# Forecast Production Assistant

# GRAPHICS METAFILE STANDARD

# Version 3.0

January 2010

## INTRODUCTION

The Forecast Production Assistant (FPA) Graphics Data Exchange Metafile Standard fulfills two purposes. First it defines a graphics data file format, which requires no prior knowledge on the part of the application using the file as to the contents of the file and, second, it contains information on the meaning of the graphical elements. This second part is important for applications such as the FPA. Unlike other programs, which display graphics images without having to be concerned with the meaning of the images, the FPA is required to understand what the image represents.

On the other hand, Graphics Data Exchange Metafiles contain no graphical display information. This is deliberate, as data display is application dependent. The way in which data is displayed is normally element dependent and should be determined by information held in a display control file associated with the application.

This standard is not intended to remain static. It will need to evolve with time to encompass the growing requirements of the increasingly complex graphics programs, which are being developed. The FPA developers encourage any organization which finds that the standard does not meet their particular requirements to contact the FPA developers. The required changed could then be incorporated into the standard and distributed. This should go a long way towards ensuring future data interchange capability.

## VERSION SUPPORT

Current versions of the FPA software suite will normally support the most recent metafile standard. However, older metafile standards may also be supported in a limited sense. Limited support will normally imply read-only capability, but can include discontinuation of support for specific features.

Specifically, FPA Version 8 provides full support (read and write capability) of Metafile Standard 3.0, and limited support (read only capability) of Metafile Standards 1.2 through 2.0. Several obsolescent features related to discrete (type b) fields are no longer supported.

## DEFINITION OF TERMS

Text shown in **bold** represents keywords, or literal text.

Text shown in *italic* represents variables to which values are to be assigned.

## FILE NAMING CONVENTIONS

FPA Metafiles may have a descriptive name, such as those used for geographical information.

FPA Metafiles may have a name based on the field element, level and time, with two formats.

The old format ( **eellll_YYYY:JJJ:HH[:MM][L]** ) is constructed from a two character element identifier (**ee**), followed by a level identifier (**llll**), separated from the time stamp by an underscore '_'. Note that the level identifier can be up to 32 characters long. The time stamp consisted of a four digit year (**YYYY**), three digit Julian day (**JJJ**), two digit hour (**HH**), and an optional two digit minute (**MM**) separated by the '**:**' character. An optional '**L**' character at the end indicated local time rather than GMT. (The two letter element identifier restricted introduction of new fields, so a new format has been added.)

The new format ( **eeee~llll~YYYY-JJJ-HH[-MM][L]** ) is constructed from an element identifier (**eeee**), a level identifier (**llll**) and a time stamp separated by the '~' character. Note that the element and level identifiers can bu up to 32 characters long. The timestamp uses replaces the '**:**' separator with '**-**', but is otherwise unchanged. (Special FPA link chain files may replace the timestamp with '**Links**'.)

Note that the element and level identifiers are set in FPA Configuration file, as described in the **FPA Administrator's Manual**.

# FORMATS

## File Format:

Metafiles are coded in extended ASCII. Each directive (See METAFILE SPECIFICATIONS) is identified by a keyword, followed by the relevant number of parameters on the same line, or spread over multiple lines as needed. The maximum line length is 1024 bytes after the keyword.

## Units:

All numerical values in a metafile are given in MKS units, where relevant, unless a conversion factor is provided for. Some of the primitives (see the **compound primitives** and the basic **barb primitive)** in the metafile specifications contain an *mcf* parameter. This is the MKS conversion factor, which is defined as the amount to multiply the value(s) by to convert them to MKS units. This feature can be used to control the data precision, and can also be used to eliminate decimal points, in order to reduce file sizes for transmission over communication circuits.

Regardless of the units and scale factors used in a metafile, application programs may convert the data to any desired units, for evaluation and/or display purposes.

## Latitude and Longitude:

Certain parameters of some commands (e.g. **projection** and **mapdef** require a latitude or a longitude to be provided. In addition, locations, origins and point streams in various primitives can optionally be given as latitude-longitude pairs (see **units** primitive).

The recognized format for latitudes and longitudes in all these cases is as follows:

| | Example: |
|---|---|
| Whole degrees: | **46** |
| Decimal degrees: | **52.75** |
| Degrees and minutes: | **60:45** or **60°45'** |
| Degrees, minutes and seconds: | **60:45:15** or **60°45'15"** |

You may indicate the hemisphere by an optional leading sign and/or trailing letter:

| | Direction: | Leading sign: | Trailing letter: |
|---|---|---|---|
| Latitude: | North of Equator | **+** | **N** |
| | South of Equator | **–** | **S** |
| Longitude: | East of Greenwich | **+** | **E** |
| | West of Greenwich | **–** | **W** |

If neither a sign nor a letter is used, the positive sense is assumed. If both a leading sign and a trailing letter are used, **–** reverses the normal sense of the letter used.

## Latitude and Longitude (obsolete compact form):

In addition to the above-recognized latitude-longitude formats, locations, origins and point streams can optionally be given as latitude-longitude pairs in compact degree-minute notation (see **units**). Note; this does not apply to parameters of the **projection** and **mapdef** commands.

The recognized format in this case is [degrees*100 + minutes]. For example, **4350** means 43 degrees and 50 minutes. In this notation, the hemisphere is only indicated by an optional sign, such that **+** refers to latitude north or longitude east, and **–** refers to latitude south and longitude west. This form is obsolescent, and is being supported only for backward compatibility with previous metafile standards.

## METAFILE SPECIFICATIONS

### Version Control:

**rev** *version.revision*

Specifies the version of the metafile standard that the file conforms to. This must be the first non-comment line in the file. If absent it is assumed that the file conforms to the unpublished standard that predates the first published release (1.0).

| | |
|---|---|
| *version* | version number |
| *revision* | version revision number |

For example the current version would be **rev 3.0**

### Reference Control:

**projection** *type ref1 ref2 ... ref5*

Specifies the map projection and relevant reference points.

| | |
|---|---|
| *type* | Keyword for projection (predefined) |
| *ref**n*** | A list of up to 5 projection dependent reference parameters |

The currently defined map projections are:

| | | |
|---|---|---|
| No Projection: | *type* | **none** |
| Latitude - Longitude: | *type* | **latitude_longitude** |
| Plate-Carée: | *type* | **plate_caree** |
| Polar Stereographic: | *type* | **polar_stereographic** |
| | *ref1* | **north** or **south** |
| | *ref2* | "true" latitude * |
| Lambert Conformal: | *type* | **lambert_conformal** |
| | *ref1* | upper reference latitude * |
| | *ref2* | lower reference latitude * |
| Mercator Equatorial: | *type* | **mercator_equatorial** |
| Rotated Latitude-Longitude: | *type* | **rotated_lat_lon** |
| | *ref1* | bottom axis latitude * |
| | *ref2* | bottom axis longitude * |
| | *ref3* | rotation [optional] |
| Oblique Stereographic: | *type* | **oblique_stereographic** |
| | *ref1* | central latitude * |
| | *ref2* | central longitude * |
| | *ref3* | secant angle [optional] |

(* - See FORMATS, Latitude and Longitude)

**mapdef** *olat olon rlon xmin ymin xmax ymax units*

Specifies the target map extent and orientation according to the previously specified **projection** command.

| | | |
|---|---|---|
| *olat* | latitude of the map "origin" | * |
| *olon* | longitude of the map "origin" | * |
| *rlon* | reference longitude | * |

| | |
|---|---|
| *xmin* | start of the x-axis in "units" |
| *ymin* | start of the y-axis in "units" |
| *xmax* | end of the x-axis in "units" |
| *ymax* | end of the y-axis in "units" |
| *units* | number of metres per map unit, or degrees per map unit for **latitude_longitude** projections |

(* - See FORMATS, Latitude and Longitude)

The meaning of *rlon* varies for each type of projection, although the interpretation is always the same. *rlon* is the longitude line which appears vertical (parallel to the y-axis). For polar projections this implies a rotation of *rlon-olon*. For other projections it is irrelevant.

The position of the "origin" within the map is controlled by the choice of *xmin*, *ymin*, *xmax* and *ymax*. For example, if *xmin* and *ymin* are both zero, then the origin corresponds to the lower left corner. If *xmin* = -*xmax* and *ymin* = -*ymax* then the origin corresponds to the map centre.

Nevertheless, all x-y co-ordinates are stated in units from the lower left corner.

If units = 1000, then x-y are in km.

If units = 1609.2, then x-y are in miles.

If units = 0.01, then x-y are in cm.

If units = 1, then x-y are in degrees for **latitude_longitude** projections.

If the data is to be used without reference to a map projection (that is just plotted in a rectangle), then *olat, olon, rlon* and *units* are all set to zero (0).

**units** *type mode*

Defines the representation of point locations in subsequent primitives until another **units** line is encountered. All primitives, which use positions, origins or point streams denoted as "*x y*" in this document, are affected.

| | | |
|---|---|---|
| *type* | **xy** | All positions and locations are given as x, y pairs, in units (as specified by the *units* parameter in the prevailing **mapdef** line) from the lower-left corner of the map definition. |
| *type* | **latlon** | All positions and locations are given as latitude, longitude pairs. The format of latitude-longitude data is described in the FORMATS section (See Latitude and Longitude). |
| *type* | **latlon** | Same as above, except the compact degree-minute notation is used (See Latitude and Longitude (obsolete compact form)). This form is obsolescent and is being supported only for backward compatibility. |
| *mode* | **dddmm** | |

**bkgnd** *file*

Identifies another metafile to be displayed at this point. Generally used to specify the background geography. Only one background may be specified. The specified file must be a metafile.

| | |
|---|---|
| *file* | metafile name |

## Ingest Control:

**source_projection** *type ref1 ref2 ... ref5*
**source_mapdef** *olat olon rlon xmin ymin xmax ymax units*

Defines the projection and map description of the original data, from which subsequent fields have been extracted. Several pairs of these commands may appear, and indicate that the data has been built from a composite of several sources.

The format is the same as used for **projection** and **mapdef** (c.f.).

**source_component** *code*

Indicates which components have been included from the original data. This relates only to vector fields that are treated as related pairs of component fields (e.g. u-wind and v-wind).

This is relevant whenever the target map is different from the original map. In these cases, the original x and y components must each be re-decomposed into components parallel to the axes in the target co-ordinate system. The target x and y components, therefore must each be built from a combination of the original x and y components.

In cases where both original component fields are not available at the same time (such as the FPA ingest process), the contributions to each target component due to the first original component is saved. The **source_component** flag is then set to indicate that only one of the components has been used. The contributions due to the second original component are handled in the same way, once the field containing the second component is encountered. At that point, the **source_component** flag is reset to indicate that both components have been used.

| | | |
|---|---|---|
| *code* | Contains **x** and **y** | Both original components have been used. |
| | Contains **x** only | Only the original x component has been used. |
| | Contains **y** only | Only the original y component has been used. |
| | Does not contain **x** or **y** | No original components have been used. |

## Application-Specific Control:

**info** *process parameter value*

Identifies parameters that can be used by an external application. It is the responsibility of an external process to recognize and deal with these parameters. The parameters are not otherwise relevant to the data contained in the metafile.

| | |
|---|---|
| *process* | arbitrary name that a process may use to select the subset of parameters that it needs |
| *parameter* | arbitrary parameter name |
| *value* | value for this parameter |

Several of these lines may be given in one metafile. The list of parameters defined in this way is maintained in a simple array, which can be obtained by a process from the structure.

## Field Control:

**field** *type elem level*

Identifies the following primitives as being members of the given field. This is the current field until the next **field** statement is encountered.

| | | |
|---|---|---|
| *type* | **continuous** | continuous (single-valued) numerical field |
| | **discrete** | discrete (area or wind-calculation) field |
| | **lchain** | link chain (time series or track) field |
| | **line** | line field |
| | **scattered** | scattered point field |
| | **vector** | continuous (xy component) numerical field |
| *elem* | field element name | |
| *level* | field level name | |

**value | bgvalue | lvalue | rvalue | nvalue** *count attribute value attribute value …*

These statements impart meaning to individual graphical elements.

The **value** statement assigns the given attributes to the basic primitive that follows it. The **bgvalue** statement assigns the given attributes to the background (i.e. default values) of the current field. The **lvalue** and **rvalue** statements assign the given attributes to sub-areas created respectively on the left and right hand sides of a dividing line (see **area divide** primitive). The **nvalue** statement assigns the given attributes to a link node on a link chain (see **node** and **lchain** primitives).

| | |
|---|---|
| *count* | number of attributes to follow |
| *attribute* | attribute name |
| *value* | value for this attribute |

The number of *attribute value* pairs must match the given *count*.

**subfields** *count subfield type subfield type …*

This statement defines the subfields that will be used in the following set of **plot** primitives (c.f.), within the current field.

| | | |
|---|---|---|
| *count* | number of attributes to follow | |
| *subfield* | subfield name | |
| *type* | type of data required for this subfield | |
| | **label** | subfield contains a text string |
| | **mark** | subfield contains a simple marker (dot, circle, cross, etc.) |
| | **barb** | subfield contains a wind (direction and speed) |
| | **int** | subfield contains an integer number |
| | **float** | subfield contains a decimal number |

The number of *subfield type* pairs must match the given *count*.

Compound Primitives (defines an entire field):

**bspline** *mcf nx ny x y angle grid val val ...*

Defines a field, which is represented by a continuous bi-variate B-spline. The B-spline is defined by an array of control vertex values. This type of field can only be identified as a continuous field by a preceding **field continuous** line.

| | |
|---|---|
| *mcf* | MKS conversion factor (multiply *val* by *mcf* to get MKS units) |
| *nx ny* | number of control vertices in x, then y sense |
| *x y* | origin relative to local co-ordinates |
| *angle* | orientation relative to local co-ordinates |
| *grid* | grid length (in prevailing units) |
| *val val …* | list of spline coefficients |

The list of spline coefficients is read into an nx by ny array in the following sense: The first value represents the lower-left corner of the map. Subsequent values increment vertically by the given grid length until ny. The next value is at the bottom of the next column and so on.

**bspline2D map** *order mcf nx ny x y angle grid val val ...*

Defines an xy component field, which is represented by a pair of continuous bi-variate B-splines. Each B-spline is defined by an array of control vertex values. This type of field can only be identified as a vector field by a preceding **field vector** line.

| | | |
|---|---|---|
| *order* | **block** | list all x component values, then all y component values |
| | **pair** | list x value, y value, x value, y value, and so on |
| *mcf* | MKS conversion factor (multiply *val* by *mcf* to get MKS units) | |
| *nx ny* | number of control vertices in x, then y sense | |
| *x y* | origin relative to local co-ordinates | |
| *angle* | orientation relative to local co-ordinates | |
| *grid* | grid length (in prevailing units) | |
| *val val …* | list of spline coefficients | |

The list of spline coefficients is read into two nx by ny arrays in the following sense: The first value represents the lower-left corner of the map. Subsequent values increment vertically by the given grid length until ny. The next value is at the bottom of the next column and so on. Note that for an *order* of **block**, all nx by ny x component values are listed, followed by all nx by ny y component values. Note that for an *order* of **pair**, a pair of x and y component values is listed for each increment.

**grid** *mcf nx ny x y angle grid val val ...*

Defines a grid point field, which is represented by an array of grid point values. This type of field can only be identified as a continuous field by a preceding **field continuous** line.

| | |
|---|---|
| *mcf* | MKS conversion factor (multiply *val* by *mcf* to get MKS units) |
| *nx ny* | number of grid points in x, then y sense |
| *x y* | origin relative to local co-ordinates |
| *angle* | orientation relative to local co-ordinates |
| *grid* | grid length (in prevailing units) |
| *val val …* | list of grid point values |

The list of grid point values is read into an nx by ny array in the following sense: The first value represents the field value at the lower-left corner of the map. Subsequent values increment horizontally to the right by the given grid length until nx. The next value is at the beginning of the next row an so on.

## Basic Primitives (defines a member of a field):

**area boundary** *np x y x y ...*

Defines an area and sets its outer boundary to the given list of points. This primitive can be a member of a discrete field (**field discrete**) only.

| | |
|---|---|
| *np* | number of points to follow |
| *x y x y …* | list of point co-ordinates that define the polygonal boundary of the area |

**area hole** *np x y x y ...*

Defines a hole in the area defined with the last **area boundary** primitive.

| | |
|---|---|
| *np* | number of points to follow |
| *x y x y …* | list of point co-ordinates that define the polygonal boundary of the hole |

**area divide** *np x y x y ...*

Defines a division of the area defined with the last **area boundary** primitive. (Note that this primitive must be preceded by both **lvalue** and **rvalue** statements (c.f.)).

| | |
|---|---|
| *np* | number of points to follow |
| *x y x y …* | list of point co-ordinates that define the division line |

**barb** *x y dir speed gust mcf label*

Defines a wind barb or arrow. This primitive can be a member of a point field (**field scattered**) only.

| | |
|---|---|
| *x y* | barb position in map co-ordinates |
| *dir* | direction in degrees true |
| *speed* | speed in required units (knots for conventional wind barb) |
| *gust* | gust speed in required units (knots for conventional wind barb) |
| *mcf* | MKS conversion factor to km/hr |

**button** *xmin ymin xmax ymax label*

Defines a "button", which is displayed as a box containing a label. This primitive can be a member of a point field (**field scattered**) only.

| | |
|---|---|
| *xmin ymin* | position of lower left corner of box in map co-ordinates |
| *xmax ymax* | position of upper right corner of box in map co-ordinates |
| *label* | label displayed inside the box |

**curve** *sense np x y x y ...*

Defines a curve and sets its path to the given list of points. This primitive can be a member of a line field (**field line**) or can be used as an alternate method to define an area boundary for a discrete field (**field discrete**).

| | | |
|---|---|---|
| *sense* | **r** | drawn in the right-handed sense |
| | **l** | drawn in the left-handed sense |
| *np* | Number of points to follow | |
| *x y x y …* | List of point co-ordinates that define the line | |

The sense of the line specifies which way to orient asymmetrical line patterns. Patterns are defined as being either right or left handed or ambidextrous. Right- or left-handed patterns would then be flipped appropriately according to the sense of the curve. The sense follows the curve in the order in which the points are given.

**label** *angle x y text*

Defines a label. This primitive can be a member of a point field (**field scattered**) only.

| | |
|---|---|
| *angle* | label orientation (degrees) |
| *x y* | label position in local co-ordinates |
| *text* | the label string itself |

**lchain** *xtime splus eplus minterp lnum*

Defines a link chain, which is represented by a time series of nodes. This primitive can be a member of a link chain field (**field lchain**) only. (Note that this primitive must be followed by *lnum* **node** primitives (c.f.))

| | |
|---|---|
| *xtime* | reference time for link chain in *yyyy:jjj:hh* or *yyyy:jjj:hh:mm* format (where *yyyy* is the year, *jjj* is the Julian day, *hh* is the hour, and *mm* is the minutes) |
| *splus* | start time of the link chain (in minutes from *xtime*) |
| *eplus* | end time of the link chain (in minutes from *xtime*) |
| *minterp* | time delta for track interpolation (in minutes) |
| *lnum* | number of nodes in link chain |

**mark** *angle x y type*

Defines a marker. This primitive can be a member of a point field (**field scattered**) only.

| | |
|---|---|
| *angle* | marker orientation (degrees) |
| *x y* | marker position in local co-ordinates |
| *type* | marker type identifier |

**node** *x y type mplus attach mtype imem*

Defines an individual location on a link chain defined with the last **lchain** primitive. This primitive can be a member of a link chain field (**field lchain**) only.

| | | |
|---|---|---|
| *x y* | data position in local co-ordinates | |
| *type* | **normal** | normal link node (position used for track interpolation) |
| | **normal-guess** | guess link node (must be moved before interpolation) |
| | **control** | control node (intermediate position used for track) |
| | **control-guess** | guess control node (must be moved before interpolation) |
| | **floating** | attribute node (intermediate position not used for track) |
| *mplus* | time of the link node (in minutes from *xtime* of **lchain**) | |

| | |
|---|---|
| *attach* | index to item that link node is attached to (default is -1) - used internally by the FPA for time linking |
| *mtype* | type of item member for link node (default is 0) - used internally by the FPA for time linking |
| *imem* | index to type of item member for link node (default is -1) - used internally by the FPA for time linking |

**spot** *x y class attach*

Defines an individual point datum. This primitive can be a member of a point field (**field scattered**) only.

| | | |
|---|---|---|
| *x y* | data position in local co-ordinates | |
| *class* | arbitrary spot prototype identifier (used for process-dependent handling, such as presentation) | |
| *attach* | Indicates which feature (normally when used as annotations to a related field) the data must attach to: | |
| | **none** | does not attach to any feature |
| | **auto** | attach to the nearest feature, as appropriate for the field type |
| | **contour** | attach to the nearest contour (continuous field) |
| | **max** | attach to the nearest maximum (continuous field) |
| | **min** | attach to the nearest minimum (continuous field) |
| | **col** | attach to the nearest saddle point (continuous field) |
| | **bound** | attach to the nearest area boundary (discrete field) |
| | **div** | attach to the nearest dividing line (discrete field) |
| | **line** | attach to the nearest line (line field) |
| | **point** | attach to the nearest point datum (scattered field) |

**plot** *x y value value ...*

Defines an array of values for a point datum. The number of values and their meaning are determined either by an application control file or by a **subfields** statement, which gives the data field prototype. This primitive can be a member of a point field (**field scattered**) only.

| | |
|---|---|
| *x y* | data position in local co-ordinates |
| *value …* | ordered set of values |

Values provided with this primitive are required to match, both in number and in type, with the subfields defined in a prevailing **subfields** statement. All **plot** primitives in the same field must use the same prototype (set of subfields), as defined by one **subfields** statement.

Note, that for a subfield that has a type equal to **barb**, two consecutive numerical values are required, representing direction then speed. Subfields with a type equal to **int** or **float** require a single numerical (integer or floating point respectively) value. All other subfields require a single alphanumeric value. Strings should be quoted (either "" or '') in order to preserve embedded blanks.

This is an alternate, and potentially more compact method for defining point data than the **spot** primitive. Whereas the **spot** primitive assigns attribute values using the **value** statement, the **plot** primitive allows the values to be given in one line. If data at different points can possess different attributes, then the **spot** primitive must be used.

## EXAMPLE METAFILES

In the following examples, data values are indicated as continuing on for the required number of values or value pairs by ... Strings containing embedded blanks are surrounded by quotes (**'…'** or **"…"**).

### A. B-spline pressure surface with labels:

In this example the map definition is for a map of the Atlantic forecast area. A pressure field is defined. A low centre mark is defined using a **spot** that is attached to a minimum in the pressure field. A contour label is also attached to an isobar.

```
rev  3.0
projection  polar_stereographic  north  60N
mapdef  26:45N  90W  85W  0  0  4000  5000  1000
units  xy

field  continuous  pressure  msl
bspline  1.0  23 28  0  0  0  200
101249 101249 101107 100983 100953 101000 101045 101110 101269 101411
101509 101573 ...
... 99980 100640 100640

field  scattered  pressure  msl
value  6
   FPA_user_label "No Label"
   FPA_auto_label "No Label"
   FPA_category "default"
   FPA_label_type "low_at_min"
   hilo_type "low"
   EVAL_spval "975"
spot  3672 3993 "hilo" min
value  5
   FPA_user_label "No Label"
   FPA_auto_label "No Label"
   FPA_category "default"
   FPA_label_type "contour"
   EVAL_contour "1020"
spot  1309 3177 "contour" contour
```

Note that the pressures are given in Pascals with a resolution of 1.0 Pascal. However, values in the contour and low labels are stated in 100's of Pascals (i.e. millibars) for final presentation.

## B. B-spline vector wind field:

In this example the map definition is for the same geographic coverage as in Example A, but with a different format for latitude and longitude. A u-v component wind field is defined.

```
rev 3.0
projection  polar_stereographic  north  60
mapdef  26.75  -90  -85  0  0  4000  5000  1000
units  xy

field  vector  uv_wind  surface
bspline2D  map  block  0.001  23 28  0  0  0  200
808 808 5992 -200 -1433 -1066 -1925 -2060 -1336 1364
3752 3741 3885 ...
... -7804 -9002 -9002
3092 3092 3882 -3770 -2027 -1843 -1627 -2215 -815 522
1578 3103 2968 ...
... -8462 -18728 -18728
```

Note that the u and v component winds are given in thousandths of m/s. The **23** by **28** grid of u component winds is listed first, followed by the **23** by **28** grid of v component winds.

## C. Grid-point temperature field (from an external source):

Continuous fields from an external source (e.g. GRIB) can be supplied as a grid instead.

```
rev 3.0
projection  polar_stereographic  north  60N
mapdef  26:45N  90W  85W  0  0  4000  5000  1000
units xy

field continuous  temperature  surface
grid  0.01  9  11  0  0  0  500
27801 27922 27956 27560 27134 26945 26245 263456
27356 ...
… 27335 27203
```

Note that the temperatures are in °K with a resolution of 0.01 K °.

D. Discrete weather field at surface with labels:

An area of weather is defined. The area is divided into a region of snow and a region of rain showers and a label is provided within each subarea.

```
rev  3.0
projection  polar_stereographic  north  60N
mapdef  26:45N  90W  85W  0  0  4000  5000  1000
units  xy

field discrete weather surface
bgvalue 4
   FPA_user_label     'Clear'
   FPA_auto_label     'Clear'
   FPA_category       'None'
   wx                 ''
area boundary 43 253 97 248 90 ...
… 253 97
lvalue 4
   FPA_user_label     'Snow'
   FPA_auto_label     'Snow'
   FPA_category       'frozen'
   wx                 '1 SN'
rvalue 4
   FPA_user_label     'Rain Showers'
   FPA_auto_label     'Showers'
   FPA_category       'Precip'
   wx                 '1–3 –SHRA FG'
area divide 12 412 83 420 75 ...
… 355 142

field scattered weather surface
value 4
   FPA_user_label     'Snow'
   FPA_auto_label     'Snow'
   FPA_category       'frozen'
   wx                 '1 SN'
spot 62 249 area none
value 4
   FPA_user_label     'Rain Showers'
   FPA_auto_label     'Showers'
   FPA_category       'Precip'
   wx                 '1–3 –SHRA FG'
spot 189 203 area none
```

## E. Fronts:

Both a warm front and a cold front are defined.

```
rev  3.0
projection  polar_stereographic  north  60N
mapdef  26:45N  90W  85W  0  0  4000  5000  1000
units  xy

field line fronts surface
value 5
   FPA_user_label      'Warm Front'
   FPA_auto_label      'Warm Front'
   FPA_category        'warm'
   FPA_line_type       'warm'
   motion              '10kt'
curve r 43 407 2202 401 2153 ...
… 223 2084
value 5
   FPA_user_label      'Cold Front'
   FPA_auto_label      'Cold Front'
   FPA_category        'cold'
   FPA_line_type       'cold'
   motion              '12kt'
curve r 55 407 2202 418 2158 ...
… 10 815
```

## F. Station plot:

In this case the map area is for Ontario but no map is included, as a **projection** and **mapdef** are not required when using "**units latlon**". The primitives are using latitude-longitude as point parameters, instead of x-y. The element name is "**sa**" and the subfields expected are ID, temperature, dew point and wind (direction in degrees true and speed in knots).

```
rev 3.0
units latlon

field scattered sa surface
subfields 4
   id          label
   tt          float
   td          float
   wd          barb
plot   51:34N  87:46W   SITE–1  234  107  130 15
plot   50:23N  86:19W   SITE–2  218   83   150 20
plot   49:92N  85:94W   SITE–3  264  127  140 17
plot   50:73N  87:01W   SITE–4  243   94   145 13
```

## G. Wind field:

Rather than defining actual wind values, FPA wind fields can define rules for calculating winds from models (such as adjusted geostrophic wind, calculated from pressure or geopotential height). There are no areas of adjusted wind in this example, but the wind background (default) is set to **–30°** cross-isobaric and **70**% of MSL geostrophic wind.

```
rev  3.0
projection  polar_stereographic  north  60N
mapdef  26:45N  90W  85W  0  0  4000  5000  1000
units  xy

field discrete actual_wind surface
bgvalue 7
   FPA_user_label          '–30° 70% G85% Vg Msl'
   FPA_auto_label          '–30°' 70% G85%'
   FPA_category            'Vg_Msl'
   FPA_wind_model          'Vg_Msl'
   FPA_wind_direction      'model –30°'
   FPA_wind_speed          'model 70%'
   FPA_wind_gust           'model 85%'
```

## H. Link chain field:

A link chain for a surface cold front is defined. The chain extends from **0** to **90** minutes from time **2009:100:00:00**, with **2** link nodes. Interpolated nodes will be **10** minutes apart (the last **30** extrapolated). The cold front is item **1** at time **0** and item **0** at time **60**.

```
rev  3.0
projection  polar_stereographic  north  60N
mapdef  26:45N  90W  85W  0  0  4000  5000  1000
units  xy

field lchain fronts surface
value 3
   FPA_user_label          'Cold Front Links'
   FPA_auto_label          'Cold Front Links'
   FPA_category            'Links'
lchain  2009:100:00:00  0  90  10  2
nvalue 3
   FPA_user_label          'Cold Front Node'
   FPA_auto_label          'Cold Front Node'
   FPA_category            'Nodes'
node  255 1605  normal  0  1  0  -1
nvalue 3
   FPA_user_label          'Cold Front Node'
   FPA_auto_label          'Cold Front Node'
   FPA_category            'Nodes'
node  307 1803  normal  60  0  0  -1
```