Sharing geological collection data with GeoCASe 2.0

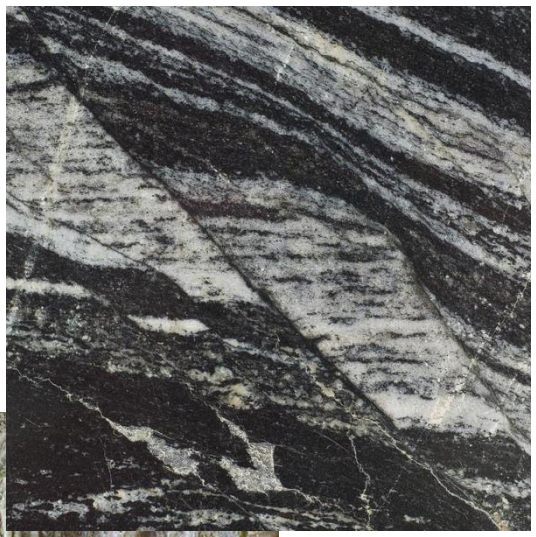
Authors: Tom Dijkema

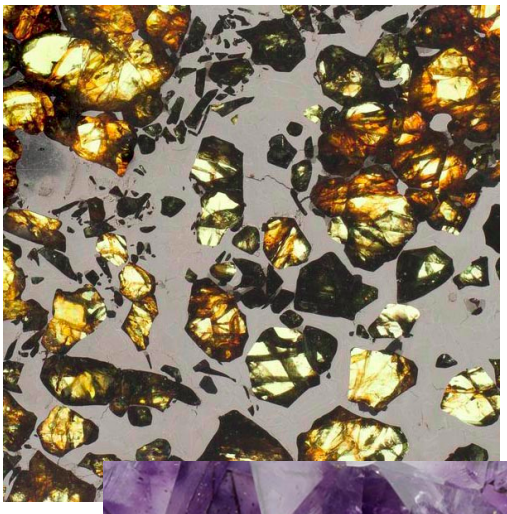
Dr. Niels Raes

Affiliations: Naturalis Biodiversity Center, Leiden, the Netherlands

Date: 14 January 2022

Version: 0.1





Abstract

Digitising geological collections is currently a well known practice. Information of specimens being copied to digital dittos makes it easier to share them with other curators, researchers and policy makers. The best known digital portal for sharing geological collections is GeoCASe. Initially built by developers from the Museum für Naturkunde in 2007, GeoCASe has seen upgrades like a new user interface in the years following up developments of developers from the Tallinn University of Technology. Data ingestion into the portal, however, is still going slow due to the technical difficulty presented to, and the capacity of curators. Preferably, mobilising geological specimens to GeoCASe should be a common practice even non-technical individuals can execute.

This manual presents a series of steps that can be followed to set up a complete pipeline for mobilising geological data to GeoCASe using the BioCASe Provider Software. These steps, however technical, are described in detail so even someone without experience should be able to follow them. Attached to the manual is the NLBIF service, specially made for inexperienced technicians. This service allows the use of a predefined instance of the BioCASe Provider Software. This resolves most of the technical steps so curators can focus on mapping their geological dataset to the ABCD-EFG standard, saving time and resources.

The manual's goal is to assist curators in sharing their collections with GeoCASe to enrich the platform for the benefit of institutions, collections, curators, researchers, policy makers and other users worldwide.

Table of Contents

A. Introduction

The Distributed System of Scientific Collections or DiSSCo is a consortium of over 170 Natural History Museums (NHMs) from 23 European countries that hold a large number of Natural Science Collections (NSCs) with an estimated 1.5 billion natural science and earth science specimens. DiSSCo is working towards the construction of a distributed research infrastructure that acts as a new business model for one European collection that digitally unifies all European natural science assets under common access, curation, policies and practices that ensure that all the data is easily Findable, Accessible, Interoperable and Reusable (FAIR principles). As a first step in the process of digitally unlocking the NSC specimens DiSSCo partners are asked to start mobilising their natural history specimen records to the Global Biodiversity Information Facility (GBIF), and their earth science specimen records to the Geoscience Collections Access Service (GeoCASe).

Many European countries have national GBIF nodes, or can make use of the European and Central Asian (ECA) nodes, to mobilise their NSC records to GBIF facilitated by GBIF's Integrated Publishing Toolkit (IPT). DiSSCo even has established a DiSSCo network at GBIF that provides access to dataset and NSC records of DiSSCo partners. Mobilising earth science records to GeoCASe is still less straightforward and requires several steps to set up a publishing pipeline. In order to facilitate the mobilisation of Earth Science specimen records this tutorial documents all steps in the process that are required to start sharing data with the GeoCASe portal.

The GeoCASe 2.0 portal

This manual aims at helping curators to mobilise their geological collections to the GeoCASe 2.0 portal. This portal, also referred to as just GeoCASe, is an international gateway for accessing and browsing geological collections. As a separate portal for geological collections, it creates value for the user by distinguishing the unique attributes of geological specimens and displaying them in a useful manner.

'***GeoCASe*** *stands for* ***G****eoscience* ***C****ollections* ***A****ccess* ***S****ervice. It is a data network and web portal designed to make collections of minerals, rocks, meteorites and fossils held in museums and research institutions universally available online, in order to foster scientific research and collaboration internationally*'

GeoCASe provides a well known endpoint within the geological community. This means that curators who mobilise their collection to GeoCASe provide from a wide audience reach, which should benefit the popularity of their collections.

To mobilise geological collections to GeoCASe, is a process that can be described as an extensive upload of collection data to a faraway database. It requires a fair amount of technical steps before the database can be reached and data can be mobilised to it. Currently, GeoCASe is built as its own provider. It harvests data from external collections, saves this data locally and displays it in the portal. Using the BioCASe Provider Software, it isolates itself from other best practices like GBIF using the Darwin Core Archive. There is a clear reason for this: because the Darwin Core Archive does not (yet) support geological attributes. The downside is it heavily impacts the mobilising process for curators, because there is often a lack of technical knowledge.

To aid curators in the mobilisation process of geological collections, this manual provides a set of detailed steps to set up a complete instance of the BioCASe Provider Software including the surrounding necessities.

Worth noting is that NLBIF (the Dutch node for GBIF) also offers a web server with a BioCASe Provider Software instance that may be used by other parties who do not have the (technical) capacity to set this up on their own. Using this server in cooperation with NLBIF, means that curators will not have to set up a new BioCASe instance, instead only have to prepare their dataset and create a new endpoint. They will also receive help from NLBIF in this process. Every chapter in this manual states the impact of this (NLBIF) service on the steps to follow and is a part of two main topics. The first topic concerning the technical set-up and the second topic about using the BioCASe Provider Software. In general, people who use the NLBIF service only have to execute the steps described in the second part.

Assistance for institutions that do not use the NLBIF web server is also available!

At Naturalis and NLBIF we value the importance of geological collections for scientific, legal and other purposes. That is why we encourage curators to mobilise their collections to GeoCASe and offer this manual, as well as our service and the web server to accommodate this.

Technical overview

The process of mobilising geological collections to GeoCASe requires a set of slightly difficult steps that 'can' take some time. These steps are classified as 'slightly difficult' because the manual estimates the reader to have a non-technical background. Thus the reader can understand all steps and in the best scenario, execute themselves. In contrast the manual depends upon the reader to have the correct knowledge about geological collections and is able to identify their different attributes during the 'mapping process'. This is important to deliver high quality data to GeoCASe and fully represent the geological collections. More on this topic later on.

Chapter B: Configuring a web server
In preparation for the BioCASe Provider Software, it is necessary to acquire a web server and configure certain properties. A web server is mandatory to install BioCASe and run it later on. It is also needed to host a database holding the geological collections. When making use of the NLBIF service, this chapter can be skipped because Naturalis offers a web server with BioCASe and a database already installed.

Chapter C: Using Docker to install BioCASe
This chapter describes how to install and use the Docker software. Docker can then be used to install and host different applications container wise, which is an efficient method. Using Docker the BioCASe Provider Software and the database can be installed and will be run on the web server. When making use of the NLBIF service, this chapter can be skipped because NLBIF already has a Docker instance running on the web server that runs BioCASe and the database.

Chapter D: Database Preparation
In this chapter the preparation of the database is discussed. It shows examples of using MySQL to prepare a valid database. Metadata tables and database users are created in preparation of the data ingestion. When making use of the NLBIF service this chapter can be skipped. It, however, could be useful to read the chapter to know how the database is built.

Chapter E: Handling the geological dataset
Extending upon the last chapter, this chapter covers the dataset preparation and ingestion into the database. Depending on the local storage method for the geological collection data, they will need to be transformed to a handable format by the database. This required chapter gets a bit technical by adding metadata to the metadata tables, creating a new table for the occurrences/specimens of the dataset and defining a relation between the meta- and collection data. This is the first chapter that is really mandatory when making use of the NLBIF service.

Chapter F: Configuring and using the BioCase Provider Software
Configuring the BioCASe Provider Software is a key component of the mobilisation process. BioCASe is an independent piece of software with its own characteristics and functionalities. It is important to handle these well. The 'mapping process' is a part of this chapter. When making use of the NLBIF service it is important to read this chapter to get to know the BioCASe Provider Software. For the curator, the 'mapping process' is of uttermost importance. Because of the curator's geological knowledge they will be able to map their data fields to those of the ABCD-EFG standard. The manual of course provides additional help in relating fields to each other.

Chapter G: Mobilising data to GeoCASe
At last, the geological collections can be mobilised to GeoCASe by using the prepared BioCASe Provider Software instance. The mobilising process is something GeoCASe itself will have to set in motion. This chapter describes how to let GeoCASe initiate the process. When making use of NLBIF service the curator can independently contact GeoCASe to initiate the mobilising process and update their collections in the portal.

At the beginning of every chapter will be mentioned how to mobilise geological collections when making use of the NLBIF service. In general, much of the technical aspects can be skipped for visual guidance on the NLBIF service and recurring chapters, see **figure 1** on the next page.

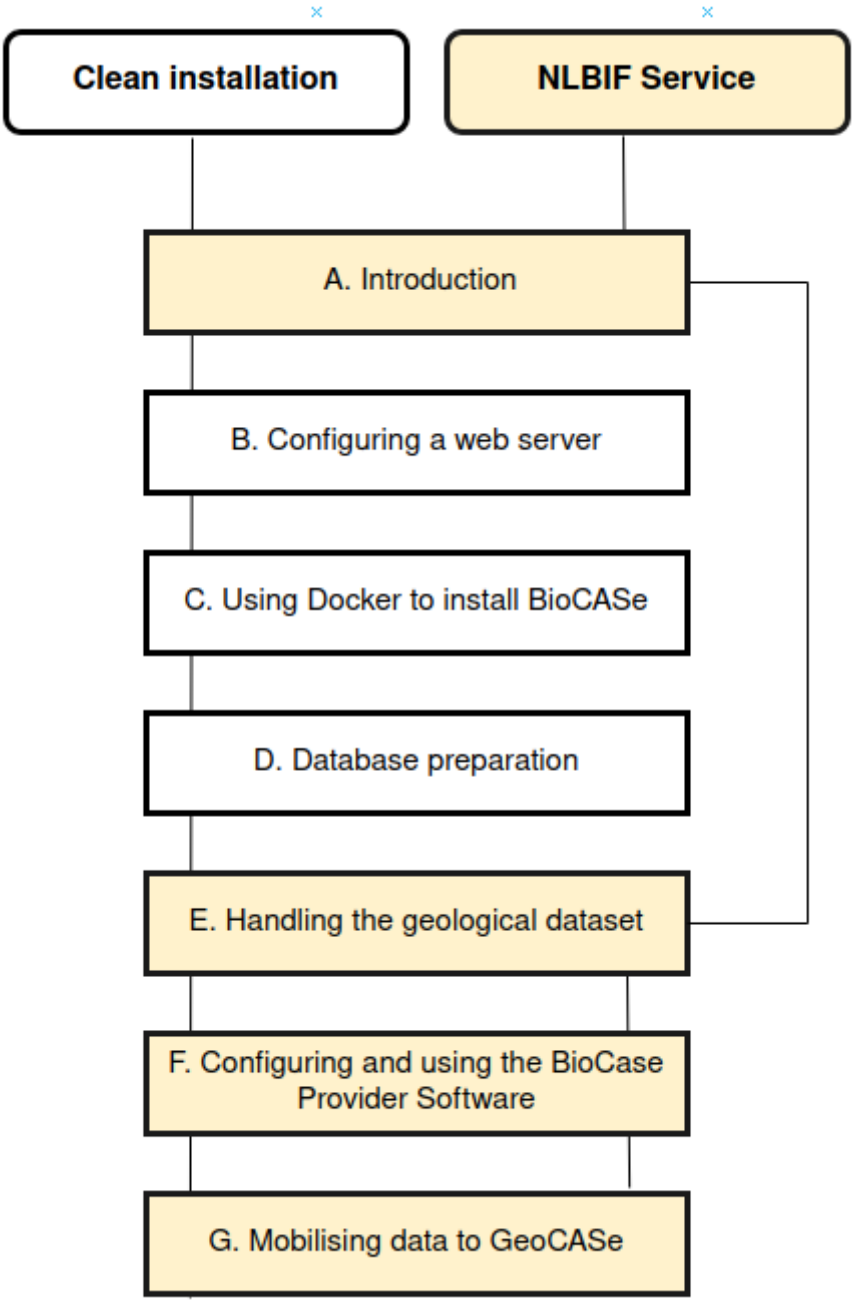


Figure 1: Read indication for when using the NLBIF service

Part 1: The technical process of creating a BioCASe instance

Image

B. Configuring a web server

To mobilise geological data to the GeoCASe 2.0 portal, it is required to install the **Bio**logical **C**ollection **A**ccess **Se**rvice (BioCASe) software on a web server. The BioCASe instance on a web server serves as the endpoint from which GeoCASe can pull the data. This chapter describes how to set up a web server in preparation of the BioCASe software installation process.

Besides operating an own web server with a BioCASe installation, it is possible to make use of the NLBIF service. NLBIF offers a web server with a BioCASe installation that can host multiple datasets from different institutions to facilitate the mobilisation of geological data to the GeoCASe. If you wish to make use of this service, please contact the contact person of Naturalis as mentioned in the appendix. You will not have to follow this chapter's steps if so and can skip to **Chapter E**.

Setting up a web server

To set up a web server, you can use the services of a server provider and order a Virtual Private Server (VPS) package. Such a package often consists of a VPS in combination with a terminal and/or operating system to manage your web server. It is important that an individual server is acquired and not a hosting package which often is used for hosting simple websites. The reason for this is that we need to edit the property settings of the web server, use different ports and install specific software. For this tutorial we used a server package from TransIP with the following specifications:

- Provider: TransIP
- Package: BladeVPS
- Operating System: CentOS
- Server operating tool: cPanel
- 1 Core
- 1024 MB RAM
- 50 GB SSD

This web server contains enough storage capacity and operating power to share data with the GeoCASe portal. If necessary, a package can be scaled up relatively easily. The operating system we use is CentOS, sometimes referred to as AlmaLinux which is a Linux type distribution. We included cPanel as a GUI (graphical user interface) to easily interact with the server. cPanel consists of two different interfaces:

A. a WHM (WebHost Manager) interface to manage the server's back-end, and
B. the core cPanel interface for the management of the server's content.

Prior to setting up a web server you have to register a domain name that provides web access to the server. We made use of a subdomain, under the existing NLBIF domain. The resulting domain is as follows: biocase.nlbif.nl.

If the installation of the server or defining the domain causes any problems, make sure to contact your web server provider to get support. It is important to set up the web server correctly to avoid problems further in the process to mobilise data to GeoCASe.

After the purchase of the VPS server the following information will be provided:

- Username and password
- WHM address, e.g. https://biocase.nlbif.nl:2087/
- cPanel address, e.g. https://biocase.nlbif.nl:2083/

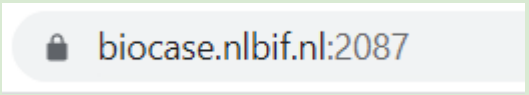
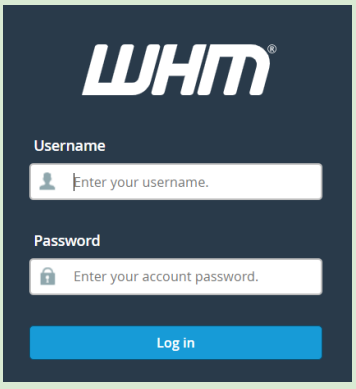
This is crucial information to get to work with the server. The new server can be accessed through the cPanel interface at port 2083 and the WHM interface at port 2087.

A **port** is a term that is used throughout this document. It refers to the different access points of the server. The cPanel interface can be accessed at port 2083 by entering the port number after the domain name separated by a colon, e.g. biocase.nlbif.nl:2083. The same goes for the WHM interface which can be reached at biocase.nlbif.nl:2087. Only one service at the time can be run on one port. Multiple services can be installed to run on the same port, but they cannot run simultaneously.

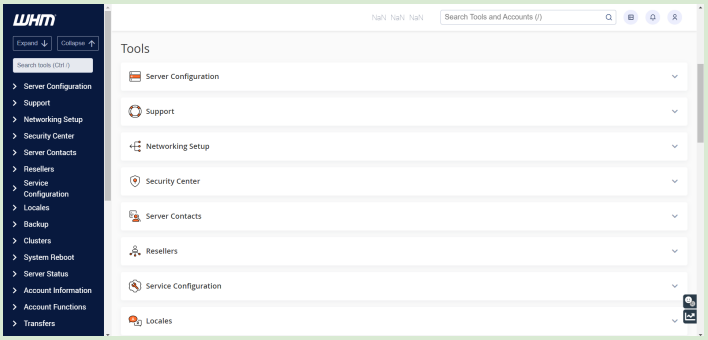
Configuring the web server

After the web server is set up it needs further configuration. First we will take a look at the WHM interface because we first need to set up a cPanel user account to access the cPanel interface.

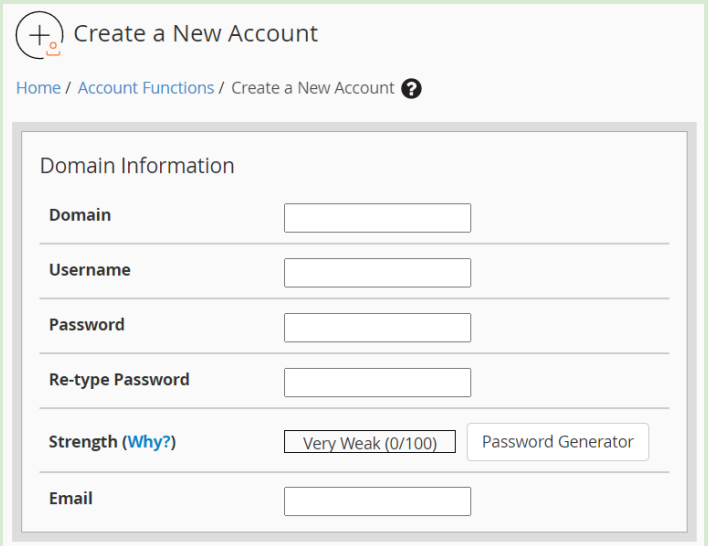
Table 1. Create a cPanel user account to login to the cPanel interface that is linked to a specific section of the domain. A cPanel user account has access to the main domain section, here biocase.nlbif.nl.

1. Go to the address of your WHM installation by typing in your domain name in the URL bar of your browser and adding the 2087 port, like for example: biocase.nlbif.nl:2087.	
2. You will see a login form where you should be able to login in with the username and password credentials that were received in the mail from your provider.	

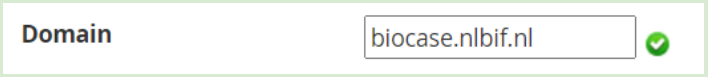
3. After logging in you will see the main page of the WHM portal. This page offers all kinds of tools that can be used.



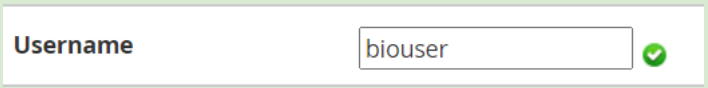
4. Type and click in the search bar on: 'Create a New Account'.



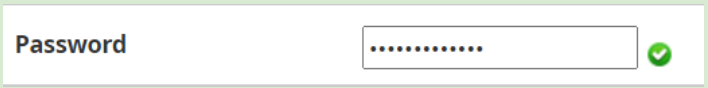
5. Enter the domain name you registered, for example biocase.nlbif.nl.



6. Provide a username for the cPanel account and fill it in.



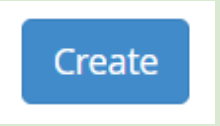
7. Generate a password (or use the Password Generator) and fill it in.



8. Retype the password.

9. Make sure to document your credentials somewhere save

10. Save the user by clicking on 'Create'



To check the user configuration, navigate to the cPanel interface that is located on port 2083, e.g. https://biocase.nlbif.nl:2083. If correct, a similar login form will pop up where you can login with the credentials that you just created.

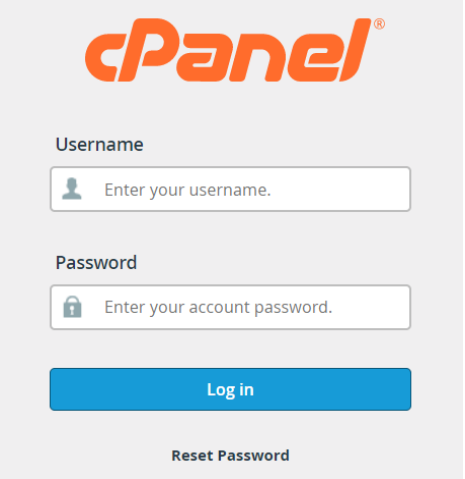
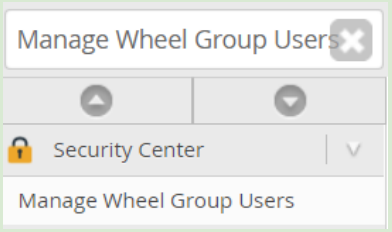
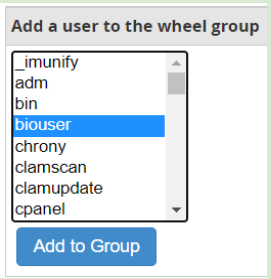


Figure 2: cPanel login page.

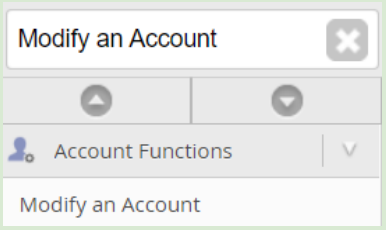
If the cPanel user is not working properly you should check if you have filled in the correct credentials and domain name. This can be done by searching and clicking in the WHM search bar onto: 'Modify an Account'. Then select the right account, if only one is existent, this account will automatically be displayed.

Next the user permissions have to be defined to make use of the 'shell'. The shell represents a terminal that is used to interact with the server. To give permissions we need to add the user to the so-called 'Wheel Group'.

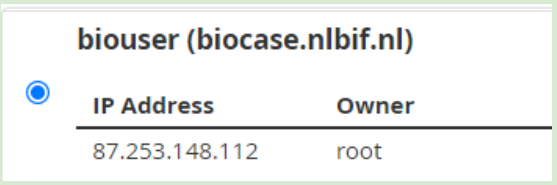
Table 2. Configure user access to the cPanel shell.

1. Search and click in the WHM search bar onto 'Manage Wheel Group Users'.	
2. Select the cPanel username you created from the list and press on 'Add to Group'.	

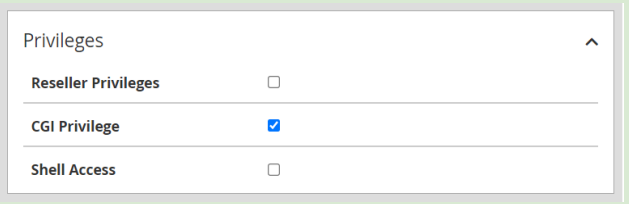
3. Search and click in the search bar onto ‘Modify an Account’.



4. Click on the account you created (if only one exists, this one will be automatically displayed) and press ‘Modify’.



5. Expand the privileges element.



6. Check the option of ‘Shell Access’.

7. Save this setting by pressing ‘Save’.

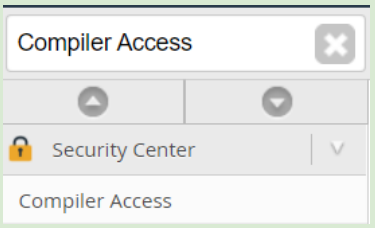
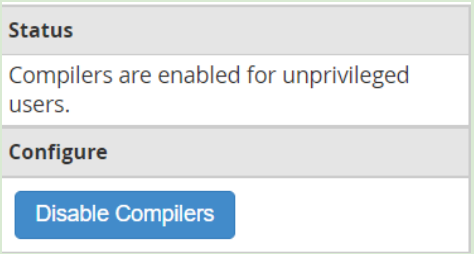
Security measures

To protect the server it is important to configure several security settings. These range from shutting down services to setting security settings. In this section we will skip through recommended security settings.

Compiler Access

Some servers are built to compile files. This, however, is not an activity that is needed to share data with GeoCASe. Furthermore, if non-intended users manage to access the server, they can manipulate the compiler. When not used, It is best to disable the '*Compiler Access*'.

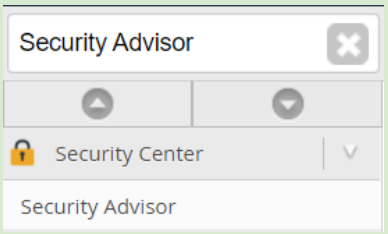
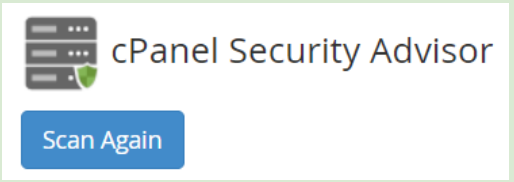
Table 3. Disabling the 'Compiler access'.

1. Search and click in the WHM portal onto 'Compiler Access'.	
2. If enabled, press 'Disable Compilers' to shut access to the compiler.	

Security advisor

An easy way to identify more security issues is by using the '*Security advisor*'. This is a built-in tool that scans the server settings and identifies possible security breaches. Identified issues are categorised in three classes: 'Important', 'Information' and 'Verified'. Important issues need attention and can often be solved by adjusting the settings. If possible, try to determine the importance of identified issues and apply the necessary changes. Issues labelled with 'Information' indicate important security settings that are set. These should be monitored regularly. Issues labelled with 'Verified' indicate which components are set well and secure.

Table 4. Using the 'Security Advisor'.

1. Search and click in the WHM portal onto 'Security advisor'.	
2. Run an automatic scan or press 'Scan Again' for a manual scan.	
3. Read the advice given by the security advisor.	

If you do not possess the correct knowledge to improve the security of the server, make sure to contact your IT-department to review the settings and security advice.

Service Manager

Next, we take a look at running services on the server. Not all services are required to mobilise data to GeoCASe. Instead they should be turned off to block unwanted visitors and improve the server's performance. This can be done by using the '*Service Manage*r' in the WHM interface.

Table 5. Shut down unnecessary services.

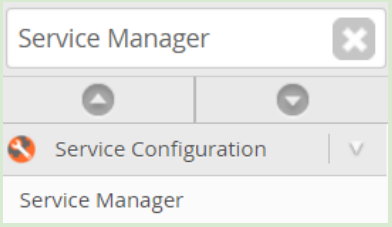
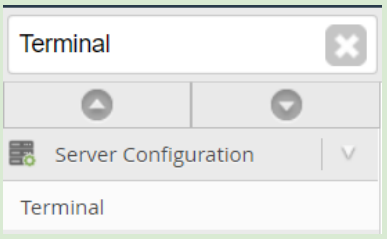
1. Search and click in the WHM portal onto 'Service Manager'.	
2. Deactivate all unnecessary services, see the list below for reference.	
3. Press 'Save' to save the settings.	

Table 6. Recommended services to be run (the rest can be turned off).

Service	Description
ChkServd	Responsible for checking, monitoring and restarting services.
ModSecLog	Parses the ModSecurity audit log and stores the events in the modsec database for later viewing or analysis.
cPanel DAV Daemon	-
cPHulk Daemon	cPHulk Brute Force Protection
Cron Daemon	-
cPanel DNS Admin Cache	cPanel DNS Admin Cache Service
Apache Web Server	Web Server
IP Aliases	-
MySQL Server	MySQL Database Server
DNS Server	PowerDNS
Name Service Cache Daemon	-
rsyslog System Logger Daemon	Enhanced System Logger Daemon

As we will make use of MySQL to set up a database on the web server, it is recommended to limit MySQL to listen only to queries that come from our web server. When not set, it is possible for foreign users to login to the database and make possible changes when they are in possession of the correct credentials. To solve this threat, we can restrict the bind-address from which MySQL listens to our server address.

Table 7. Set the MySQL bind address to our server address.

1. Search and click in the WHM portal onto 'Terminal'	
2. Activate MySQL by typing 'mysql'	mysql
3. Change the bind address variable to your servers ip address by using the following command	bind-address = [your ip address];

4. Exit mysql by typing ‘quit’	quit
5. Restart MySQL by using the following command	Service mysqld restart
6. To check if your bind address has been changed properly, type in the following command in the activated mysql command line (type mysql in the terminal) to display the variable:	mysql SHOW variables LIKE ‘bind_address’;

To further secure the gateways to our server, like MySQL, we can access the Firewall properties in the online portal of TransIP. This portal can be found by logging in to your TransIP account. In case you are using another provider, a similar portal should be available. When logged in search for Firewall settings. A page containing all the server ports should appear in which we can open or close ones. First off we want to close the unnecessary ports as we will not be using these. Make sure the Firewall is activated, otherwise these changes will have no effect.

Table 8. Close the unnecessary ports.

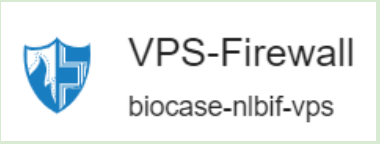
1. Open the Firewall section in your providers online environment	
2. Check if the Firewall is up and running (image translates to: ‘activate VPS-Firewall for this VPS’; ‘On’)	
3. Close all unnecessary ports, use the table below as a reference. This states all the ports that are opened on the NLBIF server	

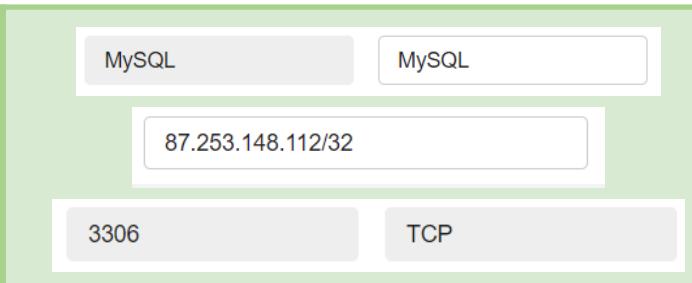
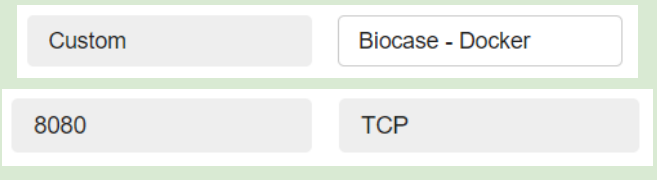
Table 9: Firewall ports of the web server.

Service	Port (range)
cPanel - FTP passive port range	30000-50000
cPanel - WHM	2086
cPanel	2082
cPanel - SSL	2083

cPanel - WHM SSL	2087
cPanel - Webmail	2095
cPanel - Webmail SSL	2096
cPanel - FTP	21
cPanel - HTTP	80
cPanel - HTTPS	443
cPanel - IMAP	143
cPanel - IMAPS	993
cPanel - SMTP	587
cPanel - SMTP	465
cPanel - SMTP	25
cPanel - SSH	22

Finally, we want to add two ports to create access to MySQL and our future BioCASe installation. MySQL should be reachable via port 3306 from any point of the server. We will install BioCASe on port 8080 as an extension of 80.

Table 10. Open a port for MySQL and BioCASe.

1. Add port 3306 to the Firewall' list that is globally reachable from the server's ip address	
2. Add port 8080 to the Firewall' list	

C: Using Docker to install BioCASe

Before we will install the BioCASe Provider Software to mobilise geological records to GeoCASe, it is good to understand how the Docker software functions. We provide this introduction to Docker because it is by far the most easy and efficient way to install and manage BioCASe. It also is recommended by the developers of BioCASe.

When you make use of the web server hosted by NLBIF with the BioCASe instance to mobilise your collection data, this section can be skipped. Using the NLBIF service allows you to add your own dataset to the BioCASe installation on the web server. You receive your own publisher details and dataset metadata information along with the records that you share with GeoCASe. However, background information on how to deploy a Docker installation of BioCASe on a web server might be useful knowledge.

A brief introduction to Docker

A common problem in the world of software development and distribution is the takeover of software. Software often requires certain packages and/or settings that not every computer or server owns by default. This can result in missing system requirements and nonfunctional software. To ensure that the BioCASe software and all of its dependencies are correctly installed a Docker instance with the BioCASe software and its dependencies can be deployed on a web server. Docker is an open source platform for developing, transferring and executing software that solves the dependency problem. It does so by offering complete application packages called *containers*[T]. These containers are easy to run and contain a complete image of the software and its dependencies. This way, no further installations than the container itself need to be done. Besides, containers can easily be updated or removed without harming the source data.

A Docker image of the BioCASe software is available for free use. This means that by using Docker we can install and sustain a BioCASe installation with all its dependencies on our web server.

Terminology

This chapter makes use of several terms that can be deemed as technical and specific. Make sure to consult the terminology list in the **Appendix** if you feel the need to get more context. Terms that appear in the terminology list will be marked in text with a ‘[T]‘.

How it works

Docker makes use of a client-server architecture as seen in **figure 3**. The client communicates with the Docker *daemon*[T] (*figure 3*; DOCKER_HOST) that handles heavy tasks like building, running and distributing containers. The *client*[T] and Docker daemon can function on the same system, but a client can also be connected with an external Docker daemon. The communication between the client and the Docker daemon is established via a *REST API*[T].

The *Registry*[T] (*figure 3*; Registry) serves as a place where images are saved. The Docker Hub is a public registry that automatically observes for available images. When Docker does not detect an image on the local system, it will automatically search in the Docker Hub for a matching one. In addition, it is possible to run a private registry in Docker Hub.

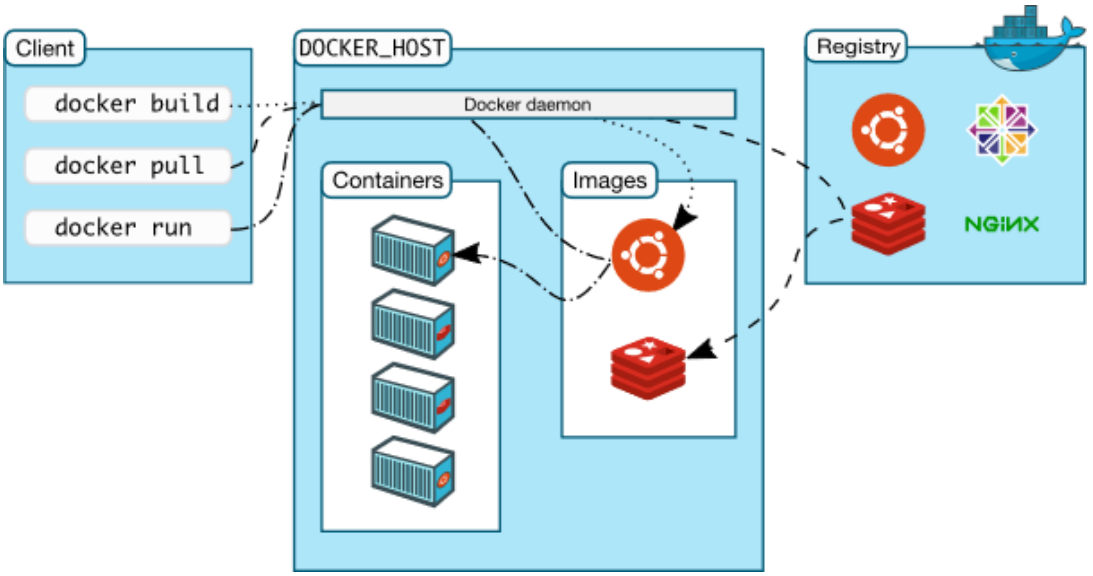


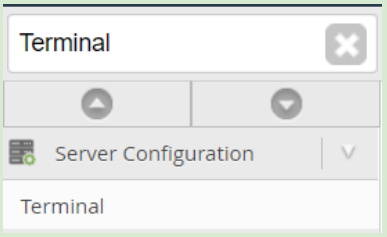
Figure 3: The Docker Architecture

When a container is executed through a command, Docker is activated. The program creates an isolated environment around the defined content of a container which has its own folder structure. Every contextual element from a container that Docker requires needs to be defined in an image. Docker itself does not add anything to the contextual mix. Because of this, the image also contains configuration options for the container like: *environment variables*[T], a standard *command line*[T] to execute and other *metadata*[T]. The above stated requirements is in fact all that is needed to set up a basic Docker application.

Installing Docker

Installing Docker is a relatively simple task that can be done in two separate ways. The first is to download an installer from the Docker website. This installer, specified for the Windows, Linux or iOS operating systems, installs a graphical interface for using Docker. The latter is to install Docker via the command line in the *terminal*[T]. Because we are going to install Docker on an online server, we use the command line in the terminal as this is the most straightforward way. As mentioned in the server installation chapter we selected CentOS as the operating system. This is a Linux distribution and the main reason why we use Linux commands in this manual.

Table 11. Install Docker via the terminal in the WHM interface.

1. Search and click on 'Terminal'	
2. When the terminal is loaded, type in the following command	sudo yum install docker-ce docker-ce-cli containerd.io

This command will have Yum install the Docker software and its dependencies. Yum is a free to use installer for Linux that, for example, can be used to install, update or remove software. We ran the command with 'sudo', which means we wanted to run the command as the administrator. To check if Docker is installed correctly, try to start the daemon by using the Docker start command, again from the terminal.

Table 12. Start Docker and see its status

1. Start the Docker daemon, type in the following command	sudo systemctl start docker
2. Docker should run by now. To test if Docker is running, type in the following command	sudo systemctl status docker
3. This should display Dockers details including an active statement that says it is running or not.	Active: active (running) since Mon 2021-12-13 22:16:25 CET; 1 months 8 days ago

If Docker does not start or run well, try to reinstall the software. This can be done by removing Docker with: sudo yum remove docker; and installing it once more with: sudo yum install docker.

Lastly we will add Docker to our start-up applications. This will automatically start Docker when the server is rebooted. This is optional, you can also choose to start Docker manually.

Table 13. Indicate Docker as a start-up application from the terminal.

1. Type the following command in the terminal	sudo systemctl enable docker.service
2. Then type in this command	sudo systemctl enable containerd.service
3. This should enable Docker as a start-up application. You can test this by rebooting the server. Use the following commands if you want to disable this type of behaviour	sudo systemctl disable docker.service sudo systemctl disable containerd.service

Basic commands

To get familiar with Docker's functionalities, this next paragraph describes the basic commands to manage a container application like BioCASel. Every described action can be executed via the terminal as done previously. It is not mandatory to execute the following actions, but they serve to get experienced with Docker. The basic commands relate to a simple standard. As an example we look into creating, starting, stopping and demolishing a car like a real live situation.

Creating: To create a car in real life, you would first need a template for the chassis. The same applies to a virtual Docker container. This template is called an *image*[T] and represents the application and all of its dependencies that are needed. To create a container with a certain image we can use a locally stored image, or grab one from the Docker Hub. If Docker does not recognize a local image, it will automatically search on the Docker Hub for a corresponding one.

There are several commands which can be used to create a container. The most straightforward one is the ***build*** command. This command builds a container from a locally stored image. It makes use of the image's location and can be given a name with the ***-t*** tag. The command shown below could be used when executing from the directory where the image is saved:

Build a container with the ***build*** command	docker build -t <container name> .

Another way to create a container is by using the ***run*** command. This command is very useful because it makes use of the auto-search function. As mentioned before, this will automatically search on the Docker Hub for an image if no image is found locally. Besides, the ***run*** function will also start the container, which we will discuss shortly. The command below will search in the Docker Hub for the BioCASe image if not found locally, download and install it into a new container:

Run a container with the ***run*** command	docker run -dp 8080:80 biocase/bps

The container will be named after the image name, in this case: biocase/bps. Important to notice is the port that we defined with ***-p*** tag. By running this command the application will be set to run on port 8080:80, which is an extension of the default port 80 for the *HTTP*[T] (Hypertext Transfer Protocol). Also, the ***-d*** indicates a container to be run as a background process. The ***run*** command can also be used to run a container that was built with the ***build*** command. Likewise the port will again be defined in the ***run*** command for the existing container.

Starting: After creating a car we would want to take it for a drive. This means we will have to start the engine by turning the key. The same applies for a Docker container. After creating, a container can be activated using the start command. This command needs an identification key to the container. A key, in Docker terms, can be for example the name, a tag or id of a container. To acquire a key, we need to display the properties of our Docker container.

Table 14. Start a container.

1. Acquire the container's properties	docker ps -a
2. With the key id, start the container with the following command:	docker start <container id>

Stopping: To stop a car from running, you turn the key anti-clockwise. The same goes for a Docker container. To stop a container use the stop command with the identification key.

Stop a container from running with the ***stop*** command	docker stop <container id>

Important to notice is that when a Docker container is stopped, all services will be shut down. This means all data stored in the Docker container will be lost. Docker supports volumes and multi-container applications to prevent data from disappearing when a Docker container is stopped. These methods will be pointed out in the advanced paragraph. The BioCASe image supports volumes so data will persist. More on that later in **Chapter D** on the BioCASe software.

Demolishing: When a car is broken and ready to be demolished it will be crushed or taken apart for parts. When a Docker container is no longer needed, it can be deleted. The ***rm*** function removes a container by using an identification key. Remember that all elements from the image will be deleted from the container. The image itself will not be removed.

Remove a container with the ***rm*** command	docker rm <container id>

By knowing how to create, start, stop and demolish Docker containers you will be able to host an application such as BioCASe and proceed to the next chapter. By interest, you may want to have a look at the advanced methods paragraph for a more in depth look at Docker's capabilities and requirements to run BioCASe.

Advanced methods

A couple of advanced methods give more insight into Docker's capabilities. It also shows how to retain data when reinstalling or removing Docker containers.

***Volumes**:* Docker volumes are an option to retain data. A volume has the ability to communicate with a specific path in the *file hierarchy*[T] of the *host machine*[T]. When Docker detects a change in a container, like adding or editing a record, the volume will communicate this action with the host machine. The host machine keeps track of this data so the information can be accessed when rebooting the container. It basically comes down to external files, such as text or CSV, that save the data and can be read when restarting a container.

There are two types of volumes. The first one, called a ***named volume***, uses external files. With this method it is also possible to create a working directory to dynamically update an application. The second one is based on setting up a ***network***. Within a network multiple images can communicate with each other. An example we will use for the BioCASe software is a network consisting of a BioCASe image and a MySQL database. The MySQL database contains the data, so when a container is restarted the data is retained. Networks can also be used for different aims such as the preservation of data and connection between multiple applications like API' for functional purposes. The establishment of a network consisting of multiple images in a single Docker container is called a multi-container application.

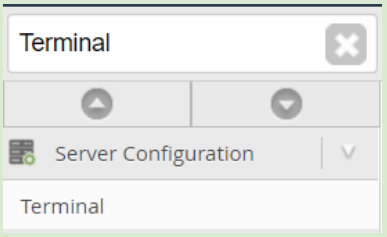
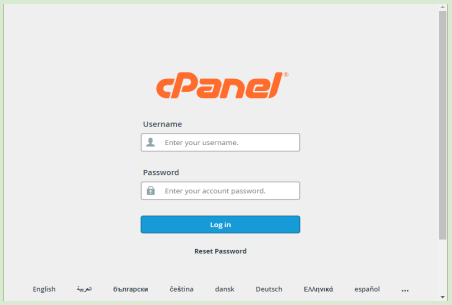
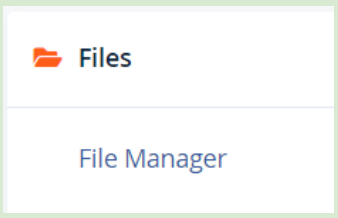
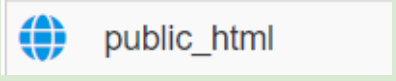
Docker Compose**:* A third method (besides ***build and ***run***) to create a Docker container is to use Docker Compose. Docker compose is an inbuilt set of functionalities that makes the creation of containers, volumes and networks simpler. It uses a single file that extracts all necessary information. For example, a Docker Compose file defines the images, port, working directory, volumes and environmental variables for network applications. BioCASe makes use of a Compose file so that no complicated commands need to be executed in the terminal. Instead, the following command can be run to set-up the entire application by letting Docker Compose read the file:

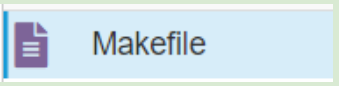
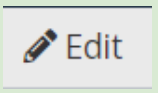
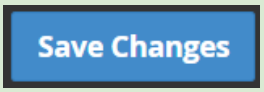
Execute a Docker Compose file with the ***up*** command	`docker-compose up -d`

Installing BioCASe with Docker

With the Docker knowledge we can install BioCASe. This process has been simplified by the developers of BioCASe who have created a *Make file*[T]. A Make file is very useful because it creates a set of rules that apply to the local environment where it is executed. These rules can be used to set-up the Docker installation of BioCASe by using a single command. The rules also create two extra command opportunities for updating and removing the installation. Before executing the Make command, make sure you have installed Docker on the web server. To note: we will be using port 8080 as extended from 80 to run the application.

Table 15. Install BioCASe with the Make file in the WHM interface.

1. Open the terminal by searching and clicking onto 'Terminal'	
2. Navigate to the public repository of the cPanel user that is linked to the registered domain, in the case of Naturalis the user is called: biouser on the domain of biocase.nlbif.nl. Enter the given commands one after another	cd home cd biouser cd public_html
3. Get the Make file by using the following command	wget http://ww2.biocase.org/svn/bps2/trunk/Makefile
4. Open and login to the cPanel interface at port 2083 in a new browser tab (do not close the WHM interface just yet)	
5. Find and click on 'file manager' in the 'Files' tab. This will open another new browser tab	
6. Select 'public_html' from the cPanel user's directory	

7. Select the Make file	
8. Press 'Edit' to open the file in the online editor	
9. Change the port variable to 8080	`PORT=8080`
10. Change the container variable to a relevant name (biocase is the default)	`CONTAINER=biocase_test`
11. Save by clicking on 'save changes'	
12. Return to the terminal in the WHM interface	
13. Type in the following command to install and start the BioCASe installation:	make install

With this last command, the Make file will create a container with the given name containing the BioCASe installation and run it on port 8080 to the extent of 80. To check if your installation is running, navigate to your domain name and add the port, followed by the BioCASe extension, as for example: biocase.naturalis.nl:8080/biocase. You can also check the container's status by using the Docker command for displaying all containers:

Check the status of a Docker container with the ***ps*** command	docker ps -a

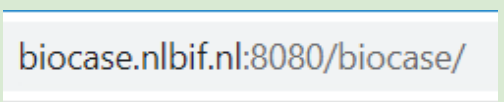
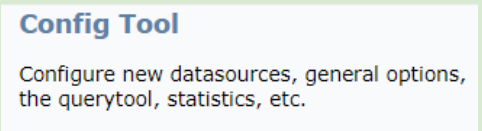
Using the Make file to create the container is not of harm to Docker's Natural commands such as: start, stop, update or remove. However, the Make file itself does also provide these methods.

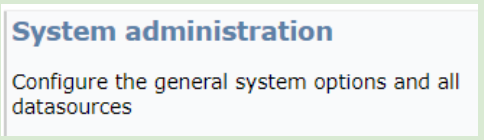
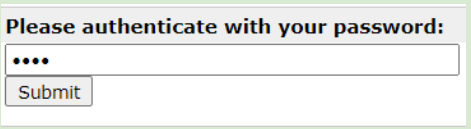
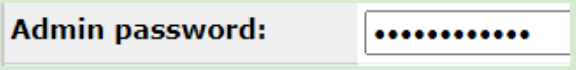
Table 16: Starting, stopping, updating and removing BioCASe with make

Start BioCASe with the ***make start*** command	make start
Stop BioCASe with the ***make stop*** command	make stop
Update BioCASe with the ***make update*** command	make update
Remove BioCASe with the ***make remove*** command	make remove

Now BioCASe is installed, navigate through the interface to get some feeling of the application. One more thing we want to do before the next step is to change the default password. A password is required to login to BioCASe's managing system. The default password that comes with every installation is: ACDC. Without changing this, everyone who knows of BioCASe could login and change settings or data sources.

Table 17. Change the BioCASe password.

1. Navigate to your BioCASe installation (for example: biocase.nlbif.nl:8080/biocase)	
2. Click on 'Config Tool'	

3. Now click on ‘System Administration’	
4. Fill in the default password: ACDC	
5. Change the password to something strong and memorable	
6. Press on ‘Update config’ to save the settings	

Now you can login with your new password. Do not forget to document the password in a safe place.

D: Database preparation

To exchange data with GeoCASe it is required to set-up an online database containing the data related to the geological collection specimens. Exchanging data with the GeoCASe portal requires a web server (**Chapter B**) with a BioCASe software installation (**Chapter C**) that connects to an online database. This chapter describes how to prepare a MySQL database that is running on the web server. This database will contain three metadata tables which are related to each other using a relational approach. A single or more MySQL users will need to be defined to access these data in the next chapter. Finally, the database will be linked to the set up BioCASe instance.

When making use of the NLBIF service, this chapter can be skipped. NLBIF provides an already prepared database containing the necessary tables, relations and users to interact with the BioCASe Provider Software. This database can be used after receiving credentials. Please contact the contact person in the appendix if so.

Creating a MySQL database

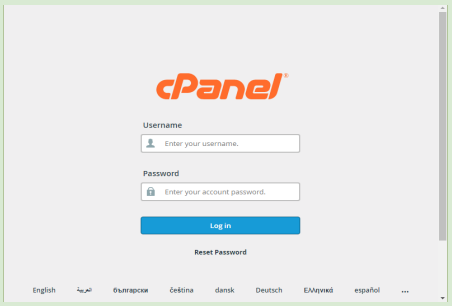
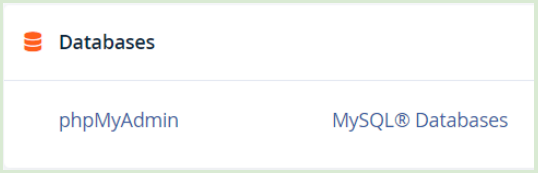
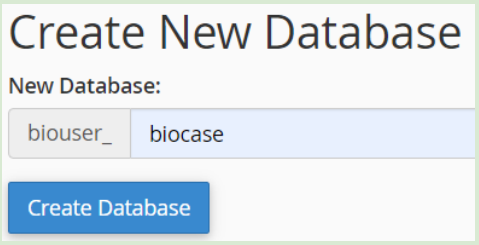
The first step in the database preparation process is to create an empty database containing three tables:

- Institutions
- collections
- collectionsMetadata

The ‘institutions’ table stores data on different publishing institutions, including their ROR IDs (Research Organisation Registry). The ‘collectionsMetadata’ table contains multiple rows that describe the characteristics of the different datasets in the database, one record per dataset. The ‘Collections’ table holds the definitions of the different geological collections. A fourth table called the ‘occurrences/specimens’ table, will contain all specimen records of one dataset and will later on be created.

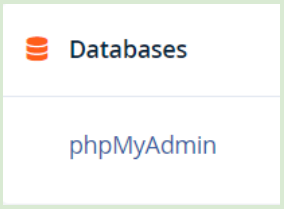
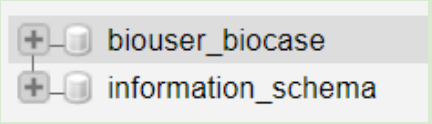
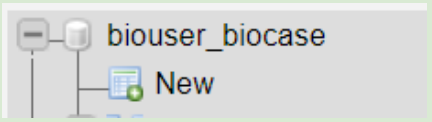
To create an online database with the tables, we first need to initiate MySQL on the web server. The easiest way is to use the graphical cPanel user interface. Next to MySQL we use phpMyAdmin, which is a built-in database management application. PhpMyAdmin allows executing otherwise complicated tasks on the command line via a graphical user interface (GUI).

Table 18. Create a new MySQL database on the web server.

1. Open up the cPanel interface on port 2083 (e.g. https://biocase.nlbif.nl:2083/) and login with your username and password.	
2. Scroll down to the databases tab and click on 'MySQL Databases'	
3. Now, create a new database with a relevant name (rather not use captions, spaces, etc.). Fill in the name and click 'Create Database' to finish	

Next, we create three tables. Only three because the 'occurrences/specimens' table will be created later using the specimen dataset (Table 18).

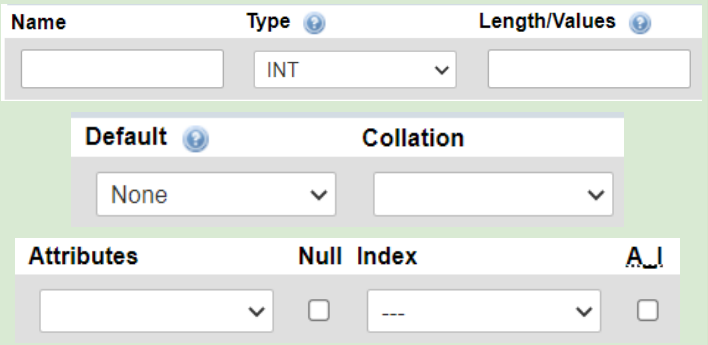
Table 19. Manually create a MySQL table.

1. Open phpMyAdmin through the cPanel interface on port 2083, e.g. https://biocase.nlbif.nl:2083/	
2. Select the created 'biouser_biocase' database	
3. Create a new table by selecting 'New'. This will be repeated three times to create three tables	

4. Fill in the table names at the top
- institutions
- collections
- collectionsMetadata

(Example uses the collectionsMetadata table)

5. Add the required variables. Pay attention to the:
- Name (name of the variable)
- Type (datatype of the variable)
- Length/Values (length of the variable)
- Default (default value of the variable)
- Null (empty value of the variable)
- A_I (auto increment value of the variable)



See **figure 4** (institutions), **figure 5** (collections) and **figure 6** (collectionsMetadata) for reference.

Click 'Go' to save the table.

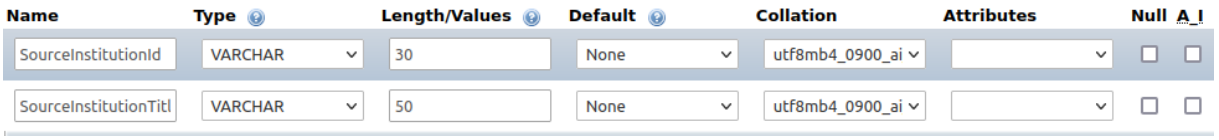


Figure 4: institutions column names and settings.

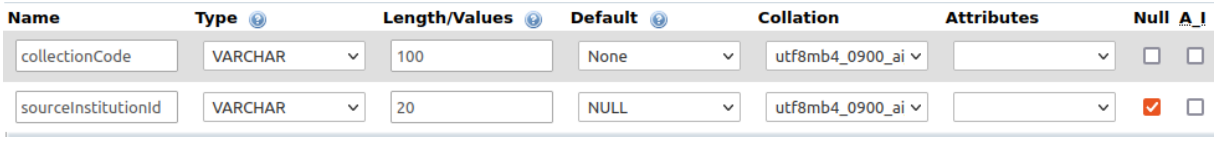


Figure 5: collections column names and settings.

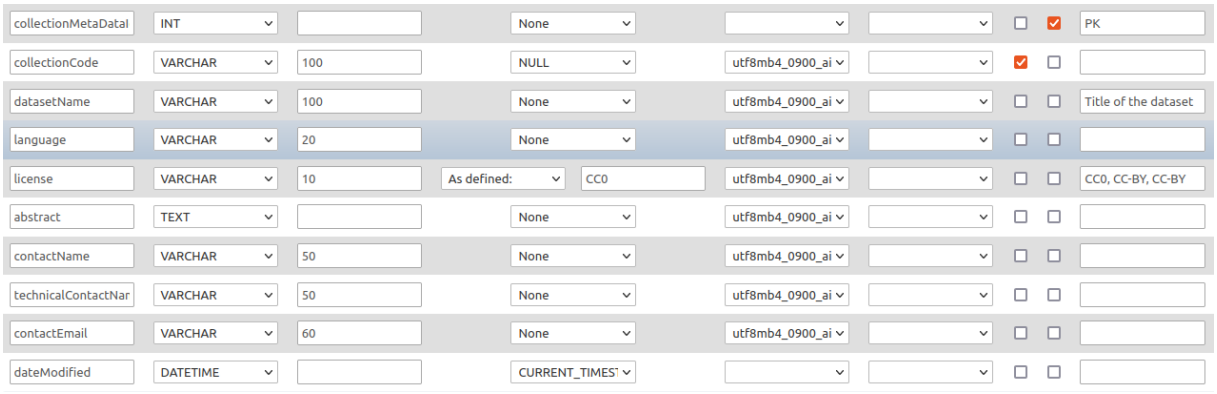


Figure 6: collectionsMetadata column names and settings.

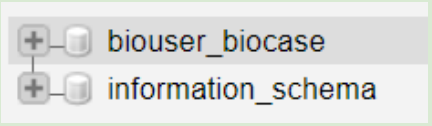
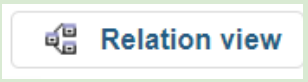
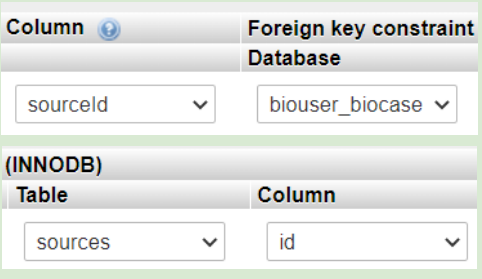
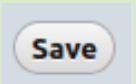
The metadata table is of uttermost importance because this table holds almost all of the bottomline required fields for using the ABCD-EFG standard. Make sure to have these fields present in your database table, otherwise you will most likely encounter issues in the future.

Defining relations between database tables

The last step in preparing the database is to create connections between the tables. These so-called relations are defined by foreign key constraints. A constraint is constructed by linking a certain field of one table to an unique field of another table. Unique fields are often identifiers as an id' like the ROR identifier. An example of a relationship is the connection between the Sources and Source_institutions tables. In this example the ROR identifier value of the Sources table can be linked to the unique ROR identifier value of the Source_institutions table which indicates that the Petrology collection belongs to the Naturalis institution. The relations are an integral part of databases and are essential for BioCASe to function.

In the following section you will create relations between the Meta, Sources and Source Institutions tables. The metadata belongs to a collection and should refer to the collection by using the collection code, or another unique identifier. As described above, the Sources need to link to the Source_institutions. A relationship between database tables is defined by selecting the database, then a table and clicking on 'Structure'. Relations to our actual collection datasets will be made in the next chapter.

Table 20: Define database relations

1. Select the database	
2. Select the Meta, Occurrences and Sources table one per one (Meta table as example in the image)	
3. Click on 'Structure'	
4. Now click on 'Relation view'	
5. Take over the values as shown in **figure 7** (Meta table) and **figure 8** (Sources table). Note that the database and table fields could be different in your database due to alternate naming	
6. Click on 'Save'	

Meta:

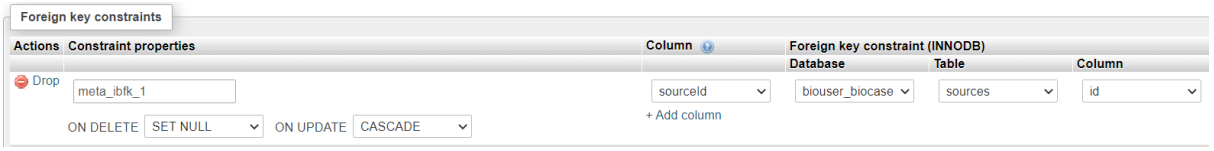


Figure 7: Relations of the meta table

Sources:

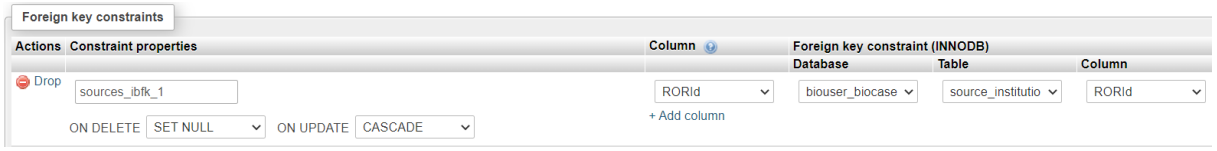
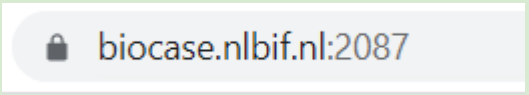
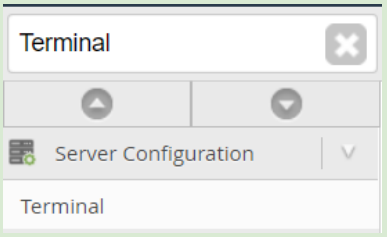


Figure 8: Relations of the sources table

Creating a MySQL user

To enable the BioCASe Provider Software to access the database, it is mandatory to create a MySQL user account with the reading privileges. This MySQL user will be able to login to the database from BioCASe and consult the stored data. In the following example we use the method that works for MySQL. If you are using a different database type, please look up an equivalent method.

Table 21: Creating a MySQL user

1. Login to the WHM portal on port 2087	
2. Search and click on 'Terminal'	
3. Start MySQL in the terminal with typing: 'mysql'	mysql
4. Use the following command to create a new user, specifically for BioCASe. Think of a relevant name and password; and make sure to document these in a safe place.	CREATE USER 'username'@% IDENTIFIED BY 'password';

Worthwhile mentioning is the use of the percentage sign, this indicates that the MySQL user has access to all sections of the server. If specified as localhost for example, this could clash with our BioCASe software. Because BioCASe runs on port 8080 extended as 80, we would not be able to connect to MySQL and use the source data.

If correct, a user is made. To check this fact, use the following command in the MySQL terminal to list all MySQL users. If your user is not present, try to add the user again.

Select all users from within MySQL with the **select** command	mysql select user from mysql.user;

To finish the task, we give the user reading privileges to be able to read out the database.

1. Again, open the MySQL terminal by typing 'mysql' in the WHM terminal	mysql
2. Select the database, the one we created in **Chapter C**	use <database name>
3. Use the following command to grant read privileges on all tables to the MySQL user	GRANT SELECT ON * . * TO 'username'@%;

If SELECT eventually does not work with the BioCASe software, check if the user will function with a grant on all privileges. This could be harmful to the database by certain applications. However, it is stated in the BioCASe documentation that it only reads the database and could therefore be trusted. Use this command to grant a user all privileges:

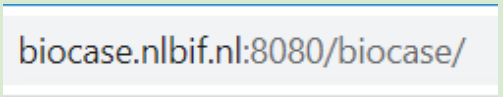
GRANT ALL PRIVILEGES ON * . * TO 'username'@%;

Data source set-up

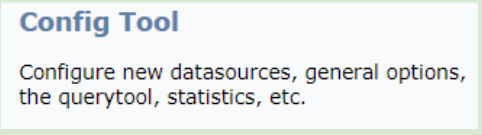
With the BioCASe software installed on the web server, as described in **Chapter C**, we can now set up a data source containing the collection data. The original source, as defined in **Chapter D**, consists of a database with three tables. In order to use this database, it is mandatory to have the credential of a valid MySQL user account. With the user account, it is possible to connect BioCASe with the MySQL database. This process requires the user account information, the MySQL database name, and the IP address of the server domain. The IP address is often found in the initial confirmation email of the set-up of your web server, or simply login to the WHM portal and open the terminal. The IP address is displayed on top (*see figure at step 6*; here the IP address is 87.253.148.112).

Connect BioCASe to the MySQL database:

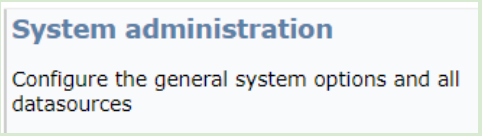
1. Open BioCASe at you domain, e.g.: http://87.253.148.112:8080/biocase/ or http://biocase.nlbif.nl:8080/biocase



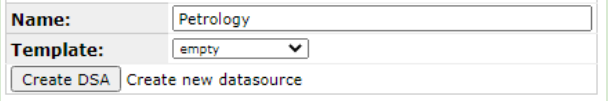
2. Click on 'Config tool'



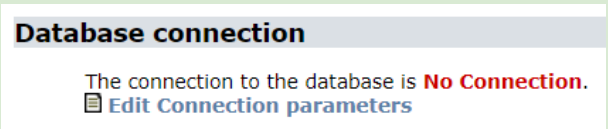
3. Now click on 'System administration' and login with your BioCASe password



4. Add a new data source and give it a name, e.g. 'Petrology' for a petrology collection, and click on 'Create DSA'.

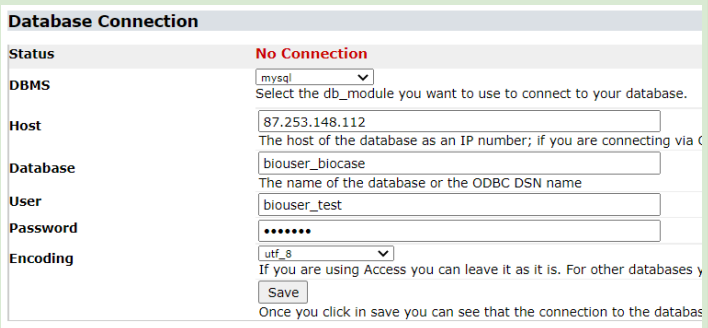


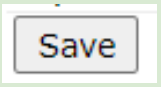
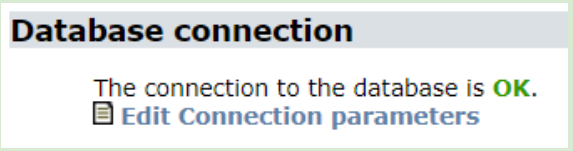
5. Click on 'Edit connection parameters'



6. Fill in the correct details as we have defined in earlier steps:

- DBMS → mysql
- Host → IP address of the web server
- Database → Name of the database
- User → Username of the MySQL user



• Password → Password of the MySQL user • Encoding → The encoding type, set to utf-8 per default	
7. Click on 'Save' to save the configuration	
8. If the connection is successful, the red painted 'no connection' will transform to a green 'OK'.	

After clicking on 'Save' the MySQL database connection parameters are saved in the configuration file of BioCASe, meaning that they will be preserved on a new start-up of the Docker container. BioCASe will also try to connect with the database.

If the connection fails, review your connection parameters on potential typing errors. Also try to give the MySQL users overall privileges as stated in the previous error section. Make sure to contact the contact person of Naturalis if this still does not work.

Mid time sum-up

If you have executed all predefined steps of Chapters B, C and D, you should by now have a technical set-up that is ready to handle the BioCASe Provider Software to mobilise geological datasets to the GeoCASe 2.0 portal. A short sum-up of the topics that were discussed in this technical section of the manual:

- How to configure a web server
- Security measures for the web server
- Introduction to the Docker software
- Hands on with Docker to install the BioCASe Provider Software
- Preparation and set-up of the database
- Inserting metadata into the database
- Creation of a MySQL user
- Connecting the BioCASe Provider Software to the database

The separation of the technical section from the next section, that is more focused on how to use the BioCASe Provider Software as a user, caused some overlap between chapters. The reason for this is the current structure of this manual that minimises the technical debt, a regular user that just wants to mobilise their collections to GeoCASe and makes use of the NLBIF service, has to go through. Important to know for all readers is that every new source institution or collection has to register some metadata in the database as the text in the above sum-up in bold indicates. Please make sure to at least read the paragraph about inserting metadata from Chapter D when you are representing a new institution or collection and are making use of the NLBIF service.

Please contact the contact persons in the appendix when you do not know how to do this or need permission to enter the database.

Part two of this manual will focus on how to prepare a dataset in the database, configure it in the BioCASe Provider Software and finally mobilise it to the GeoCASe 2.0 Portal. Users of the NLBIF service are encouraged to read all coming chapters.

Part 2: Mobilising geological collections to GeoCASe 2.0

Image

E: Handling the geological dataset

This chapter will explain how to prepare a geological dataset for further usage. The prepared dataset will be used to populate the database the BioCASe Provider Software will encapsulate. Besides the dataset, metadata about the collection and/or source institution will be inserted into the database. Finally, a relation will be defined between the dataset and the metadata.

The steps in this chapter are required to execute when making use of the NLBIF service. Please do not hesitate to ask for help when you get stuck. Contact the contact person in the appendix if so.

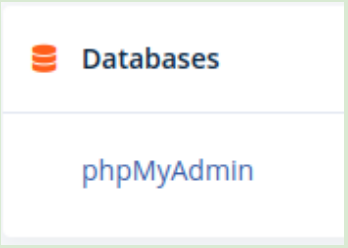
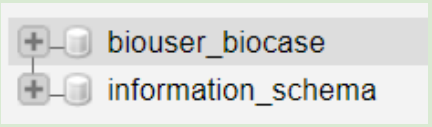
Inserting metadata

Let us start with adding some required metadata to the database. We start with this part because it is the most technical and complex to understand if you did not read the first part of this manual. This part is only necessary when you represent a new source institution or collection.

There are three metadata tables: Meta, Sources and Source_institutions. These can be found in the cPanel instance of the web server, found at port 2083 of the web server (for example at *biocase.nlbif.nl:2083*). This instance allows us to navigate to the database for inserting data into the tables. To insert metadata into the tables, we use the insert function. This function can be found by selecting the database name in phpMyAdmin, which is a database management interface, and activating the 'Insert' tab (Fig. #). Then we can fill in the table fields with the corresponding information. Examples of Naturalis' tables are shown below. Important to notice are the id' which are used to create a relation later on. We want to point from our future specimens table to the metadata using these id'.

Note: Credentials for using cPanel and MySQL (the database) when making use of the NLBIF Service will be provided to you upon contact.

1. Navigate to the CPanel interface of your server (or NLBIF's when using the NLBIF service)	

2. Login to cPanel using your credentials	
3. Scroll down and click on 'phpMyAdmin', this will redirect you to the database interface in a new tab	
4. Select the database	
5. Select the Meta, Sources and Source_institutions tables, one for one (Meta table as example in image)	
6. Insert the required variables from your source institution or collection as visualised per table in the images below	

Meta:

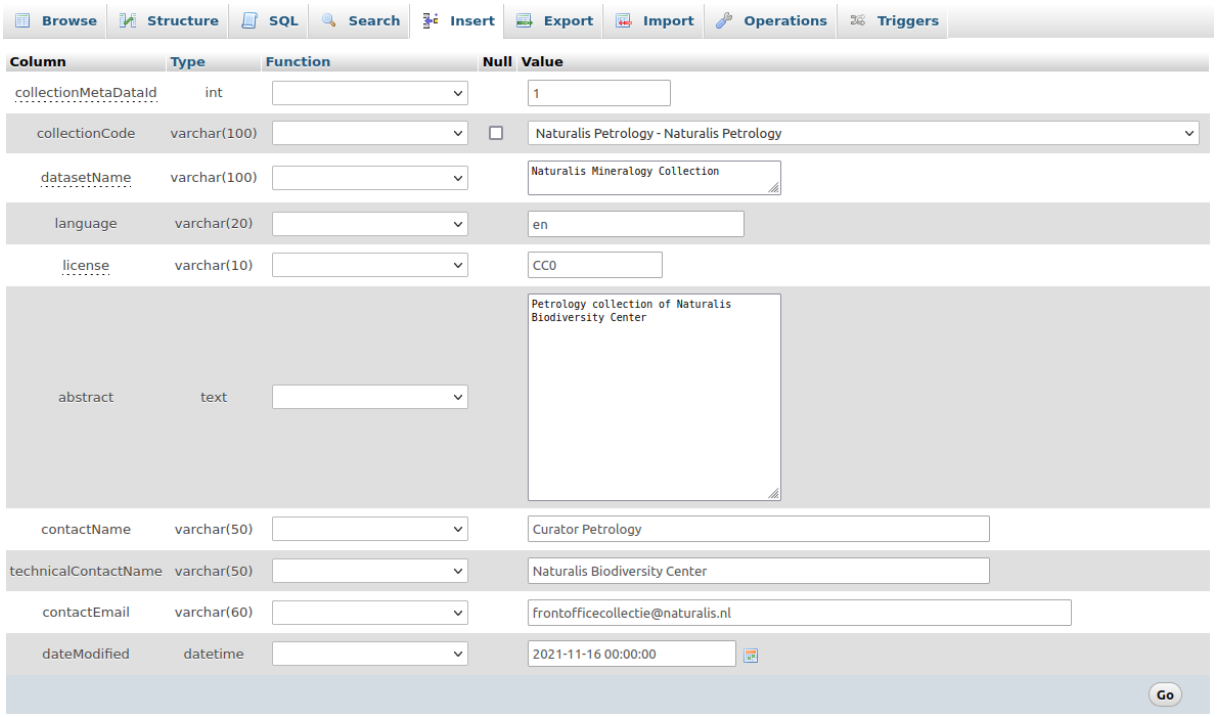


Figure 9: Inserting content to the Meta table.

Sources:

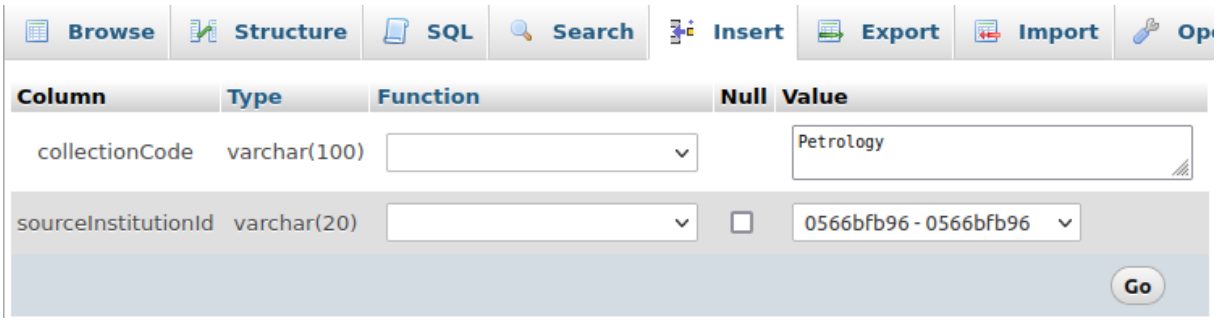


Figure 10: Inserting content to the Sources table

.

Source_institutions:

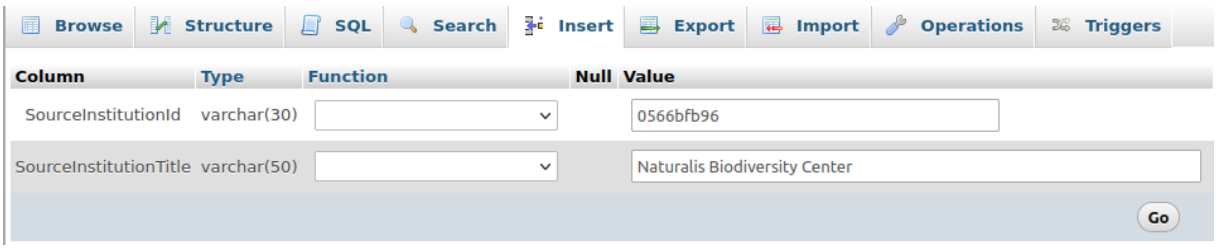
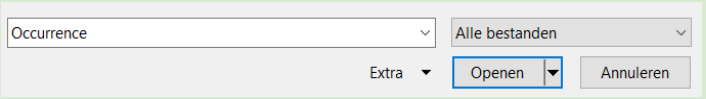
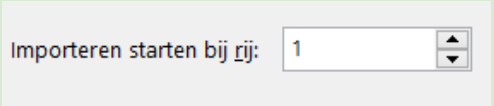
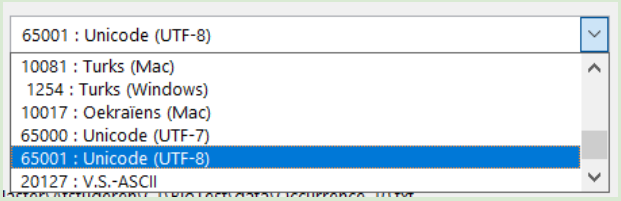
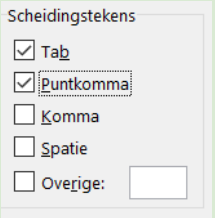


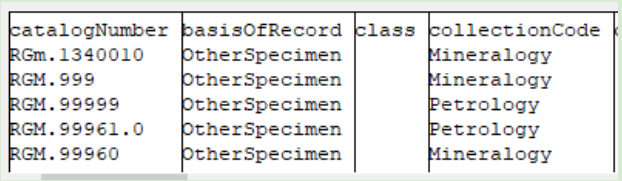
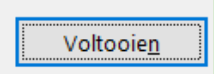
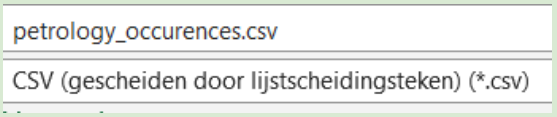
Figure 11: Inserting content to the Source_institutions table

Preparing the dataset

With the metadata done, we can now work on our geological dataset. Before you upload data to the database and create a occurences/specimens table containing these data, it is important to prepare the dataset you want to mobilise. Here the Petrology dataset of Naturalis Biodiversity Center is used as an example. This dataset is a DarwinCore text file export from the Naturalis Collection Management System. Because plain text files are not supported by phpMyAdmin the data needs to be transformed to CSV format. This can be done by using Microsoft Excel. If you already have stored your data in a CSV file, you can skip this step. When stored in an Excel file, please also convert it to a CSV because these files are much smaller and better handable by the web server.

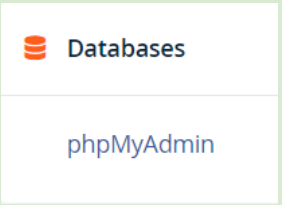
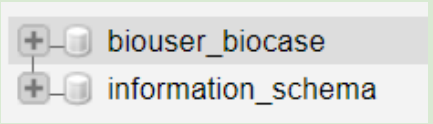
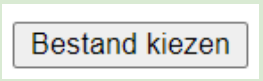
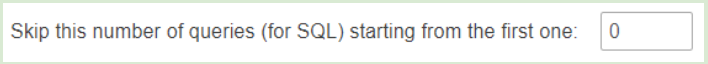
Table 19. Convert *.txt to *.csv

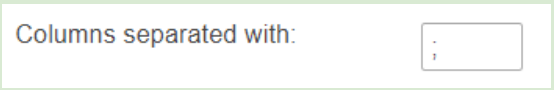
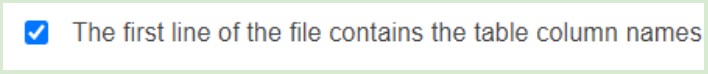
1. Open Microsoft Excel	
2. Click on open file and select the DarwinCore *.txt file (make text files visible in MS Excel by checking 'show all files')	
3. In the opening editor, start importing rows at line one (image translates to: start import at row: 1)	
4. Check 'my files contain headers' (image translation) and proceed to the next step	
5. Set the original document type to: 65001: Unicode (UTF-8)	
6. Depending on the type of separator used in the text file, check the separator, Naturalis uses a comma as the default separator (you can recognize the correct separator if Excel shows the rows separated by lines in the preview image)	

	
7. Press proceed and at last complete to convert to a Excel file (image translates to: complete)	
8. Save the file by default and choose 'CSV UTF-8 (comma separated)' as the file type. The image translates to: CSV (list separator)	

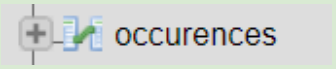
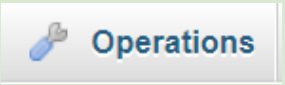
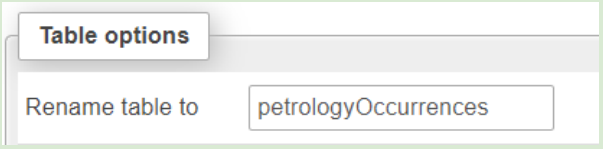
Importing the dataset into the database

Now it is time to create the fourth table: the occurrences/specimens table, based upon the CSV file.

1. Again, open phpMyAdmin in the cPanel interface	
2. Select the created database	
3. Choose the 'Import' tab from the above row	
4. Press 'Choose file' (image translation) and select the CSV file we generated.	
5. Make sure the import starts at row zero since this contains our headers	
6. Check if the format is set to CSV, if not select CSV	

7. Define the separator you used for creating the CSV, in the case of Naturalis a point comma (;)	
8. Check the option that declares that our first line contains headers and thus the names of the fields	
9. Click 'Go' and wait till the program is finished (this can take a few minutes depending on the size of the dataset)	

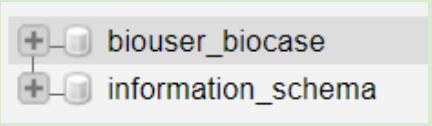
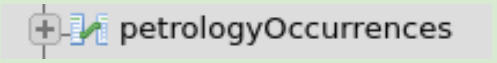
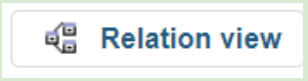
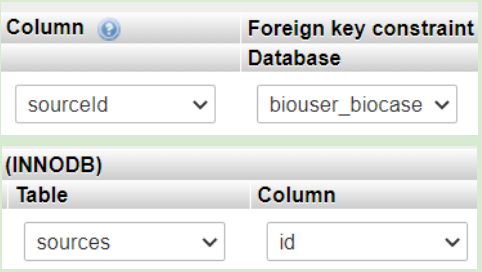
The table is now created and added to the online database. Change the name of the table to a descriptive name, e.g. petrology_occurences. We prefer that you also mention your institution credentials or name in this name like: 'umuFossilOccurences' (University of Utrecht fossil occurences). Do not use spaces or special characters, camelcase is encouraged.

1. Select the database and then the just created table to rename (will be named after your original datafile)	
2. Click on 'Operations'	
3. Fill in the new name at 'Rename table to'	
4. Press 'Go' to save the name	

If the upload process of the CSV file fails, try to identify the error phpMyAdmin returns. A possible reason is a limited timeout. This means the process of uploading is taking longer than the maximum allowed time frame for executing processes. Another reason can be that a wrong separator was given. Check if you used the correct separator for the CSV file and retry the upload. When reuploading check If the headers of the data table were created. If so, start at line one and do not check the option that declares the first line contains headers.

Define a relation

To link our dataset data to the metadata we defined earlier, we need to define a relation between these tables and our newly created occurrences/specimen table in the database. Our technical set-up allows us to only need to define one single relation.

1. Again, select the database in phpMyAdmin	
2. Select the occurrences/specimens table that holds your dataset	
3. Click on 'Structure'	
4. Now click on 'Relation view'	
5. Take over the values as shown in **figure 10** (Meta table), **figure 11** (Occurrences table) and **figure 12** (Sources table). Note that the database and table fields could be different in your database due to alternate naming	

Occurrences (specimens):

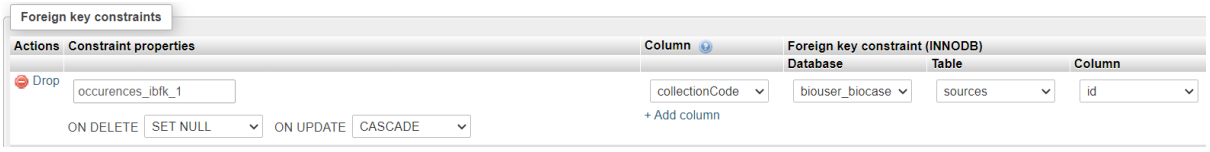


Figure 12: Relations of the occurrences (specimens) table

Additional datasets

Now that the relationships between the tables of our MySQL database are defined it is suited for use by the BioCASe software. In the case of a new dataset, e.g. a Mineralogy collection, this will have to be defined separately in a seconds Occurrences table. This table will be created in the same manner the occurrences table was created for the Petrology collection, including the definition of relations. Additionally, a new record has to be added to the Meta table that describes the metadata for the Mineralogy collection. Also a new record in the Sources table would be created representing the Mineralogy collection. No changes need to be made to the Source_institutions tables, unless a collection from a different institution will be added.

F: Configuring and using the BioCase Provider Software

After all preparations we can finally configure and use the BioCASe Provider Software to mobilise geological data to GeoCASe. This process contains some specific settings that will be discussed to optimise the usefulness of the data. Once installed and configured, we will be able to register our BioCASe instance at GeoCASe and publish the data online. Please refer to the terminology in the **Appendix** for terms used in this chapter.

When making use of the NLBIF service, it is mandatory to create a new data source in BioCASe. Also, the configuration of the geological dataset and mapping process needs to be done in BioCASe. This configuration can be done relatively easily and could be a good experience on how to handle your data in this online environment. As the NLBIF Service will be making use of the existing server and BioCASe installation, it is not necessary to install a new variant of the BioCASe software on a different location.

Please do not hesitate to ask for help when you get stuck. Contact the contact person in the appendix if so.

The purpose of BioCASe

BioCASe and its software called the BioCASe Provider Software, is software which creates an access point to online database services from where data can be requested. Much like API's[T], certain commands can be given to request data via an XML[T] protocol. XML (Extensible Markup Language) is a type of structured syntax[T] that can easily be read by both humans and computers. BioCASe supports a number of different data standards in which data can be requested that are widely recognized in the biological and geological communities, e.g. DarwinCore and ABCD (Access to Biological Collection Data) with its EFG (Extension for Geosciences) extension. By using BioCASe, geological collections can be shared in the public domain through a recognized data standard in an usable online endpoint. At present the use of BioCASe is a requirement when sharing data with the GeoCASe portal.

How it works

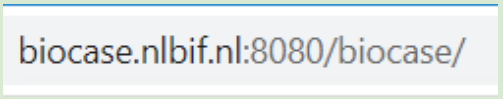
BioCASe is a so-called PyWrapper[T], a software package that allows primary biodiversity/geology data publishing from any arbitrarily structured database. BioCASe makes use of a database to create the access point to request data. We have seen this database before in the preparational steps. With this database, the next step is to configure BioCASe. First, a connection with the database needs to be made which requires the creation of a database user with admin privileges. Secondly, the database tables can be configured in BioCASe by for instance defining the primary keys and relations between tables. A primary key identifies an unique record in a database table. The 'RORid' for example is the primary identifier variable for institutions in the Source_institutions table. Lastly, the data should be mapped to a data standard. Because GeoCASe is an Earth Science Collections Portal we will focus on the ABCD-EFG standard

Database table configuration

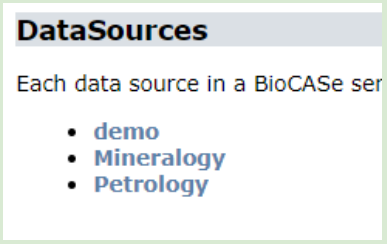
The second step is to configure our database tables in the BioCASe Provider Software. This is a requirement because BioCASe will use these definitions to query the database. This is why the relationships between tables in the MySQL database need to be correctly defined as described in **Chapter D**. The configuration of the relationships between the database tables is initiated with the setup of the BioCASe connection and represents a one to one copy from the MySQL database. A few elements need further definition: primary keys, foreign keys and variable types.

Configuring the database tables in BioCASe:

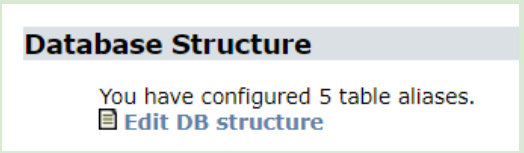
1. Open BioCASe at your domain, as for example: biocase.nlbif.nl:8080/biocase



2. Click on the data source you added in the previous step, for example a Petrology dataset (or create a new one, see **page 0**)

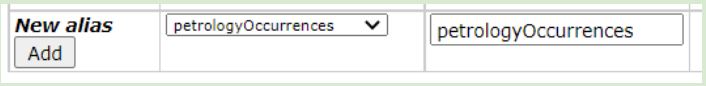


3. Next click on 'Edit DB structure' (login if necessary)

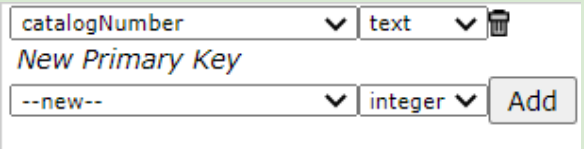


This page shows the entire configuration of the database. **Figure 13** shows a table in which the elements can be defined. After configuration of the table relationships between the Primary keys and the Foreign keys, BioCASe renders an image of the database structure (*figure at step 7*).

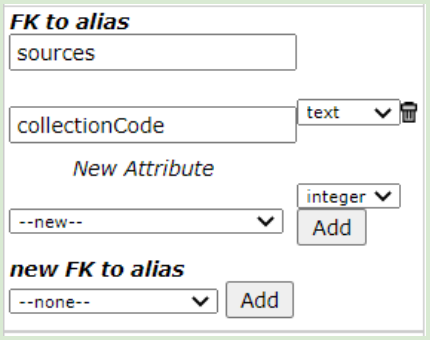
4. Add the tables by selecting the name from the drop down menu and clicking 'Add'



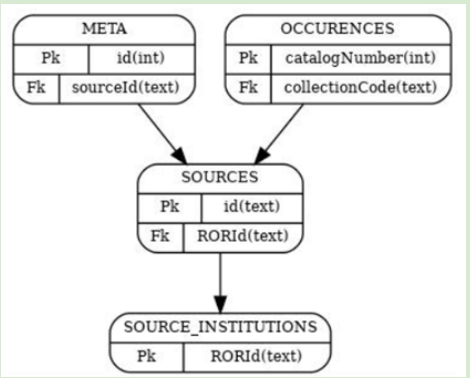
5. Define the primary key (i.e. unique record Id) for each table. Use the example as an indicator (*figure 12*). Mind the field types; e.g. the catalogNumber of records in the 'occurrences' table is alphanumeric hence 'text'



6. Define the foreign keys, as in relations, from each table to the other tables (*figure 13*)



7. Check if the rendered image complies with the example figure. Names may differ based on your dataset, but types should be similar)



8. Click on 'Save' to save the table configuration

	Table/View	Alias	Primary Key Attribute(s)	Foreign Key(s)
Delete alias	meta	meta	id integer New Primary Key --new-- integer Add	FK to alias sources sourceId text New Attribute --new-- integer Add new FK to alias --none-- Add
Delete alias	occurences	occurences	catalogNumber text New Primary Key --new-- integer Add	FK to alias sources collectionCode text New Attribute --new-- integer Add new FK to alias --none-- Add
Delete alias	source_institutions	source_institutions	RORId text New Primary Key --new-- integer Add	new FK to alias --none-- Add
Delete alias	sources	sources	id text New Primary Key --new-- integer Add	FK to alias source_institutions RORId text New Attribute --new-- integer Add new FK to alias --none-- Add

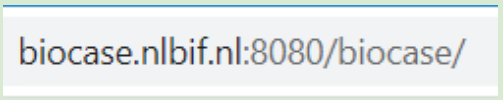
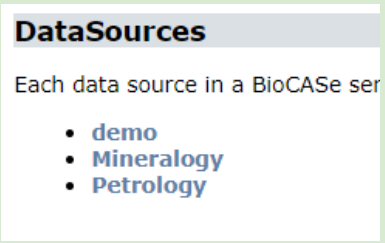
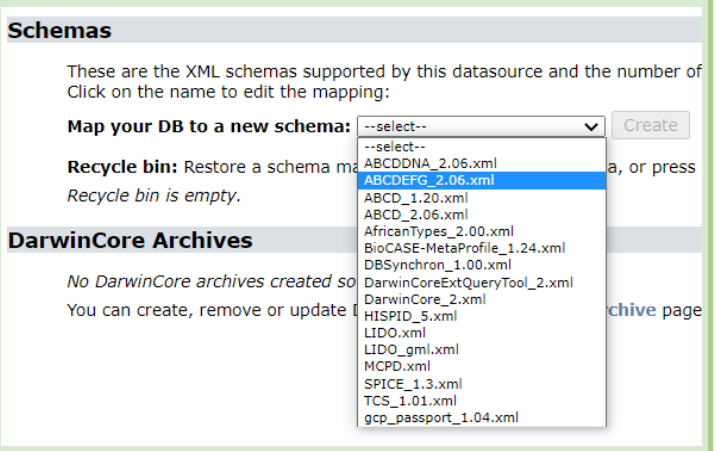
Figure 13: Table configuration in BioCASe

Mapping data to the ABCD-EFG data standard

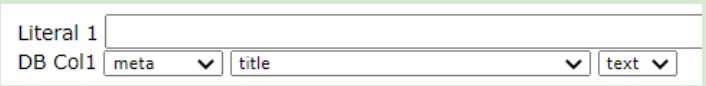
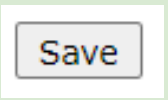
The last step of this chapter describes the mapping process that the BioCASe Provider Software offers. This process declares the data fields of the chosen data standard. As such, we create connections between the database fields and the data standard. Once the database fields are mapped to the data standard the data can be exported from BioCASe to XML documents.

First, a data standard has to be selected. We use the ABCD-EFG standard as this standard is widely recognized in the geological community and is used by the GeoCASe portal. The ABCD-EFG data standard proposes a set of mandatory elements which were prepared in **Chapter D** when the database tables were created. The other fields in the dataset, which are a lot, are made up of optional fields. However, when defining certain variables like the licence type for the object, some additional variables are mandatory like 'language'. Required fields are marked in red.

Mapping required fields for using the ABCD-EFG data standard:

1. Open BioCASe at you domain, e.g. biocase.nlbif.nl:8080/biocase	
2. Click on your configured dataset	
3. In the schemas section, select ABCDEFG_2.06.xml from the dropdown menu (Fig. x) and click 'Create'	

This will open the mapping section displaying all the premature fields of the ABCD-EFG standard. Now, link the fields, that are marked red, to their counterparts in the database. We will be showing an example of the title field from the metadata.

1. Click on the plus icon next to 'title', this will open a new window	
2. In this window on the first row, select the 'metadata table' and then the 'title' field. Also make sure to set the field 'type correspondent' to the database, in this case to text	
3. Press 'Save' to save the mapping. This will close the window and reload the page displaying BioCASe	

Congratulations, you have just mapped your first element! If correct, the overview page of the mapping will display this change. The email field will have turned green and the value we bound from the meta table will be displayed. Now repeat the above steps for all the visible, red fields. In the end the result should look something like **figure 14**.

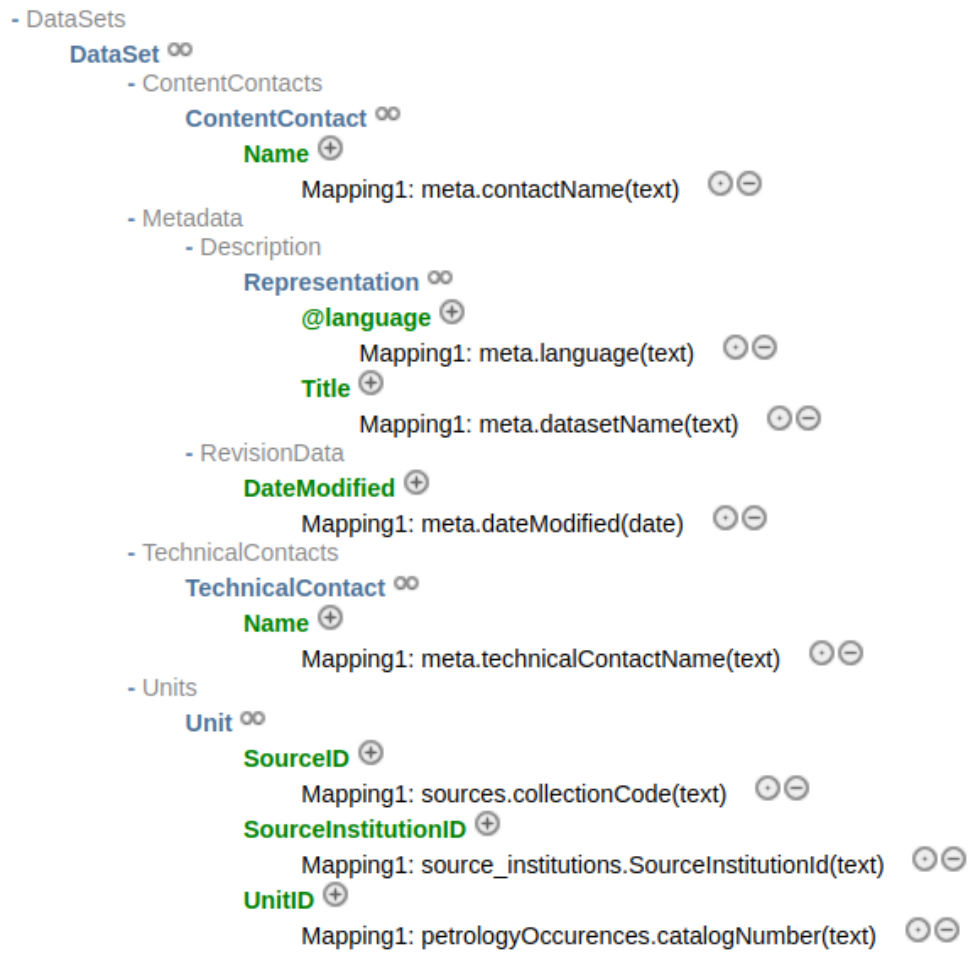
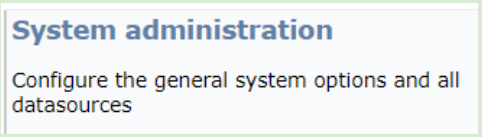
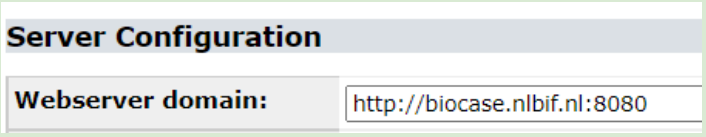


Figure 14: Pre required ABCD-EFG fields

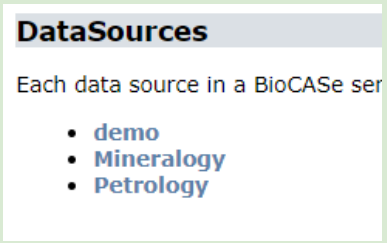
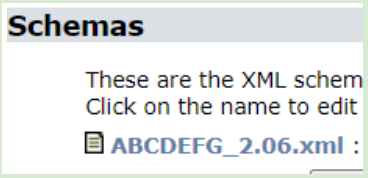
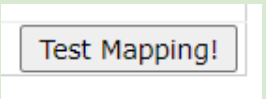
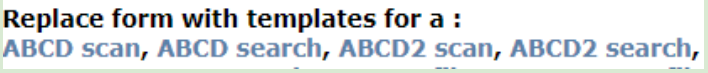
With these fields filled we are able to create the most simplest export of our data. This is a good time to test the basic fields because we have used all four base tables from the database. If this test succeeds, it can be assumed our database configuration functions correct and we can proceed with the mapping. To execute the test we will be using the manual query form that is built into BioCASe. To use this query form we will first need to define our endpoint. This endpoint represents the URL from which data can be retrieved by using the BioCASe protocol.

Configure the endpoint:

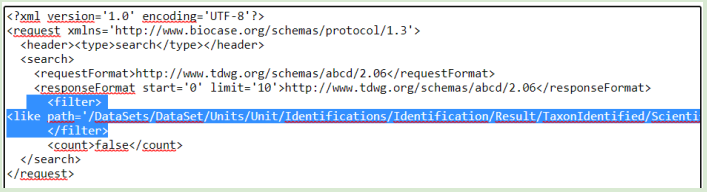
1. Open the system administration section from the home menu of BioCASe	
2. In the server configuration section, change the web server domain to the domain of your server installation as for example: http://biocase.nlbif.nl:8080	

With our endpoint correctly defined we can execute a request via the BioCASe protocol and test our initial mapping.

Test our mapping:

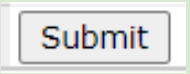
1. Click on your configured datasource	
2. Now click on the ABCDEFG_2.06.xml schema	
2. From the mapping overview page of the data source in BioCASe, click on 'test mapping'	
3. From the templates menu, select 'ABCD2 search'	

4. Remove the filter portion of the statement



5. Make sure the wrapper option contains the endpoint of your server, with the addition of the PyWrapper and data source statement

6. Press 'submit' to execute the query



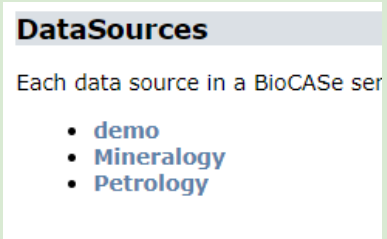
After processing the query, BioCASe will display its response. This response should contain the fields that you defined in the initial mapping. If the result contains errors, read why the error is given and try to look this up in the mapping, database configuration or endpoint definition.

Furthermore, warnings indicate things that did not process well such as mapped fields which are empty in the database. They are not mandatory and therefore do not stop the protocol from working. The return script from BioCASe is pretty readable for humans, please try to fix as many of the warnings as possible.

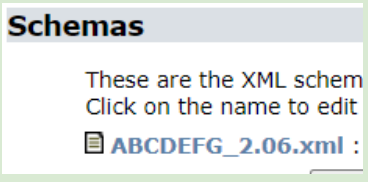
When satisfied with the initial result, you can continue to map more fields out of the broad selection the ABCD-EFG standard has to offer. The method for mapping stays the same, however, you will need to enable the visibleness of these fields. To set an example we will be mapping another field. We have chosen to map the scientific name, because this is an often recurring field.

Map the scientific name:

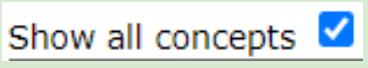
1. Click on your configured datasource

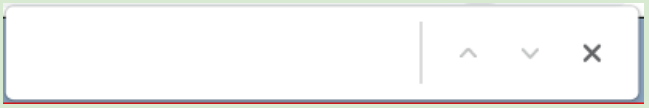
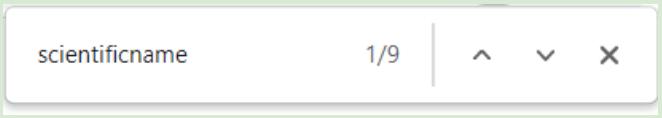
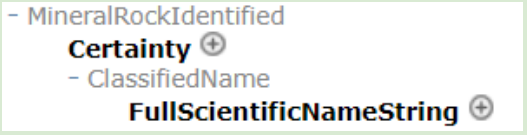
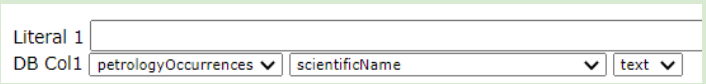
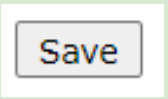


2. Navigate to the mapping overview page in BioCASe from the schemas section



3. Check the checkbox that says



'show all concepts'. This will reload the page and show all the fields of the ABCD-EFG standard	
4. Press on CTRL and F on your keyboard to initiate your browsers search function	
5. Type and search for 'scientificname'. This should give multiple hits, each for a different kind of natural group. We want to map the scientific name for a geological collection.	
6. Filter through the search results until you find a scientific name element called: 'FullScientificNameString' in the category of 'MineralRockIdentified'	
7. Press the plus button next to 'FullScientificNameString'	
8. Bind the value of the occurrences (specimens) table to the field	
9. Click on 'Save' to save the mapping	

Now we have mapped the scientific name to the correct equivalent of the ABCD-EFG standard. Bear in mind that the sorting process is very important. You could want to be mapping the scientific name for geological purposes to 'MineralRockIdentified', but end up mapping it to 'Mycology'. It is important to scan the names of the groups you map to and ask for someone to review the mapping (especially if you lack knowledge of the biological and geological terms).

Repeat the mapping process for all the fields you can or want to map. **Chapter G** offers a very useful list of the most used fields for mapping geological data to the GeoCASe portal. Use this list to check the quality of the data you are going to submit.

G: Mobilising data to GeoCASe

The last step of this manual is the mobilisation of geological data to the GeoCASe portal. This step is relatively short but of most importance for making the data available for biodiversity scientists.

When using the NLBIF service, it is best to send an email to GeoCASe together with the contact person of NLBIF. It is important to discuss the way of handling and presenting data in GeoCASe, such as incorporating the data under the institution name of Naturalis or keeping it bound to a separate institution. Since the NLBIF server already is registered at GeoCASe, it would only be necessary to include new details as the endpoint of the data source.

The mobilisation process cut short

GeoCASe, which is short for: Geoscience Collection Access Service, is a data network including an internet portal that is developed for creating an universal point where geological data can be found by biodiversity scientists and other interested parties. From this point of view it has a similar goal to DiSSCo, but focused on the geological aspect of data from institutions. GeoCASe will someday be a part of the DiSSCo network. Hence, why it is a useful step to mobilise data to GeoCASe as a preparation for the future infrastructure that can request this data.

Mobilising data to GeoCASe can be done using a XML based tool that transforms data to a readable data format. ABCD-EFG and DarwinCore are both supported. As mentioned earlier we will be using ABCD-EFG, as this is a widely accepted data standard in the geological community. With the preparations of all previous chapters, we are able to transform our source data to ABCD-EFG and enable GeoCASe to pull this data.

Publishing data to GeoCASe

The method for publishing data to GeoCASe is probably the most simple we have discussed yet. We simply have to communicate the URL of the endpoint of the BioCASe installation to the people who manage the GeoCASe portal. The organisation who manages GeoCASe is called TalTech, also known as the Technical University of Tallinn. They manage GeoCASe via a back-end tool which also can be used to add new collections to the portal. By providing the endpoint URL of the BioCASe installation, TalTech will be able to pull the source data via a similar request as we have seen in **Chapter E** when using the query tool. The request will activate the BioCASe protocol and let it generate a response. This response will be used to generate the data pages in the portal.

To send the endpoint URL, you can make use of regular email. Send an email containing your information, such as the institution name and include the endpoint URL to the following address: geocase@mfn.berlin. This address refers to a German region. This is because the project originates in Berlin. After receiving the email, the people behind GeoCASe will get in contact with you and handle your data. Suggestions could be made for improving the data mapping or providing additional information regarding your institution.

If everything worked, congratulations! Your data should be available at the GeoCASe portal. Navigate to geocase.org, choose search and filter on your institution's or data's details to see the data in the GeoCASe 2.0 portal's view.

Appendix

Contact persons / NLBIF service

Summary of possible occurring errors

Chapter B:

If the installation of the server or defining the domain causes any problems, make sure to contact your web server provider to get support. It is important to set up the web server correctly to avoid problems further in the process to mobilise data to GeoCASe.

If the cPanel user is not working properly you should check if you have filled in the correct credentials and domain name. This can be done by searching and clicking in the WHM search bar onto: 'Modify an Account'. Then select the right account, if only one is existent, this account will automatically be displayed.

If you do not possess the correct knowledge to improve the security of the server, make sure to contact your IT-department to review the settings and security advice.

Chapter C:

If Docker does not start or run well, try to reinstall the software. This can be done by removing Docker with: sudo yum remove docker; and installing it once more with: sudo yum install docker.

Chapter D:

If the upload process of the CSV file fails, try to identify the error phpMyAdmin returns. A possible reason is a limited timeout. This means the process of uploading is taking longer than the maximum allowed time frame for executing processes. Another reason can be that a wrong separator was given. Check if you used the correct separator and retry the upload. When reuploading check If the headers of the data table were created. If so, start on line one and do not check the option that declares the first line contains headers.

Chapter E:

If SELECT eventually does not work with the BioCASe software, check if the user will function with a grant on all privileges. This could be harmful to the database by certain applications. However, it is stated in the BioCASe documentation that it only reads the database and could therefore be

trusted. Use this command to grant a user all privileges:

GRANT ALL PRIVILEGES ON * . * TO 'username'@%;

If the connection fails, review your connection parameters on potential typing errors. Also try to give the MySQL users overall privileges as stated in the previous error section. Make sure to contact the contact person of Naturalis if this still does not work.

Furthermore, warnings indicate things that did not process well such as mapped fields which are empty in the database. They are not mandatory and therefore do not stop the protocol from working. The return script from BioCASe is pretty readable for humans, please try to fix as many of the warnings as possible.

MIDS for mapping geological data

xxx

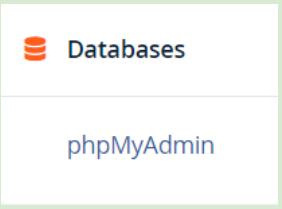
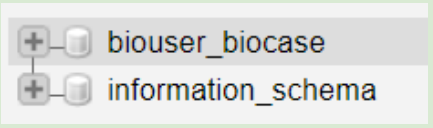
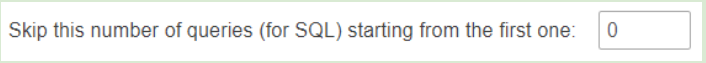
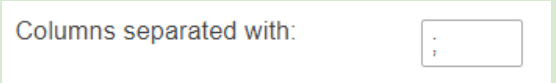
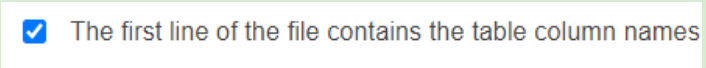
Mobilising multimedia items to GeoCASe

When, as an institution, you are also publishing media items along with your geological dataset to GeoCASe, it is good to know how this slightly different approach is executed. In basic, there are no completely new steps that need to be executed as adding media builds upon the existing steps. The images, assuming they are hosted on another existing web server, can be added to the database as their links will point to the original media item. An export of the images (preferably in CSV format) from your image web server will do the trick. This file will, like the occurrences/specimens file, be uploaded to the database to create a new table containing the links to the media items. Then, the new table can be configured in the BioCASe Provider Software, whereafter the media items can be mapped to the multimedia concepts of the ABCD-EFG standard.

Let's start with adding the multimedia table to the database. As mentioned, you can use an export file of your image server (or other storage) for this. Please convert your file to a CSV file as these are best handled by the database.

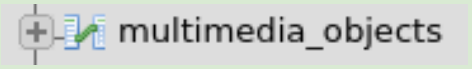
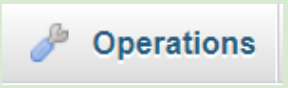
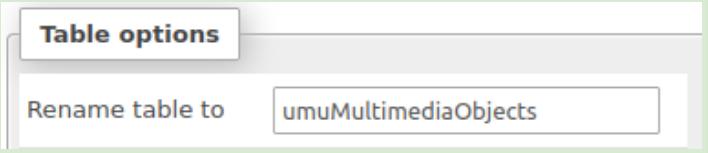
Add the multimedia table

1. Navigate to the CPanel interface of your server (or NLBIF's when using the NLBIF service)

2. Login to cPanel using your credentials	
3. Open phpMyAdmin in the cPanel interface	
4. Select the database	
5. Choose the 'Import' tab from the above row	
6. Press 'Choose file' (image translation) and select the CSV file containing the media items with url	
7. Make sure the import starts at row zero since this contains our headers	
8. Check if the format is set to CSV, if not select CSV	
9. Define the separator you used for creating the CSV, in the case of Naturalis a point comma (;)	
10. Check the option that declares that our first line contains headers and thus the names of the fields	
11. Click 'Go' and wait till the program is finished (this can take a few minutes depending on the size of	

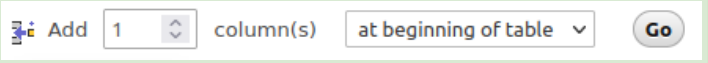
the dataset)	

The multimedia table is now created and added to the online database. Change the name of the table to a descriptive name. We prefer that you also mention your institution credentials or name in this name like: ‘umuMultimediaObjects’ (University of Utrecht multimedia objects). Do not use spaces or special characters, camelcase is encouraged.

1. Select the database and then the just created table to rename (will be named after your original datafile)	
2. Click on ‘Operations’	
3. Fill in the new name at ‘Rename table to’	
4. Press ‘Go’ to save the name	

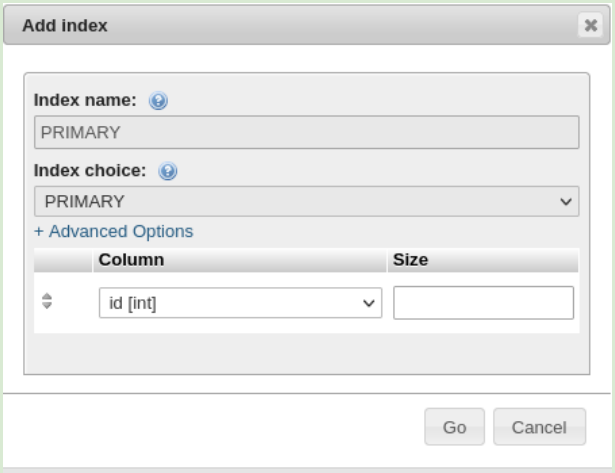
It is also important to add an identifier field to this table, as we need to be able to relate the multimedia items to the belonging occurrences/specimens. The identifier will have the auto increment behaviour so it starts at an id of ‘1’ and will count upwards when new records are inserted. By setting the identifier as an unique key, we also prevent it from being duplicated.

Adding an identifier field

1. In phpMyAdmin, select the ‘Structure’ tab from the top bar	
2. At a new column to the table by opting to add a single new field at the beginning of the table and click on ‘Go’	

3. Fill in a name for the field, most simple being: 'id'. Set the length to '6' and make sure to check the 'AI' checkbox. This will activate the auto increment behaviour. Activating this will trigger this pop-up. Press 'Go' to continue

Use **figure 15** as a reference for filing in the fields



4. Click on 'Save' to save the identifier field

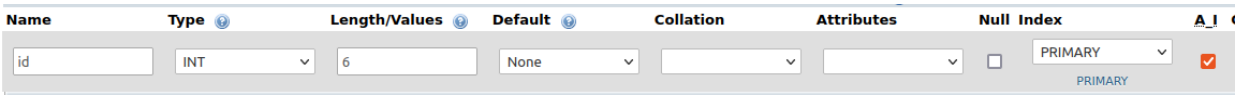
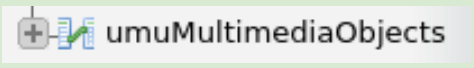


Figure 15: Identifier field of the multimedia table

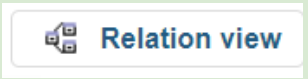
Now the last thing for the table in the database is to add a relation between our new multimedia table and the occurrences/specimens table. This is because the multimedia items are made from an occurrence/specimen and thus belong to them.

1. In phpMyAdmin, select the multimedia table

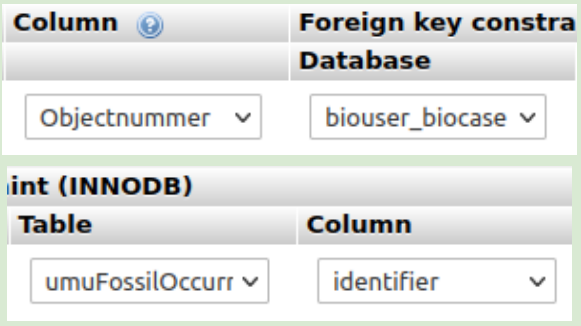


2. Click on 'Structure'

3. Now click on 'Relation view'



4. Take over the values as shown in **figure 16**. Note that the database and table fields could be different in your database due to alternate naming

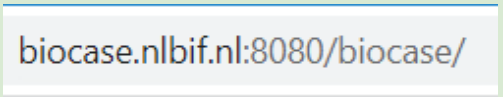
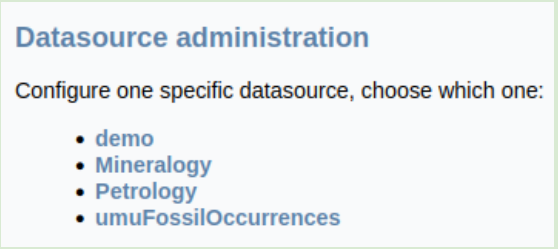
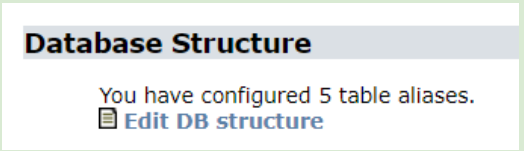
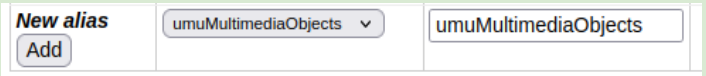
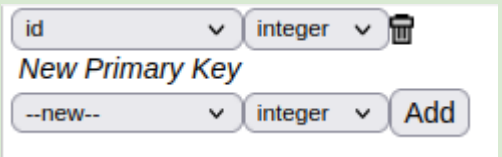


5. Click on ‘Save’	

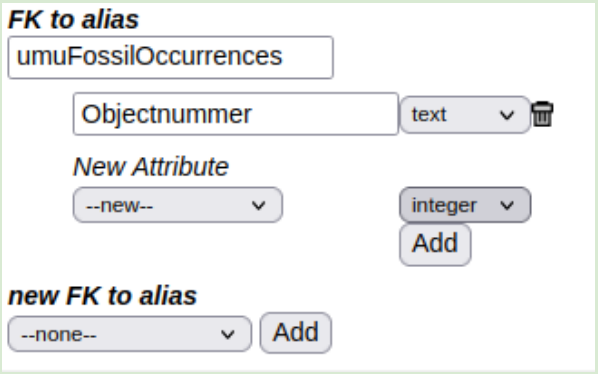
Figure 16: Relation between multimedia and occurrences/specimens

With the table created, we can move to the BioCASe Provider Software and configure it in the existing structure of the collection. Important to notice is that we are giving an example for a single collection. If you have multiple collections you could create multiple tables containing the multimedia items sorted per collection or create a general table for your institution containing all multimedia items.

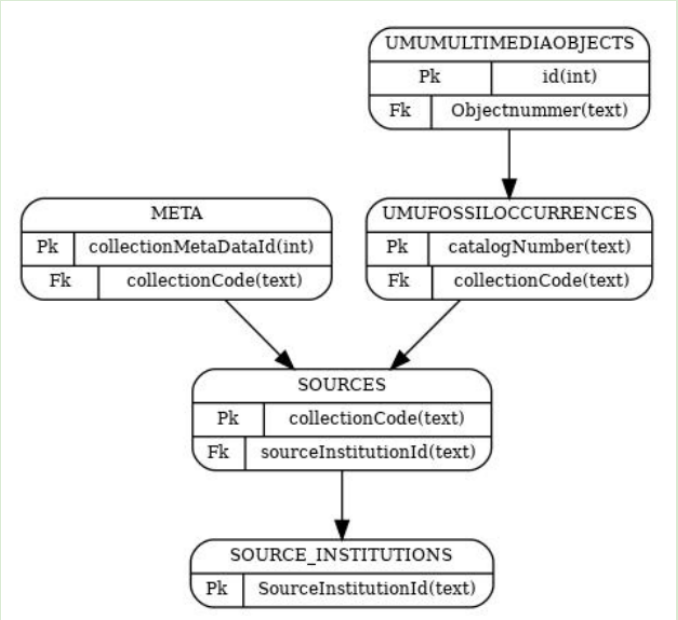
Configuring the multimedia table in BioCASe:

1. Open BioCASe at your domain, as for example: biocase.nlbif.nl:8080/biocase	
2. Click on the data source you added in the previous step, for example the ‘umuFossilOccurrences’ dataset	
3. Next click on ‘Edit DB structure’ (login if necessary)	
4. Add the multimedia table by selecting the name from the drop down menu and clicking ‘Add’	
5. Define the primary key (i.e. unique record Id) for each table. Use the example as an indicator. Mind the field types; e.g. the id of the multimedia table is a number, hence ‘integer’	

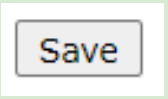
6. Define the foreign key, this represents the relation of the multimedia items to an occurrence/specimen. Now it becomes feasible why we use this approach with a separate table, because of the relation we can add multiple multimedia items to a single occurrence/specimen.



7. Check if the rendered image complies with the example figure. Names may differ based on your dataset, but types should be similar)



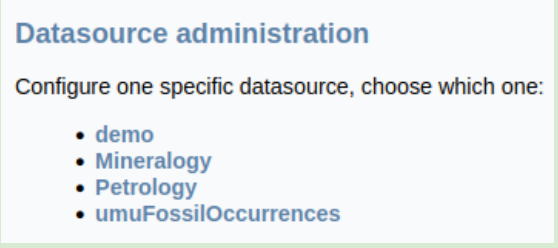
8. Click on 'Save' to save the table configuration

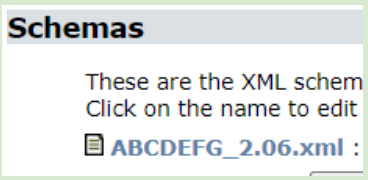
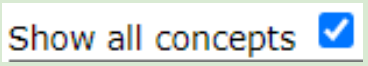
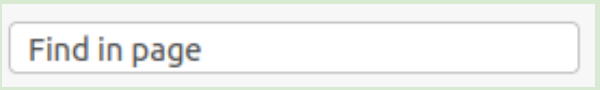
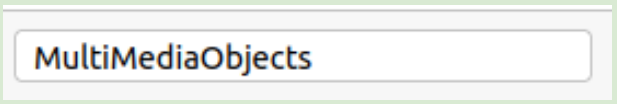
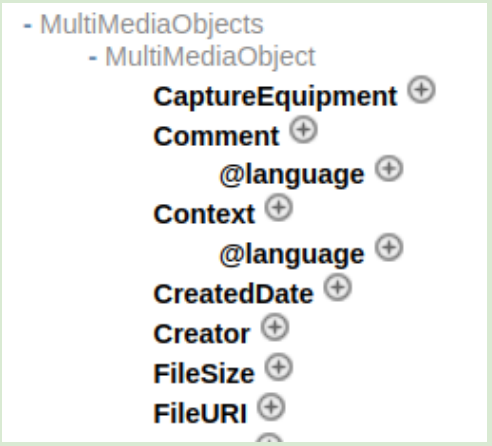
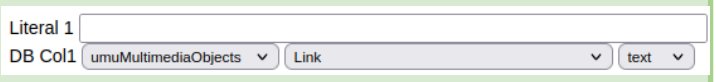
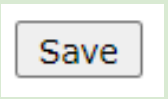


With the table configuration in BioCASe complete, it is now possible to map the multimedia items to the concepts of the ABCD-EFG standard. We only have to do this once, because if an occurrence/specimen has multiple multimedia items attached to it, they will automatically be sorted by the relation we defined.

Mapping the multimedia items in BioCASe

1. Click on your configured datasource



2. Navigate to the mapping overview page in BioCASe from the schemas section	
3. Check the checkbox that says 'show all concepts'. This will reload the page and show all the fields of the ABCD-EFG standard	
4. Press on CTRL and F on your keyboard to initiate your browsers search function	
5. Type and search for 'MultiMediaObjects'. This should immediately redirect you to the multimedia section of the mapping	
7. Press the plus button next to 'FileURI'	
8. Bind the value of the multimedia table to the field	
9. Click on 'Save' to save the mapping	

Besides mapping the 'FileURI' element, it is necessary to map two multimedia licence related items named 'Text' and 'Language'. Text should contain the value of the licence like for example: 'CC-0', language should hold the language of the licence text, in this case 'en' for English. See **figure 17** for further guidance.

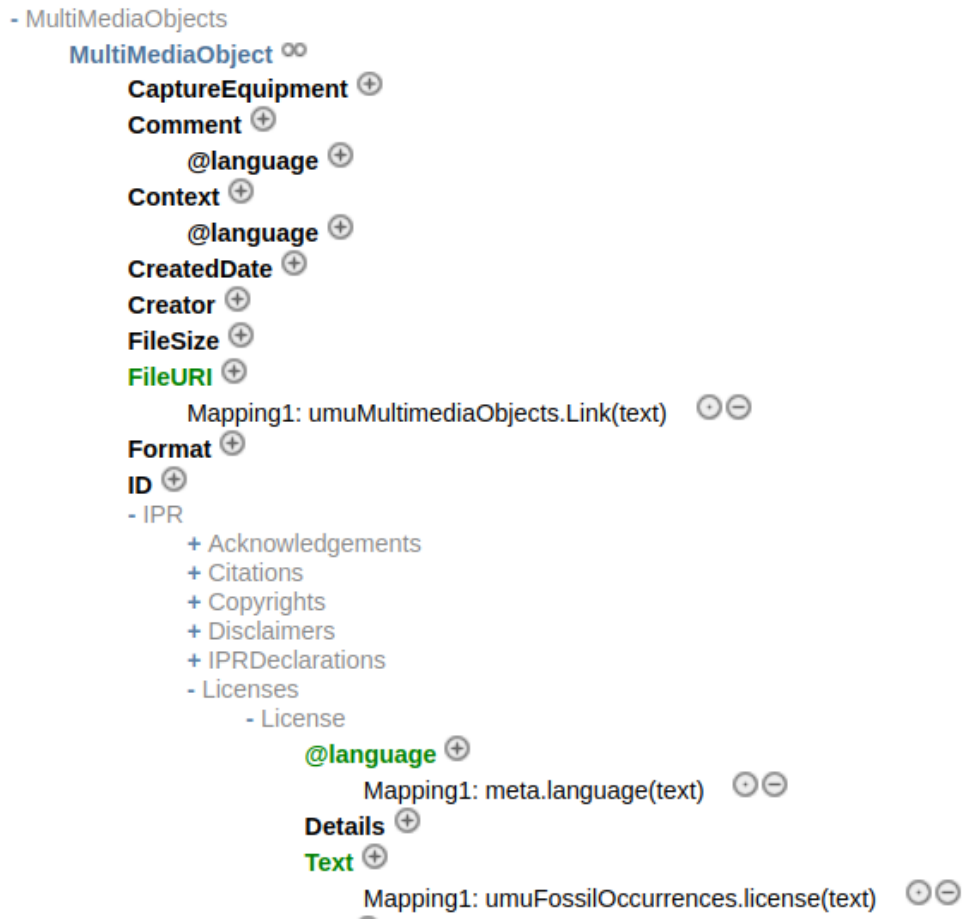


Figure 17: Required Multimedia fields

Congrats! By finishing the mapping you should have applied the multimedia part of ABCD-EFG to the endpoint's response. Go and test your mapping using the query editor (like on **page #**). If you receive a valid response, you can once again contact the people at GeoCASe to pull the data from our endpoint. This will refresh the representation of the dataset in the GeoCASe database and adds the multimedia items as well.

Terminology

API: An API, or sometimes referred to as an endpoint, is often a gateway to receiving an application's data or to execute certain methods for example adding, editing and removing records in the application's database. API methods can be accessed by exposing certain paths (endpoints) of a web domain. The BioCASe Provider Software also creates an endpoint for datasets that are defined using the ABCD-EFG standard. This would for the example of the Petrology dataset be: http://biocase.nlbif.nl:8080/biocase/pywrapper.cgi?dsa=Petrology.

(Docker) Client: The Docker client functions as the primary way to interact with Docker. Commands can be sent via the client to Dockerd, the daemon, to execute certain actions such as creating, updating or removing containers.

Command line: The command line is a developer's way of interacting (interfacing) with the host machine and source system. Most machines own a command line because it is one of the most basic programming methods.

(Docker) Container: When using Docker an application will be stored inside of a container. A container consists of an isolated environment that runs on the operating system of the host such as Linux or Windows. A container contains all dependencies of an application as defined in the image.

(Docker) Daemon: The daemon is the engine of Docker. It listens for commands from the client and on noticing, will execute these. A daemon is also capable of communicating with other daemons.

(Docker) Dockerfile: A Dockerfile represents the template for the construction of an image. During the process of building an image in a container, the Dockerfile also indicates which commands should be executed. For example to install certain dependencies. Docker only uses Dockerfiles that are identified as relevant for building the layers of the image. A layer represents a part of the application. Building a container can be done by using multiple layers which translates to Dockerfiles. The process of building layer after layer allows Docker to be a light and relatively fast program due to the distribution of available workspace of the web server.

Environment variables: Environmental variables are variables that can be passed to an application from out of the direct scope (the application's files). These variables are often defined based upon the kind of environment the application is installed in. Certain environments require different conditions. The environment variables can be set in a Dockerfile for example, instead of manually changing the internal variables of an application, which is not a best practice.

File hierarchy: File hierarchy refers to the local file layout of an application or host machine. It is basically the same as how your computer's storage is structured with folders like: documents, downloads and desktop. The file hierarchy in an application has an impact on the reach of certain files when they are nested in deeper folders.

Host machine: The host machine can be seen as the physical machine that runs a web server or application, but could also refer to a virtual machine that practically does the same thing. Eventually all host machines, physically or virtually, are part of a physical set-up often housed in a datacenter.

HTTP: HTTP, or Hypertext Transfer Protocol, is the protocol for communication between a webclient and a web server. Using this protocol the client, robotic or being used by a user, can request certain

operations from the web server. HTTPS is a secure extension that encrypts the requests and responses between client and web server.

(Docker) Images: In the context of Docker an image does not mean a picture, but rather a readable template with exact instructions on how to build a container. In most cases, an image is based on an existing application with some favourable changes. In addition, it is possible to create an image from scratch by using a Dockerfile. This can be done by creating an empty Dockerfile and filling it with instructions on how to build the image.

Makefile: A Makefile is kinda similar to a Docker file as in it can create an application and set-up an environment to run it in. It functions upon plain terminal commands that are executed in a certain order. Makefiles can be run with the program 'Make', which is an Unix based program and not a standard present piece of software on every host machine.

Metadata: Metadata is a specific group of data that tells something about the context of a dataset. The name, format and publishing date of a dataset are all examples of metadata properties.

PyWrapper Software: A PyWrapper is a commonly used term for a piece of Python software that is built around a database. Performing actions upon the database and exposing database data are core functionalities. The BioCASe Provider Software is a PyWrapper software whose purpose it is to expose collection data from the database to for example: GeoCASe.

(Docker) Registry: A Docker registry is where images are stored. The Docker Hub is a public registry where everyone can share and make use of images. It is also possible to make a private registry to save images.

REST API: An API is a web service based endpoint that can be used by applications to extract certain information. A REST API (Representational State Transfer) is a type of API that makes use of the 'RESTful' architecture. This basically represents an optimised way of building the inner structure.

Syntax: The syntax of a programming language defines how a programming language looks to the user. It is the barrier between code that is being written by the programmer and the machine that translates it to binary for execution. Syntax for the common language HTML exists for example out of different blocks of elements, each referring to a part of a web page like: header, title and body.

Terminal: A terminal is a common term for a window that contains the command line. In the WHM interface, for example, opening the terminal will give access to the command line.

XML: The Extensible Markup Language is standard of the World Wide Web Consortium for formerly handling data in a structured manner. An XML file can hold a whole dataset's field range and display data properties that are children, relevant to their parents (nested approach). XML is thus both readable for machines as humans.