# Meteorology / Help / ARL Data Format

The following sections describe the ARL packed data format in a little more detail to permit the development of customized applications. Library routines are provided to simplify the task of creating a model compatible data file. A meteorological data file is composed of one or more time periods. Each time period begins with one or more ASCII index records that summarize the valid time, the grid definition, the variables, and level information. Each subsequent record contains one horizontal data field, consisting of 50 ASCII bytes of time, variable, and level information for that record, followed by X times Y bytes of data, where X and Y are the number of data points in the horizontal and vertical directions, respectively. Floating point or integer data are packed as one byte per variable. Precision is maintained by packing the differences between adjacent grid points rather than packing the absolute values. In any time period, although not required, the surface data precede the upper-level data fields. All records are of the same length to permit the model to read the file in a "direct access" mode. Data files can be read on most computing platforms without any transformation and appended to each other using routine operating system commands such as "cat" or "type". Only binary transfers or copies are permitted. All the routines discussed in this section can be found in the *source* directory.

**Valid Meteorological Data Types**

Meteorological variables are identified to the model by a unique four-character identification field that is written to the first 50-byte header portion of each data record. Some of the variables that can be decoded by the model and their units are identified below.

**Sample Surface Level Parameters**

| Description | Units | Identification |
|---|---|---|
| Pressure at surface | hPa | PRSS |
| Pressure at mean sea level | hPa | MSLP |
| Temperature at surface | K | TMPS |
| Total precipitation (6-h) | m | TPP6 |
| U-Momentum flux | $N/m^2$ | UMOF |
| V-Momentum flux | $N/m^2$ | VMOF |
| Sfc sensible heat flux | $W/m^2$ | SHTF |
| Latent heat flux | $W/m^2$ | LTHF |
| Downward short wave flux | $W/m^2$ | DSWF |
| Temperature at 2 m | K | T02M |
| Relative humidity at 2 m | % | RH2M |
| U-component of wind at 10 m | m/s | U10M |
| V-component of wind at 10 m | m/s | V10M |
| Total cloud cover | % | TCLD |

**Sample Upper-Level Parameters**

| Description | Units | Identification |
|---|---|---|
| U wind component (respect to grid) | m/s | UWND |
| V wind component (respect to grid) | m/s | VWND |
| Geopotential height | gpm* | HGTS |
| Temperature | K | TEMP |
| Pressure vertical velocity | hPa/s | WWND |

Data may be obtained from any source, however there are certain minimum data requirements to run the model: surface pressure or terrain height, u-v wind components, temperature, and moisture. In addition, precipitation is required for wet removal calculations. Not required, but certainly necessary to improve vertical mixing calculations, would be some measure of low-level stability. This may take the form of a near-surface wind and temperature or the fluxes of heat and momentum.

It is also important to have sufficient vertical resolution in the meteorological data. Some of the NOAA higher resolution data files have five or more levels in the boundary layer (<850 hPa) in addition to wind and temperatures near the surface, usually at 2 and 10 m agl. These surface values are especially important when the data are only available on absolute pressure surfaces, rather than terrain following sigma surfaces, to avoid interpolation to the surface between data levels when the local terrain is well above sea-level.

**Creation of a Meteorological Input Data File**

One may prepare meteorological data from any number of different sources to be in a suitable format for the model using a series of routines described in this section. In general it is assumed one has access to a meteorological data source, either the data fields are already on a grid, such as those output from a meteorological model, or perhaps from observations that have been interpolated to a grid. Some example conversion programs are provided within the GUI to convert from either NOAA or ECMWF GRIB format data files to the HYSPLIT format.

The meteorological data are processed in time-sequence, calling the subroutines provided, to create a HYSPLIT compatible output file. These subroutines will pack the data and write the index record. The index record, which precedes the data records for each time period, can only be written after the data records are processed. The packing routines must first be initialized by setting the appropriate grid parameters and defining all the meteorological variables that will be written to the file. Multiple output grids may be defined and written simultaneously by invoking the *PAKSET* routine with a different unit number for each grid. The grid parameters are all defined in a configuration file, which should be in the directory from which the procedure is invoked:

*CALL PAKSET (kunit,fname,krec,nx,ny,nz)*

*Kunit* is the Fortran unit number to which the data records will be written. *Fname* is the character string of the name of the configuration file. It can be any name, but it will default to *metdata.cfg*. The file is opened internally on *kunit* to read the configuration file. This routine needs to be called once for each grid. *Krec* is the starting record number (of the index record) to which output will be written. It is normally set to 1 unless you want to start writing in the middle of a file. The remaining parameters (*nx, ny, nz*) are returned by the subroutine and define the horizontal and vertical grid dimensions. These values can be used to set variable dimensions. It is your responsibility to open *kunit* for output after having completed the *pakset* calls:

*OPEN(file=myfile, unit=kunit, form='unformatted', access='direct', recl={50+nx*ny})*

The individual data records are packed and written by a call to *PAKREC,* once for each variable at each level. The routine calculates the record offset from the index record according to the variable and level information provided in the arguments and writes the record according to the order specified in *metdata.cfg*. The data can be supplied in any order. Note that although the level indicator (LL) goes from 1 to the number of levels, one is subtracted before it is written to the 50 byte header to be consistent with the definition of surface data to be at level "0". All the records in a time period may be initialized according to the value of the *kini* flag. Initialization fills in the time variable for all records and assigns the variable identification field as *null*.

*CALL PAKREC (kunit,rvar,cvar,nx,ny,nxy,kvar,iy,im,id,ih,mn,ic,ll,kini)*

kunit - integer unit number of the defined file
rvar - input array of real*4 data values
cvar - character*1 array of packed data values
nx,ny - integer horizontal grid dimensions
nxy - integer product nx*ny and length of cvar
kvar - character*4 descriptor of variable being written
iy,im - integer year and month
id,ih - integer day and hour
mn - integer minutes (usually 0)
ic - integer forecast hour (hours from initialization)
ll - integer level indicator (1 to NZ)
kini - integer initialization flag (0-no; 1-yes)

When all the data records for a time period have been written, it is necessary to close that time period by writing its index record:

*CALL PAKNDX(kunit)*

At this point your program can return to *PAKREC* if data records for additional time periods are to be added to the file.

The key to the process is creating the proper configuration file for the data set you want to create. Some of the sample data decoders provided dynamically configure the packing configuration file based upon the command line input information. A sample *metdata.cfg* file for NOAA's global model output is shown below. An extract of the global one-degree latitude-longitude GRIB model data have been interpolated to a 100 km resolution Lambert Conformal projection 100x100 point grid centered about 45N 90W. The configuration file format is such that the first 20 characters are a dummy identification field followed by the data.

**Example Meteorological Packing Configuration File**

Col 1 (A20) .... Col 21 (A4,2I4,12F10,3I4)
Data Source..... GFSX
Grid Number..... 99
Z-Coordinate..... 2
Pole Latitude..... 45.0
Pole Longitude..... -90.0
Reference Latitude. 45.0
Reference Longitude -90.0
Reference grid size 100.0
Orientation..... 0.0
Tangent Latitude... 45.0
Synch point X..... 50.5
Synch point Y..... 50.5
Synch Latitude..... 45.0
Synch Longitude.... -90.0

Reserved..... 0.0
Number X points.... 100
Number Y points.... 100
Number of Levels... 16

Format: F6.,I3,(1X,A4)

Level 01:.0000 6 PRSS MSLP TPP6 U10M V10M T02M
Level 02:1000. 6 HGTS TEMP UWND VWND WWND RELH
Level 03: 975. 6 HGTS TEMP UWND VWND WWND RELH
Level 04: 950. 6 HGTS TEMP UWND VWND WWND RELH
Level 05: 925. 6 HGTS TEMP UWND VWND WWND RELH
Level 06: 900. 6 HGTS TEMP UWND VWND WWND RELH
Level 07: 850. 6 HGTS TEMP UWND VWND WWND RELH
Level 08: 800. 6 HGTS TEMP UWND VWND WWND RELH
Level 09: 750. 6 HGTS TEMP UWND VWND WWND RELH
Level 10: 700. 6 HGTS TEMP UWND VWND WWND RELH
Level 11: 600. 6 HGTS TEMP UWND VWND WWND RELH
Level 12: 500. 6 HGTS TEMP UWND VWND WWND RELH
Level 13: 400. 6 HGTS TEMP UWND VWND WWND RELH
Level 14: 300. 6 HGTS TEMP UWND VWND WWND RELH
Level 15: 200. 6 HGTS TEMP UWND VWND WWND RELH
Level 16: 100. 6 HGTS TEMP UWND VWND WWND RELH

It is important that the information contained in this file is correct because it not only controls the writing of the packed meteorological data file, but much of the information is written into the index record of each time period. The model decodes this information to set up the internal processing of the meteorological data. Starting with HYSPLIT V4.5, the model is also capable of using meteorological data on a latitude-longitude grid. Previous versions were limited to data on a conformal map projection. Data on a regular latitude-longitude grid still need to be converted to the ARL packed format. Modifications to the packing configuration file required to support a latitude-longitude grid are noted below. A complete description of *metdata.cfg* format follows:

**Description of the Meteorological Packing Configuration File**

Record 1 consists of a four-character string that identifies the source of the meteorological data. This string will be passed through to many of the output graphics.

Record 2 is an optional integer identification of the meteorological data grid. It was used extensively in previous meteorological data file formats. It is not used in Hysplit4 applications.

Record 3 is an integer number that identifies the vertical coordinate system. Only four coordinate types are recognized: 1-pressure sigma; 2-pressure absolute; 3-terrain sigma; 4-hybrid sigma.

Records 4 & 5 identifies the pole position of the grid projection. Most projections will either be defined at +90 or -90 depending upon the hemisphere. The longitude would be the point 180 degrees from which the projection is cut. Lat-Lon Grids only: contains the latitude and longitude of the grid point with the maximum grid point value. Note that lat-lon grids grids should be defined with reference to the dateline.

Records 6 & 7 is the reference position at which the grid spacing is defined. Lat-Lon Grids only: contains the grid spacing in degrees latitude and longitude.

Record 8 is the grid spacing in km at the reference position. Lat-Lon Grids only: a value of zero signals that the grid is a lat-lon grid.

Record 9 is the grid orientation or the angle at the reference point made by the y-axis and the local direction of north.

Lat-Lon Grids only: value always = 0.

Record 10 is the angle between the axis and the surface of the cone. For regular projections it is equal to the latitude at which the grid is tangent to the earth's surface. A polar stereo-graphic grid would be tangent at either 90 or -90, while a Mercator projection is tangent at 0 latitude. A Lambert Conformal projection would be in between the two limits. An oblique stereo-graphic projection would have a cone angle of 90. Lat-Lon Grids only: value always = 0

Records 11 & 12 are used to equate a position on the grid with a position on the earth as given in the following two records:

Records 13 & 14. In this example, the position indicated is the center of the grid located over the North Pole. Any position is acceptable. It need not even be on the grid. Lat-Lon Grids only: contains the latitude and longitude of the 1,1 position grid point.

Record 15 is not currently used.

Records 16 & 17 identify the number of grid points in each direction.

Record 18 is the number of levels in the vertical, including the surface level.

Record 19, through the number of levels, identifies the height of each level in appropriate units according the definition of the vertical coordinate, the number of variables at that level, and the four character identification string for each variable. The height units are as follows for each coordinate:

1-sigma (fraction)
2-pressure (mb)
3-terrain (fraction)
4-hybrid (mb: offset.fraction)

**Decoding Meteorological Data Files**

One may want to develop other applications for HYSPLIT compatible meteorological data files. For these situations, some lower-level routines are provided in the source code library. The key to reading the meteorological files is decoding the index record. The format for this record is given below. Complete descriptions are similar to the variables in the discussion above.

FORMAT DESCRIPTION

- (A4) Data Source
- (I3) Forecast hour (>99 the header forecast hr = 99)
- (I2) Minutes associated with data time
- (12F7) 1- Pole Lat, 2- Pole Lon, 3- Tangent Lat, 4- Tangent Lon, 5- Grid Size, 6- Orientation, 7- Cone Angle, 8- Xsynch point, 9- Ysynch point, 10- Synch point lat, 11- Synch point long, 12) Reserved
- (3I3) 1) Number x points, 2) Number y points, 3) Number levels
- (I2) Vertical coordinate system flag
- (I4) Length in bytes of the index record, excluding the first 50 bytes

LOOP through the number of data levels

- (F6) height of the first level
- (I2) number of variables at that level

LOOP through the number of variables

- (A4) variable identification

(I3) rotating checksum of the packed data
- (1X) Reserved space for future use

Once the index record has been read and decoded you have sufficient information to read and decode the data records. A data un-packer is provided to convert the packed character*1 array to a real*4 array. It can also be used to extract a sub-grid from the full domain through specification of the sub-grid lower left corner:

*CALL PAKINP (rvar,cvar,nx,ny,nx0,ny0,lx,ly,prec,nexp,var1,ksum)*

rvar - real output array of integer dimensions lx,ly
cvar - character*1 packed input array of length nx*ny
nx,ny- integer dimensions of the full grid
nx0 - integer sub-grid position of left edge in nx units
ny0 - integer sub-grid position of lower edge in ny units
lx - integer first dimension of sub-grid length
ly - integer second dimension of sub-grid length
prec - real precision of packed data array
nexp - integer scaling exponent of packed data array
var1 - real value of array at position (1,1)
ksum - integer rotating checksum of packed data array

If the entire grid is to be unpacked then *nx0=ny0=1* and *nx=lx*, *ny=ly*. The checksum (*ksum*) that is returned should be compared with the corresponding value in a table generated from reading the index record. If you are not going to compare the checksum, set *ksum = -1*, this will save a little computer time. Due to the sub-grid option the checksum cannot be computed in the regular unpacking loop, but requires a second pass through the data. The checksum pass is enabled when *ksum=0*. It will then return a non-zero value. If you don't reset it to zero, no further checksums will be computed.

If you want to create your own packed data by converting a real*4 array to the character*1 packed data array use the following:

*CALL PAKOUT(rvar,cvar,nx,ny,nxy,prec,nexp,var1,ksum)*

Although the structure of the packed data may seem complex, unpacking is a very simple process, the basic elements of which are summarized in the Fortran code shown below. The value of each element is the sum of the initial value and the difference stored in that element location.

```
SUBROUTINE UNPACK(CPACK,RVAR,NX,NY,NXY,NEXP,VAR1)

    CHARACTER*1 CPACK(NXY)
    REAL*4 RVAR(NX,NY)
    SCALE=2.0**(7-NEXP)
    VOLD=VAR1
    INDX=0
    DO J=1,NY
    DO I=1,NX
    INDX=INDX+1
    RVAR(I,J)=(ICHAR(CPACK(INDX))-127.)/SCALE+VOLD
    VOLD=RVAR(I,J)
    END DO
    VOLD=RVAR(1,J)
    END DO
    RETURN
```

---

Table of Contents