“,”message”:””“}

X: success, error, fail

CHAPTER

FIVE

INSTALL NEW OBM SERVER GUIDE

5.1 New OBM server installation using Docker

Docker installation tutorial

5.2 Common errors and solutions after new installations or updates

Common errors

5.3 Server configuration

Server configuration

5.4 Local variables for a project

Several low-level settings coming from the local_vars.php.inc file which can be updated by the server admin

5.4.1 The local_vars.php.inc file

Database connection definitions

```
// Please change the passwords for an other random string
define('gisdb_user','YOUR_PROJECT_ADMIN');
define('gisdb_pass','xxxxxxx');
define('gisdb_name','POSTGRES_DB_NAME');
define('gisdb_host','POSTGRES_HOST_NAME');
```

Project's sql table name

```
define('PROJECTTABLE','your_database_table_name');
//define('PROJECTTABLE',basename(__DIR__));
```

Project data restriction settings

```
// `public` data read/mod for everybody
// `login` data read/mod only for logined users
// `group` data read/mod only for group members
define('ACC_LEVEL','public');
define('MOD_LEVEL','group');
```

Language settings

```
// the corresponding language file should be exists
// see the language file inclusion in the prepare_vars.php
define('LANG','hu'); # en, ro, ru, ...
```

Path and URL settings

```
// On openbiomaps.org is /projects
// else maybe empty
define('PATH','/biomaps/resources');
define('URL',sprintf("%s%s",$_SERVER['SERVER_NAME'],PATH));
```

mapserver variables

```
define('PRIVATE_MAPSERV',sprintf("%s/private/proxy.php",URL));
define('PUBLIC_MAPSERV',sprintf("%s/public/proxy.php",URL));
define('PRIVATE_MAPCACHE',sprintf("%s/private/cache.php",URL));
define('PUBLIC_MAPCACHE',sprintf("%s/public/cache.php",URL));
// Standalone installation
define('MAPSERVER','http://localhost/cgi-bin/mapserv.fcgi');
// Docker installation
define('MAPSERVER','http://mapserver/cgi-bin/mapserv');
// Using Mapcache needs further settings, see Mapserver documentation
define('MAPCACHE','http://localhost/mapcache');
define('MAP','PMAP');
// MAP's JS OBJ variables
// should move into postgresql vars
define('PRIVATE_MAPFILE','private.map');
```

Invitations

```
// If 0, only admin can send invitations
// otherwise the specified number of active invitation can be, so can't send more the xx␣
↪invitations at once
// default is 11
define('INVITATIONS',0);
```

MAIL settings, if no local mail agent…

```
//define('SMTP_AUTH',true);
// *local smtp server example*
//define('SMTP_HOST','mail.your-smtp-server.org');
//define('SMTP_USERNAME','MAIL USER');
//define('SMTP_PASSWORD','xxxxxx');
//define('SMTP_PORT','PORT-NUMBER');
//define('SMTP_SENDER','mail_user@your-smtp-server.org');
// *Google example, it not works, updates needed!!!*
```

(continues on next page)

(continued from previous page)

```
//define('SMTP_HOST','smtp.gmail.com');
//define('SMTP_USERNAME','your-user@gmail.com');
//define('SMTP_PASSWORD','xxxxxxxxx');
//define('SMTP_SECURE','tls');
//define('SMTP_PORT','587');

// *Deprecated features...*
//define('TURN_OFF_LAYERS','layer_data_points');
//define('SHINYURL',false);
//define('RSERVER',false);
```

Which page loaded after log in? profile, mainpage, map default is map

```
define('LOGINPAGE','map');
define('TRAINING',false);
```

MainPage configuration

```
define('MAINPAGE',array(
  //'custom_skeleton'=>1,
  'template'=>'gridbox', //intropage
  'content1'=>'map',   // map | upload-table | slideshow
  'sidebar1'=>'column_dinpi.altema|custom_countries|members|uploads|data|species|species_
↪stat', // members uploads data species species_stat
  'system_footer'=>'on',
  'system_header'=>'off',
  //'restrictaded_pages'=>array('map','id','history','profile','data','table','editrecord
↪','qtable','query','show','LQ','metadata')
));
```

Which style folder used

```
define('STYLE',array(
  'template'=>'evolvulus'
));
```

Footer configuration

```
define('FOOTER',array(
  'links'=>'map|upload|about|terms|usage|privacy',
  'languages'=>'languages',
  'partners' => array(
          array('img'=>'obm_logo.png','size'=>'110','url'=>'https://openbiomaps.org'),
          array('img'=>'unideb_logo.png','size'=>'','url'=>'https://unideb.hu')
      ))
);
```

Encrypt hash

```
// *used by the read_table module to encrypt the table name, ...*
define('MyHASH','password-string');
```

Developer options

```
// *Switch to an other GIT branch*
define('branch','testing');
// *Extra logging for PDS actions*
define('DEBUG_PDS', true);
```

local_vars.php.inc

CHAPTER

SIX

PWA APPLICATION

6.1 What is the OBM PWA application?

Wants to learn more about PWA (Progressive Web Application) apps? Start here: https://web.dev/progressive-web-apps/

Our PWA APP is an online-offline hybrid application for supporting fieldwork. Width this app you can read the online database easily. How does it work?

While you are online, you can see data as a layer above the base map. Practically it is a cluster styled layer, where the number of feature points is the label in the cluster symbols. There is a filter/query option on the map to fetch a lot of data from the database. The default filtering option is the viewport, so applying this filter will fetch all records of data from the database that you can see on the map. Practically, it is a bad idea to zoom out to a much larger area than you really need because a huge amount of fetched data can freeze your app. After the fetching is finished, the cluster layer style will have a slight change which indicates that these features are available on your device. The displaying method is still clustered due to displaying numerous features that can freeze the app. When you click on a cluster symbol, the attribution appears in a scrollable modal dialogue, so you can read all attributes of the clicked features.

The PWA app runs in the browser but can operate without the browser window. So it looks like a standalone mobile application. The fetched data is stored in offline storage, but the base map actually is not, but maybe can be cached if you browse it before an offline usage.

When you visit the app's URL from the CHROME or OPERA browser, it will offer to install it as a desktop app. Use this option to access the offline usage features of the app, and the window size will be a bit larger without the browser frame.

Features

- Show your location (yellow dot)
- Show the GPS accuracy (grey circle around the location symbol)
- Show your tracklog
- Turn off-on tracklog
- Zoom to your location
- Query features from the online database by drawing a circle, polygon or actual viewport
- Display fetched data's attributes

Limitations

- It only supports POINT features
- Base maps can't be stored offline
- Fetching a large number of records (>50.000) can cause problems for offline storing and more…

Where is it?

- https://YOUR_SERVER/projects/YOUR_PROJECT/pwa/

6.2 Configuration settings for PWA application

Few settings are needed on the project admin interface. On the Maps settings page, you have to create a new MapServer layer in the *private map* file:

```
LAYER
      NAME "my_cluster"
      TYPE point
      STATUS on

      CONNECTIONTYPE postgis
      CONNECTION "host=localhost dbname=gisdata password={xxxxx} user=YOUR_PROJECT_admin␣
↪options='--client_encoding=UTF8'"

      PROJECTION
          "init=epsg:4326"
      END

      METADATA
          "wms_title"            "YOUR_PROJECT Cluster layer"
          "wms_srs"              "epsg:4326 epsg:900913"
      END

      DATA "obm_geometry FROM (SELECT * FROM YOUR_PROJECT WHERE ST_GeometryType(obm_
↪geometry)='ST_Point') as new_table USING UNIQUE obm_geometry USING srid=4326"

      CLUSTER
          MAXDISTANCE 50
          REGION "ellipse"
      END

      LABELITEM "Cluster_FeatureCount"
      CLASSITEM "Cluster_FeatureCount"

      CLASS
          NAME 'Clustered points'
          MAXSCALEDENOM 100000
          EXPRESSION ("[Cluster_FeatureCount]" != "1")
          STYLE
              SYMBOL "circle"
              SIZE 30
              COLOR 51 153 204
              OUTLINECOLOR 30 30 30
              OUTLINEWIDTH 1
          END
          LABEL
              #FONT arial
              #TYPE TRUETYPE
```

(continues on next page)

(continued from previous page)

```
            SIZE 8
            COLOR 255 255 255
            ALIGN CENTER
            PRIORITY 10
            BUFFER 1
            PARTIALS TRUE
            POSITION cc
        END
    END
    CLASS
        NAME "Single point"
        MAXSCALEDENOM 100000
        EXPRESSION "1"
        STYLE
            SIZE 12
            SYMBOL "circle"
            COLOR 000 130 255
            OUTLINECOLOR 30 30 30
            OUTLINEWIDTH 1
        END
        TEXT "[NAME_OF_YOUR_LABELING_COLUMN]"
        LABEL
            #FONT arial
            #TYPE TRUETYPE
            SIZE 8
            COLOR 0 0 0
            OUTLINECOLOR 255 255 255
            ALIGN CENTER
            PRIORITY 9
            BUFFER 1
            PARTIALS FALSE
            POSITION ur
        END
    END
    TOLERANCE 50
    UNITS PIXELS
END #wms cluster layer
```

The *NAME_OF_YOUR_LABELING_COLUMN* is a column name which will be used as a label. Most common, is the species-name column.

The *YOUR_PROJECT* is the target table name which will used. Most commonm is the base project table.

MAXSCALEDENOM 100000 means that no features displayed over 1:100.000 zoom level which is generally a good pratcice to prevent overloading your mapserver when it tries to calculates millions of clusters...

The CONNECTION string should be set up properly according to your server. If you use a Docker, these settings are most probably good for you, except for the password. Copy and paste the CONNECTION setting from another working layer.

CONNECTION "host=localhost dbname=gisdata password={xxxxx} user=YOUR_PROJECT_admin options='–client_encoding=UTF8'"

More, you have to create an SQL query on the SQL QUERY SETTINGS page:

```
SELECT obm_id, obm_geometry, NAME_OF_YOUR_LABELING_COLUMN %selected%
FROM YOUR_PROJECT
%morefilter%
WHERE ST_GeometryType(obm_geometry)='ST_Point' AND %qstr%
```

As you can see, there is a predefined filter that uses only POINT data, which is because clustering cannot merge line and polygon data.

6.3 Installation

Load the following url at once to make your app ready to use:

https://YOUR_SERVER/projects/YOUR_PROJECT/pwa/setup.php

Make sure you connect using https (secure connection), otherwise the application will not work properly!

CHAPTER

SEVEN

MOBILE APPLICATION DOCUMENTATION

7.1 How does the mobile app work?

7.1.1 Server choice

7.1.2 Log in

7.1.3 Project choice

7.1.4 Form selection

7.1.5 Pin form to the home screen

7.1.6 Map screen

7.1.7 Settings

Tracklog settings

Drawing angle settings

Sound feedback

Data upload

GPS usage

Saves

Language selection

7.2 Mobile application settings on the server

7.2.1 Forms

CHAPTER

EIGHT

DEVELOPER HINTS

CHAPTER

NINE

FREQUENTLY ASKED QUESTIONS

9.1 What is OpenBioMaps?

OpenBioMaps is a software and services platform for managing biological data. It can be used on its own or as a service. It can be used to create database-based projects that can be used simultaneously by multiple users with different devices and access privilege levels. If you do not wish to maintain your own server, you can use it as a service, as some institutions also operate servers that host research or citizen-science projects for no charge.

9.2 What is OpenBioMaps consortium?

OpenBioMaps is a consortium of public institutions and social organizations. Their aim is to develop OpenBioMaps software and maintain freely available services based on it. The members of the consortium are equal partners and they have all contributed to achieving this objective in some ways. The partnership can be extended, if the parties wishing to join are willing to accept the system's fundamentals, satisfy the specified conditions set and the partners accept the new member.

Current OpenBioMaps partners:

University of Debrecen

contact: Dr. Miklós Bán

Danube-Ipoly National Park Directorate

contact: Zsolt Baranyai

Eötvös Loránd University

contact: Dávid Ritter

WWF World Wildlife Fund for Nature Hungary

contact: Katalin Sipos

Eszterházy Károly University

contact: Dr. Erika Pénzesné Kónya

Milvus Group Association

contact: Edgár Papp

Danube-Dráva National Park Directorate

contact: Ákos Gáborik

Fertő-Hanság National Park Directorate

contact: Gábor Takács

The OpenBioMaps consortium was established on September 1, 2015. The OpenBioMaps Consortium Agreement will be available here.

9.3 How can I contact the consortium?

By email:

management@lists.openbiomaps.org

9.4 How can I create/find a new database-project?

If you are already a member of a project on a server, you can use the web interface to create a new database project on the same server. This is a simple process that can be done by filling in a form.

9.5 How can I upload data?

Simply answered: using data upload forms. Or perhaps using any PostgreSQL client, although this solution is only recommended for occasional imports of large amounts of data.

9.6 How can I access data?

- Via the web interface with map or text queries.
- Using a PostgreSQL client.
- Using the OpenBioMaps R package.
- Using data sharing via the web interface.
- Data export via the web interface.

9.7 How can I sign up for an OpenBioMaps project?

Registration requires an invitation. Any registered member can invite new users. In addition, by default, an invitation request form is available from the login interface, which can be filled in by new hosts to request an invitation to join a project. In addition, there is an OpenID module that allows registration and login with a Google Account.

For more information on registration and invitations, please contact the creators or administrators of the database you wish to join.

9.8 Is there a programmable interface for developers?

Yes. The Project Data Service (PDS) allows you to query projects and user data on a per-project basis through URL requests from databases.

Example: https://openbiomaps.org/pds.php?scope=get_project_list

For more information visit the API documentation.

9.9 What language support is available?

There are no language restrictions, but the OpenBioMaps is currently available in Hungarian, English Romanian, Spanish, and partially in Russian. Additional languages or translations can be added through the https://translate.openbiomaps.org interface.

Each project can have individual language settings and associated translations.

9.10 How can I contribute to OpenBioMaps?

- By creating/establishing a database project
- Uploading data to a database project
- By creating a new OpenBioMaps server
- Hosting database-project on your server
- Adding new languages or improving existing translations
- Software development
- Financial support

9.11 Should I pay for anything?

All components and services of OpenBioMaps are completely free of charge, but some of the development is not voluntary work, i.e. we pay the developers, so all support for the development is gratefully accepted!

9.12 How and where does the OpenBioMaps store the data?

Each OpenBioMaps server stores the data in its own database and file system.

9.13 Is there any backup solution?

No centralized backup, as there is no centralized data management in OpenBioMaps. Each server has its own backup solution, but some servers use each other's storage capacity for archiving.

9.14 I lost my password, how can I get a new one?

Don't worry, it's very easy to get a new password.

Follow the "lost password" link on the login page.

There you can enter your login email address. Once you submit it, you will receive an email from the system containing a link that you can follow to log in to your account and set a new password.

9.15 Pink squares appear on the map page

This may be due to some kind of configuration error, which may be related to the map layers or the settings of the data queries.

9.16 What is the RUM?

RUM is an acronym for database openness classes:

Read - Upload - Modify

Each element can have a value of [-] or [0] or [+].

where

[-] is not public, [0] is partially public and the [+] is public

and the colors are: [-] black, [0] red and [+] green

e.g.

RUM partial public read, public upload and no public modify

9.17 Is it possible to assign a DOI to databases?

Yes, all databases in a finalized state can receive a DOI using the DataCite DOI Service.

All databases have a DOI metadata page like:

https://dinpi.openbiomaps.org/projects/danubefish/index.php?metadata

Our DOI prefix in DataCite is: 10.18426

The DOI suffixes are automatically generated and they are unique.

In every database, it is possible to assign additional DOI-s for datasets.

9.18 Where can I find the list of the existing OpenBioMaps servers?

The servers that have registered can be found in the OpenBioMaps database at https://openbiomaps.org/projects/openbiomaps_network.

9.19 How to use the OpenBioMaps mobile app?

On Iphone or Android (currently, only the Android version works). Users need to be logged in on their own server to access the data upload forms available in their project. After logging in and downloading the forms, the app can be used offline. The current base map is Google-based and only works offline if the target area is downloaded for offline use from the Google Terrain Map application.

The mobile application lists the servers that are registered in the https://openbiomaps.org/projects/openbiomaps_network database.

9.20 Where can I find the OpenBioMaps R package?

For now, only available as a developer package here: https://github.com/OpenBioMaps/obm.r

`Download docs as pdf|Download docs as epub`