How Bator does read OPERA Radar files processed by Odyssee (HDF5)

Bator is able to read OPERA radars HDF5 files processed by Odyssee. These files must respect EUMETNET OPERA v2.0, v2.1, or v2.2 information models for implementation with the HDF5 file format. Furthermore, only PVOL and SCAN data file are handled.

1. Used writing rules

➔ HDF5 keywords are written in **bold**.
➔ Label used in OPERA radar files are written in *italic*.
➔ Variables and symbols used in Bator are written using `Courier New`.

2. Validation of the file and memory allocations

This operation (performed by `PrefetchHdf5()` and `ValidOdim()` subroutines) requires the param.cfg and NAMELIST files (see documentation concerning these two files for more information). It is composed of the following steps :

➔ selection of the appropriate template (in param.cfg file),
➔ check that the *Conventions* **attribute** matches any of the allowed values,
➔ count the number of elevations found in the file, get all *nrays*, *nbins*, *rscale*, and *rstart* **attributes** in order to allocate the required memory for the `ZENT`, `ZENTSUP`, `ZWAGON` arrays (in Bator.F90).

3. Getting required data from file

A *PVOL* file may contain several *dataset*, with different *startdate*, *starttime*, *nrays* **attributes**, which contain one or more required data types (*DBZH*, *TH*, *VRAD*,...) at the same elevation. In this case, we have to choose one (the closest from the analysis date) to get a proper cylinder of observations. This task is one of the aims of the first part of the `odim()` subroutine. The different stages decided at MF in order to select observations are listed below.

a) *Required top level attributes.*

These required **attributes** are components of *what*, *where*, and *how* top level **groups**. They are stored in the `Radar` structure.

If one top level attribute is missing, the data file will be rejected.
Only *NOD* identifier is considered and must be defined in source attribute.

b) *Other top level attributes*

When we read a *SCAN* data file, the OPERA convention allows to have **attributes** which are specific to *dataset* and *data* **groups** at the top level, as supplemental components of *what* and *where* top level **groups**. So, Bator gets these **attributes** (`GetDAttributes()` subroutine) if they exist and stores them in the `Radar%Attrib` structure.

c) *Filling the FullDatasetList structure*

Bator parses the data file getting all **attributes** from *dataset*, *data* and *quality* **groups** to store them in the `FullDatasetList` array.

When parsing ends, all components of `FullDatasetList()%Gdata()%Attrib` and `FullDatasetList()%Gquality()%Attrib` structures are filled.

d) Selection of the most popular nrays

- ➔ Populates each *nrays* value found with matching *data* **groups** whose *quantity* takes the value *DBZH*, *TH*, *VRAD*, or *VRADH*.
- ➔ Selects the 2 most "populated" *nrays* which must be proportional.
- ➔ Keeps the *data* **groups** matching the selected *nrays* values (and then their elevations). The others are rejected and the corresponding `FullDatasetList()%Gdata()%Attrib` are reinitialized.

e) Selection of the closest elevations to the analysis date

When parsing the `FullDatasetList()GData` array, if several *dataset* **groups** have the same *elevation* value, only the closest to the analysis date is kept. Bator uses the `SelectedElangles` array to store the result.

The resulting `SelectedElangles` for a given elevation value will be a mix of the different quantities found in the datasets groups which match this elevation.

f) Elevations Sort, getting data, and thinning along rays

Sorts selected elevations in the ascending order, gets the data of each *quantity* (and associated flags) and thins them along the ray according to the required resolution. The `Radar%FinalElev` array is used to store the result. This array has to be used in the second part of the `odim()` subroutine.