

Project Title:	Baltic+ BalticAIMS
Document Title:	D3.1 Service Chain Verification Report SVR
Version:	1.1 Public version
Author(s) and affiliation(s):	Martin Böttcher, Carole Lebreton, Carsten Brockmann, Uwe Lange, BC Mikko Kervinen, Sampsa Koponen, Jenni Attila, Hanna Alasalmi, Vesa Keto, SYKE
Version history:	0.1 10.02.2022 Template 1.0 24.03.2022 Version submitted for SRR review 1.1 06.04.2022 Public version
Distribution:	ESA, Project team, public

Contents

Glossary.....	3
1 Introduction	4
1.1 Purpose and scope	4
1.2 Document overview.....	4
2 Overview	5
2.1 Showcases and datasets	5
2.2 Services and system elements.....	5
3 Adding data to BalticAIMS.....	7
3.1 How to add gridded data.....	7
3.2 How to add feature data.....	12
3.3 How to add files for download	18
3.4 Naming convention	20
4 Accessing data of the BalticAIMS services.....	22
4.1 OGC interfaces WMTS, WFS, WMS	22
4.2 TARKKA integration	24
4.3 QGIS configuration for xcube data and geodb data	27
4.4 xcube viewer.....	36
4.5 Jupyter Notebooks.....	38
5 References	41

Glossary

API	Application programming interface
B#	Band (number)
CHL	Chlorophyll a
CLC	Copernicus Land Cover
CLMS	Copernicus Land Monitoring Service
CRS	Coordinate reference system
EO	Earth Observation
FI	Finnish showcase area
GIS	Geographic Information System
GO	Gotland showcase area
HELCOM	Helsinki Commission
HR-OC	High resolution ocean color
MV	Mecklenburg-Vorpommern
OGC	Open Geospatial Consortium
RGB	Red Green Blue
ROI	Regions of interest
SPM	Suspended Particulate Matter
TUR	Turbidity
VM	Virtual Machine
WFS	Web Feature Service
WMS	Web Mapping Service
WMTS	Web Mapping Tile Service
WCS	Web Coverage Service

1 Introduction

1.1 Purpose and scope

This document is the service chain verification report for the BalticAIMS services. While the document shows how the services are verified it at the same time contains the instructions how to add data to the system and how to use the interfaces and data from the various (user) clients.

This document is complemented by the BalticAIMS service readiness report [SRR, D3.1] that in addition to the configuration status explains how the installation and configuration of the different services has been done. The two documents are the response to the three WP 2 documents Service portfolio definition [Portfolio], Data and platform provisioning plan [Platform], and Service delivery chain specification [ChainSpec].

1.2 Document overview

After this formal introduction

section 2 provides an overview of showcases and datasets, services, and system elements that implement them

section 3 explains how to add gridded data, feature data, and plain files to BalticAIMS, with reports on verification

section 4 explains how to access data of the different BalticAIMS OGC interfaces and interactive interfaces of TARKKA, QGIS, xcube viewer, and Jupyter notebooks, with reports on verification

2 Overview

This section provides an overview of showcases and datasets, services, and system elements that implement them.

2.1 Showcases and datasets

The 5 show cases are

- A: Provide EO based information to be used in user legacy systems for spatial planning
- B: Monitor the effects of nutrient flow from the drainage basin to the coastal waters
- C: Monitoring the impacts of coastal activities
- D: Combination of Coastal Zone mapping and CMEMS coastal water quality material
- E: Monitoring of temperature anomalies

Each show case is further elaborated in several user stories with concrete applications. Details are provided in D2.1.

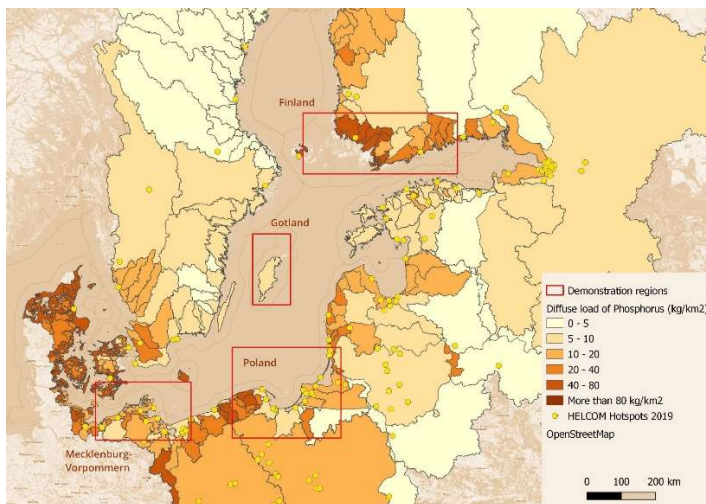


Figure 2-1: BalticAIMS showcase areas

There are 4 areas the show cases will be applied to in different combinations:

1. Archipelago Sea, Finland, and Helsinki
2. Gotland
3. Mecklenburg-Vorpommern
4. Poland

Datasets for the different showcases and areas are either gridded (time series of) image data, or tabular (series of) feature data. They are collected in BalticAIMS, sometimes processed or post-processed for BalticAIMS, and sometimes collected from other sources and made available in a harmonised analysis-ready way in BalticAIMS.

- Examples of gridded data are the high-resolution water quality datasets of the central Baltic Sea systematically generated by SYKE, or the Corine Land Cover classes provided at 4 time steps. Gridded data is made available in the xcube service.
- Examples of feature data are the N and P load from agriculture made available by HELCOM, or the coastal water body catchments provided by SMHI. Tabular feature data is made available in the geodb service.

2.2 Services and system elements

The BalticAIMS services used by the different show cases are:

- Showcase A makes available EO data for spatial planning using the user systems, mainly GIS. The GIS will read from BalticAIMS data interfaces, in particular WCS and WFS, to access data cubes and GeoDB. Simple static files are served from BalticAIMS geo file server as well.
- Showcase B provides data to analyse the effects of nutrient flow from drainage basins to coastal water. Main service used is TARKKA. The showcase may also use the OGC services provided by BalticAIMS.
- Showcase C provides data to analyse the impact of coastal activities. TARKKA demonstrates the service. GIS interfaces may be used as well.
- Showcase D maps coastal zones. It uses the BalticAIMS data service to include selected layers into the analysis.
- Showcase E makes available EO data to analyse temperature anomalies. It mainly uses TARKKA to demonstrate the service.

BalticAIMS data services are based on TARKKA, xcube, and geodb. The involved elements are shown in Figure 2-2.

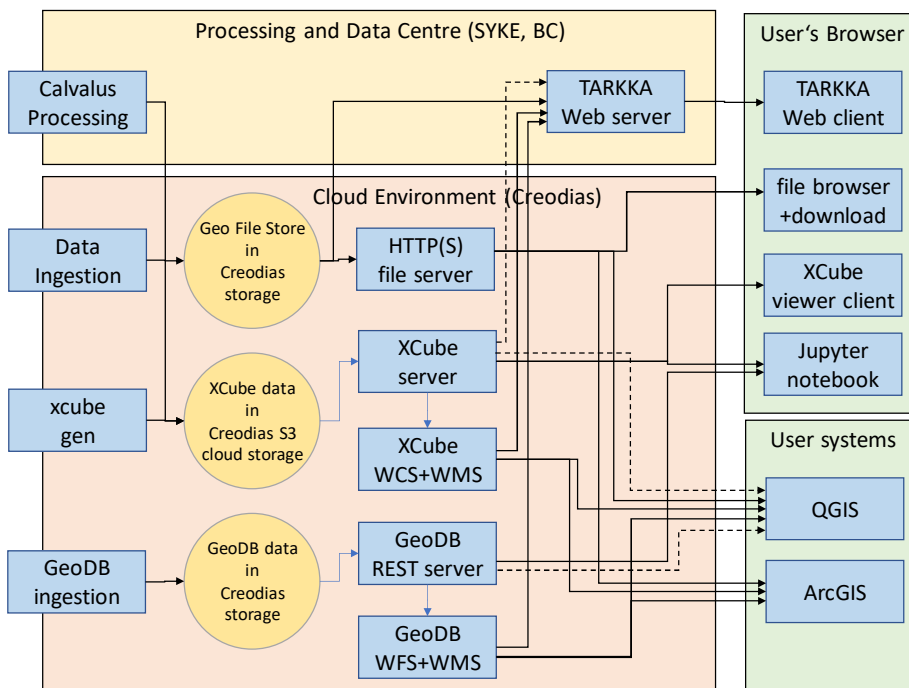


Figure 2-2: BalticAIMS system elements

- Raster time series data is either ingested and prepared with xcube gen, or it is processed by BalticAIMS and stored in XCube zarr format. This data is accessed via XCube server and viewer or via OGC WMS or WCS either by TARKKA or by user GIS applications.
- Feature data is inserted into GeoDB and served via OGC WFS either to TARKKA or to user GIS applications.
- Unstructured or simple file data can also be stored in a structured geo file store and served via HTTP for download or direct access by user GIS applications.

3 Adding data to BalticAIMS

This section explains how to add gridded data, feature data, and plain files to BalticAIMS, with reports on verification.

3.1 How to add gridded data

The general procedure to add a dataset available as a time series of single files to the BalticAIMS data cube is the following:

- Upload complete dataset to BalticAIMS object storage `s3://balticaims/inputs/<datasetid>/` on Creodias
- Adapt naming and metadata in `xcube-gen` configuration file with respect to dataset
- For each file in object storage
 - Retrieve daily observation to local storage
 - `s3cmd -get s3://balticaims/inputs/<datasetid>/<daily_geotiff_file>`
 - Append dataset to cube with
 - `xcube --traceback gen -c <xcube-gen-config-file> <daily_geotiff_file>`
- Finally optimize cube and create overviews
 - `xcube optimize -IC <cubename>.zarr`
 - `xcube prune <cubename>.zarr`
 - `xcube level -t 128 <cubename>.zarr`
- Transfer optimized cube and overviews to object storage
 - `s3cmd put -r <cubename>.zarr s3://balticaims/cubes/`
 - `s3cmd put -r <cubename>.levels s3://balticaims/cubes/`

The following subsections explain scripts for cube generation and their configuration in the `xcube` server.

3.1.1 Scripts for cube generation

There are several scripts for generating data cubes already. They can be used as templates for new data cubes. One of the scripts is explained here.

```
#!/bin/bash

#/hroc/calvalus/projects/hroc/l3-compo-v01/3bal/2021-07/23/20210723_P1D_CMEMS_HROC_L3-rgb_BAL__100m-v01.2.nc
#/hroc/calvalus/projects/hroc/l3-compo-v01/3bal/2021-07/23/20210723_P1D_CMEMS_HROC_L3-tur-spm-chl_BAL__100m-v01.2.nc

mkdir -p /balticaims/inputs/hroc-l3-compo
for y in 2020 2021; do
  for m in {01..12}; do
    s3cmd -c ~/.dot-s3cfg-hrocpartners get --skip-existing \
      -r s3://hroc/calvalus/projects/hroc/l3-compo-v01/3bal/$y-$m \
        /balticaims/inputs/hroc-l3-compo/
    for d in {01..31}; do
      #if [[ $y$m$d < 20200816 || $y$m$d > 20200831 ]]; then continue; fi
      ta=/balticaims/inputs/hroc-l3-compo/$y-$m/$d/$y$m${d}_P1D_CMEMS_HROC_L3-tur-spm-
chl_BAL__100m-v01*.nc
      ra=/balticaims/inputs/hroc-l3-compo/$y-$m/$d/$y$m${d}_P1D_CMEMS_HROC_L3-rgb_BAL__100m-
v01*.nc
      mosaic=/balticaims/inputs/hroc-l3-compo/$y$m${d}_P1D_CMEMS_HROC_L4-
turspmchlrgb_BAL__100m-v01.2.nc
      if [ -e $ta -a -e $ra ]; then
        ~/xcube-gen-scripts/mosaiccompowithrgb.py $ta $ra $mosaic
      else
        echo $ta
        echo $ra
        echo "no input for $y-$m-$d"
        continue
      fi
    done
  done
done
```

```
fi
echo "cubify $mosaic"
xcube --traceback gen -c ~/xcube-gen-configs/ba-hroc-l3d-config.yml --sort $mosaic
rm -f $ta $ra $mosaic
done
done; done
```

Figure 3-1: Script to generate a cube for 2 years of daily HR-OC data for the complete Baltic Sea

- There are three loops for year, month, and day
- Two composites are retrieved from HR-OC cloud storage on Creodias, one with TUR, SPM, CHL, and one with the RGB bands. They are combined into a single file if both are available.
- A call to xcube gen adds them to the cube. The call uses a configuration shown in Figure 3-2.

```
output_size: [12594,14040]
output_region: [9,53,31,66]

output_writer_params:
  chunksizes:
    lon: 2160
    lat: 2160
    time: 1

output_path: "/balticaims/cube/ba-hroc-l3d.zarr"

output_variables:
- TUR
- SPM
- CHL
- B4
- B3
- B2
```

Figure 3-2: xcube gen configuration for the HR-OC cube for the complete Baltic Sea area

The configuration

- specifies region, shape, and chunk sizes
- the output path of the cube as a preliminary location on the VM
- the variables to be added to the cube

In this case the default NetCDF accessor can be used because the input files already come with the expected metadata to extract the time information.

More examples of such workflow can be found in

https://github.com/sykefi/BalticAIMS/blob/master/xcube-gen-scripts/make_syke_cubes.sh

and xcube gen configuration in

https://github.com/sykefi/BalticAIMS/blob/master/xcube-gen-configs/fi-syke-EO_HR_WQ_S2_TURB-1d-config.yml

This is an input dataset available in GeoTIFF. For GeoTIFF an accessor is required to allow xcube gen to extract the (meta)data.

3.1.2 Extension of xcube data accessors

The xcube library was extended by adding a plugin *xcube_gen_balticaims* with input processor type *balticaims-geotiff* to support addition of daily geotiff products in different coordinate systems to datacubes. Currently plugin is customized to support Corine land cover and SYKE eo-products and can be used to import products of type 'SST', 'turb', 'CLC', 'ALGAE' and 'cdom'. This means that the plugin can be used to import datasets where any of these tags can be found in the file name. For additional types the accessor can be extended to support more types of inputs.

The implementation is listed in the following figures with explanations after each part. The implementation is stored in GitHub and installed in the Conda environment on the balticcube VM.

```
from xcube.core.dsio import DatasetIO
import xcube.core.timecoord as timecoord

class BalticAIMSGeoTiffInputProcessor(DefaultInputProcessor):
    """
    Input processor for BalticAIMS GeoTiff inputs.
    """

    def __init__(self, **parameters):
        super(XYInputProcessor, self).__init__('balticaims-geotiff', **parameters)

    @property
    def default_parameters(self) -> Dict[str, Any]:
        default_parameters = super().default_parameters
        default_parameters.update(input_reader='geotiff')
        return default_parameters

    def get_time_range(self, dataset: xr.Dataset) -> Tuple[float, float]:
        # TODO parse from file name
        return DefaultInputProcessor().get_time_range(dataset)
```

Figure 3-3: Accessor implementation (part 1)

- This BalticAIMSGeoTiffInputProcessor specifies balticaims-geotiff as input format parameter.
- The implementation of BalticAIMSGeoTiffInputProcessor parses the time information from the file name as it is not contained in the metadata. The get_time_range method recognises several time formats in file names and is sufficient for the file types encountered so far.

```
class GeoTiffDatasetIO(DatasetIO):
    """
    A dataset I/O that reads from / writes to GeoTIFF files.
    """

    def __init__(self):
        super().__init__('geotiff')

    def fitness(self, path: str, path_type: str = None) -> float:
        ext = _get_ext(path)
        ext_value = ext in {'.tif', '.tiff'}
        type_value = 0.0
        if path_type == "file":
            type_value = 1.0
        elif path_type is None:
            type_value = 0.5
        else:
            ext_value = 0.0
        return (3 * ext_value + type_value) / 4
```

Figure 3-4: Accessor implementation (part 2)

- The first two methods of GeoTiffDatasetIO are rather formal, support to read .tif files

```
def read(self, input_path: str, **kwargs) -> xr.Dataset:
    filename = input_path[input_path.rfind('/')+1:]
    start, stop = timecoord.get_timestamps_from_string(filename)
```

```
varname = self._get_varname_from(filename)
if not start:
    raise Exception('cannot find time in ' + filename)
if not stop:
    stop = start
time_var = xr.Variable('time', [start])
# da = xr.open_rasterio(input_path, chunks={'y':512,'x':512}, **kwargs)
with rioxr.set_options(export_grid_mapping=True):
    if varname == 'CLC':
        da = rio.open(input_path)
        vrt = rio.vrt.WarpedVRT(da, crs="EPSG:4326")
        da = rioxr.open_rasterio(vrt, chunks={'y':1024,'x':1024}, mask_and_scale=True,
lock=False)
    else:
        da = rioxr.open_rasterio(input_path, chunks={'y':1024,'x':1024},
mask_and_scale=True, lock=False, **kwargs)
        if da.spatial_ref.attrs['crs_wkt'] != 'GEOGCS["WGS
84",DATUM["WGS_1984",SPHEROID["WGS
84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwic
h",0],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]],AXIS["Latitude",NORTH],AXIS["
Longitude",EAST],AUTHORITY["EPSG","4326"]]'':
            da = da.rio.reproject('EPSG:4326')
        das = xr.concat([da], dim=time_var)
        ds = das.to_dataset('band')
        ds = ds.rename({'y':'lat','x':'lon',1:varname,'spatial_ref':'crs'})
        ds = ds.chunk(chunks=1024)
        ds['time'].attrs['standard_name'] = 'time'
        ds['lat'].attrs['standard_name'] = 'latitude'
        ds['lon'].attrs['standard_name'] = 'longitude'
        ds[varname].attrs['grid_mapping'] = 'crs'
        ds.attrs['filename'] = filename
        ds.attrs['time_coverage_start'] = str(start)
        ds.attrs['time_coverage_stop'] = str(stop)
return ds
```

Figure 3-5: Accessor implementation (part 3)

- The read method is the core of the implementation. It opens the input file with rioxarray and reprojects the content to the target CRS, which currently is EPSG:4326. It uses a virtually warped intermediate for CLC inputs, and `rio_reproject` for other inputs. It adds the time dimension and names the spatial dimensions `lat` and `lon`.

```
def _get_varname_from(self, filename: str) -> str:
    candidates = [ 'SST', 'turb', 'CLC', 'ALGAE', 'cdom' ]
    for c in candidates:
        if c in filename:
            return c
    return 'band1'

def write(self, dataset: xr.Dataset, output_path: str, **kwargs):
    raise NotImplementedError

def append(self, dataset: xr.Dataset, output_path: str, **kwargs):
    raise NotImplementedError

def update(self, output_path: str, global_attrs: Dict[str, Any] = None, **kwargs):
    raise NotImplementedError

def _get_ext(path: str) -> Optional[str]:
    _, ext = os.path.splitext(path)
    return ext.lower()
```

Figure 3-6: Accessor implementation (part 4)

- There are a few utility functions. One of them is extracting the variable name from filename. This is required because geotiff does not have the concept of variables and names of them. For extensions, we recommend to use the same approach to use as variable name a substring of the filename. If this is not desired, the mapping can be added to this implementation.

The accessor is installed by

```
eval "$(/balticaims/software/miniconda3-ba/bin/conda shell.bash hook)"
conda activate xcube
cd /balticaims/software/xcube-gen-balticaims
python setup.py install
```

3.1.3 Configuration of xcube server

The xcube server configuration is located at

~/ba-server-config-v1.yml

which is a symlink to a file maintained under GitHub.

For each new data cube there is an entry in this file in Datasets. If the cube requires a new style then there is also an entry for the style. The style may refer to colour tables.

```
Datasets:
- Identifier: FI_SYKE_TUR_1D
  Title: Finland SYKE S2 turbidity 1D
  FileSystem: obs
  Path: "balticaims/cubes/fi-syke-EO_HR_WQ_S2_TURB-1d.zarr"
  Endpoint: "https://cf2.cloudferro.com:8080"
  Region: "RegionOne"
  Style: syke_legends
...
Styles:
- Identifier: syke_legends
  ColorMappings:
    ALGAE:
      ColorFile: "ba-syke-algae-index-legend.cpd"
      ValueRange: [1., 4.]
    SST:
      ColorFile: "ba-syke-sst-legend.cpd"
      ValueRange: [0.0, 25.0]
    turb:
      ColorFile: "ba-syke-turbidity-legend.cpd"
      ValueRange: [0.0, 60.0]
    dummy:
      ColorFile: "ba-syke-petrol-legend.cpd"
      ValueRange: [0.0, 60.0]
```

Figure 3-7: xcube server configuration file ba-server-config-v1.yml with datasets and styles

- The dataset has a unique identifier, a title to be displayed in the viewer, the access information for the S3 bucket (unless served from local Posix file system), and the reference to the style to be used
- The style contains colour mappings for different variables. In this example they refer to a cpd file with a mapping from values to colours. The cpd files are maintained on GitHub and linked in ~/ .

3.1.4 Verification for added grid datasets

Verification is mainly done by accessing the data for example with the xcube viewer. This is described in section 4.4. An initial test can be done by accessing the xcube server capabilities at

<https://xcube.balticaims.eu/wmts/1.0.0/WMTSCapabilities.xml>

Figure 3-8 shows parts of the result with a layer available in xcube server after adding the cube.



```
<Contents>
  > <TileMatrixSet>
    <Layer>
      <ows:Identifier> FI_SYKE_TUR_ID.turb </ows:Identifier>
      <ows:Title> FI_SYKE_TUR_ID/Turbidity </ows:Title>
      <ows:Abstract />
      > <ows:WGS84BoundingBox>
      > <Style isDefault="true">
        <Format> image/png </Format>
      > <TileMatrixSetLink>
        <ResourceURL format="image/png" resourceType="tile" template="http://xcube.balticaims.eu/wmts/1.0.0/tile/FI_SYKE_TUR_ID/turb/{TileMatrix}/{TileRow}/{TileCol}.png" />
      > <Dimension>
    </Layer>
  > <TileMatrixSet>
    <Layer>
      <ows:Identifier> FI_HROC_L4D_1M.B2 </ows:Identifier>
      <ows:Title> FI_HROC_L4D_1M/Top of atmosphere reflectance at 492nm from Sentinel-2 MSI </ows:Title>
      <ows:Abstract />
      > <ows:WGS84BoundingBox>
      > <Style isDefault="true">
        <Format> image/png </Format>
      > <TileMatrixSetLink>
        <ResourceURL format="image/png" resourceType="tile" template="http://xcube.balticaims.eu/wmts/1.0.0/tile/FI_HROC_L4D_1M/B2/{TileMatrix}/{TileRow}/{TileCol}.png" />
      > <Dimension>
```

Figure 3-8: WMTS GetCapabilities response with layers provided by xcube server

3.2 How to add feature data

Feature datasets to be inserted into geodb are often provided as either shape files or csv files. For insertion, the data is read from the source file into a geopandas GeoDataFrame object. This is inserted into a newly created geodb collection.

The account for adding feature collections for BalticAIMS is baadmin. This is different from the account(s) to read and use the data to avoid that users inadvertently delete shared data.

Files to be inserted can be temporarily placed in a subfolder of the “inputs” directory of the balticcube VM on Creodias. From there, the data can be accessed by a notebook that inserts the data. After that, the files can be moved to cloud storage or deleted.

3.2.1 Notebooks for geodb insertion

There are several examples of notebooks that demonstrate how to insert feature data into geodb. To insert a new dataset copy, rename, and update one of the insert_....ipynb notebooks.

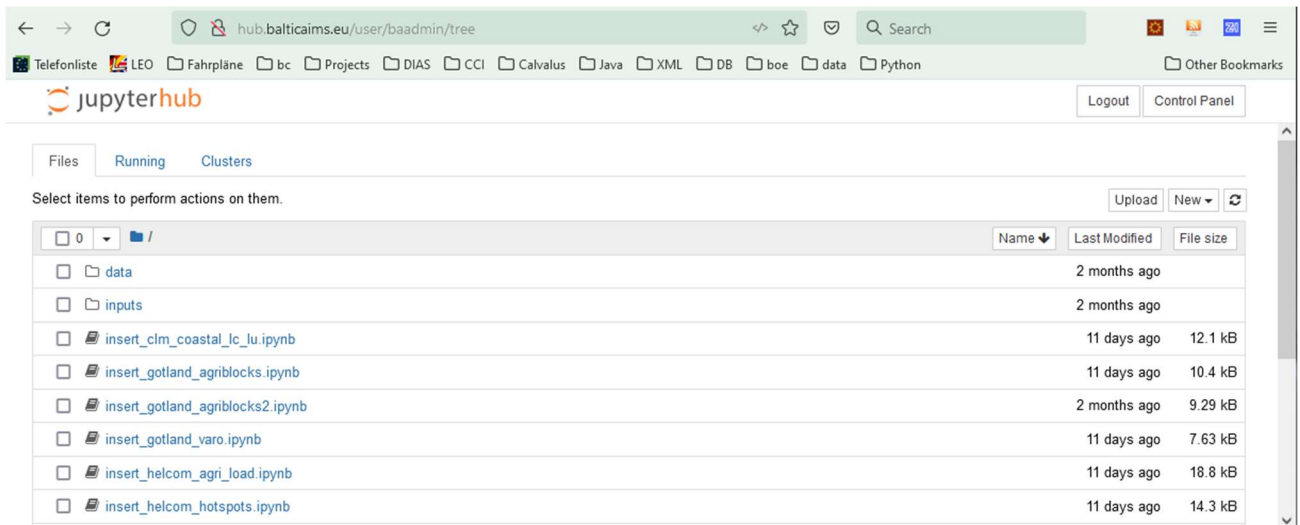


Figure 3-9: Existing notebooks that insert feature data into geodb, to be used as templates

The following two figures show one of these notebooks. They all have a similar structure.

Ingest HELCOM agricultural load

```

In [2]: import geopandas
import shapely
from xcube_geodb.core.geodb import GeoDBClient

In [3]: geodb = GeoDBClient()
geodb

Out[3]: <xcube_geodb.core.geodb.GeoDBClient at 0x7f816d492110>

In [4]: if geodb.collection_exists("helcom_plc_agricultural_load_nitrogen", None):
geodb.revoke_access_from_collection("helcom_plc_agricultural_load_nitrogen", "bauser")
geodb.drop_collection("helcom_plc_agricultural_load_nitrogen")
if geodb.collection_exists("helcom_plc_agricultural_load_phosphorus", None):
geodb.revoke_access_from_collection("helcom_plc_agricultural_load_phosphorus", "bauser")
geodb.drop_collection("helcom_plc_agricultural_load_phosphorus")

In [5]: gdf_n = geopandas.read_file('/balticaims/inputs/helcom-agri-load/nitrogen/diffuse_loads.shp')
gdf_n.iloc[:4]

Out[5]:
   SourceCode  NAME SubBasin COUNTRY MONITORING N_diffuse_ N_back_km N_agri_km Shape_Leng Shape_Area geometry
0  SCPL00026  PREGOLYA  BAP      PL          M          0.000      0.000      0.0  1.038793e+06  7.521853e+09  POLYGON Z ((5147587.467
3533023.865 0.000, 514...
1  SCRU00052  Nemunas  BAP      RUS          M          0.000      0.000      0.0  5.536196e+05  1.495622e+09  POLYGON Z ((5137264.907
3563800.000 0.000, 513...
2  SCRU00095  BAPRULAND  BAP      RUS          U          317.482    187.832      0.0  1.239330e+06  5.429763e+09  MULTIPOLYGON Z
(((4956900.000
3547100.000 0.00...
3  SCRU00096  GUFRLAND  GUF      RUS          U          336.222    284.739      0.0  2.469696e+06  9.982831e+09  MULTIPOLYGON Z
(((5341400.000
4196000.000 0.00...

In [6]: collections = {
    "helcom_plc_agricultural_load_nitrogen":
    {
        "crs": 3035,
        "properties":
        {
            "SourceCode": "varchar",
            "NAME": "varchar",
            "SubBasin": "varchar",
            "COUNTRY": "varchar",
            "MONITORING": "varchar",
            "N_diffuse_": "float",
            "N_back_km": "float",
            "N_agri_km": "float",
            "Shape_Leng": "float",
            "Shape_area": "float"
        }
    }
}
geodb.create_collections(collections, clear=True)
    
```

Figure 3-10: Example notebook to insert agricultural nitrogen load data

- The first two code blocks 2 and 3 declare software to be used and connect to the geodb with hidden connection parameters specific for the account “baadmin” logged in into Jupyterhub.
- Block 4 clears existing entries from the collections to be populated, in this case for nitrogen and phosphorus.
- Block 5 reads the shapefile and lists the first entries. Each entry has a geometry and several other columns.
- Block 6 creates a schema for the columns provided in the output of the previous block. This schema needs to be adapted to the data format to be ingested.
- The code continues in the following figure.

```
In [7]: # convert to 2d
gdf_n.set_geometry([shapely.wkb.loads(shapely.wkb.dumps(geom, output_dimension=2)) for geom in gdf_n['geometry']], inplace=True)
gdf_n.iloc[0]

Out[7]: SourceCode          SCPL00026
NAME          PREGOLYA
SubBasin      BAP
COUNTRY       PL
MONITORING    M
N_diffuse_    0.0
N_back_km     0.0
N_agri_km     0.0
Shape_Leng    1038792.91891
Shape_Area    7521852722.54
geometry      POLYGON ((5147587.466700001 3533023.865, 51475...
Name: 0, dtype: object

In [8]: geodb.insert_into_collection('helcom_plc_agricultural_load_nitrogen', gdf_n)
geodb.grant_access_to_collection("helcom_plc_agricultural_load_nitrogen", "bauser")

Processing rows from 0 to 310

Out[8]: <xcube_geodb.core.message.Message at 0x7f816d493cd0>
```

Figure 3-11: Example notebook to insert agricultural nitrogen load data (continued)

- This dataset contains 3-D coordinates. They are converted to 2-D by block 7.
- Block 8 inserts the GeoDataFrame with all its entries into the collection created in block 6. It also grants access to the “read-only” user.

This notebook in fact continues to insert the phosphorus dataset as well. This is not shown here.

To publish a feature collection in Geoserver, another step is added in the notebook. Figure 3-12 shows this for three CLMS collections.

```

jupyterhub insert_clm_coastal_lc_lu Last Checkpoint: 03/10/2022 (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted | geodb
Memory: 161.3 MB

Publishing the three collections as WFS

In [3]: import os
import dotenv
from geo.Geoserver import Geoserver

In [16]: def publish(collection, database):
    dotenv.load_dotenv(dotenv.find_dotenv(".env"))
    url = os.getenv("GEOSERVER_URL")
    user = os.getenv("GEOSERVER_USER")
    password = os.getenv("GEOSERVER_PASSWORD")
    geo = Geoserver(url, username=user, password=password)
    #print(geo.get_workspace(workspace=database))
    #print(geo.get_featurestore(store_name=database, workspace=database))
    response = geo.publish_featurestore(workspace=database,
                                        store_name=database,
                                        pg_table=database + '_' + collection)

    if response is not None:
        raise Exception(400, response)
    print(geo.get_layer(layer_name=database + '_' + collection, workspace=database)["layer"]["name"])

In [13]: publish('clm_coastal_lc_lu_2012', 'baadmin')
baadmin_clm_coastal_lc_lu_2012

In [14]: publish('clm_coastal_lc_lu_2018', 'baadmin')
baadmin_clm_coastal_lc_lu_2018

In [15]: publish("clm_coastal_lc_lu_changes_2012_2018", "baadmin")
baadmin_clm_coastal_lc_lu_changes_2012_2018
    
```

Figure 3-12: Publishing feature collections at Geoserver using the Geoserver API

- The publish method gets the collection name and the user name and inserts it into Geoserver.

3.2.2 Verification for added feature collections

Insertion into geodb is verified with notebooks as well. These notebooks are provided with the account bauser (password ...).

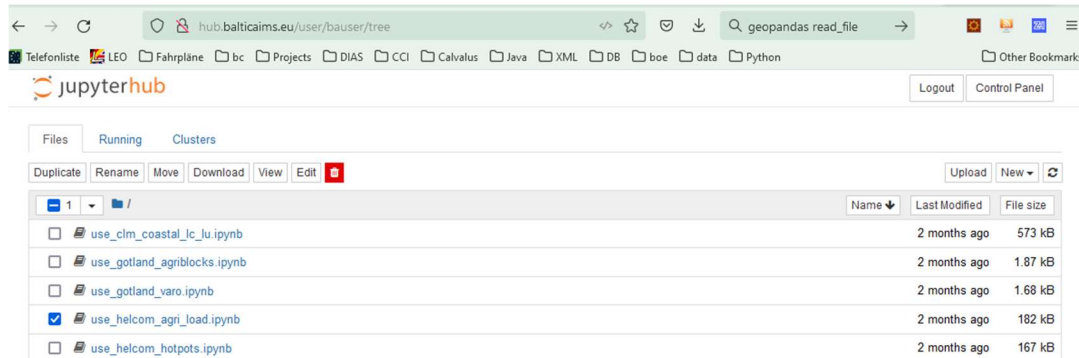


Figure 3-13: Existing notebooks to verify and use data from geodb, to be used as templates for new datasets

The notebook for verification finds the collection in geodb, loads the data, lists some entries, and plots a map of some parameter of the data.

Use HELCOM agricultural load

```
In [6]: import geopandas
        from xcube_geodb.core.geodb import GeoDBClient
        geodb = GeoDBClient()
        geodb.get_my_collections()

Out[6]:
```

	owner	database	collection
0	baadmin	baadmin	helcom_hotspots_2019
1	baadmin	baadmin	helcom_plc_agricultural_load_nitrogen
2	baadmin	baadmin	helcom_plc_agricultural_load_phosphorus

Figure 3-14: Example notebook to verify insertion of agricultural nitrogen into geodb (part 1)

- The first block (6) connects to geodb server with the hidden credentials of the bauser account logged in in Jupyterhub. Among them is helcom_plc_agricultural_load_nitrogen.


```
In [9]: gdf = geodb.get_collection('helcom_plc_agricultural_load_nitrogen', database='baadmin')
gdf.iloc[:2] # the first lines of the table

Out[9]:
```

	id	created_at	modified_at	geometry	sourcecode	name	subbasin	country	monitoring	n_diffuse_	n_back_km	n_agri_kr
0	1	2022-01-20T18:27:19.514571+00:00	None	POLYGON ((5147587.467 3533023.865, 5147574.325...	SCPL00026	PREGOLYA	BAP	PL	M	0.0	0.0	0.0
1	2	2022-01-20T18:27:19.514571+00:00	None	POLYGON ((5137264.907 3563800.000, 5137200.000...	SCRU00052	Nemunus	BAP	RUS	M	0.0	0.0	0.0

```
In [8]: gdf.plot(column='diffuse_km', cmap='tab20', figsize=(10,15))

Out[8]: <AxesSubplot:>
```

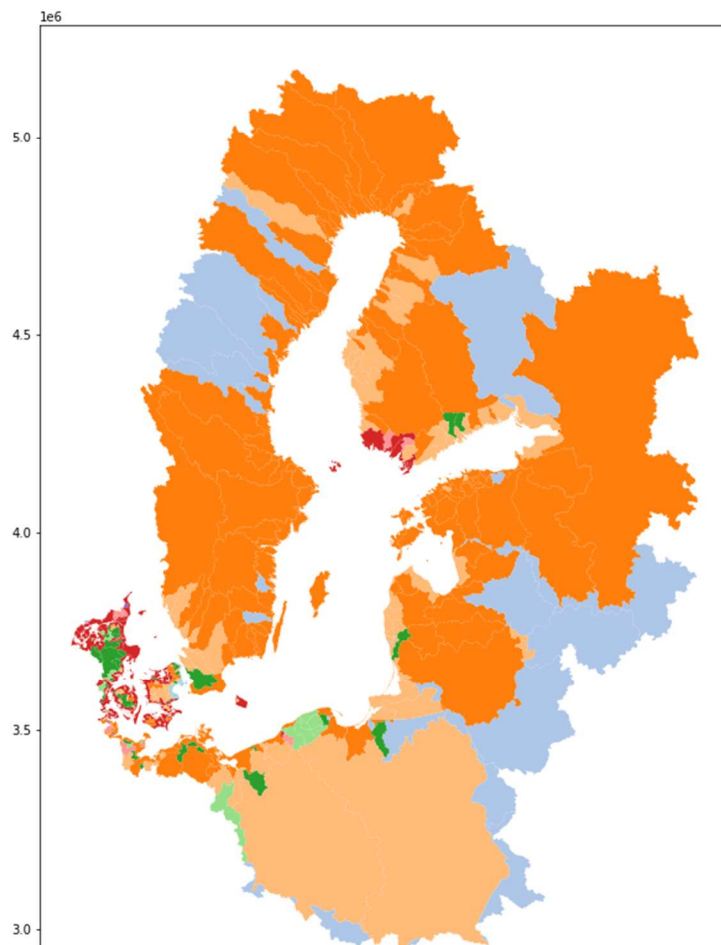


Figure 3-15: Example notebook to verify insertion of agricultural nitrogen into geodb (continued)

- The second block (9) lists a few entries of the collection from geodb.
- The third block (8) plots one of the parameters as a map.

This successfully verifies that the data is available and can be accessed from geodb. The feature datasets inserted so far are shown in Figure 3-16.

	owner	database	collection
0	baadmin	baadmin	clm_coastal_lc_lu_2012
1	baadmin	baadmin	clm_coastal_lc_lu_2018
2	baadmin	baadmin	gotland_agriblocks_2021
3	baadmin	baadmin	gotland_varo_2016
4	baadmin	baadmin	helcom_hotspots_2019
5	baadmin	baadmin	helcom_plc_agricultural_load_nitrogen
6	baadmin	baadmin	helcom_plc_agricultural_load_phosphorus

Figure 3-16: List of feature datasets available from BalticAIMS geodb (March 2022)

The access to WFS for published datasets is verified with the GetCapabilities URL of the WFS
<http://geoserver.balticaims.eu/geoserver/wfs?request=GetCapabilities>

It lists the published collections:

```

<ows:Constraint name="ImplementsSpatialJoins">
<ows:Constraint name="ImplementsTemporalJoins">
<ows:Constraint name="ImplementsFeatureVersioning">
<ows:Constraint name="ManageStoredQueries">
<ows:Constraint name="PagingIsTransactionSafe">
<ows:Constraint name="QueryExpressions">
</ows:OperationsMetadata>
<FeatureTypeList>
  <FeatureType xmlns:baadmin="https://geodb.balticaims.eu/baadmin">
    <Name> baadmin:baadmin_clm_coastal_lc_lu_2012 </Name>
    <Title> baadmin_clm_coastal_lc_lu_2012 </Title>
    <Abstract />
    <ows:Keywords>
    <DefaultCRS> urn:ogc:def:crs:EPSG::3035 </DefaultCRS>
    <ows:WGS84BoundingBox>
    </FeatureType>
  <FeatureType xmlns:baadmin="https://geodb.balticaims.eu/baadmin">
  
```

Figure 3-17: Result of GetCapabilities of WFS shows published collections

3.3 How to add files for download

Most data we use for the showcases are either gridded data or tabular feature data. These data shall be served by xcube and geodb to make them available only with suitable protocols for tools integration. In a few cases though users may prefer to download data, or data may not fit into the grid or feature schema. Such files can be offered on the Geo File Server.

3.3.1 Location for files to be served

Files are added to the Geo File Server by creating a proper directory structure and by copying them into that directory using suitable names for the files, preferably their original names. Using Corine Land Cover gridded data as example, the procedure is as follows:

- copy the CLC files to the balticcube VM
- login to the balticcube VM
- create directory under /balticaims/data/

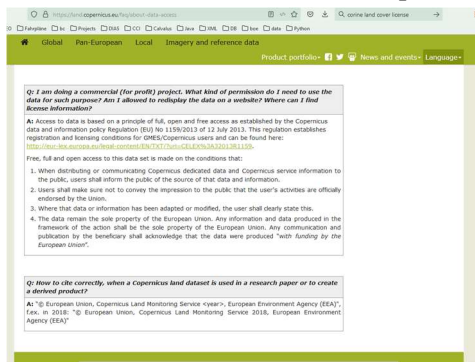
- move files into this directory

```
scp *CLC*tif eouser@balticcube:  
ssh eouser@balticcube  
mkdir -p /balticaims/data/rasterdata/land/CorineLandCover  
mv *CLC*tif /balticaims/data/rasterdata/land/CorineLandCover
```

3.3.2 Location for data licenses and references

All datasets served by one of the BalticAIMS services xcube, geodb, and geo file server shall have a copy of their license or terms of use available on the geo file server in a licenses sub-directory. The license files shall be prefixed with the dataset name. To provide the license file the procedure is:

- download the license text from the data provider



- check that the license allows re-distribution of the data
- upload the license file to the balticcube VM
- move it into the licenses directory, prefix the file name with the dataset and the word License

```
scp CELEX_32013R1159_EN_TXT.pdf eouser@balticcube:  
ssh eouser@balticcube  
mv CELEX_32013R1159_EN_TXT.pdf /balticaims/data/licenses/CorineLandCover-License-  
CELEX_32013R1159_EN_TXT.pdf
```

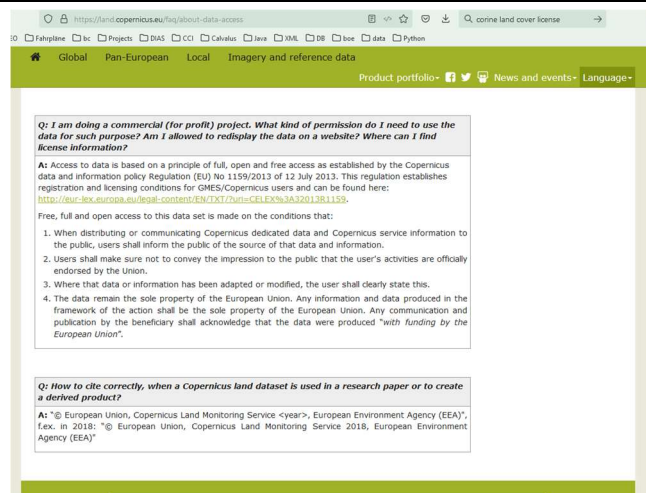


Figure 3-18: Download of license text for Corine Land Cover data from data provider CLMS

3.3.3 Verification of file download

To verify that the data provided on the Geo File Server can be downloaded,

- use a browser and access <https://data.balticaims.eu>

- navigate to some file to be downloaded
- download or open it

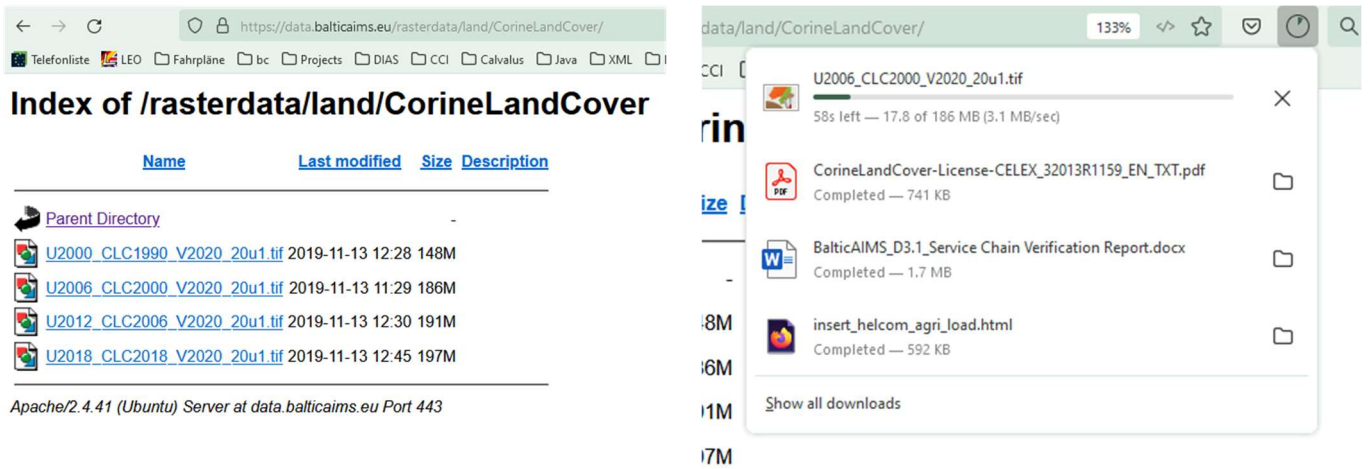


Figure 3-19: Download of a data file from the BalticAIMS Geo File Server

The same can be done for license texts for the datasets.

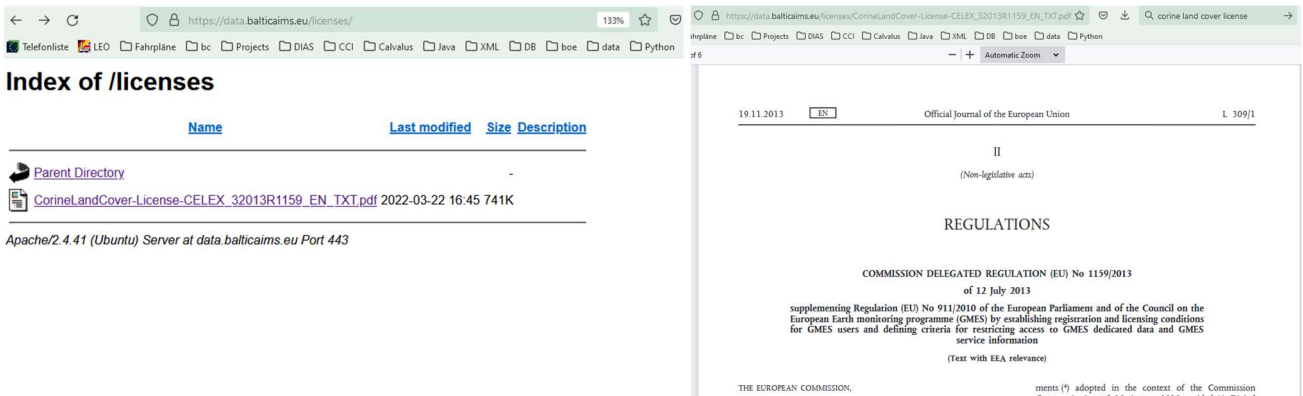


Figure 3-20: Access to license text for Corine Land Cover data

3.4 Naming convention

3.4.1 Naming of xcube datasets

Individual data cubes are stored on balticaims object storage `s3://balticaims/cubes/<cube name>.zarr` and named according to rule

`<region>-<record name>-<aggregation level>.zarr`

where

- region: identifier for different regions [*mv* | *po* | *fi* | *go*]
- record name: adapted entry from dataset table
- Aggregation level, optional, identifier for aggregation level and temporal resolution, eg. *14d-monthly*

For example:

`s3://balticaims/cubes/fi-hroc-14d-monthly.zarr`

3.4.2 Naming of geodb collections

GeoDB Collections are named according to Record Name in the Datasets table. Attributes are named by names listed in the datasets table in column “Variables required”, or by their original name if not specified differently.

<region>_<record name>_<variable name>_<time>

where

- region: optional identifier for regional datasets
- record name: adapted entry from dataset table
- variable name: optional variable name from dataset table
- time: optional time-stamp information for temporal datasets

For example:

helcom_plc_agricultural_load_phosphorus
clm_coastal_lc_lu_2018, helcom_hotspots_2019

3.4.3 Naming of files to be downloaded

Archiving rules for the raster datasets to be shared on the geofile server are

/balticaims/geofiles/rasterdata/<topic>/<record name>/[<region>]/[<temporal coverage>]/<filename>.tif

where

- topic: one of land, water, (extended if helpful)
- record name: entry from Dataset table
- region: optional, if there are separate datasets for different regions, e.g Gotland
- temporal coverage: optional, if there are separate datasets for different periods, e.g. 2018
- filename: either the original file name as provided by the data source, or a suitable name to identify the file

For example:

https://data.balticaims.eu/rasterdata/land/HR_Land_Cover/fin/clc2012_fi20m/

4 Accessing data of the BalticAIMS services

This section explains how to access data of the different BalticAIMS OGC interfaces and interactive interfaces of TARKKA, QGIS, xcube viewer, and Jupyter notebooks, with reports on verification.

4.1 OGC interfaces WMTS, WFS, WMS

The OGC Web Mapping Tile Service WMTS provided by xcube server serves time series of gridded data to suitable clients. The OGC Web Feature Service WFS and Web Mapping Service WMS provided by Geoserver as front end of geodb serves spatial feature data. How these interfaces are used by the clients TARKKA and QGIS is subject to the next two subsections 0 and 4.3. This section describes the machine-to-machine WEB API.

4.1.1 Interface description

WMTS of xcube has the base address <https://xcube.balticaims.eu/wmts/1.0.0/> and has two main entry points:

- GetCapabilities returns the list of layers available at xcube server.
- GetTile is called in a parameterized or RESTful HTTP GET call specifying layer and tile position. It returns the projected image of the requested layer and tile.

WFS of geodb has the base address <https://geoserver.balticaims.eu/geoserver/wfs> and three entry points:

- GetCapabilities returns the feature types (collections) available
- DescribeFeatureType returns the schema of a feature type (collection) with its attribute names
- GetFeature retrieves the value set of a feature type and returns an XML record

WMS of geodb has the base address <https://geoserver.balticaims.eu/geoserver/wms> and two entry points:

- GetCapabilities returns the list of collections available at geodb
- GetMap returns a gridded representation of the data of a geodb collection

4.1.2 Verification

The capabilities documents are retrieved at

<https://xcube.balticaims.eu/wmts/1.0.0/WMTSCapabilities.xml>

<https://geoserver.balticaims.eu/geoserver/wfs?request=GetCapabilities>

<https://geoserver.balticaims.eu/geoserver/wms?request=GetCapabilities>

Figure 4-1, Figure 4-2, and Figure 4-3 show the results of these queries respectively. The other entry points are verified by TARKKA and QGIS below, presenting data that is based on these services.

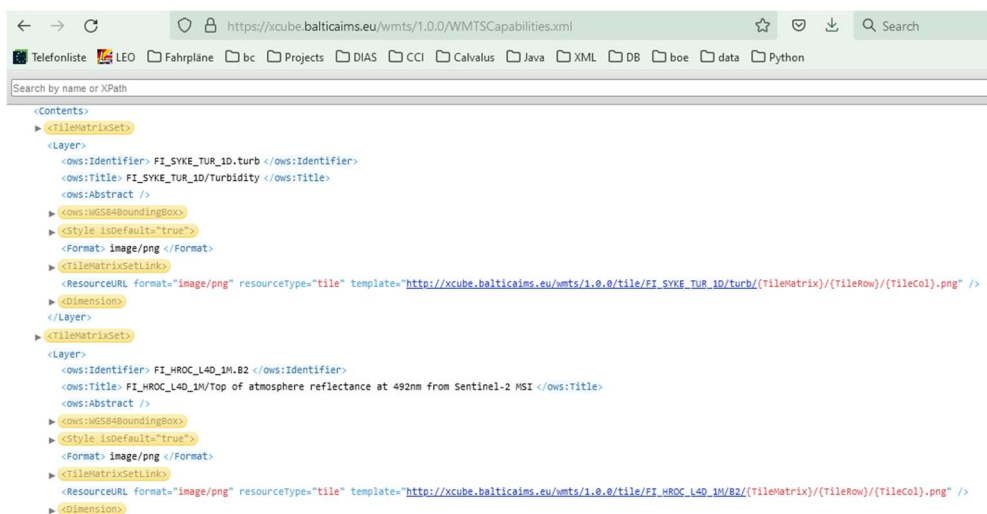
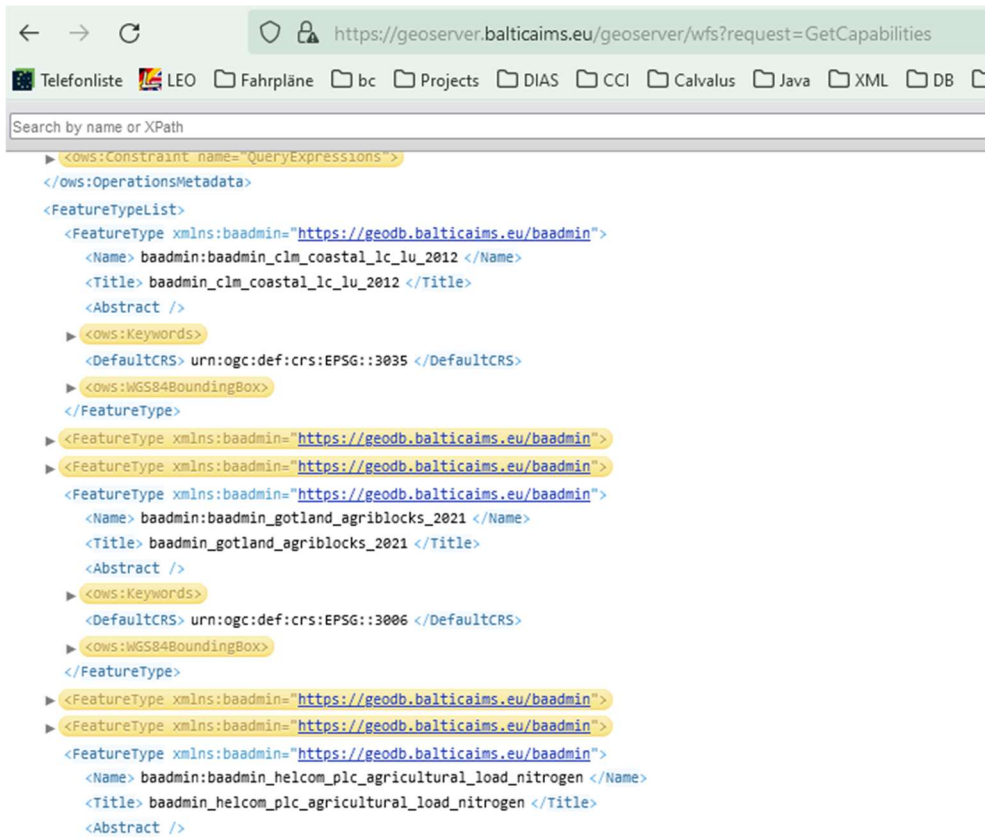
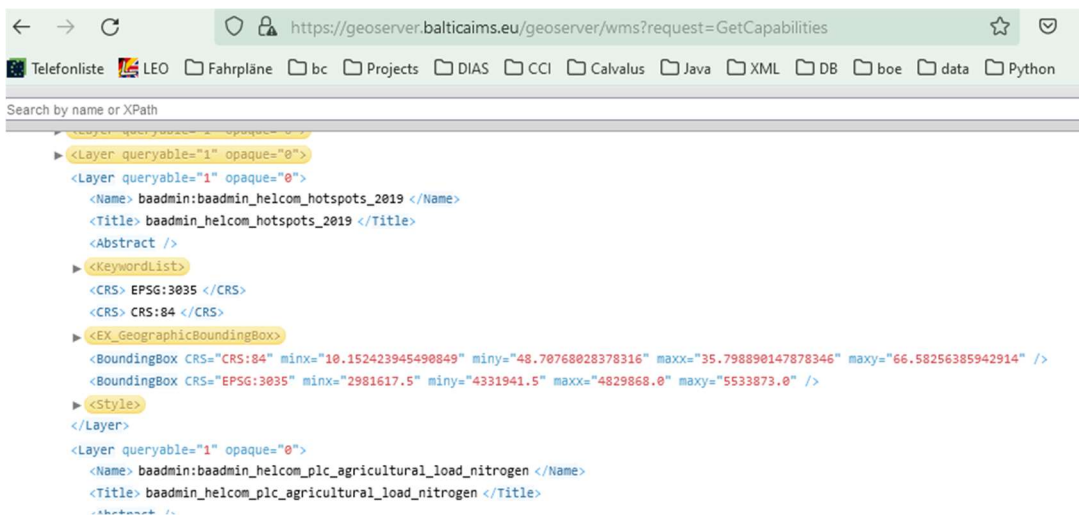


Figure 4-1: WMTS GetCapabilities response with layers provided by xcube server



```
<ows:Constraint name="QueryExpressions" />
</ows:OperationsMetadata>
<FeatureTypeList>
  <FeatureType xmlns:baadmin="https://geodb.balticaims.eu/baadmin">
    <Name> baadmin:baadmin_clm_coastal_lc_lu_2012 </Name>
    <Title> baadmin_clm_coastal_lc_lu_2012 </Title>
    <Abstract />
    <ows:Keywords>
      <DefaultCRS> urn:ogc:def:crs:EPSG::3035 </DefaultCRS>
    <ows:WGS84BoundingBox>
    </FeatureType>
  <FeatureType xmlns:baadmin="https://geodb.balticaims.eu/baadmin">
  <FeatureType xmlns:baadmin="https://geodb.balticaims.eu/baadmin">
    <FeatureType xmlns:baadmin="https://geodb.balticaims.eu/baadmin">
      <Name> baadmin:baadmin_gotland_agriblocks_2021 </Name>
      <Title> baadmin_gotland_agriblocks_2021 </Title>
      <Abstract />
      <ows:Keywords>
        <DefaultCRS> urn:ogc:def:crs:EPSG::3006 </DefaultCRS>
      <ows:WGS84BoundingBox>
    </FeatureType>
  <FeatureType xmlns:baadmin="https://geodb.balticaims.eu/baadmin">
  <FeatureType xmlns:baadmin="https://geodb.balticaims.eu/baadmin">
    <FeatureType xmlns:baadmin="https://geodb.balticaims.eu/baadmin">
      <Name> baadmin:baadmin_helcom_plc_agricultural_load_nitrogen </Name>
      <Title> baadmin_helcom_plc_agricultural_load_nitrogen </Title>
      <Abstract />
```

Figure 4-2: WFS GetCapabilities response with collections provided by geodb



```
<ows:Constraint name="QueryExpressions" />
</ows:OperationsMetadata>
<Layer queryable="1" opaque="0">
  <Layer queryable="1" opaque="0">
    <Name> baadmin:baadmin_helcom_hotspots_2019 </Name>
    <Title> baadmin_helcom_hotspots_2019 </Title>
    <Abstract />
    <KeywordList>
      <CRS> EPSG:3035 </CRS>
      <CRS> CRS:84 </CRS>
    <EX_GeographicBoundingBox>
      <BoundingBox CRS="CRS:84" minx="10.152423945490849" miny="48.70768028378316" maxx="35.798890147878346" maxy="66.58256385942914" />
      <BoundingBox CRS="EPSG:3035" minx="2981617.5" miny="4331941.5" maxx="4829868.0" maxy="5533873.0" />
    <Style>
  </Layer>
  <Layer queryable="1" opaque="0">
    <Name> baadmin:baadmin_helcom_plc_agricultural_load_nitrogen </Name>
    <Title> baadmin_helcom_plc_agricultural_load_nitrogen </Title>
    <Abstract />
```

Figure 4-3: WMS GetCapabilities response with the same collections provided by geodb

4.2 TARKKA integration

4.2.1 Interface description

TARKKA is a highly customizable web application framework for visualizing spatiotemporal data both on map and as statistical timeseries. The app builds on the React and OpenLayers frameworks and OGC-compliant data providers, with customizations implemented for several external data providers such as the Xcube Server used in BalticAIMS. Although the web app frontend presented for the end user is the most visible part of the TARKKA platform, the actual data handling relies heavily on backend APIs and data processing workflows not visible to the end user.

TARKKA provides the user with abilities to select which data the user wants to utilize, along with both temporal and spatial controllers. The user can access timeseries data for regions of interest (ROI) by selecting the ROI using map tools.

The TARKKA app is agnostic about the data production chain. Both in-situ and Earth Observation data can be utilized simultaneously, as well as supporting GIS-datasets (e.g. regional divisions) to further tailor the users' viewpoint.

The first version of TARKKA was published in 2017 within a Finnish government funded VESISEN project (TARKKA-1.0). Parallel to the BalticAIMS project (2021/Q4-2022/Q4), the TARKKA framework is undergoing life-cycle management, which will result in a modernized version of the TARKKA framework (TARKKA-2.0). The BalticAIMS use cases are built on top of the TARKKA-2.0 at its current development state.

4.2.2 BalticAIMS extensions for TARKKA

To access spatial data for map visualizations, OGC-compliant WMS or WMTS services are used to provide well-performing data access. For the use cases in BalticAIMS, a data interface for Xcube Server WMTS was implemented. To access timeseries data, TARKKA can utilize OData-standard compliant APIs, or any data providers providing the timeseries data in JSON/GeoJSON format. For the use cases in BalticAIMS, a data interface for the Xcube Server Timeseries API was implemented.

By default TARKKA also supports OGC-compliant interfaces that are also used within BalticAIMS to access data from external data sources directly.

For each BalticAIMS use case, relevant data sources are combined as a *theme* that the user may select to visualize and analyze the corresponding datasets. The theme selector is provided as a drop-down menu in TARKKA EO Map Viewer tab.

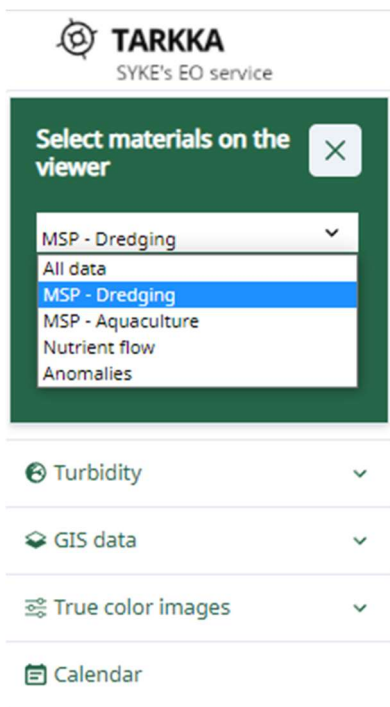


Figure 4-4: TARKKA theme selector

4.2.3 Verification

Currently **Use case C1: Dredging** is implemented in TARKKA. This use case utilizes data sources and interfaces described in Table 4-1. Additional use cases can be added in the theme selector based on this first example easily.

Table 4-1: Data sources and interfaces utilized in TARKKA for Use case C1.

Dataset	Origin	Interface type
Raster data		
Turbidity, SYKE	BalticAIMS (xcube)	Xcube WMTS
Turbidity, HROC	BalticAIMS (xcube)	Xcube WMTS
Sentinel-2 true color images (10m)	Sentinel hub service with SYKE modifications	WMS
Sentinel-3 true color images (300m)	Sentinel hub service with SYKE modifications	WMS
Water surface temperature, AVHRR & SLSTR (1km)	SYKE open data (GeoServer)*	WMS
Time series data		
Turbidity, SYKE	BalticAIMS (xcube)	Xcube API
Turbidity, HROC	BalticAIMS (xcube)	Xcube API
Monitoring stations data of Finnish coast and lakes (VESLA)	SYKE open data*	Odata

GIS data		
Dredging areas, Helsinki	SYKE (GeoServer)	WMS
Nature reserves, Finland	SYKE open data*	WMS
Reference stations, point data	SYKE (GeoServer)	WMS
Reference stations' areas for EO data	SYKE (GeoServer)	WMS
WFD water bodies of Finland	SYKE open data*	WMS
Main drainage basins of Finland	SYKE open data*	WMS
Basemaps		
OpenStreetMap	Basemap native in OpenLayers library	WMS
NLS basemap of Finland	National Land Survey of Finland, open data	WMTS
NLS terrain maps of Finland	National Land Survey of Finland, open data	WMTS

*https://www.syke.fi/en-US/Open_information

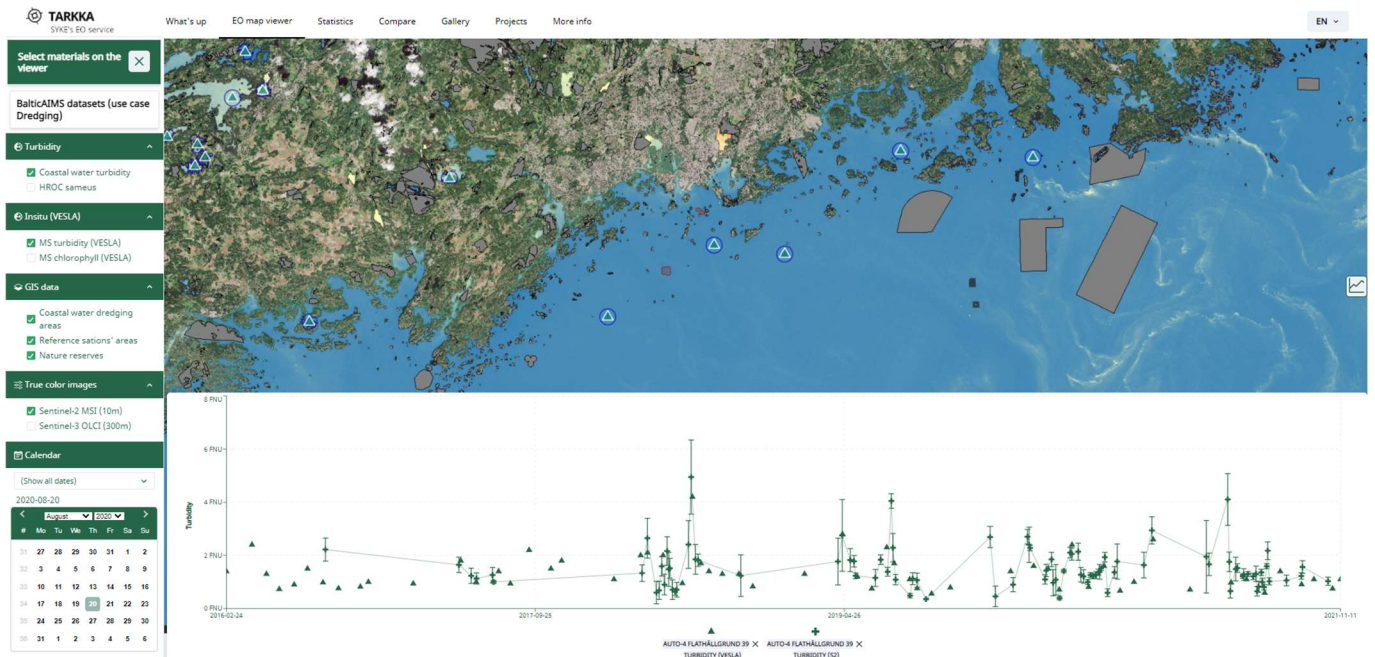


Figure 4-5: A screenshot of TARKKA presenting turbidity data from BalticAIMS Xcube both on map and as a time series. The map view contains also in-situ measurement station locations, nature reserve areas and the dredging areas for the Use Case C1.

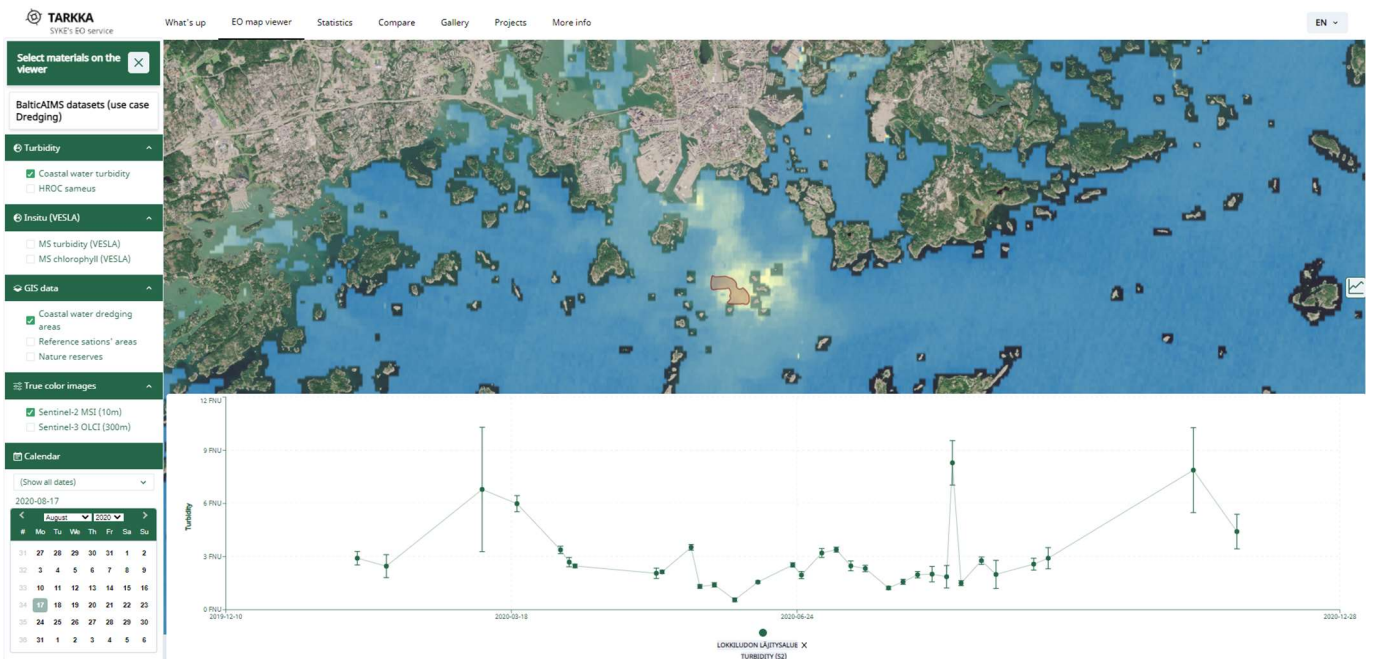


Figure 4-6: Another view presenting the actual dredging event, with the timeseries extracted from the dumping site

4.3 QGIS configuration for xcube data and geodb data

This section illustrates how to access the BalticAIMS web services using the desktop GIS application QGIS (4.3.1). QGIS is used to check the performance of the OGC services as well as the availability of service operations (4.3.4). The QGIS version used for tests is 3.24.0-Tisler.

4.3.1 Accessing the xcube WMTS OGC web services

xcube serves its data by an OGC WMTS service. This is directly accessible from GIS applications. The xcube WMTS serves collections of time series, with several layers each. Table 4-2 lists the currently available collections, layers and time ranges.

Table 4-2: BalticAIMS xcube WMTS: collections, layers, and temporal extent (2022-03-15).

Collection	Title	ToA 492 nm	ToA 560 nm	ToA 665 nm	CHL/CHL_mean	SPM/SPM_mean	TUR/TUR_mean	Other	Temporal extent
FI_SYKE_TUR_1D	Finland SYKE S2 turbidity 1D	X	X	X	X	X	X		2016-08-24 – 2021-10-31
FI_HROC_L4D_1D	Finland HROC L4D daily	X	X	X	X	X	X		2020-02-03 - 2021-08-31
FI_HROC_L4D_1M	Finland HROC L4D monthly	X	X	X	X	X	X		2020-02-01 - 2021-08-01
FI_CORINE_LC and	Finland Corine LC							CLC	1990, 2000, 2006, 2018
FI_HR_CORINE_LC	Finland Corine LC 20m							CLC	2000, 2006, 2012, 2018
FI_ALGAE_INDEX_1D	Finland SYKE MR Algae bloom index 1D							ALGAE	2016-07-04 - 2021-09-09
FI_S2_ALGAE_INDEX_1D	Finland SYKE HR Algae bloom index 1D							ALGAE	2017-07-23 - 2021-09-01
FI_SLSTR_SST_1D	Surface Temperature							SST	2017-05-01 - 2021-11-15

Collection	Title	ToA 492 nm	ToA 560 nm	ToA 665 nm	CHL/CHL_mean	SPM/SPM_mean	TUR/TUR_mean	Other	Temporal extent
HROC_SST_1D_Helsinki	Helsinki HROC SST 1D							SST	2021-04-01 - 2021-09-28
GO_HROC_L4D_1D	Gotland HROC L4D daily	X	X	X	X	X	X		2020-02-03 - 2021-08-31
GO_HROC_L4D_1M	Gotland HROC L4D monthly	X	X	X	X	X	X		2020-02-01 - 2021-08-01
GO_CORINE_LC	Gotland Corine LC							CLC	1990, 2000, 2006, 20181
GO_SYKE_TUR_1D	Gotland SYKE S2 turbidity 1D						X		2000-01-01 - 2021-10-31
GO_ALGAE_INDEX_1D	Gotland SYKE MR Algae bloom index 1D							ALGAE	2016-07-04 - 2021-09-09
GO_S2_ALGAE_INDEX_1D	Gotland SYKE HR Algae bloom index 1D							ALGAE	2017-07-23 - 2021-09-01
GO_SLSTR_SST_1D	Gotland SYKE SLSTR SST 1D							SST	2017-05-01 - 2021-11-15
MV_HROC_L4D_1D	Meck.Vorp. HROC L4D daily	X	X	X	X	X	X		2020-01-17 - 2021-08-31
MV_HROC_L4D_1M	Meck.Vorp. HROC L4D monthly	X	X	X	X	X	X		2020-01-01 - 2021-08-01
MV_CORINE_LC	Meck.Vorp. Corine LC							CLC	1990, 2000, 2006, 2018
BA_HROC_L3_5D_cloud	Baltic Sea HROC L3 5D (test cloud storage)	X	X	X	X	X	X	hroc_mask_mean	2021-07-01 - 2021-09-29
BA_HROC_L3_1D	Baltic Sea HROC L3 1D (test)	X	X	X	X	X	X	hroc_mask	2021-07-01 - 2021-09-30

Figure 4-7 shows one layer of one of these datasets, the mean Chlorophyll-a concentration of the MV HROC L4D 1D collection for August 2021.

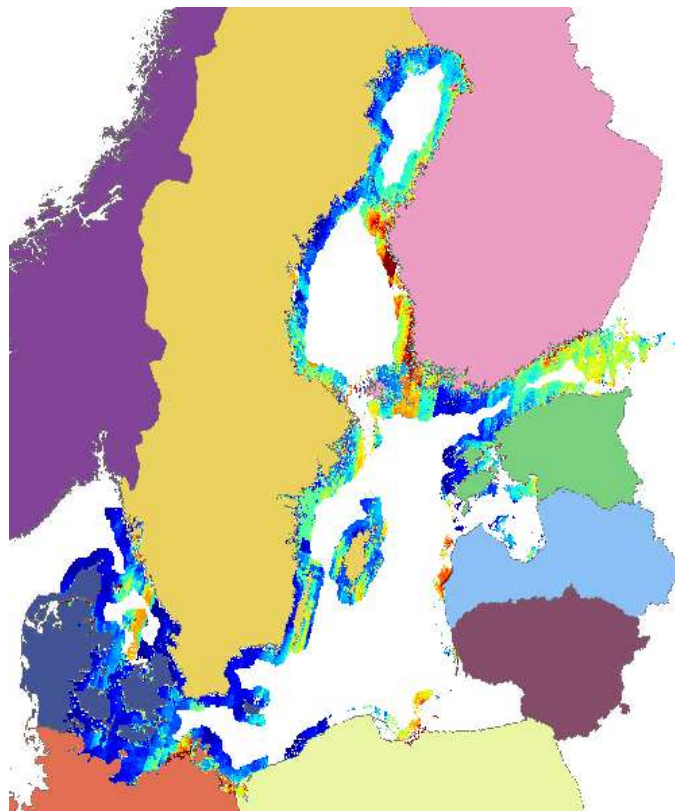


Figure 4-7: BalticAIMS WMTS: mean Chlorophyll a concentration in the Baltic Sea area (August 2021, no scale bar available).

Figure 4-8 shows the forms to configure access to the BalticAIMS xcube server in QGIS. The menu Layer => Add layer => WMTS layer leads to the forms. Required entries and selections are URL (1), the layer name (2) and the time dimension (3). The URL can be copied from the BalticAIMS web site.

(1) Create or choose connection

(2) Choose layer

MV_HROC_L4D_1M.TUR_mean	image/png	MV_HROC_L4D_1M/Turbidity derived from MSI L2R using HR-OC L2W processor
-------------------------	-----------	---

(3) Select time dimension

Dimension	Wert	Zusammenfassung	Voreinstellung	Wert
1 time	time		current	2020-01-01T00:00:00.000000000

Figure 4-8: WMTS connection: Access a WMTS layer by configuring the URL, layer name and time dimension.

This example shows how a single time step is accessed. In section 4.3.3 below the access of a time series is described.

4.3.2 Accessing the geodb WMS and WFS OGC web services

Environmental data for the purpose of BalticAIMS use cases are stored in a BalticAIMS geoDB and served via a Geoserver instance (<https://geoserver.balticaims.eu/geoserver/wms?request=getCapabilities>). Currently, 8 layers are provided (Table 4-3).

Table 4-3: BalticAIMS geoserver: Environmental data layer about agriculture, hot spots and land cover.

Layer	Number of features (size)
baadmin_clm_coastal_lc_lu_2012	4425760 (7.7 GB)
baadmin_clm_coastal_lc_lu_2018	4425760 (7.8 GB)
baadmin_clm_coastal_lc_lu_changes_2012_2018	232176 (323 MB)
baadmin_gotland_agriblocks_2021	32464 (36 MB)
baadmin_gotland_varo_2016	56 (188 kB)
baadmin_helcom_hotspots_2019	162 (466 kB)
baadmin_helcom_plc_agricultural_load_nitrogen	310 (50 MB)
baadmin_helcom_plc_agricultural_load_phosphorus	310 (50MB)

As presented in Figure 4-9 for the layer `baadmin_helcom_hotspots_2019` and `baadmin_clm_coastal_lc_lu_2018`, the WMS uses default styles for their symbolisation, i. e. grey polygons and red squares.

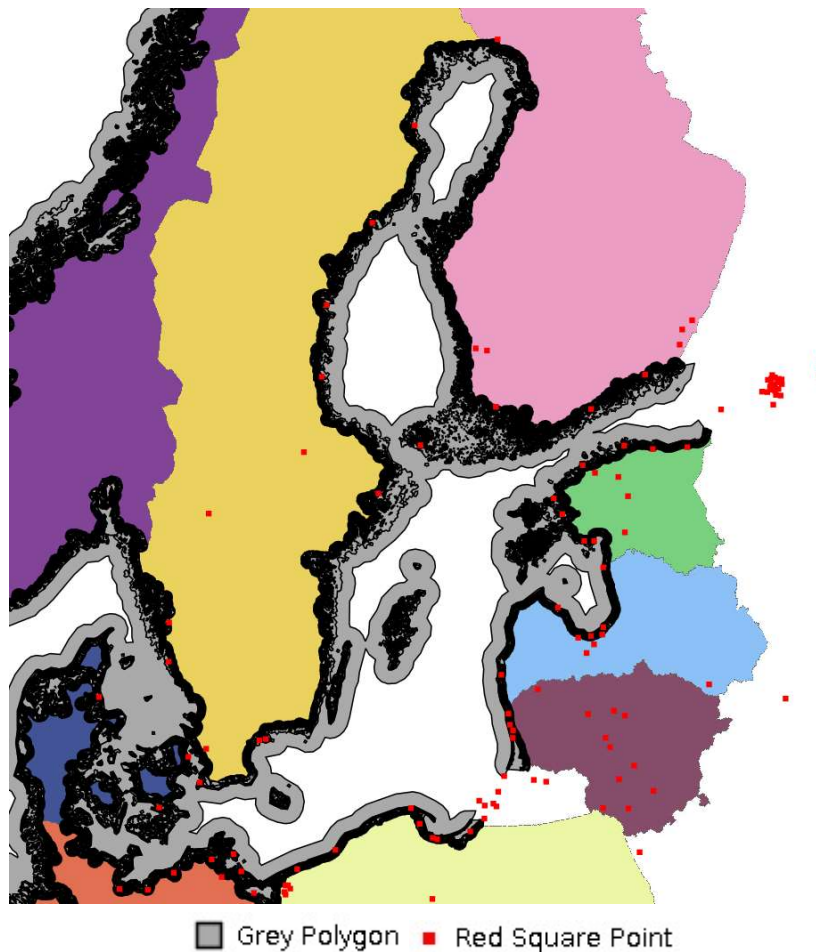
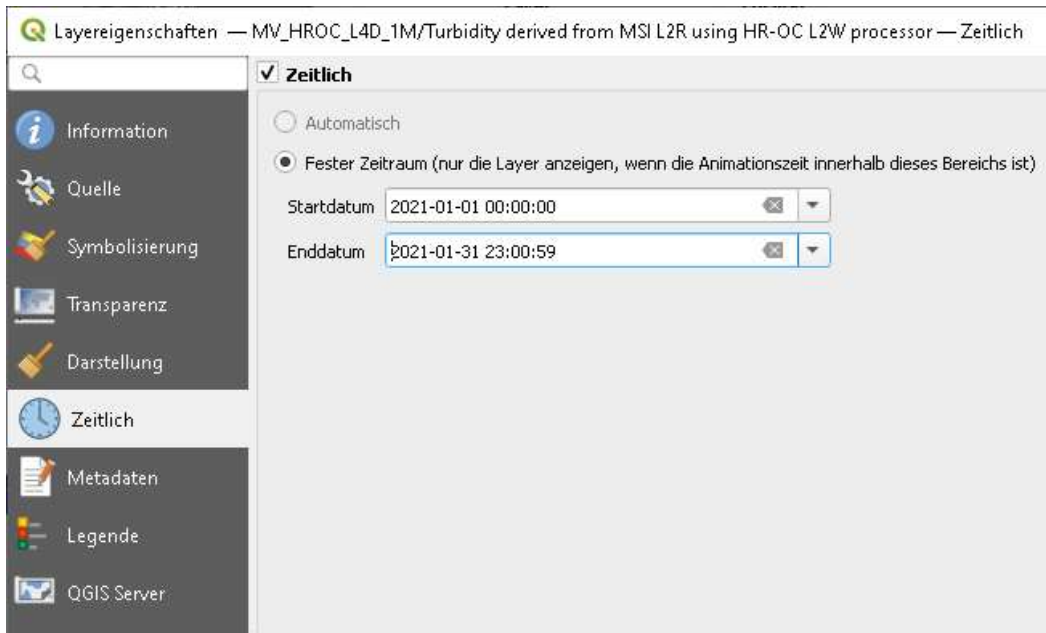


Figure 4-9: BalticAIMS geoserver: Layer `baadmin_helcom_hotspots_2019` (red squares) and `baadmin_clm_coastal_lc_lu_2018` (grey polygons).

4.3.3 QGIS time series animations

QGIS has a time control bar for datasets. To create an animation for a selected layer the corresponding time range must be selected in the layer's properties as depicted in Figure 4-10 (1). The corresponding layer will be marked by a clock icon in the table of content (2) and the animation can be started using the time control toolbar (3).

(1) Layer properties



(2) "Time aware" WMTS layer in the table of content ¹⁾



(3) Time control toolbar for animation

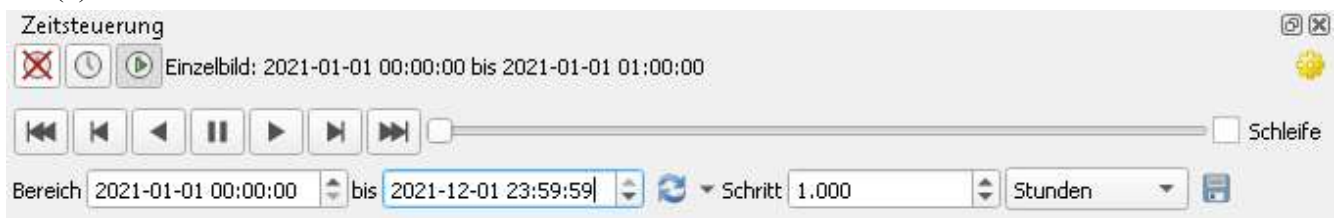


Figure 4-10: WMTS layer properties to define the time range of the layer for a time series animation.

4.3.4 WMTS verification

To verify the services, their performance using QGIS as client as well as the available service operations have been measured. The QGIS project used for verification contains monthly mean turbidity and SPM layers of FI, GO and MV for August 2021. For BA turbidity and SPM layers have been added showing the 5-days mean value (see Figure 4-11). For MV further timesteps have been included covering the time period from January to August 2021. They have been used to create an animation.

¹⁾ Not to be mixed-up with WMS-T service layer, i. e. WMS services with (real) time awareness.

- WMTS
 - Corine
 - MV_CORINE_1C/CLC
 - HROC_SST_1D_Helsinki/SST
 - GO_HROC_L4D_1D/Chlorophyll-a concentration derived from MSI L2R using HR-OC L2W processor
 - FI_HR_CORINE_1C/CLC
 - FI_CORINE_1C/CLC
 - Turbidity
 - 5 days
 - BA_HROC_L3_5D_cloud/Turbidity derived from MSI L2R using HR-OC L2W processor (2021-09-29, 5 days mean)
 - 1 month
 - FI_HROC_L4D_1M/Turbidity derived from MSI L2R using HR-OC L2W processor (Aug 2021)
 - GO_HROC_L4D_1M/Turbidity derived from MSI L2R using HR-OC L2W processor (Aug 2021)
 - MV_HROC_L4D_1M/Turbidity derived from MSI L2R using HR-OC L2W processor (Jan 2021)
 - MV_HROC_L4D_1M/Turbidity derived from MSI L2R using HR-OC L2W processor (Feb 2021)
 - MV_HROC_L4D_1M/Turbidity derived from MSI L2R using HR-OC L2W processor (Mar 2021)
 - MV_HROC_L4D_1M/Turbidity derived from MSI L2R using HR-OC L2W processor (Apr 2021)
 - MV_HROC_L4D_1M/Turbidity derived from MSI L2R using HR-OC L2W processor (May 2021)
 - MV_HROC_L4D_1M/Turbidity derived from MSI L2R using HR-OC L2W processor (Jun 2021)
 - MV_HROC_L4D_1M/Turbidity derived from MSI L2R using HR-OC L2W processor (Jul 2021)
 - MV_HROC_L4D_1M/Turbidity derived from MSI L2R using HR-OC L2W processor (Aug 2021)
 - SPM
 - 5 days
 - BA_HROC_L3_5D_cloud/Suspended Particulate Matter derived from MSI L2R using HR-OC L2W processor (2021-09-29, 5 days mean)
 - 1 month
 - MV_HROC_L4D_1M/Suspended Particulate Matter derived from MSI L2R using HR-OC L2W processor (Aug 2021)
 - GO_HROC_L4D_1M/Suspended Particulate Matter derived from MSI L2R using HR-OC L2W processor (Aug 2021)
 - FI_HROC_L4D_1M/Suspended Particulate Matter derived from MSI L2R using HR-OC L2W processor (Aug 2021)
- Background
 - GISCO 2010 Country

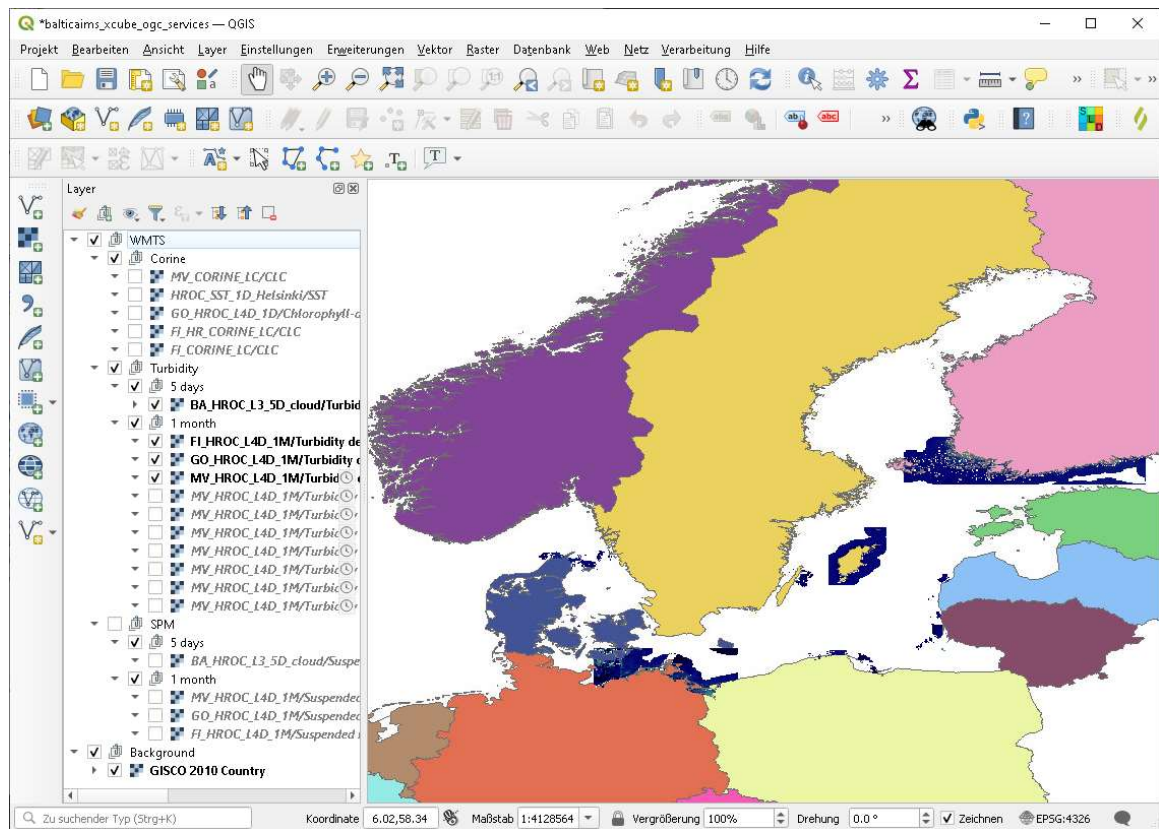


Figure 4-11: QGIS project for WMTS layer: Table of content and map display.

Measurement results:

- Opening the project with layer visibility turned off takes 12 sec.
- The duration to visualise the layer varies between 1 and 3 seconds (see Table 4-4).
- Visualisation was slowest for layer FI_HR_CORINE_LC/CLC taking approx. 3 sec.

These durations are considered acceptable.

Additional observations:

- A legend is not provided by the WMTS service and the info button in QGIS is thus inactive. These findings indicate that the service operations GetLegendGraphic and GetFeatureInfo are not provided by the service (for confirmation see: <https://xcube.balticaims.eu/wmts/1.0.0/WMTSCapabilities.xml>).
- As a result, the layer cannot be used appropriately to analyse the data in desktop GIS applications or data portals not specifically configured.

The addition of the two operations is suggested for further development.

4.3.5 WFS and WMS verification

Measurement results

The BalticAIMS geoDB data are presented by a Geoserver instance as WFS and WMS. The performance for all layers and both services is described in Table 4-4. Selected results are explained below the table.

Table 4-4: geoDB web mapping and web feature service: Time for visualisation in QGIS.

Layer	Number of features (sizeMB)	Full extent		Gotland (1:750.000)			
		Time for visualisation (OpenLayers, WMS ²⁾)	Time for visualisation (QGIS, WFS)	Time for visualisation (QGIS, WMS)	WMS GetFeatureInfo (QGIS, WMS)	Time for visualisation (QGIS, WFS)	WFS GetFeatureInfo
baadmin_clm_co astal_lc_lu_2012	4425760 (7.7 GB)	approx. 1 min	Layer and table: > 1 h	13 sec	5 seconds	Layer: 1.5 min Table: > 5 min ³⁾	17 sec
baadmin_clm_co astal_lc_lu_2018	4425760 (7.8 GB)	approx. 1 min	Layer and table: > 1 h	13 sec	8 seconds	Layer: 1.5 min Table: >> 5 min ³⁾	8"
baadmin_clm_co astal_lc_lu_changes_2012_2018	232176 (323 MB)	2 sec	Layer: 3 min Table: ?	1.5 sec	< 1 sec	Layer: 8 sec Table: 1 min ⁴⁾ , 5 sec ³⁾	1"

²⁾ Source:

<http://geoserver.balticaims.eu/geoserver/web/wicket/bookmarkable/org.geoserver.web.demo.MapPreviewPage?l&filter=false>
 => OpenLayers

³⁾ Loading features in current extent (Gotland) only.

⁴⁾ Loading all features.

Layer	Number of features (sizeMB)	Full extent		Gotland (1:750.000)			
		Time for visualisation (OpenLayers, WMS ²⁾)	Time for visualisation (QGIS, WFS)	Time for visualization (QGIS, WMS)	WMS GetFeatureInfo (QGIS, WMS)	Time for visualisation (QGIS, WFS)	WFS GetFeatureInfo
baadmin_gotland_agriblocks_2021	32464 (36 MB)	1 sec	50 seconds	5 sec	< 1 sec	Layer: 25 sec Table: > 2 h ⁴⁾ , 12 sec ³⁾	immediately
baadmin_gotland_varo_2016	56 (188 kB)	2 sec	3 seconds	2 sec	< 1 sec	Layer: 1 sec Table: 1 sec ⁴⁾	immediately
baadmin_helcom_hotspots_2019	162 (466 kB)	2 sec	20 seconds	< 1 sec	< 1 sec	Layer: 2 sec Table: 1 sec ⁴⁾	not appl. (otherwise immediately)
baadmin_helcom_plc_agricultural_load_nitrogen	310 (50 MB)	2 sec	Layer: 30 seconds Table: 3 seconds	3 sec	1 sec	Layer: 3 sec Table: 1 sec ⁴⁾	immediately
baadmin_helcom_plc_agricultural_load_phosphorus	310 (50MB)	2 sec	Layer: 20 seconds Table: 3 seconds	< 1 sec	< 1 sec	Layer: 3 sec Table: 1 sec ⁴⁾	immediately

- Visualisation performance for the WFS is very low due to the high number of geometries and the complexity of a few geometries having up to 4 Mio. points. Only the small layer baadmin_gotland_varo_2016 containing 56 simple geometries can be visualised with 1 or 3 seconds.
- Opening a QGIS project containing all WMS layers with layer visibility set to invisible takes 2 sec independent of the map scale used at startup.
- In contrast, the QGIS project for all WFS layers took 12 seconds to start.
 1. Open QGIS project with invisible layer (1:750.000): 12 sec without object counts (> 1 h with object counts!)
 2. Without the two 7 GB layer: 3.5 min with object counts (5 sec without object counts)

Data export

In QGIS the data can be exported to shapefiles (not tested for the land cover layer) if the info button is active. The WFS layer can be downloaded as shapefiles directly using the URLs listed in Table 4-5.

Table 4-5: BalticAIMS WFS layer: Geoserver links for shapefile download.

Layer	Download URL
baadmin_clm_coastal_lc_lu_2012	https://geoserver.balticaims.eu/geoserver/baadmin/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=baadmin%3Abaadmin_clm_coastal_lc_lu_2012&maxFeatures=50&outputFormat=SHAPE-ZIP

baadmin_clm_coastal_lc_lu_2018	https://geoserver.balticaims.eu/geoserver/baadmin/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=baadmin%3Abaadmin_clm_coastal_lc_lu_2018&maxFeatures=50&outputFormat=SHAPE-ZIP
baadmin_clm_coastal_lc_lu_changes_2012_2018	https://geoserver.balticaims.eu/geoserver/baadmin/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=baadmin%3Abaadmin_clm_coastal_lc_lu_changes_2012_2018&maxFeatures=50&outputFormat=SHAPE-ZIP
baadmin_gotland_agriblocks_2021	https://geoserver.balticaims.eu/geoserver/baadmin/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=baadmin%3Abaadmin_gotland_agriblocks_2021&maxFeatures=50&outputFormat=SHAPE-ZIP
baadmin_gotland_varo_2016	https://geoserver.balticaims.eu/geoserver/baadmin/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=baadmin%3Abaadmin_gotland_varo_2016&maxFeatures=50&outputFormat=SHAPE-ZIP
baadmin_helcom_hotspots_2019	https://geoserver.balticaims.eu/geoserver/baadmin/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=baadmin%3Abaadmin_helcom_hotspots_2019&maxFeatures=50&outputFormat=SHAPE-ZIP
baadmin_helcom_plc_agricultural_load_nitrogen	https://geoserver.balticaims.eu/geoserver/baadmin/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=baadmin%3Abaadmin_helcom_plc_agricultural_load_nitrogen&maxFeatures=50&outputFormat=SHAPE-ZIP
baadmin_helcom_plc_agricultural_load_phosphorus	https://geoserver.balticaims.eu/geoserver/baadmin/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=baadmin%3Abaadmin_helcom_plc_agricultural_load_phosphorus&maxFeatures=50&outputFormat=SHAPE-ZIP

The WMS capabilities XML file lists one style for the layer `baadmin_clm_coastal_lc_lu_2012` called generic. An optional layer content specific style is not provided yet.

```
<Layer queryable="1" opaque="0">
  <Name>baadmin:baadmin_clm_coastal_lc_lu_2012</Name>
  <Title>baadmin_clm_coastal_lc_lu_2012</Title>
  <Abstract/>
  <KeywordList>
    <Keyword>features</Keyword>
    <Keyword>baadmin_clm_coastal_lc_lu_2012</Keyword>
  </KeywordList>
  <CRS>EPSG:3035</CRS>
  <CRS>CRS:84</CRS>
  <EX_GeographicBoundingBox>
    <westBoundLongitude>-51.840412358600766</westBoundLongitude>
    <eastBoundLongitude>65.94168980893207</eastBoundLongitude>
    <southBoundLatitude>25.753066127198494</southBoundLatitude>
    <northBoundLatitude>71.9102658214829</northBoundLatitude>
  </EX_GeographicBoundingBox>
  <BoundingBox CRS="CRS:84" minx="-51.840412358600766" miny="25.753066127198494" maxx="65.94168980893207" maxy="71.9102658214829" />
  <BoundingBox CRS="EPSG:3035" minx="921932.3125" miny="1248146.0" maxx="5429336.0" maxy="6911689.0" />
  <Style>
    <Name>generic</Name>
    <Title>Generic</Title>
    <Abstract>Generic style</Abstract>
    <LegendURL width="20" height="20">
      <Format>image/png</Format>
      <OnlineResource xlink:type="simple" xlink:href="https://geoserver.balticaims.eu/geoserver/ows?service=WMS&request=GetLegendGraphic" />
    </LegendURL>
  </Style>
</Layer>
```

4.3.6 Verification summary

Table 4-6 summarizes the verification results of the previous two subsections.

Table 4-6: BalticAIMS WMTS, WFS and WMS: Verification results concerning performance and service operations.

Service	Visualisation	FeatureInfo	Legend	Style	Download
WMTS	Good	Not available (raster data values)	Missing	Available	Not applicable
WFS	Bad	Available	Not applicable	Missing	Available
WMS	Good	Available	Available	Missing	Not applicable

Without any further configuration, the WFS does not display the large Shapefiles with an acceptable performance. Since visualisation itself can be achieved by the WMS as well, the remaining WFS function with bad performance is the access to the feature table, which is not possible via WMS. Therefore, use cases should be checked for the requirement of "feature table access". Based on use cases it can be analysed:

1. Can the attribute table be limited to visible features only to enhance performance?
2. If yes, in which scale range shall the WFS attribute table be accessible?
3. Can performance be increased by adding a spatial index?
4. Can performance be enhanced by splitting the large shapefiles by country?
5. Shall the large features, having more than 4 Mio. points, be split in smaller polygons?
6. For which layer does the user require data from multiple countries at the same time?
7. Shall visibility of layer be limited to a certain scale range to avoid loading of the whole layer's features?

The following service functions are not available:

- The raster data (WMTS) need the GetFeatureInfo operation to display the pixel values of Chlorophyll-a, SPM, turbidity and other layers when hovering over the image.
- The WMTS does not provide a legend via the operation GetLegendGraphics and thus, the colours of the raster layer cannot be interpreted in absolute terms. To provide this function makes it unnecessary to configure the legend manually.
- For the Geoserver WMS of geodb the GetLegendsGraphic operation is available. However, the layer's styles are not stored and provided by geodb and thus they are not defined. This makes adding a collection simpler but finally lacks this type of information.

4.4 xcube viewer

The xcube viewer is a web application that displays time series of gridded data hosted by xcube server(s). To make data accessible to users the respective datasets are added to the xcube server as described in section 3.1. No additional configuration is required for the viewer.

4.4.1 Viewer functions

Figure 4-12 shows the xcube viewer web application available at <https://viewer.balticaims.eu> .

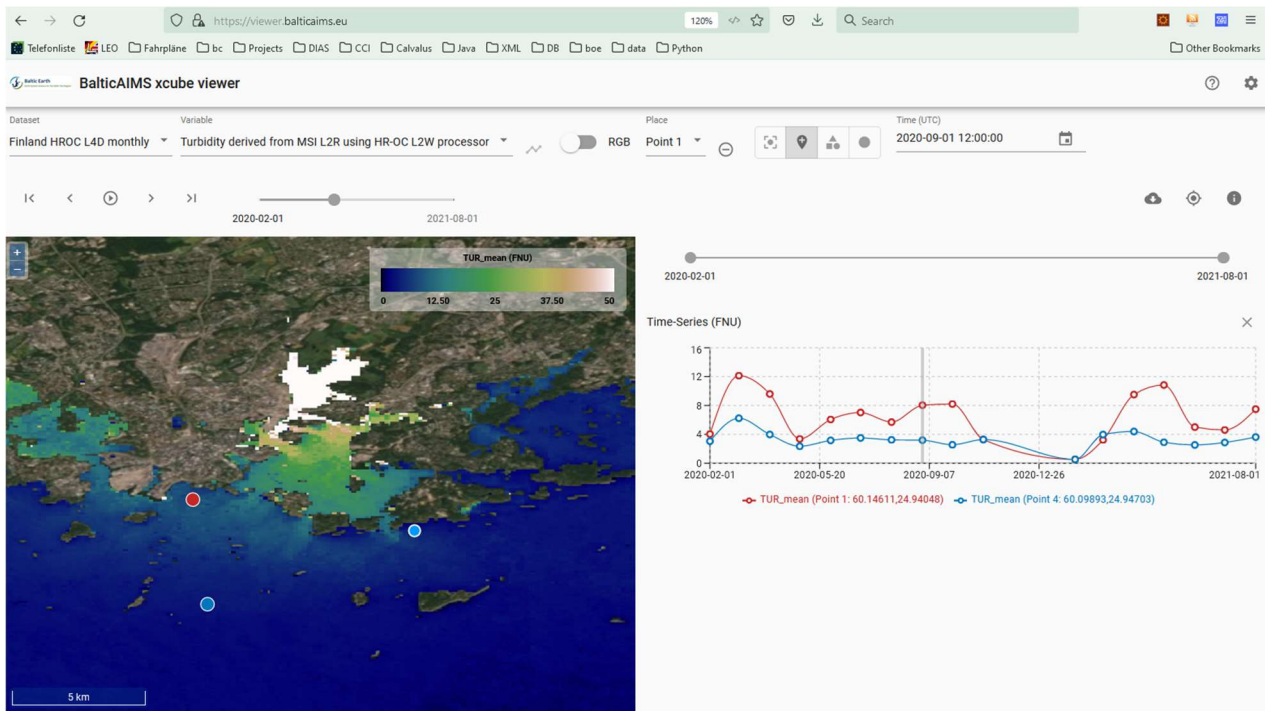


Figure 4-12: xcube viewer showing a dataset variable and the time series at two selected points

Functions of the viewer with respect to BalticAIMS data are

- to select BalticAIMS datasets and their variables
- to display a time step of a time series on a map, with zoom and pan and time series play (left)
- to display the time series of selected points or regions in a plot (right)
- to switch to RGB for datasets that provide that

4.4.2 Verification

The viewer provides access to the BalticAIMS collections for the three regions populated with data. The Finland HR-OC L4 monthly dataset has layers TUR, SPM, and CHL and the RGB bands of Sentinel-2.

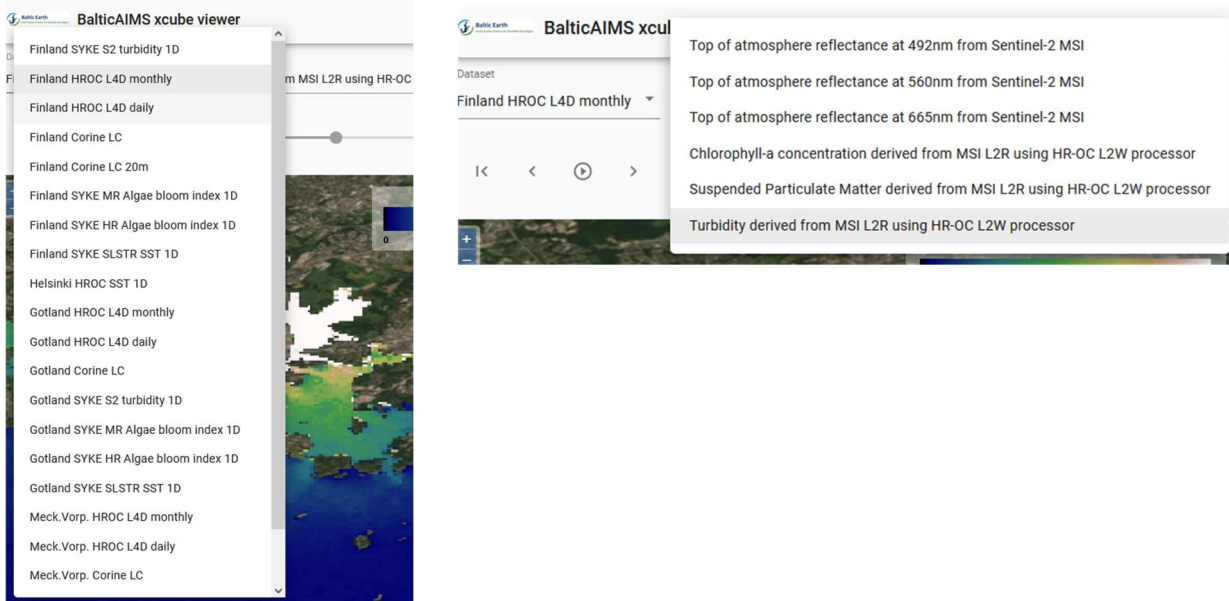


Figure 4-13: Datasets and layers can be selected in the xcube viewer

4.5 Jupyter Notebooks

The JupyterHub of BalticAIMS is a multi-user service with some prepared notebooks for geodb data access and visualisation. Additional notebooks can be developed by users starting from the existing notebooks as templates and examples.

While displayed in the user's browser, the notebooks in fact run on the balticcube VM. They access the data of geodb and xcube locally without network transfer to the user. This is different from the other modes of access that to some extent transfer selected data to the client applications via the API.

4.5.1 Using an existing notebook

After login (as bauser, password ..., Figure 4-14) the example notebooks are listed (Figure 4-15) and one can be started and run.

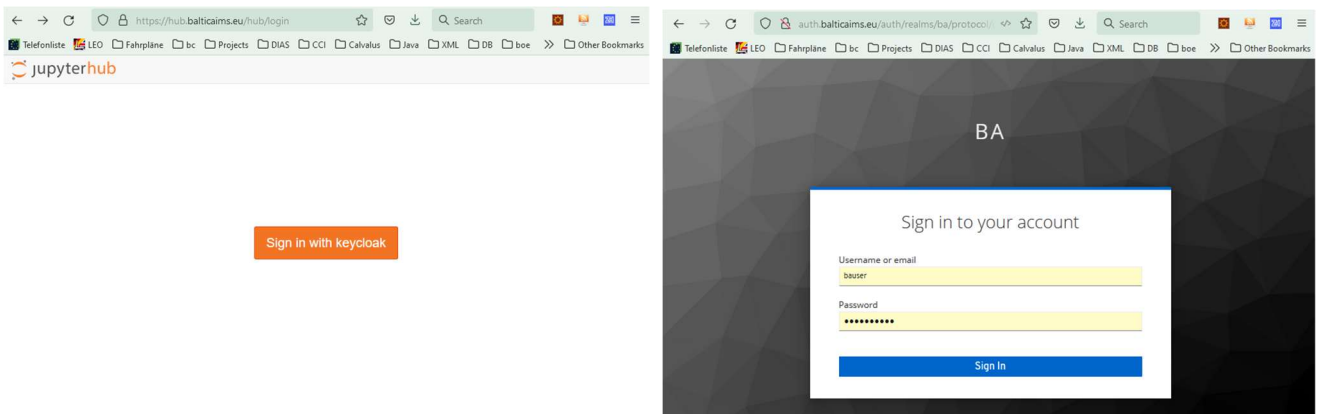


Figure 4-14: Login as bauser into BalticAIMS JupyterHub

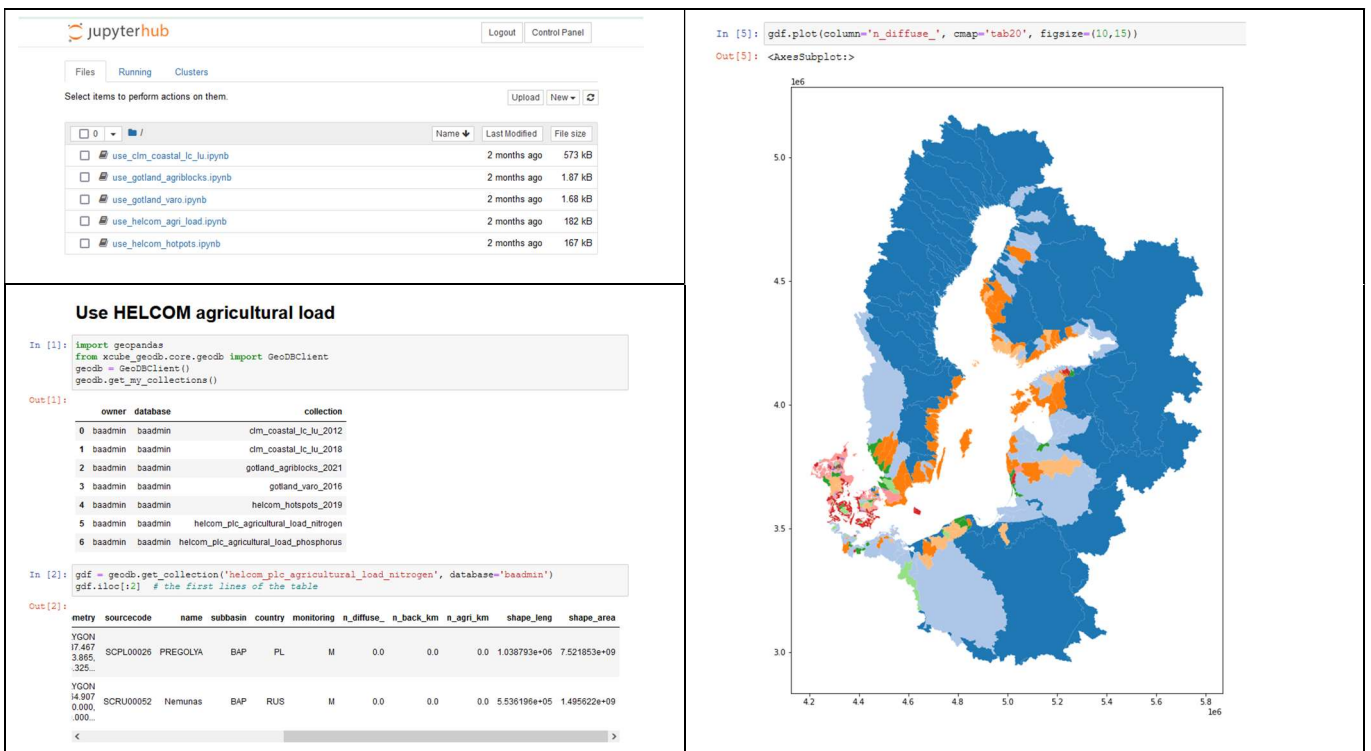


Figure 4-15: List of existing notebooks and run of one of them

For notebooks accessing geodb it is sufficient to import and instantiate the GeoDBClient. The user account logged in in JupyterHub has configured the credentials to access geodb. They are implicitly used when instantiating GeoDBClient.

The geodb API can be used to list collections and to load data of a collection into a geopandas GeoDataFrame, which is a representation of a tabular data structure with geographic features. The geodb API is described in <https://xcube-geodb.readthedocs.io/en/latest/core.html>.

4.5.2 Creating a new notebook

A new notebook can either be created as a duplicate of an existing notebook (Figure 4-16), or as a notebook from scratch (Figure 4-17).

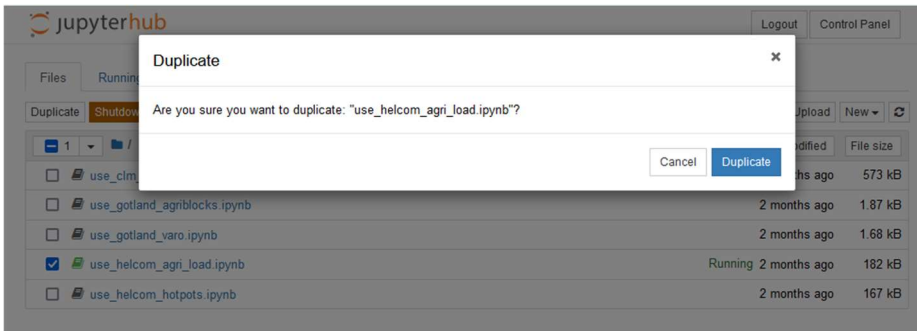


Figure 4-16: Duplicating an existing notebook



Figure 4-17: Creating a new notebook from scratch

Notebooks can be tested while writing them.

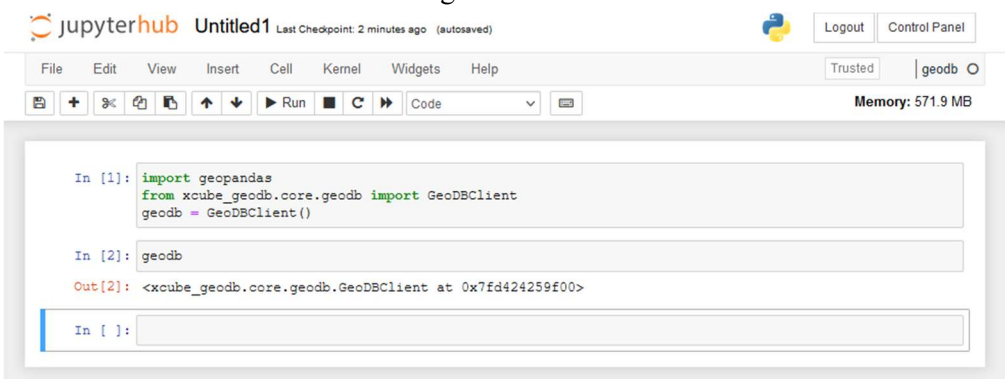


Figure 4-18: Stepwise development and test of a new notebook with geodb access

4.5.3 Verification

Verification of notebooks has been demonstrated in section 3.2 on how to add feature data.

5 References

Doc ID	Title	Origin, Version, Date
[Portfolio]	D2.1 BalticAIMS service portfolio definition	ESA project report, v1.1, 05.10.2021
[Platform]	D2.2 BalticAIMS data and platform provisioning plan	ESA project report, v1.0, 16.09.2021
[ChainSpec]	D2.3 BalticAIMS service delivery chain specification	ESA project report, v1.0, 16.09.2021
[SRR]	D3.2 BalticAIMS service readiness report	ESA project report, v1.0, 24.03.2022