AdriaClim

WP4
“INFORMATION SYSTEM AND PRODUCTS”

Activity 4.3
Big data repository and networking services

Deliverable 4.3.2
Software and Documentation on available services

CMCC

Final version

Public document

March 2023

Table of contents

1. *Aims and content of the document*

The present deliverable shows how to set up and configure the main component of the data repository, the ERDDAP data server introduced in the deliverable 4.3.1.

This document is organized as follows:

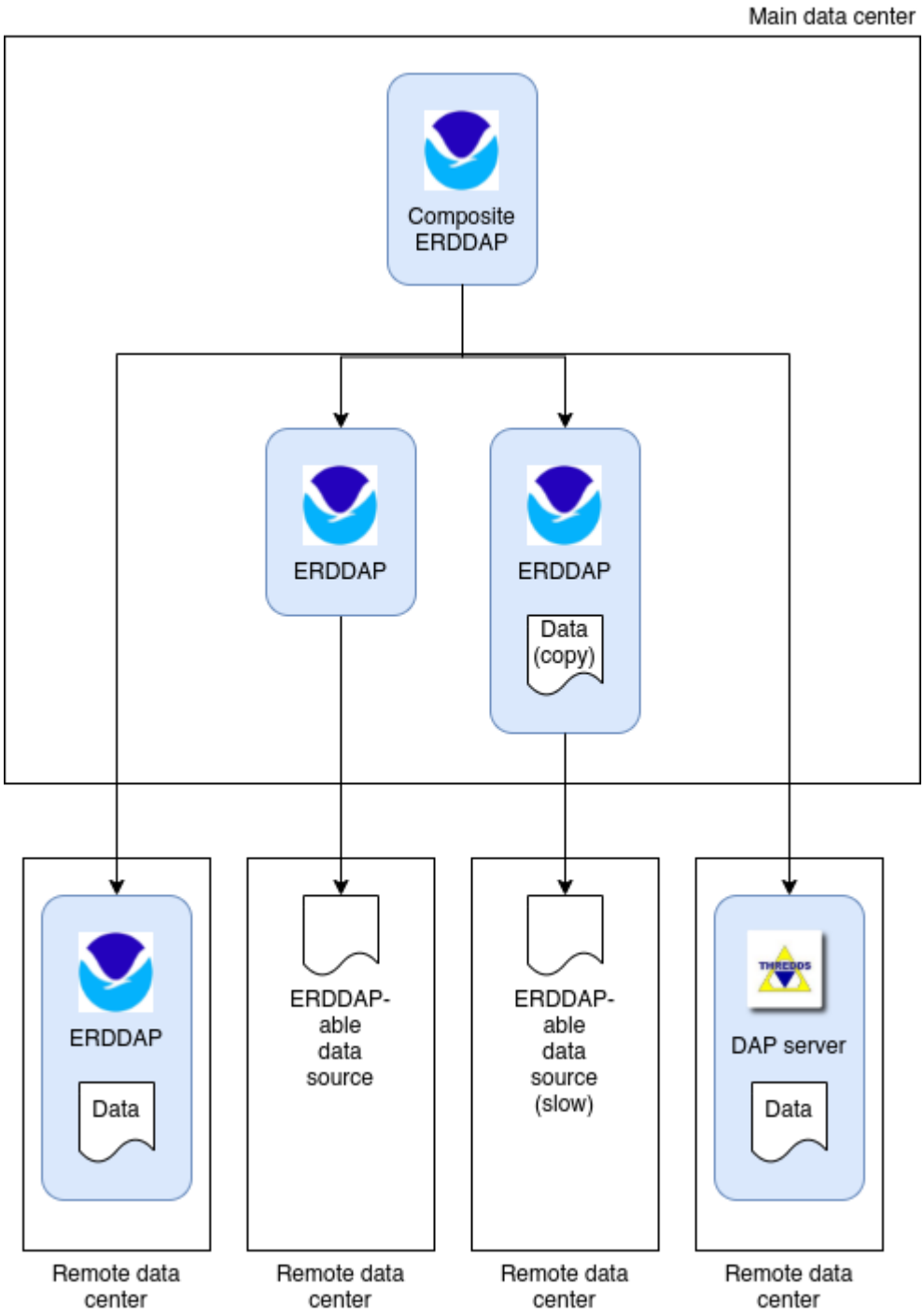
- **section 2** is a quick recap of the data repository architecture and any other software involved;
- **section 3** illustrates how to deploy the ERDDAP data server, as Docker container;
- **section 4** goes in depth on ERDDAP datasets configuration, for both a regular node and the central composite one, consumed by the geoportal;
- **section 5** shows a few examples of typical usage, under an end-user point of view.

2. Overview

As described in the deliverable 4.3.1, the data repository architecture is based on a federation of ERDDAP nodes:

- the central composite node, that routes queries to remote ERDDAP nodes which actually contain the data. The composite node only stores metadata;
- the remote ERDDAP/OpenDAP servers, i.e. regular nodes hosting the data. Each partner is responsible for its own data and maintains full control over them.

Please note that the central composite node must be based on ERDDAP, but the other nodes don't have such strict constraint: if they host a different type of DAP server (for example based on THREDDS), the central node can still interact with them, read metadata from their datasets and redirect users to those remote data (see the figure below, from deliverable 4.3.1).



In addition to ERDDAP data server capabilities, an ad-hoc script has been developed to help researchers and users in downloading dataset files, even if they are private datasets accessible after authentication.

The script can be executed as follows:

```
./erddap-downloader.sh -u <USERNAME> -p <PASSWORD> -d <DATASET_PATH>
```

Optional parameters are:

- `-e <ERDDAP_URL>`: ERDDAP URL (default: https://erddap.cmcc-opa.eu/erddap/)
- `-r <ERDDAP_REGEX>`: regular expression for filtering sub paths and filenames
- `-t <DST_DIR>`: destination directory

For example, suppose we want to download some datasets from the ERDDAP node

`https://erddap.cmcc-opa.eu/erddap/`

and we have identified the following files for downloading:

`https://erddap.cmcc-opa.eu/erddap/files/wrfhydro_historical_2aff_7940_c642/1992/`

The command would be:

```
./erddap-downloader.sh \
  -u <USERNAME> \
  -p <PASSWORD> \
  -d "wrfhydro_historical_2aff_7940_c642/1992"
```

- 3. ERDDAP deployment

ERDDAP is an open source data server written in Java, and then it can be installed on any machine - and operating system - that supports Java and Tomcat. Furthermore a Docker image is maintained and provided by a third-party company, for deploying ERDDAP as Docker container: this is the way described in this section.

The source code is hosted as Git repository on Github at the following URL:

`https://github.com/BobSimons/erddap`

while the Git repository of the Docker image is available at

`https://github.com/axiom-data-science/docker-erddap`

The Docker image is also uploaded to - and publicly pullable from - the Docker Hub Container Image Library:

`https://hub.docker.com/r/axiom/docker-erddap`

Before running the Docker container, and before starting to add datasets, a minimal configuration is required.

ERDDAP provides an initial compressed archive file, `erddapContent.zip`, available at the release page of the Github repository:

`https://github.com/BobSimons/erddap/releases`

The archive contains:

- a `setup.xml` file where to configure all the settings which specify how ERDDAP works;
- a `datasets.xml` file where to add ERDDAP datasets, as described in the next section;
- an `erddapStart2.css` file for CSS style settings. The file must be renamed and moved to the path `<tomcat>/content/erddap/images/erddap2.css`.

Since the datasets configuration is described further on in the deliverable, and the latter file is just related to web styling, this section will be focused on `setup.xml`.

Starting from ERDDAP version 2.13, the same parameters defined in the file can be set or overridden via environment variables, by specifying them as `ERDDAP_parameter`.

At least the following parameters must be set:

- `bigParentDirectory`: the absolute path on the server to a directory on a drive with appropriate space. ERDDAP will use this directory for at least a) caching information about the datasets to speed up reloading them in the future; b) caching data files; c) storing logs;
- `baseUrl`: the start of the public URL, to which `/erddap` is appended;
- `baseHttpsUrl`: HTTPS variant of baseUrl, required when authentication is active;
- `emailEverythingTo`: daily status reports and all error messages are emailed to this email account;
- `admin*`: information about the ERDDAP administrator (institution, address, country, email, etc.);
- `email*`: email account information used for sending emails to the `emailEverythingTo` and `emailDailyReportsTo` email addresses and for sending emails related to subscriptions (SMTP host, SMTP port, username, etc.);
- `flagKeyKey`: ERDDAP has a service that lets remote users set a flag to notify ERDDAP to try to reload a dataset. These requests use a key which is generated based on *baseUrl/erddap*, a *datasetID*, and *flagKeyKey*.

There are two additional environment variables for specifying the amount of memory to be allocated, `ERDDAP_MIN_MEMORY` and `ERDDAP_MAX_MEMORY`. Both environment variables will fall back to a single `ERDDAP_MEMORY` variable, which in turn falls back to "4G" by default. Alternatively, `ERDDAP_MAX_RAM_PERCENTAGE` sets the maximum Java heap size to a percentage of the memory available to the container. This option sets the JVM option `-XX:MaxRAMPercentage`.

Assuming we have configured the following directories:

- ERDDAP content: `/srv/erddap/content`, containing at least the files `setup.xml`, `datasets.xml` and `images/erddap2.css`
- ERDDAP (internal) data: `/srv/erddap/data`
- Data files (NetCDF, CSV, …): `/srv/erddap/datasources`

and ERDDAP's "bigParentDirectory" is `/erddapData/`, this is an example of a command for running ERDDAP via Docker container, v. 2.18:

```
docker run -d --restart unless-stopped \
  -p 80:8080 \
  -p 443:8443 \
  -v /srv/erddap/content:/usr/local/tomcat/content/erddap \
  -v /srv/erddap/data:/erddapData \
  -v /srv/erddap/datasources:/datasources \
  -e ERDDAP_MIN_MEMORY=4G \
  -e ERDDAP_MAX_MEMORY=8G \
  axiom/docker-erddap:2.18
```

If we want to set or to override the configuration parameters defined in `setup.xml` through environment variables, we can add them as additional Docker command line options, for example:

```
docker run -d --restart unless-stopped \
  -p 80:8080 \
  -p 443:8443 \
  -v /srv/erddap/content:/usr/local/tomcat/content/erddap \
  -v /srv/erddap/data:/erddapData \
  -v /srv/erddap/datasources:/datasources \
  -e ERDDAP_MIN_MEMORY=4G \
  -e ERDDAP_MAX_MEMORY=8G \
  -e ERDDAP_bigParentDirectory=/erddapData/ \
  -e ERDDAP_baseUrl=https://erddap.cmcc-opa.eu \
  -e ERDDAP_baseHttpsUrl=https://erddap.cmcc-opa.eu \
  -e ERDDAP_flagKeyKey=arandomstringetc \
```

```
  -e ERDDAP_emailEverythingTo=erddapadmin@example.com \
  -e ...(other ERDDAP_email* parameters)... \
  -e ERDDAP_adminInstitution=CMCC \
  -e ...(other ERDDAP_admin* parameters)... \
  axiom/docker-erddap:2.18
```

- 4. ERDDAP datasets configuration

Whenever we want to set up a new dataset in ERDDAP, the file `datasets.xml` must be edited and a chunk of XML added, starting from the "vanilla" version of the file available in the *zip* archive mentioned in the previous section, `erddapContent.zip`.

The structure of the file consists of some XML tags for dataset-wide configuration, and a list of `<dataset>` tags, one for each dataset to be provided:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<erddapDatasets>
  <angularDegreeUnits>...</angularDegreeUnits>
  <angularDegreeTrueUnits>...</angularDegreeTrueUnits>
  <cacheMinutes>...</cacheMinutes>
  <commonStandardNames>...</commonStandardNames>
  <convertInterpolateDatasetIDVariableExample />
  <convertInterpolateDatasetIDVariableList />
  <convertToPublicSourceUrl />
  <decompressedCacheMaxGB>...</decompressedCacheMaxGB>
  <decompressedCacheMaxMinutesOld>...</decompressedCacheMaxMinutesOld>
  <drawLandMask>...</drawLandMask>
  <emailDiagnosticsToErdData>...</emailDiagnosticsToErdData>
  <graphBackgroundColor>...</graphBackgroundColor>
  <ipAddressMaxRequests>...</ipAddressMaxRequests>
  <ipAddressMaxRequestsActive>...<ipAddressMaxRequestsActive>
  <ipAddressUnlimited>...<ipAddressUnlimited>
  <loadDatasetsMinMinutes>...</loadDatasetsMinMinutes>
  <loadDatasetsMaxMinutes>...</loadDatasetsMaxMinutes>
  <logLevel>...</logLevel>
```

```
  <nGridThreads>...</nGridThreads>
  <nTableThreads>...</nTableThreads>
  <palettes>...</palettes>
  <partialRequestMaxBytes>...</partialRequestMaxBytes>
  <partialRequestMaxCells>...</partialRequestMaxCells>
  <requestBlacklist>...</requestBlacklist>
  <slowDownTroubleMillis>...</slowDownTroubleMillis>
  <subscriptionEmailBlacklist>...</subscriptionEmailBlacklist>
  <unusualActivity>...</unusualActivity>
  <updateMaxEvents>...</updateMaxEvents>
  <standardLicense>...</standardLicense>
  <standardContact>...</standardContact>
  <standardDataLicenses>...</standardDataLicenses>
  <standardDisclaimerOfEndorsement>...</standardDisclaimerOfEndorsement>
  <standardDisclaimerOfExternalLinks>..</standardDisclaimerOfExternalLinks>
  <standardGeneralDisclaimer>...</standardGeneralDisclaimer>
  <standardPrivacyPolicy>...</standardPrivacyPolicy>
  <startHeadHtml5>...</startHeadHtml5>
  <startBodyHtml5>...</startBodyHtml5>
  <theShortDescriptionHtml>...</theShortDescriptionHtml>
  <endBodyHtml5>...</endBodyHtml5>

  <user username="..." password="..." roles="..." />

  <dataset>...</dataset>
</erddapDatasets>
```

As an example, let's assume we want to configure a dataset from a NetCDF file whose header is:

```
netcdf ADRIACLIM2_1d_19931030_grid_U {
dimensions:
    time_counter = UNLIMITED ; // (1 currently)
```

```
    bnds = 2 ;
    lon = 432 ;
    lat = 331 ;
    depthu = 120 ;
variables:
    double time_counter(time_counter) ;
       time_counter:standard_name = "time" ;
       time_counter:long_name = "Time axis" ;
       time_counter:bounds = "time_counter_bnds" ;
       time_counter:units = "seconds since 1900-01-01 00:00:00" ;
       time_counter:calendar = "gregorian" ;
       time_counter:axis = "T" ;
    double time_counter_bnds(time_counter, bnds) ;
    float lon(lon) ;
       lon:standard_name = "longitude" ;
       lon:long_name = "Longitude" ;
       lon:units = "degrees_east" ;
       lon:axis = "X" ;
    float lat(lat) ;
       lat:standard_name = "latitude" ;
       lat:long_name = "Latitude" ;
       lat:units = "degrees_north" ;
       lat:axis = "Y" ;
    float depthu(depthu) ;
       depthu:long_name = "Vertical U levels" ;
       depthu:units = "m" ;
       depthu:positive = "down" ;
       depthu:axis = "Z" ;
       depthu:bounds = "depthu_bnds" ;
       depthu:name = "depthu" ;
    float depthu_bnds(depthu, bnds) ;
    float sozotaux(time_counter, lat, lon) ;
       sozotaux:standard_name = "surface_downward_x_stress" ;
```

```
        sozotaux:long_name = "Surface Downward X Stress" ;
        sozotaux:units = "N/m2" ;
        sozotaux:_FillValue = 1.e+20f ;
        sozotaux:missing_value = 1.e+20f ;
        sozotaux:online_operation = "average" ;
        sozotaux:interval_operation = "120 s" ;
        sozotaux:interval_write = "1 d" ;
        sozotaux:cell_methods = "time: mean (interval: 120 s)" ;
    float uocetr_eff(time_counter, depthu, lat, lon) ;
        uocetr_eff:standard_name = "ocean_volume_x_transport" ;
        uocetr_eff:long_name = "Effective ocean transport along i-axis" ;
        uocetr_eff:units = "m3/s" ;
        uocetr_eff:_FillValue = 1.e+20f ;
        uocetr_eff:missing_value = 1.e+20f ;
        uocetr_eff:online_operation = "average" ;
        uocetr_eff:interval_operation = "120 s" ;
        uocetr_eff:interval_write = "1 d" ;
        uocetr_eff:cell_methods = "time: mean (interval: 120 s)" ;
    float vozocrtx(time_counter, depthu, lat, lon) ;
        vozocrtx:standard_name = "sea_water_x_velocity" ;
        vozocrtx:long_name = "Sea Water X Velocity" ;
        vozocrtx:units = "m/s" ;
        vozocrtx:_FillValue = 1.e+20f ;
        vozocrtx:missing_value = 1.e+20f ;
        vozocrtx:online_operation = "average" ;
        vozocrtx:interval_operation = "120 s" ;
        vozocrtx:interval_write = "1 d" ;
        vozocrtx:cell_methods = "time: mean (interval: 120 s)" ;

// global attributes:
// (...)
```

The NetCDF file comes from the AdriaClim RESM NEMO dataset (historical, daily).

There are some variables that depend on 3 dimensions (time, latitude, longitude), and others that depend on 4 dimensions (time, latitude, longitude, depth). As a consequence, we must set up two different ERDDAP datasets, because in EDDGrid datasets, all data variables must use all of the axis variables. So if a source dataset has some variables with one set of dimensions, and other variables with a different set of dimensions, we will have to make two datasets.

ERDDAP provides a command line tool for generating a draft of the XML code to be added in the configuration file, `GenerateDatasetsXml.sh`. Usually the generated code is accurate enough to be used "as-is", sometimes on the other hand a few additional information must be added: keywords, metadata, the institution, the variable units and so on. It also depends on the amount of metadata already present in the data files, typically NetCDF ones.

The tool is interactive (but it can be launched in a non-interactive way, too): once executed, it first asks to specify the `EDDType` (ERDDAP Dataset Type) of the dataset. It can be a regular type (`EDDGridFromNcFiles`, `EDDTableFromAsciiFiles`, etc.) or a special/pseudo dataset type (`EDDGridFromDap`, `EDDGridFromErddap`, etc.): for a complete list see deliverable 4.3.1 and the ERDDAP documentation.

The script then asks a series of questions specific to the selected EDDType. The questions gather the information needed for ERDDAP to access the dataset's source.

To generate the XML configuration for the dataset related to the variables depending on 3 dimensions, we execute the tool as follows (via Docker container as well):

```
docker run --rm -it \
  -v /srv/erddap/data:/erddapData \
  -v /srv/erddap/datasources:/datasources \
  axiom/docker-erddap:2.18 \
  bash -c "cd webapps/erddap/WEB-INF/ && bash GenerateDatasetsXml.sh"
```

We will be prompted for the following options:

```
Which EDDType (default="EDDGridFromDap")
? EDDGridFromNcFiles
Parent directory (default="")
? /datasources
File name regex (e.g., ".*\.nc") (default="")
? .*_U\.nc
```

```
Full file name of one file (or leave empty to use first matching fileName) (default="")

?

Group (without trailing slash) (or "" for all/any or "[root]" for just the root group) (default="")

?

DimensionsCSV (or "" for default) (default="")

? time_counter,lat,lon

ReloadEveryNMinutes (e.g., 10080) (default="")

?

cacheFromUrl (default="")

?
```

After a while, the tool will produce the desired output, along with other useful information (omitted for readability):

```
<dataset  type="EDDGridFromNcFiles"  datasetID="datasources_57b6_6cb0_1f21" active="true">

      <reloadEveryNMinutes>10080</reloadEveryNMinutes>

      <updateEveryNMillis>10000</updateEveryNMillis>

      <fileDir>/datasources/</fileDir>

      <fileNameRegex>.*_U\.nc</fileNameRegex>

      <recursive>true</recursive>

      <pathRegex>.*</pathRegex>

      <metadataFrom>last</metadataFrom>

      <matchAxisNDigits>20</matchAxisNDigits>

      <fileTableInMemory>false</fileTableInMemory>

      <!-- sourceAttributes>

      (...)

      </sourceAttributes -->

      <addAttributes>

      <att name="_NCProperties">null</att>

      <att name="cdm_data_type">Grid</att>

      <att name="Conventions">CF-1.6, COARDS, ACDD-1.3</att>
```

```
<att name="history_of_appended_files">(...)</att>
<att name="infoUrl">???</att>
<att name="institution">???</att>
<att name="keywords">atmosphere, atmospheric, counter, data, downward, earth, Earth Science &gt; Atmosphere &gt; Atmospheric Winds &gt; Wind Stress, grid, ocean, oceanography, physical, physical oceanography, science, sozotaux, stress, surface, surface_downward_x_stress, time, time_counter, u, wind, winds</att>
<att name="keywords_vocabulary">GCMD Science Keywords</att>
<att name="license">[standard]</att>
<att name="NCO">netCDF Operators version 4.9.3 (Homepage = http://nco.sf.net, Code = https://github.com/nco/nco)</att>
<att name="standard_name_vocabulary">CF Standard Name Table v70</att>
<att name="summary">ocean U grid variables. Data from a local source.</att>
<att name="uuid">null</att>
</addAttributes>
<axisVariable>
<sourceName>time_counter</sourceName>
<destinationName>time</destinationName>
<!-- sourceAttributes>
      <att name="_ChunkSizes" type="int">512</att>
      <att name="axis">T</att>
      <att name="bounds">time_counter_bnds</att>
      <att name="calendar">gregorian</att>
      <att name="long_name">Time axis</att>
      <att name="standard_name">time</att>
      <att name="units">seconds since 1900-01-01 00:00:00</att>
</sourceAttributes -->
<addAttributes>
      <att name="_ChunkSizes">null</att>
      <att name="bounds">null</att>
      <att name="ioos_category">Time
      <att name="source_name">time_counter</att>
```

```
        <att name="units">seconds since 1900-01-01T00:00:00Z</att>
    </addAttributes>
    </axisVariable>
    <axisVariable>
    <sourceName>lat</sourceName>
    <destinationName>latitude</destinationName>
    <!-- sourceAttributes>
        <att name="axis">Y</att>
        <att name="long_name">Latitude</att>
        <att name="standard_name">latitude</att>
        <att name="units">degrees_north</att>
    </sourceAttributes -->
    <addAttributes>
        <att name="ioos_category">Location</att>
    </addAttributes>
    </axisVariable>
    <axisVariable>
    <sourceName>lon</sourceName>
    <destinationName>longitude</destinationName>
    <!-- sourceAttributes>
        <att name="axis">X</att>
        <att name="long_name">Longitude</att>
        <att name="standard_name">longitude</att>
        <att name="units">degrees_east</att>
    </sourceAttributes -->
    <addAttributes>
        <att name="ioos_category">Location</att>
    </addAttributes>
    </axisVariable>
    <dataVariable>
    <sourceName>sozotaux</sourceName>
    <destinationName>sozotaux</destinationName>
    <dataType>float</dataType>
```

```
        <!-- sourceAttributes>
                <att name="_ChunkSizes" type="intList">1 331 432</att>
                <att name="_FillValue" type="float">1.0E20</att>
                <att name="cell_methods">time: mean (interval: 120 s)</att>
                <att name="interval_operation">120 s</att>
                <att name="interval_write">1 d</att>
                <att name="long_name">Surface Downward X Stress</att>
                <att name="missing_value" type="float">1.0E20</att>
                <att name="online_operation">average</att>
                <att name="standard_name">surface_downward_x_stress</att>
                <att name="units">N/m2</att>
        </sourceAttributes -->
        <addAttributes>
                <att name="_ChunkSizes">null</att>
                <att name="colorBarMaximum" type="double">0.5</att>
                <att name="colorBarMinimum" type="double">-0.5</att>
                <att name="ioos_category">Physical Oceanography</att>
        </addAttributes>
        </dataVariable>
</dataset>
```

This chunk of XML code (`<dataset>...</dataset>`) can be then added to `datasets.xml` configuration file.

The file can be changed (e.g., add datasets, change metadata, etc.) while ERDDAP is running. The changes will be detected the next time the file is read (as often as `loadDatasetsMinMinutes`).

Regarding the datasets on composite node, they are configured as follows, assuming that we want to add ("link") the ones hosted on CMCC node (`https://erddap.cmcc-opa.eu/erddap`):

```
Which EDDType (default="")
? EDDTableFromErddap
URL of remote ERDDAP (ending in ("/erddap") (default="")
```

```
? https://erddap.cmcc-opa.eu/erddap

Keep original datasetIDs (true|false) (default="true")

?
```

The generated XML code looks like

```
(...)

<dataset type="EDDTableFromErddap" datasetID="datasources_7bbb_3e06_ef2e"
active="true">

<sourceUrl>https://erddap.cmcc-
opa.eu/erddap/tabledap/datasources_7bbb_3e06_ef2e</sourceUrl>


</dataset>

<dataset type="EDDTableFromErddap"
datasetID="indicators_csv_2534_2591_3010" active="true">

<sourceUrl>https://erddap.cmcc-
opa.eu/erddap/tabledap/indicators_csv_2534_2591_3010</sourceUrl>


</dataset>

<dataset type="EDDTableFromErddap"
datasetID="indicators_csv_afd8_8217_3d9b" active="true">

  <sourceUrl>https://erddap.cmcc-
opa.eu/erddap/tabledap/indicators_csv_afd8_8217_3d9b</sourceUrl>


</dataset>

<dataset type="EDDTableFromErddap"
datasetID="indicators_wsdi_aba0_0062_8939" active="true">

<sourceUrl>https://erddap.cmcc-
opa.eu/erddap/tabledap/indicators_wsdi_aba0_0062_8939</sourceUrl>

</dataset>

(...)
```

In general, some constraints have to be considered before adding datasets on ERDDAP.

Since the GIS world is often map oriented, GIS programs usually offer support for reprojecting the data, i.e., plotting the data on a map with a different projection.

Currently, ERDDAP does not have tools to reproject data. An external tool should be used to make a variant of the dataset, where data has been reprojected from its original form onto a rectangular (latitude longitude) array suitable for ERDDAP.

For NetCDF files, the variables representing the dimensions (time, lat, lon, depth) must be uni-dimensional. For example, if latitude and longitude are

```
dimensions:
      (...)
      lon = 432 ;
      lat = 331 ;
      (...)
```

then the corresponding variables must be

```
float lon(lon) ;
      lon:standard_name = "longitude" ;
      lon:long_name = "Longitude" ;
      lon:units = "degrees_east" ;
      lon:axis = "X" ;
float lat(lat) ;
      lat:standard_name = "latitude" ;
      lat:long_name = "Latitude" ;
      lat:units = "degrees_north" ;
      lat:axis = "Y" ;
```

In ERDDAP, the longitude, latitude, altitude (or depth), and time (LLAT) variable `destinationNames` are special. LLAT variables are made known to ERDDAP if the axis variable's (for EDDGrid datasets) or data variable's (for EDDTable datasets) destinationNames is "longitude", "latitude", "altitude", "depth", or "time".

For CSV files, the optimal solution is to have a column for each dimension (time, latitude, longitude and optionally depth), and one or more columns for actual data. The rows should be prefixed by an “header row” containing the names of the corresponding columns.

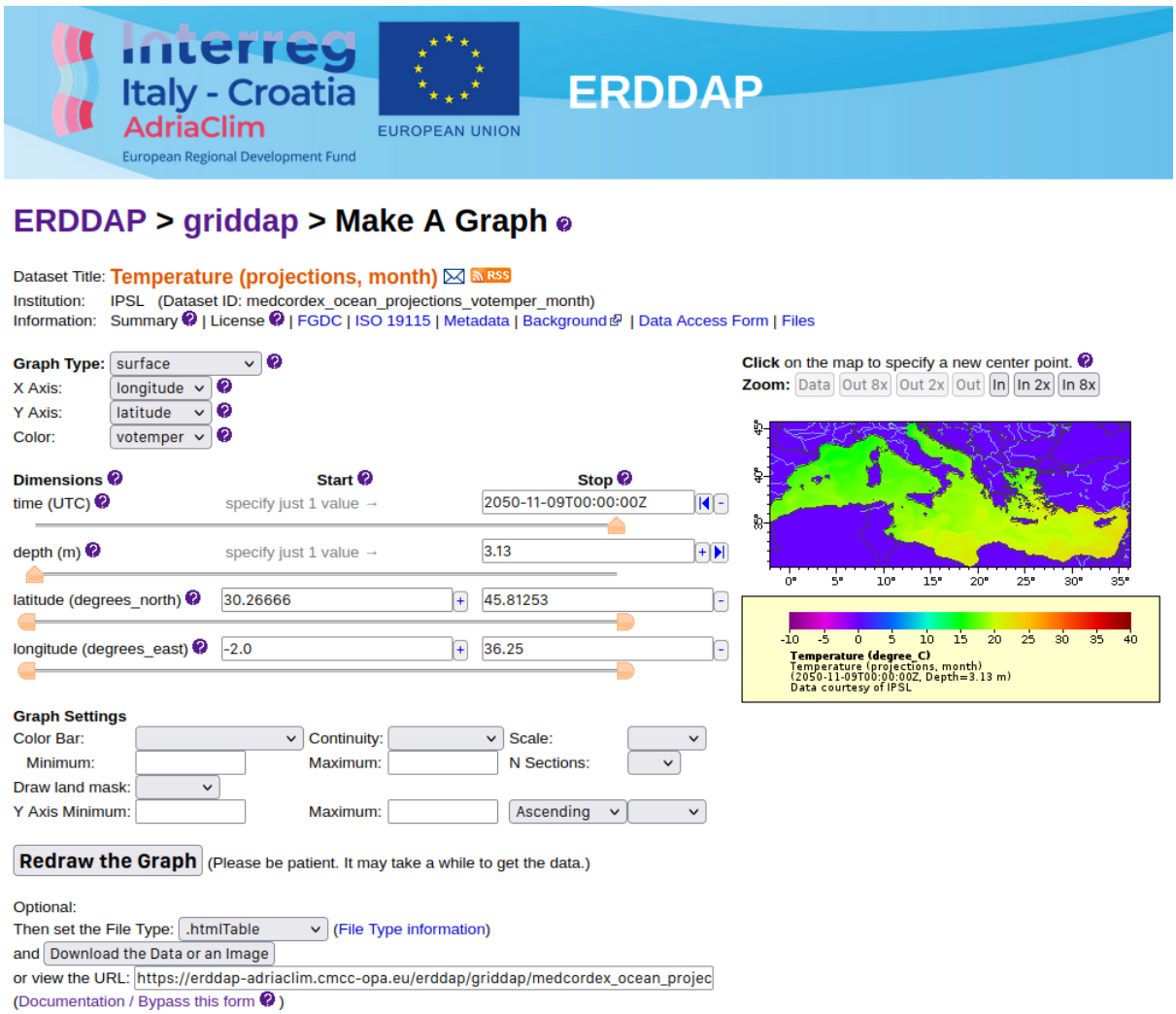
- 5. ERDDAP end-user usage examples

- Form to request graphs and maps

The “griddap” and “tabledap” components allow the user to request a data subset, graph, or map from gridded and tabular datasets, via a specially formed URL.

The following example has been taken from the AdriaClim composite node, the dataset is “Temperature (projections, month)”, from MedCordex.

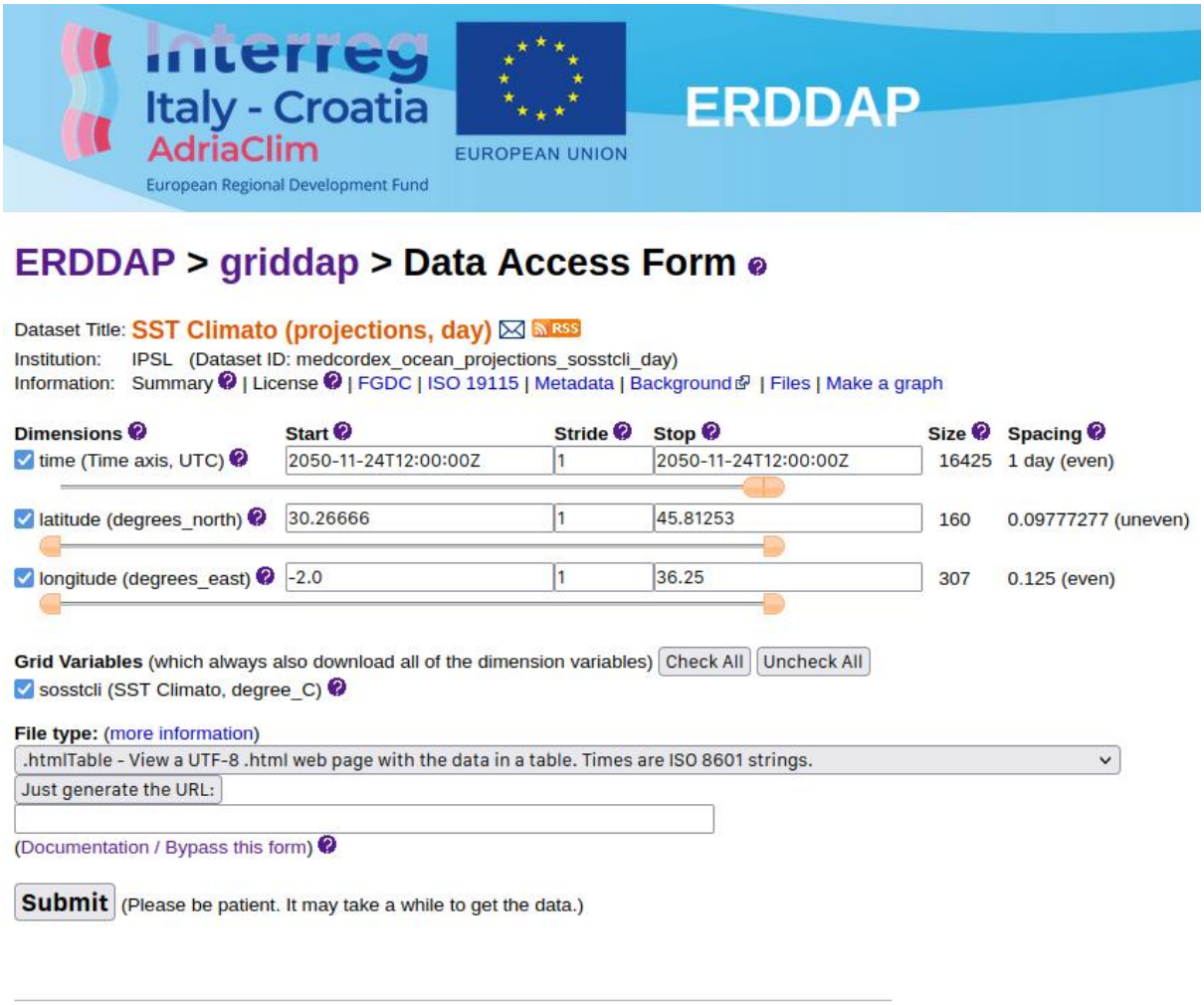
It is possible to select a timestep, a depth, a latitude/longitude range. The resulting data can be visualized on the web page, or downloaded in different formats (CSV, NetCDF, PDF, PNG and more).



- Data Access Form

Data Access Forms (*datasetID*.html) are used for requesting a subset of data in lots of different file formats.

In this example, a user can subset data coming from the MedCordex dataset “SST Climato (projections, day)”.



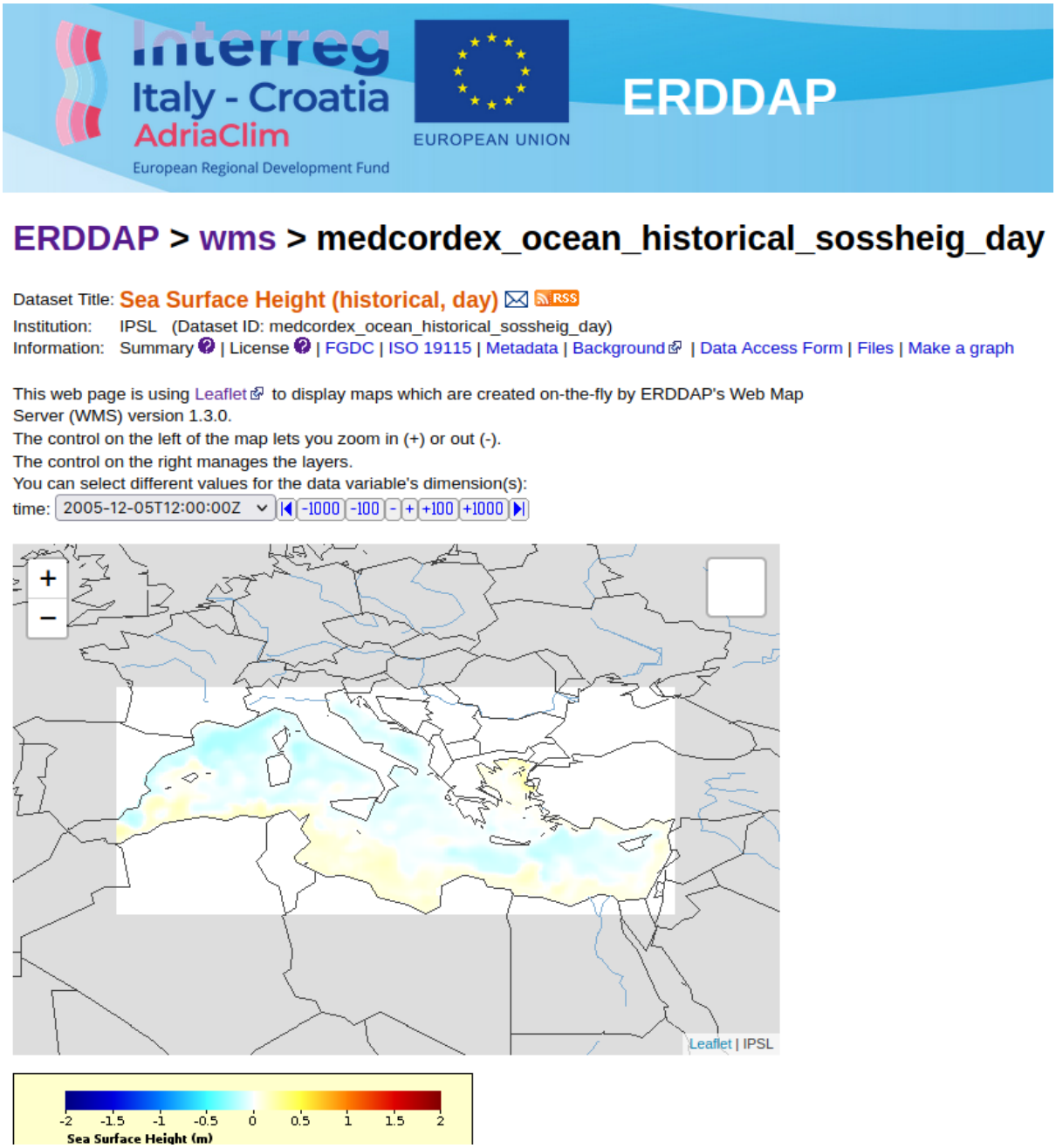
- Web Map Service (WMS)

ERDDAP's Web Map Service (WMS) lets a user request an image with data plotted on a map. WMS is an Open Geospatial Consortium (OGC) and ISO (19128) standard for "the creation and display of registered and superimposed map-like views of information that come simultaneously from multiple remote and heterogeneous sources".

In the next picture, the web page of the WMS service is shown, the dataset is “Sea Surface Height (historical, day)”, from MedCordex.

The WMS service is usually accessed “machine-to-machine” or through a WMS client, starting from the Capabilities XML file: in this example the URL is

`https://erddap-adriaclim.cmcc-opa.eu/erddap/wms/medcordex_ocean_historical_sossheig_day/request?service=WMS&request=GetCapabilities&version=1.3.0`



- RESTful access

For every ERDDAP web page with a form that an end-user as a human with a browser can use, there is a corresponding ERDDAP web service that is designed to be easy for computer programs and scripts to use.

Requests for user-interface information from ERDDAP (for example, search results) use the web's universal standard for requests: URLs sent via HTTP GET. A user can form these URLs by hand and enter them in the address text field of the browser, for example

https://erddap-adriaclim.cmcc-opa.eu/erddap/search/index.json?page=1&itemsPerPage=1000&searchFor=temperature

- Advanced search

ERDDAP offers several ways to search for interesting datasets, for example full text search, search by category (also known as *faceted* search), and Advanced Search.

Advanced Search combines all of the search techniques and adds searches for datasets that have data within longitude, latitude, and time ranges, so a user can search for datasets based on many different criteria simultaneously.

The following picture shows the AdriaClim's ERDDAP main page:

Interreg Italy - Croatia AdriaClim
European Regional Development Fund
EUROPEAN UNION

ERDDAP

ERDDAP > Advanced Search

Directions: Specify as many or as few search criteria as you want, then click `Search`.
Only the datasets that match **all** of the search criteria will appear in the results.

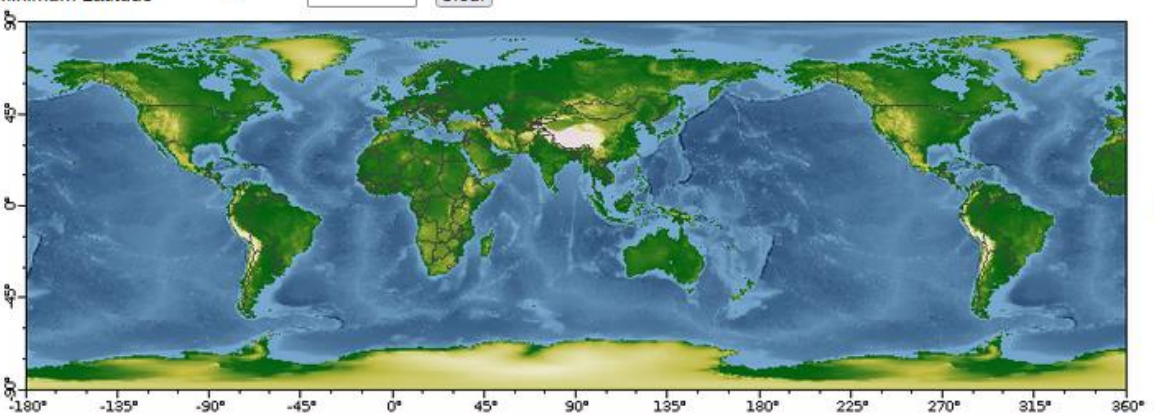
Full Text Search for Datasets

Search for Datasets by Category

protocol = (ANY)
cdm_data_type = (ANY)
institution = (ANY)
ioos_category = (ANY)
keywords = (ANY)
long_name = (ANY)
standard_name = (ANY)
variableName = (ANY)

Search for Datasets that have Data within Longitude, Latitude, and Time Ranges

Maximum Latitude =
Min and Max Longitude =
Minimum Latitude = Clear



Minimum Time =
Maximum Time =

Search