



Using OpenHI.net

Contents

1	Introduction	2
2	Using OpenHI GeoData in QGIS	3
2.1	Downloading Geodata.....	3
2.2	Example: Using available OpenHI geodata layers in Qgis	3
3	Managing a station	9
3.1	Basic Configuration/Capabilities.....	9
3.2	Uploading and timeseries processing	11
4	Διαχείριση ενημερώσεων σταθμών	14
4.1	Εισαγωγή λογαριασμών αλληλογραφίας για λήψη ενημερώσεων.....	14
4.2	Εισαγωγή ορίων για ειδοποίηση	15
5	Οδηγίες χρήσης loggertodb – συστήματος σύνδεσης με τη βάση δεδομένων Ehydrois του Openhi.net.....	16
5.1	Usage.....	16
5.2	Quick start.....	16
5.2.1	Installation.....	16
5.2.2	How to run it	16
5.2.3	Authentication.....	17
5.2.4	Configuration file examples.....	17
5.2.5	Running automatically	18
5.3	Configuration file reference.....	19
5.3.1	General parameters.....	19
5.3.2	File parameters	19
5.4	Supported formats	21
5.5	Daylight saving time	23



1 Introduction

Open Hydrosystem Information Network, (<https://openhi.net/en/>) is a national integrated information infrastructure for the surface water bodies of Greece, with [free access](#) to relative hydrological environmental and geographical data.

The main operations are described in this manual. In **Section 2** the extraction of OpenHI Layers is shown along with its usage in in a typical GIS. **Section 3 and 4 describe** capabilities of managing a station with its timeseries and warnings of its own data for the respective administrator. In **Section 5** the association of station with the OpenHI for automatic data update is described.



2 Using OpenHI GeoData in QGIS

OpenHI offers free geodata with respect to hydrological bodies along with its associated measurement network installed on terrain, lake and rivers.

The layers of available geodata in Openhi.net which adopt the [Inspire](#), directive are :

- Watercourse
- Main watercourse
- StandingWater)
- RiverBasin
- DrainageBasin
- StationBasin)
- WatercourseLink
- hydroNode

2.1 Downloading Geodata

Data are available through Web Feature Services ([WFS](#)), wrt Open Geospatial Consortium ([OGC](#)) from the following address:

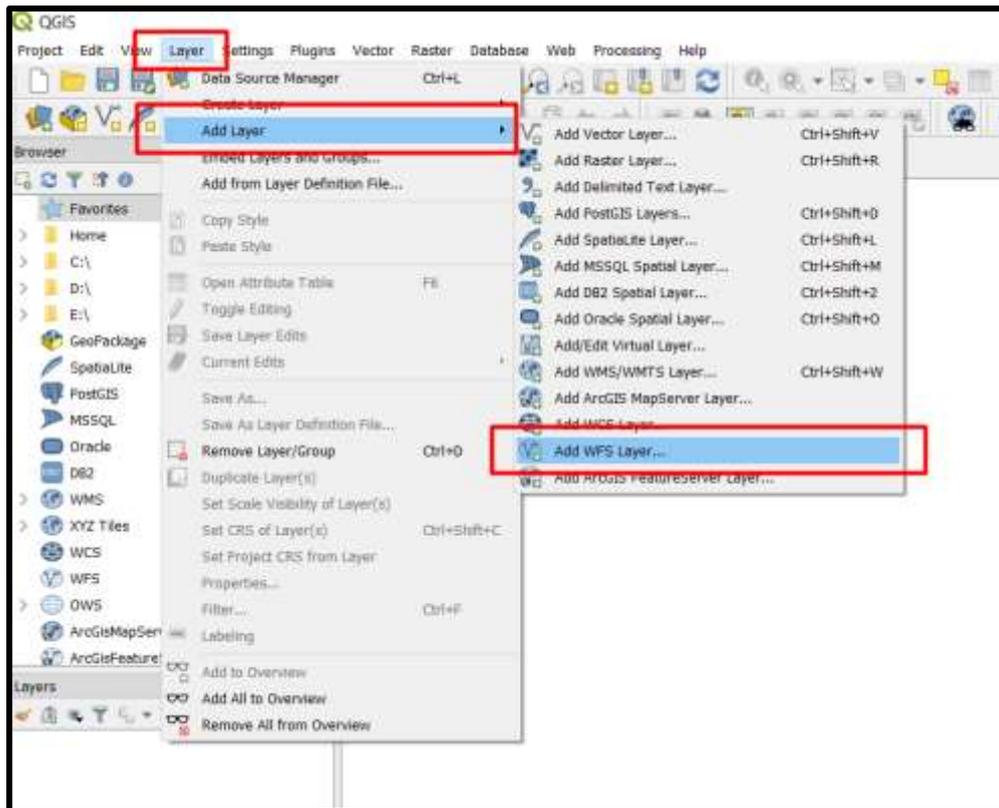
<https://system.openhi.net/cgi-bin/mapserv?map=/opt/enhydris-openhi/enhydris-openhigis/mapserver/openhigis.map>

2.2 Example: Using available OpenHI geodata layers in Qgis

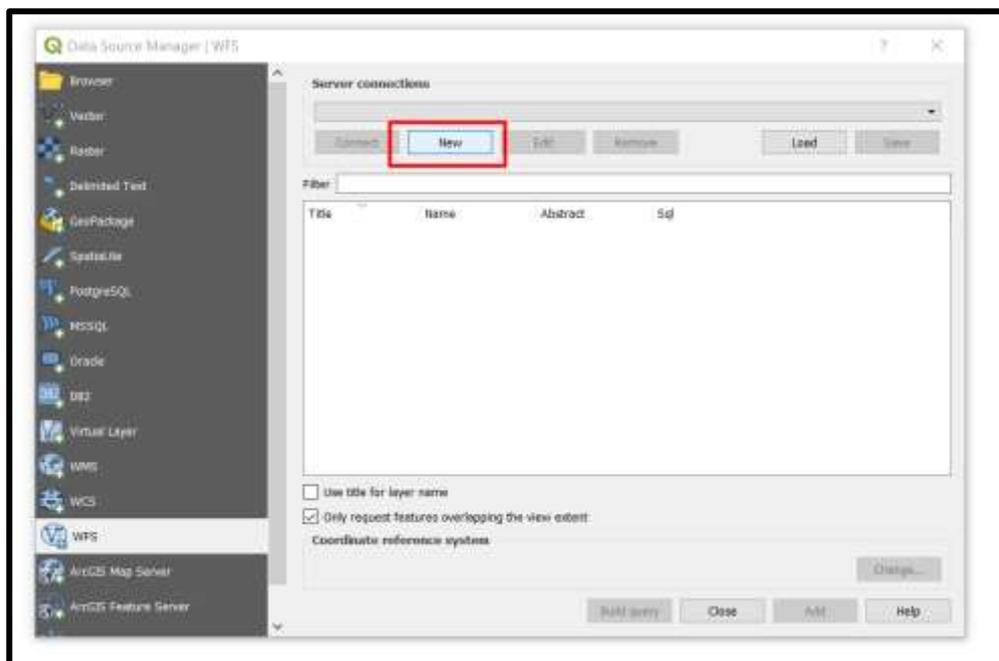
We are going to present the necessary steps to insert the OpenHI layers in the opens source [QGIS](#) (Quantum Geographic Information System).

The necessary steps are:

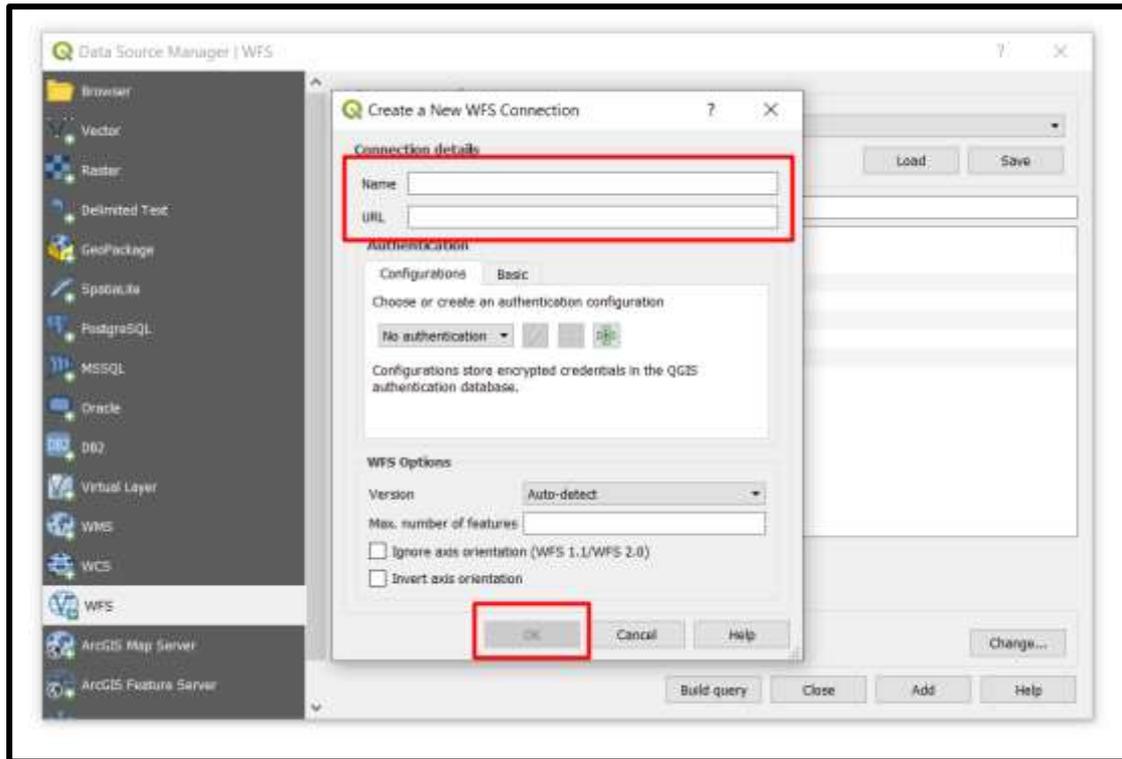
Layer > Add Layer > Add WFS layer:



Select new:



Add Name › Define address URL for the WFS › ok:

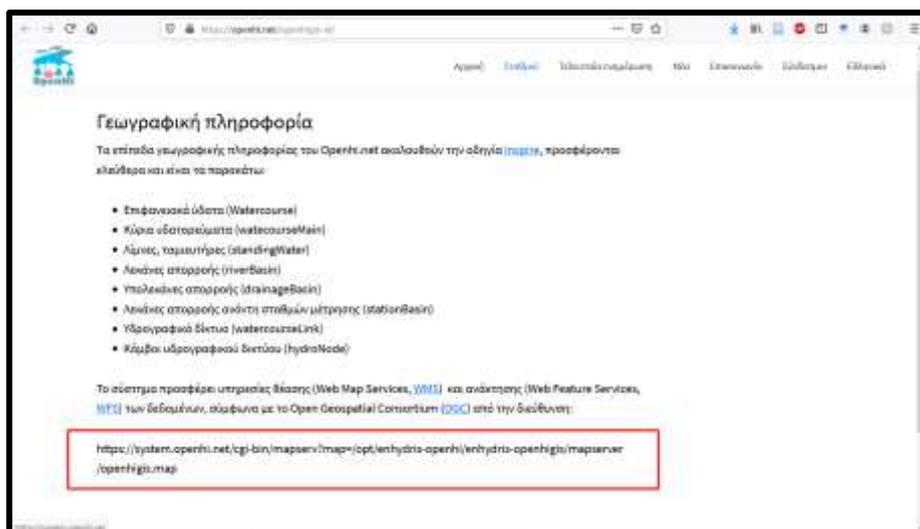


Note:

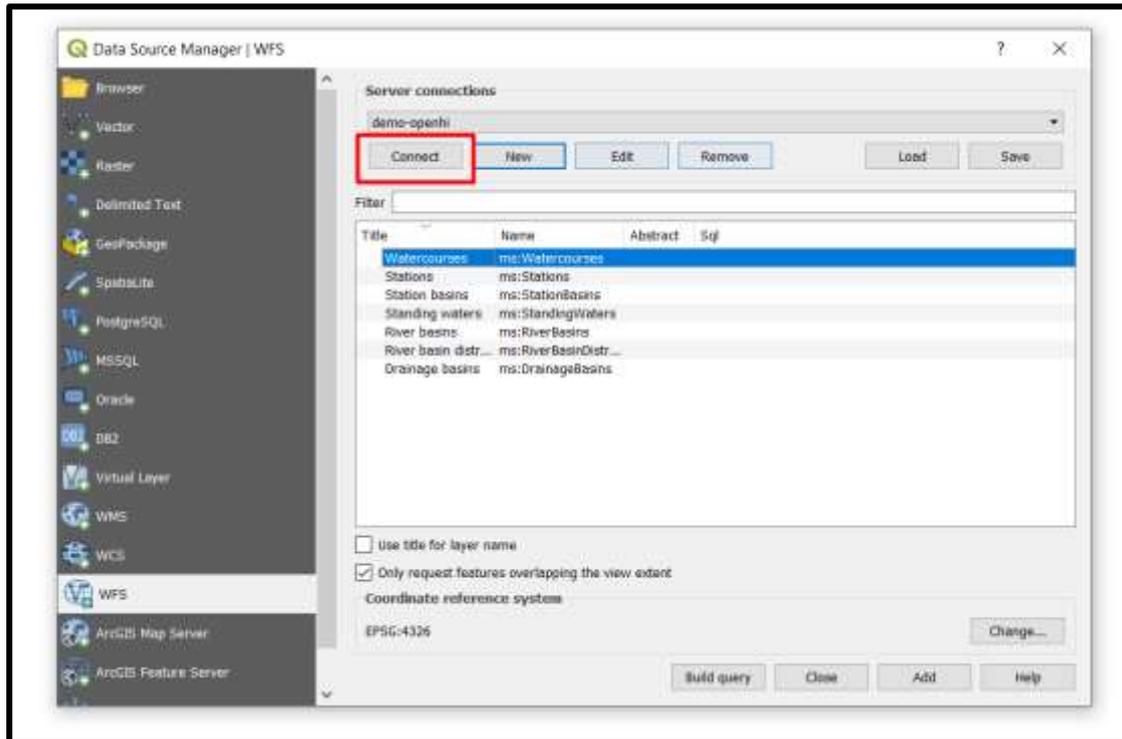
The WFS link of OpenHi is :

<https://system.openhi.net/cgi-bin/mapserv?map=/opt/enhydris-openhi/enhydris-openhigis/mapserv/openhigis.map>

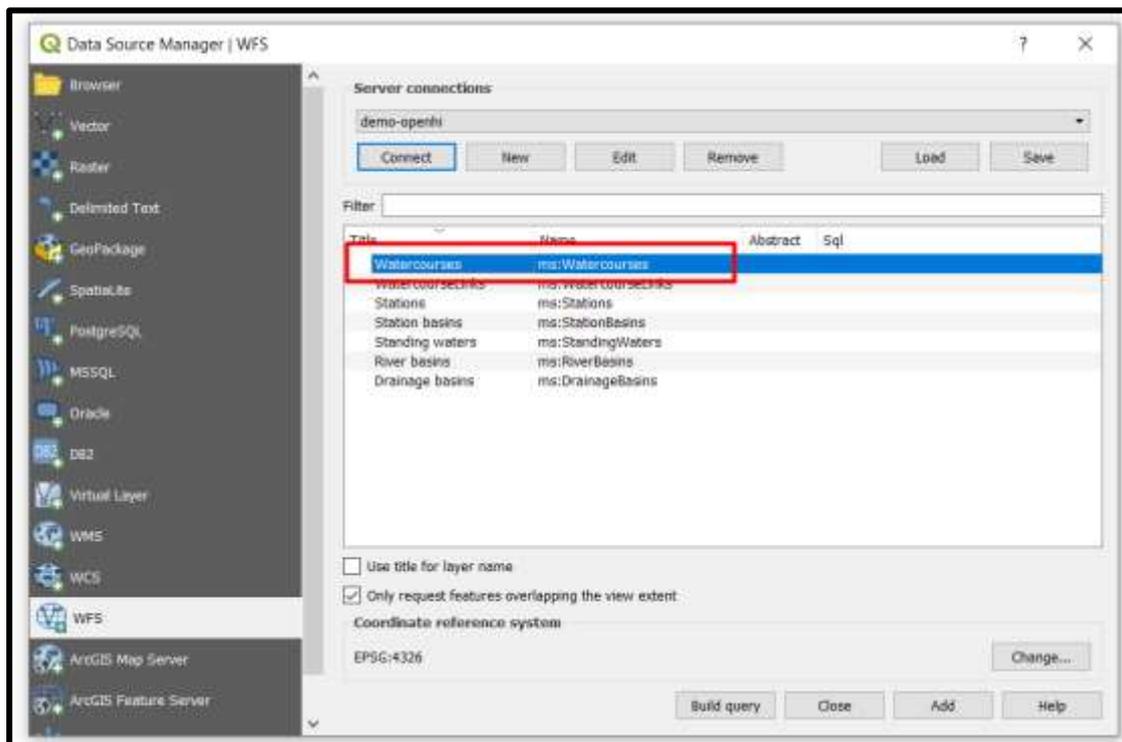
as well as from <https://openhi.net/openhigis-el/>, as shown in the following image:



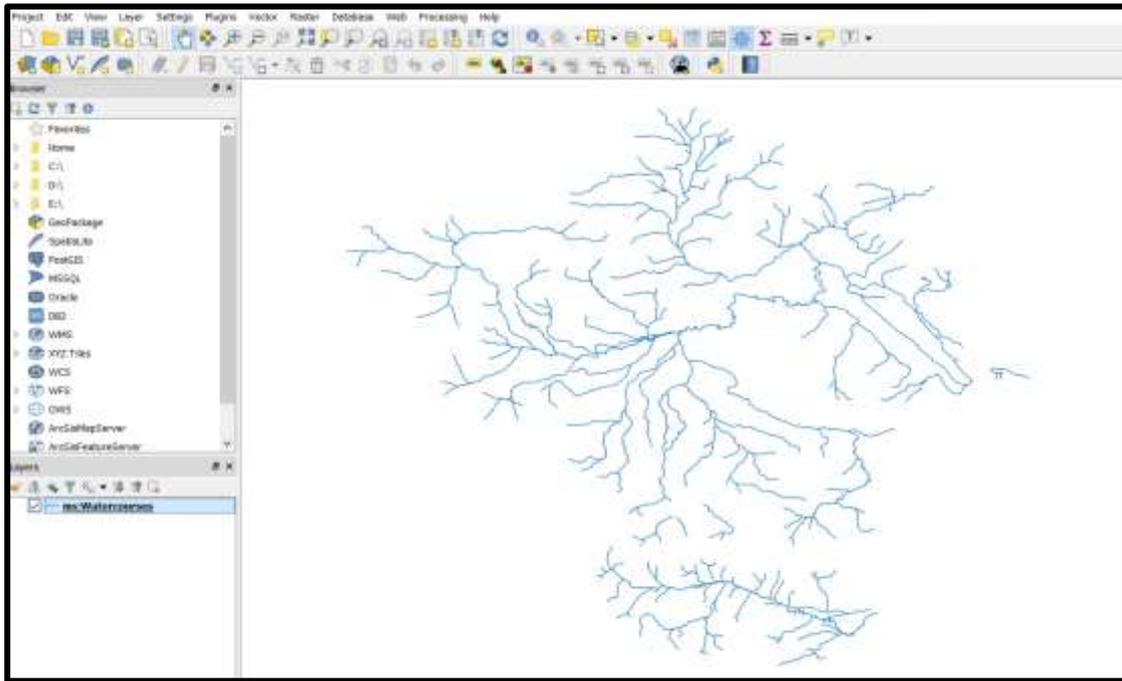
Select connect:



For every layer we click on the name › add:

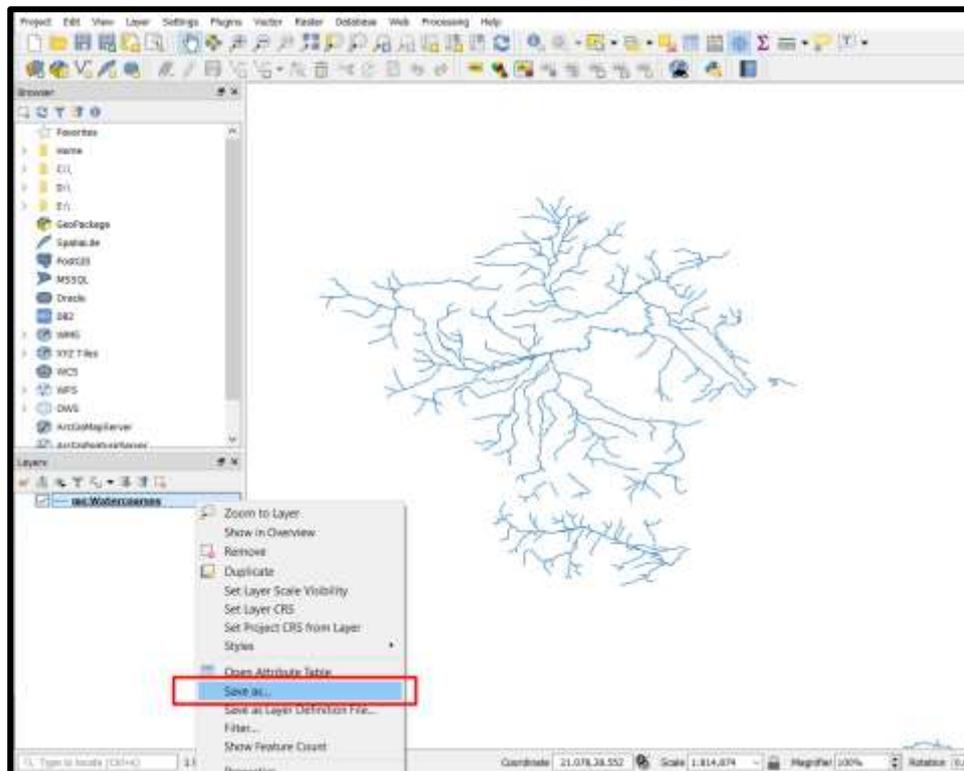


Consequently, on the basic canvas the following image appears:



The retrieval of data is performed:

Right click on a layer › save as:

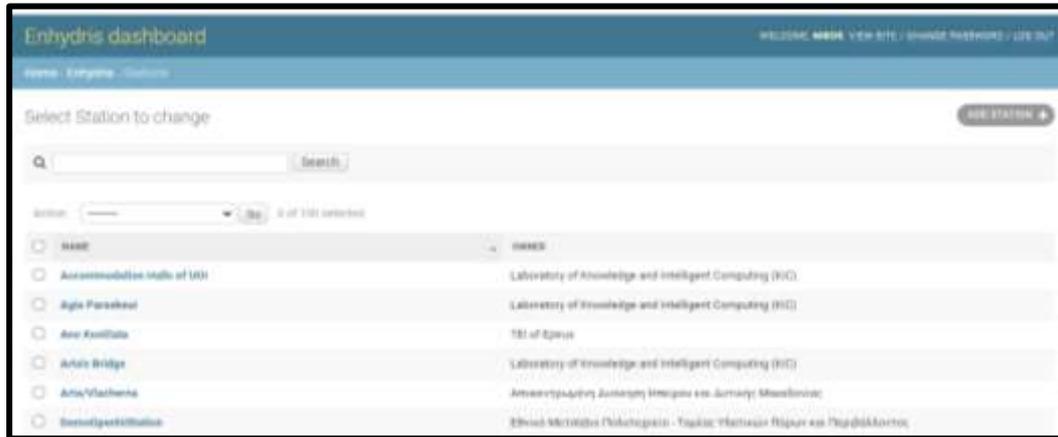


Select the desired format

3 Managing a station

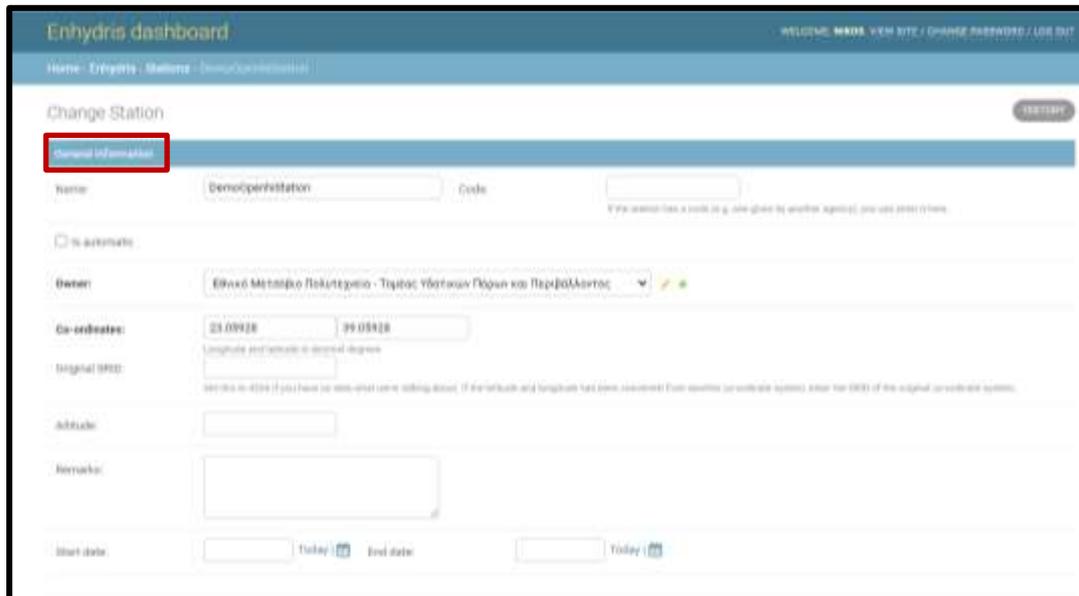
3.1 Basic Configuration/Capabilities

Home > Enhydris > Stations – Station Selection σταθμού



Every Station has the following info:

1. General information



2. Permissions





3. Files

A screenshot of the 'FILES' form in the OpenHi interface. The form has a blue header bar with the title 'FILES' in white. Below the header, there are several input fields: 'Description' with a text box, 'Date' with a date picker, and 'Content' with a 'Choose file' button and the text 'No file chosen'. At the bottom, there is a large text area for 'Remarks'. The 'FILES' title in the header is highlighted with a red box.

4. Log entries

A screenshot of the 'LOG ENTRIES' form in the OpenHi interface. The form has a blue header bar with the title 'LOG ENTRIES' in white. Below the header, there are several input fields: 'User' with a text box, 'Date' with a date picker, 'Type' with a dropdown menu, and 'Report' with a large text area. The 'LOG ENTRIES' title in the header is highlighted with a red box.

5. Time series groups

A screenshot of the 'TIME SERIES GROUPS' form in the OpenHi interface. The form has a blue header bar with the title 'TIME SERIES GROUPS' in white. Below the header, there is a list of time series groups, each with a 'SHOW' button. The groups are: 'Time series group: Level', 'Time series group: KE', and 'Another Time Series Group'. At the bottom, there are three buttons: 'Delete', 'Save and exit profile', and 'Save and continue editing'. The 'TIME SERIES GROUPS' title in the header is highlighted with a red box.



3.2 Uploading and timeseries processing

In TIMES SERIES GROUPS UI, the station administrator may select from the following options for timeseries:

Timeseries Group: Παροχής (1) metadata, (2) range check, (3) time consistency check (4) timeseries (5) Curve interpolation (6) aggregations	Timeseries group: Discharge METADATA (SHOW) RANGE CHECK (SHOW) TIME CONSISTENCY CHECK (SHOW) TIME SERIES (SHOW) CURVE INTERPOLATIONS (SHOW) AGGREGATIONS (SHOW)
---	--

The following UI appears

1. Metadata

2. range check

With four thresholds (lower, soft lower, soft upper, upper). The values beyond lower or upper are not transferred in the resulting calculated timeseries



3. Time consistency check



Thresholds are “time difference-value” while Symmetric is selected in case increase or decrease check is desired on a given time lag:

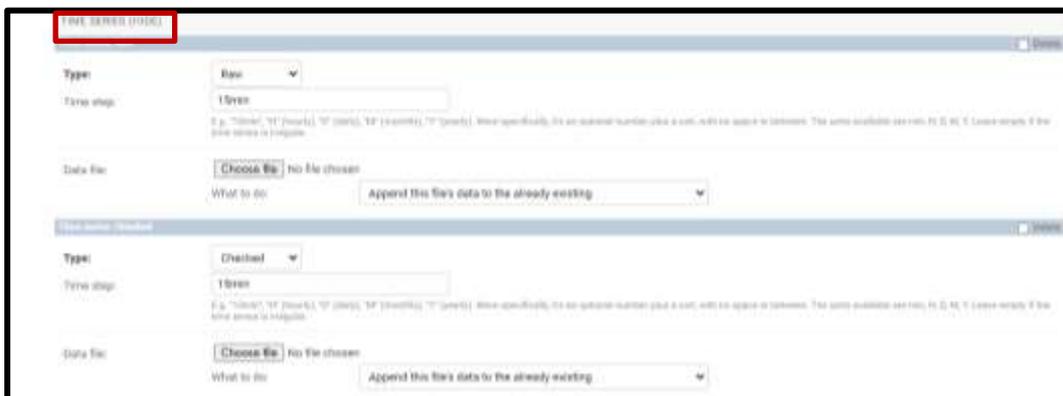
15min 0.1 it will be considered invalid if it higher or lower over 0.1 from the previous 15 min reading/value

30min 0.15 it will be considered invalid if it higher or lower over 0.15 from the previous 15 min reading/value.

1H 0.3 it will be considered invalid if it higher or lower over 0.3 from the previous 1 hour reading/value

4. Timeseries

The date to be imported should adhere to **yyyy-mm-dd hh:mm** format(i.e. 2014-12-25 23:00) in CSV (comma delimited) files (attention: when opening the file with a notepad++ tool.



Five different timeseries types:

Raw, checked, processed, regularized, aggregated

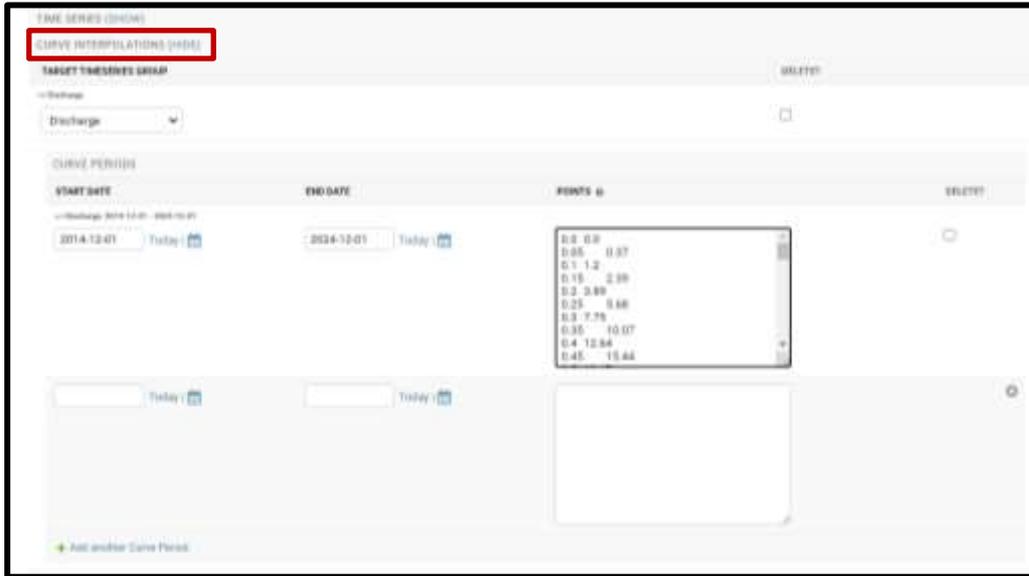
Corrected Timeseries

- A. Origin data are located in the raw timeseries
- B. If checks (range, time consistency) a checked timeseries is generated as shown in images..



- C. If values of check change the checked timeseries is updated **for the newly feed data OLNy**. **Warning:** If new limits are desired to applied for past data readings, the checked timeseries should be deleted and rebuild all the way through.
- D. Checked timeseries can be downloaded.
- E. H checked timeseries is updated from the time instance the checks were enforced.

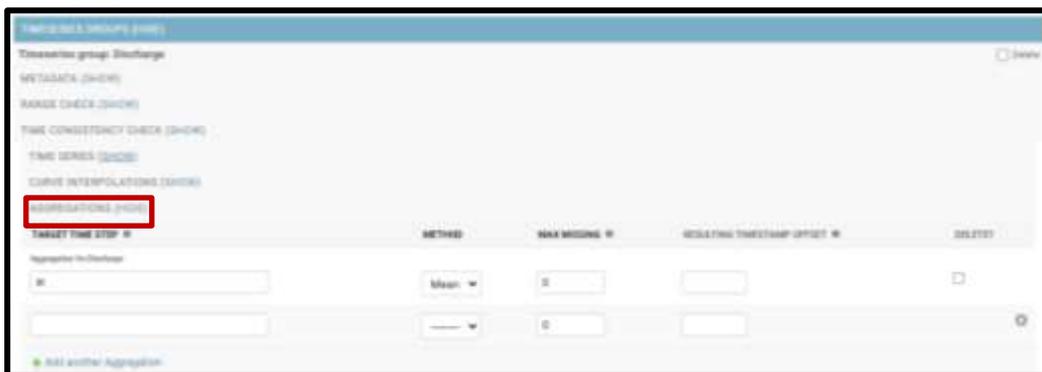
5. Curve interpolations



A stage reading can be converted to supply via the curve interpolation dueing a specified time range.

Curve interpolation is updated **for the newly feed data OLNy**. **Warning:** If new curves are desired to applied for past data readings, the resulting timeseries should be deleted and rebuild all the way through

6. Aggregations





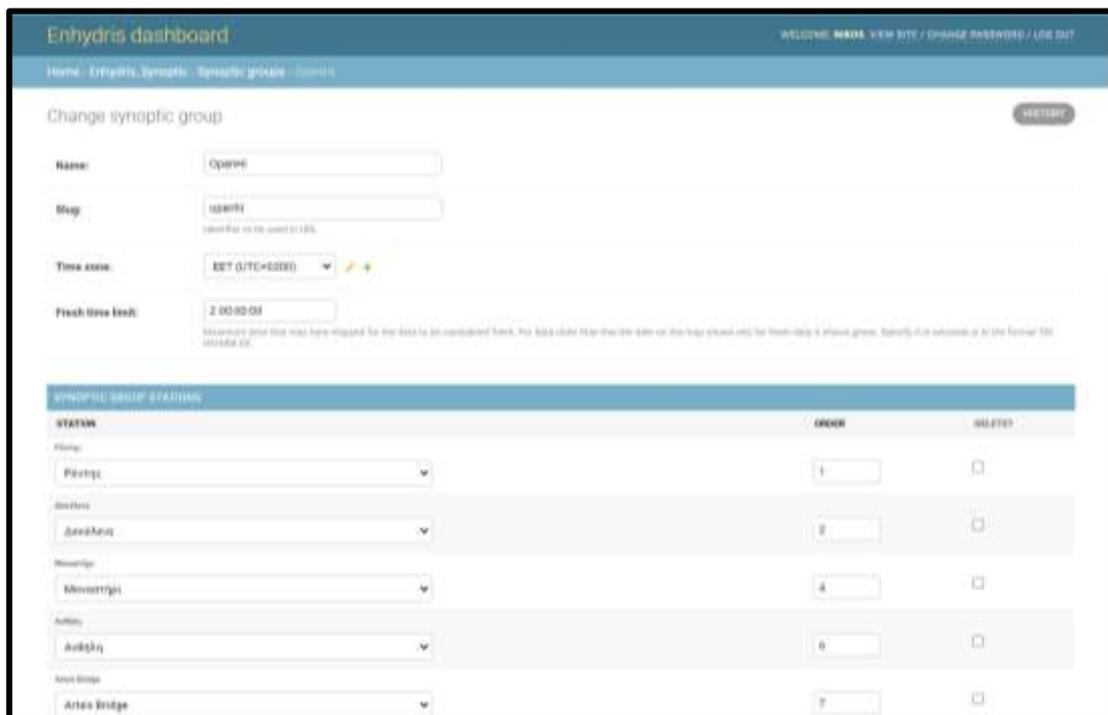
4 Notification Management

4.1 Using mail info for receiving notifications

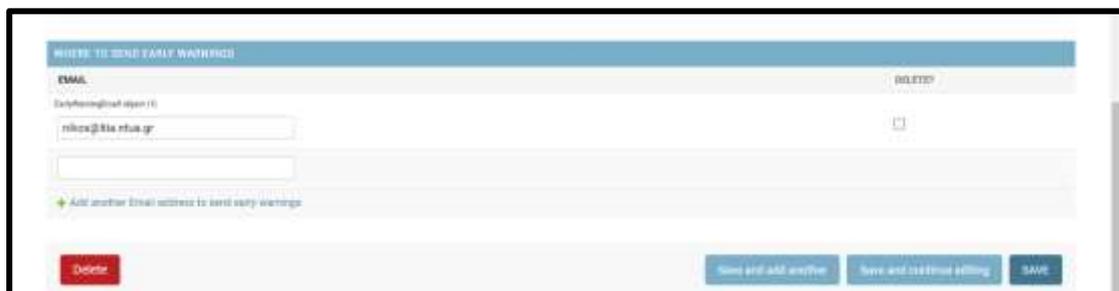
[Home](#) > [Enhydris_Synoptic](#) > [Synoptic groups](#)



Selecting Openhi



At the end of Page: Email data for notification





5 Using loggertodb – for registering a stations using the Enhydris Component of Openhi.net

5.1 Usage

loggertodb reads a data file (or several data files), connects to Enhydris, determines which records in the file are newer than those stored in Enhydris, and appends them. The details of its operation are specified in the configuration file specified on the command line.

5.2 Quick start

5.2.1 Installation

Windows: loggertodb is just a single executable, loggertodb.exe. You download it and run it; there's no installer.

Download loggertodb.exe from <https://github.com/openmeteo/loggertodb/releases/>.

Linux: Simply execute this:

```
pip3 install loggertodb
```

5.2.2 How to run it

First, you need to create a configuration file with a text editor such as vim, emacs, notepad, or whatever. Create such a file and name it, for example, /var/tmp/loggertodb.conf, or, on Windows, something like C:\Users\user\loggertodb.conf, with the following contents (the contents don't matter at this stage, just copy and paste them from below):

```
General]
base_url = https://openmeteo.org/
auth_token = 123456789abcdef0123456789abcdef012345678
loglevel = INFO
```

Then, open a command prompt and give it this command:

Unix/Linux:

```
loggertodb /var/tmp/loggertodb.conf
```

Windows:

```
C:\Program Files\Loggertodb\loggertodb.exe
C:\Users\user\loggertodb.conf
```

(the details may differ; for example, in 64-bit Windows, it may be C:Program Files (x86) instead of C:Program Files.)

If you have done everything correctly, it should show an error message similar to “No stations have been specified”. This means that, apart from the “General” section you have to add more sections to the configuration file.



5.2.3 Authentication

loggertodb needs to logon to Enhydriis, and for this it needs an authentication token to be specified as the value of the `auth_token` parameter. You can get a token at the `/api/auth/login/` URL of Enhydriis, such as <https://openmeteo.org/api/auth/login/>.

5.2.4 Configuration file examples

The following instructs `loggertodb` to use the single data file `zeno.data` and upload its data to `openmeteo.org`; the first field of each line (after the date and time) will be uploaded to time series group 232, the second to 233, and so on. The last field of each line will not be uploaded (symbolized with the 0):

```
[General]
loglevel = WARNING
logfile = /var/log/loggertodb/itiameteo.log
base_url = https://openmeteo.org/
auth_token = 123456789abcdef0123456789abcdef012345678

[NTUA]
station_id = 1334
path = /var/local/openmeteo/logger_data_files/ntua/zeno.data
storage_format = simple
date_format = %y/%m/%d %H:%M:%S
fields =
232,233,247,248,237,238,236,9141,5461,6659,9139,6661,240,6539,6541
,230,234,0
```

The following instructs `loggertodb` to use two data files (one for meteorological station PRASINOS, one for VILIA; these are just labels to make it easy for you to read the file; that are not used anywhere). While reading that each line's fields, the value "NAN" instead of a number will be interpreted as an empty (or missing, or null) value. The `timezone` parameter is used for daylight saving time adjustments (see [Daylight saving time](#)):

```
[General]
loglevel = WARNING
logfile = /var/log/loggertodb/defkalion.log
base_url = https://openmeteo.org/
auth_token = 123456789abcdef0123456789abcdef012345678

[PRASINOS]
station_id = 1345
path =
/var/local/openmeteo/logger_data_files/defkalion/prasino.data
storage_format = simple
date_format = %d/%m/%Y %H:%M:%S
fields = 9180,9182,9184,9178
null = NAN
timezone = Europe/Athens

[VILIA]
station_id = 1347
path = /var/local/openmeteo/logger_data_files/defkalion/vilia.data
storage_format = simple
date_format = %d/%m/%Y %H:%M:%S
fields = 9172,9174,9176,9170
null = NAN
timezone = Europe/Athens
```



The next is very similar to the previous one, but it's for Windows, it uses a star for null values, and the fields in the files are delimited with commas instead of spaces. In addition, the sixth field of each line (after the date and time) is not uploaded:

```
[General]
loglevel = INFO
logfile = C:\a2a\loggertodb-kostilata.log
base_url = https://openmeteo.org/
auth_token = 123456789abcdef0123456789abcdef012345678

[ANO_KOSTILATA]
station_id = 1387
path = C:\a2a\ano_kostilata_20130601.txt
storage_format = simple
delimiter = ,
date_format = %d-%m-%Y %H:%M:%S
fields = 9290,9285,9292,9294,9295,0,9291,9289,9288,9286
null = *
timezone = Europe/Athens

[KATO_KOSTILATA]
station_id = 1388
path = C:\a2a\ano_kostilata_20130601.txt
storage_format = simple
delimiter = ,
date_format = %d-%m-%Y %H:%M:%S
fields = 9279,9274,9281,9283,9284,0,9280,9278,9277,9275
null = *
timezone = Europe/Athens
```

Finally, an example of a configuration that uses the files produced by Davis WeatherLink. In this case, C:\WeatherLink\komboti is the directory that contains the .WLK files (it is necessary to read more below about [WDAT5 units](#) and [the WDAT5 format](#)):

```
[General]
loglevel = INFO
logfile = C:\WeatherLink\komboti\loggertodb.log
base_url = https://openmeteo.org/
auth_token = 123456789abcdef0123456789abcdef012345678

[KOMBOTI]
station_id = 1389
path = C:\WeatherLink\komboti
storage_format = wdat5
outsideTemp = 1256
hiOutsideTemp = 1257
rain = 1652
timezone = Europe/Athens
temperature_unit = F
rain_unit = inch
```

5.2.5 Running automatically

You probably want to have `loggertodb` automatically update the data. To do this, either run it periodically (from `cron` on Unix and `Task Scheduler` on Windows), or, if the software you



use to download the data from the meteorological station has the feature, add `loggertodb` as a trigger.

5.3 Configuration file reference

The configuration file has the format of INI files. There is a `[General]` section with general parameters, and any number of other sections, which we will call “file sections”, each file section referring to one file to be processed; this makes it possible to process many files in a single `loggertodb` execution using a single configuration file and fewer HTTP requests.

5.3.1 General parameters

`loglevel`

Can have the values `ERROR`, `WARNING`, `INFO`, `DEBUG`, indicating the amount of output requested from `loggertodb`. The default is `WARNING`.

`logfile`

The full pathname of a log file. If unspecified, log messages will go to the standard error.

`base_url`

The base url of the Enhydris installation to connect to, such as <https://openmeteo.org/>.

`auth_token`

The token `loggertodb` will use to authenticate with Enhydris. Obviously the user to whom the token corresponds must have write permissions for all time series that will be uploaded.

5.3.2 File parameters

`station_id`

The id of the station.

`path`

The full pathname of the data storage.

`storage_format`

The format of the data storage. See [Supported formats](#).

`fields`

(Not for the `wdat5` format.) A series of comma-separated integers representing the ids of the time series groups to which the data file fields correspond (time series groups are what Enhydris lists as “Data” in the page for a station). A zero indicates that the field is to be ignored. The first number corresponds to the first field after the date (and



possibly other fixed fields depending on data file format, such as the subset identifier) and should be the id of the corresponding time series group, or zero if the field is dummy; the second number corresponds to the second field after the fixed fields, and so on.

Each time series group contains variations of the same time series, such as initial, checked and aggregated. `loggertodb` uploads the data to the “initial” time series of the group. If such a time series does not exist, it is created.

`nfields_to_ignore`

This is used only in the `simple` format; it’s an integer that represents a number of fields before the date and time that should be ignored. The default is zero. If, for example, the date and time are preceded by a record id, set `nfields_to_ignore=1` to ignore the record id.

`subset_identifiers`

Some file formats mix two or more sets of measurements in the same file; for example, there may be ten-minute and hourly measurements in the same file, and for every 6 lines with ten-minute measurements there may be an additional line with hourly measurements (not necessarily the same variables). `loggertodb` processes only one set of lines each time. Such files have one or more additional distinguishing fields in each line, which helps to distinguish which set it is. `subset_identifiers`, if present, is a comma-separated list of identifiers, and will cause `loggertodb` to ignore lines with different subset identifiers. (Which fields are the subset identifiers depends on the data file format.)

`null`

Indicates how null values are represented in the source file. For example, if `null = *`, then a `*` in place of a number in the source file is interpreted as a missing value.

If the value is a number, e.g. `null = -9999`, then any string whose numeric value is that number will be interpreted as a missing value, e.g. `-9999`, `-9999.00` and `-9999.000000` will all be interpreted as missing values. The comparison is made with a tolerance of `1e-6`.

(`nullstr` is a deprecated synonym of `null`.)

`delimiter`, `decimal_separator`, `date_format`

Some storage formats may be dependent upon regional settings; these formats support `delimiter`, `decimal_separator`, and `date_format`. `date_format` is specified in the same way as for [strftime\(3\)](#).

`ignore_lines`



For storage formats that are text files, it specifies a regular expression that, if it matches, the line will be ignored. This is useful to ignore header lines or otherwise lines that shouldn't be processed.

encoding

For storage formats that are text files, it specifies the encoding. The default is utf8. [List of possible encodings](#).

timezone

See [Daylight saving time](#).

temperature_unit, rain_unit, wind_speed_unit, pressure_unit, matric_potential_unit

In the wdat5 format, you can select some of the units; C or F for temperature, mm or inch for rain and evapotranspiration, m/s or mph for wind speed, hPa or inch Hg for pressure, centibar or cm (of water) for matric potential. The defaults are C, mm, m/s, hPa, centibar.

outsideTemp, hiOutsideTemp, etc.

Only for wdat5 format; see its description below.

5.4 Supported formats

Don't create yet another conversion script

Many people think they should create a script to convert their file to a format that will be acceptable to `loggertodb` and then use `loggertodb` to read it. Don't do that. Don't have yet another script and yet another file—it increases the complexity of the system. If `loggertodb` does not support your existing file directly, contact us so that we add it (or add it yourself if you speak Python, the API is documented).

The following formats are currently supported:

simple

The `simple` format is lines of which the first one or two fields are the date and time and the rest of the fields hold time series values. If the first field (after stripping any double quotation marks) is more than 10 characters in length, it is considered to be a date and time; otherwise it is a date only, and the second field is considered to be the time; in this case the two fields are joined with a space to form the date/time string. The field delimiter is white space, unless the `delimiter` parameter is specified. The date and/or time and the values can optionally be enclosed in double quotation marks. The format of the date and time is specified by the `date_format` parameter (enclosing quotation marks are removed before parsing; also if the date and time are different fields, they are joined together with a space before being parsed). If



`date_format` is not specified, then the date and time are considered to be in ISO8601 format, optionally using a space instead of T as the date/time separator, and ignoring any seconds. If `date_format` is specified, it must include a second specifier if the times contain seconds, but these seconds are actually subsequently ignored.

The `nfields_to_ignore` parameter can be used to ignore a number of fields in the beginning of each line; this is useful in some formats where the date and time are preceded by a record id or other field.

If `path` contains one of the characters `*?[]`, it is considered to be a pattern that matches many files whose concatenation (ignoring any headers) would be the complete list of records. [glob](#) is used to find the matching files. `loggertodb` does not assume the filenames are ordered in any way; it determines the order by opening all the files and reading a date from each one.

CR1000

Date and time in ISO8601, the first two fields after the date are ignored (they are a record number and a station id), and uses subset identifiers in the next field. It is not clear whether it is debugged and works properly, neither whether its features are a matter of different data logger model or different data logger configuration.

deltacom

The `deltacom` format is space-delimited lines of which the first field is the date and time in ISO8601 format `YYYY-MM-DDTHH:mm`, and the rest of the fields are either dummy or hold time series values, optionally followed by one of the four flags `#`, `$`, `%`, or `&`.

lastem

The `lastem` format is dependent on regional settings, and uses the `delimiter`, `decimal_separator`, and `date_format` parameters. It is lines delimited with the specified delimiter, of which the first three fields are the subset identifiers, the fourth is the date, and the rest are either dummy or hold time series values.

pc208w

The `pc208w` format is comma-delimited items in the following order: subset identifier, logger id (ignored), year, day of year, time in `HHmm`, measurements.

wdat5

The `wdat5` format is a binary format used by Davis WeatherLink; the files have a `wlk` extension. When using it, set `path` to the directory name where your `wlk` files are stored (one file per month).

You can specify time series group ids like this:

```
outsideTemp = 1256
```



```
hiOutsideTemp = 1257  
rain = 1652
```

The full list of variables is outsideTemp, hiOutsideTemp, lowOutsideTemp, insideTemp, barometer, outsideHum, insideHum, rain, hiRainRate, windSpeed, hiWindSpeed, windDirection, hiWindDirection, numWindSamples, solarRad, hiSolarRad, UV, hiUV, leafTemp1, leafTemp2, leafTemp3, leafTemp4, extraRad, newSensors1, newSensors2, newSensors3, newSensors4, newSensors5, newSensors6, forecast, ET, soilTemp1, soilTemp2, soilTemp3, soilTemp4, soilTemp5, soilTemp6, soilMoisture1, soilMoisture2, soilMoisture3, soilMoisture4, soilMoisture5, soilMoisture6, leafWetness1, leafWetness2, leafWetness3, leafWetness4, extraTemp1, extraTemp2, extraTemp3, extraTemp4, extraTemp5, extraTemp6, extraTemp7, extraHum1, extraHum2, extraHum3, extraHum4, extraHum5, extraHum6, extraHum7.

Many of these fields may be reserved by Davis for future use or they may not be used in the particular installation; just don't use them. It is also recommended to ignore the calculated values such as ET (evapotranspiration). More information about the meaning of the parameters can be found in the Davis manuals and in the WeatherLink README file.

odbc

The sane place for loggers and logger software to store meteorological data is a plain text file. Databases shouldn't be used for that purpose. However, I've come across a system which was using MS Access, so I wrote this. It's only tested on Windows and MS Access, though in theory it should be usable anywhere. In that case, `path` is not actually a file name but an ODBC connection string, such as `DRIVER=Microsoft Access Driver (*.mdb);DBQ=C:\Somewhere\mydb.mdb`. `table` specifies the database table in which the data is stored; each variable should be in a plain text column, and there should also be an id column indicating order. `date_sql` is an SQL expression that selects the date and time from the table (the resulting date and time format is defined by `date_format`). `data_columns` is a comma-separated list of (text) columns to retrieve from the table; `fields` must have as many entries as `data_columns`.

You see that this was a hack made for a specific installation, but if you are unfortunate enough to really need it, we can elaborate it further.

5.5 Daylight saving time

Important

Set your loggers to permanently use your winter time or any time that does not change.

In case this was not understood:

Set your loggers to permanently use your winter time or any time that does not change.

Loggertodb contains limited functionality to deal with cases where your loggers change time to DST. However, you should never, ever, use that functionality. Instead, you should



configure your loggers to not do such an insane thing. If you use some kind of software+hardware stack that makes it necessary to configure your loggers to change to DST (something completely unnecessary, you can perfectly and easily store everything in one time zone and display it in another time zone), call your supplier and tell them they suck.

If you ignore this warning and set your loggers to use DST, don't expect loggertodb to do miracles. It can help of course, and it might work while things work smoothly. But whenever your government changes the date or time of the DST switch, or whenever something else goes wrong, you will be trying to fix a big mess instead of doing something useful. Really, you should get a life and set your loggers to permanently use your winter time or any time that does not change.

A time series is composed of records with timestamps. If we don't know exactly what these timestamps mean, the whole time series is meaningless. So, assuming you are in Germany, do you know exactly what 2012-10-28 02:30 means? No, you don't, because it might mean two different things. It could mean 02:30 CEST (00:30 UTC) or 02:30 CET (01:30 UTC). (In fact, several makes of loggers discard one of the two ambiguous hours during the switch from DST, meaning that if an incredible storm occurs at that time, you will lose it. Insane but true.)

In order to avoid insanity, Enhydris has a simple rule: all time stamps of any given time series must be in the same offset from UTC. So you can store your time series in your local time, in UTC time, in the local time of the antipodal point, whatever you like; but it may not switch to DST. If you have a time series that switches to DST, you must convert it to a constant UTC offset before entering it to Enhydris.

If you are unfortunate enough to have loggers that switch to DST, and are unable to change their configuration, `loggertodb` can attempt to convert it for you. The `timezone` parameter should be set to a string like "Europe/Athens":

```
timezone = Europe/Athens
```

(The list of accepted time zones is that of the [Olson database](#); you may find [Wikipedia's copy](#) handy.)

`loggertodb` assumes that the time change occurs exactly when it is supposed to occur, not a few hours earlier or later. For the switch towards DST, things are simple. For the switch from DST to winter time, things are more complicated, because there's an hour that appears twice. If the ambiguous hour occurs twice, `loggertodb` will usually do the correct thing; it will consider that the second occurrence is after the switch and the first is before the switch. If according to the computer's clock the switch hasn't occurred yet, any references to the ambiguous hour are considered to have occurred before the switch.

The `timezone` parameter is used only in order to know when the DST switches occur. The timestamp, after removing any DST, are entered as is. The time zone database field isn't checked for consistency, neither is any other conversion made.