Interface Control Document

for

Meteorological Data Volume (MDV)

XML format

Research Applications Laboratory

2008-01-24

National Center for Atmospheric Research
Boulder, Colorado, USA

MDV XML Interface Control Document

1 Introduction

The Meteorological Data Volume (MDV) format for gridded data was developed by the Research Applications Laboratory at NCAR in the early 1990s. At the time a number of gridded data formats were in use at RAL. To simplify the data systems, it was decided to standardize on a single gridded data format for RAL use.

No public data standard available at the time was considered suitable in terms of data encapsulation and internal compression. MDV evolved as a data format unique to RAL and NCAR. It is an effective format for gridded data, with good meta-data support and an efficient internal compression capability which allows for selected decompression of a single plane from a single data field.

Since the MDV format was developed primarily for use at NCAR, a more widely-accepted standard is needed for transferring MDV data to other organizations. The XML/binary version of MDV was developed for this purpose. It combines the advantages of XML with the efficiency of storing large gridded data field in binary format.

This document describes the MDV XML format.

2 MDV concepts

MDV is a general purpose data file format for storing two- and three-dimensional gridded data.

MDV files each contain data for a single time. Time searching and retrieval is handled by a time-based file naming convention. Support is provided for date/time stamping for creation, expiration, and forecasting times.

MDV provides capabilities for managing multiple data fields in a single file. For example, one MDV file might contain radar data with fields of reflectivity and radial velocity, or model data with temperature, humidity and wind speed as separate fields.

In the (X,Y) dimension, MDV supports a number of projection types, including Lambert Conformal Conic, Polar Stereographic, the simple Latitude-Longitude grid (also known as Simple Cylindrical) and Cartesian and polar coordinates for radar data.

In the vertical dimension, MDV supports a number of vertical coordinate types, including height in km or ft, pressure levels, flight levels for aviation, sigma levels for numerical models and elevation angles for radar data.

Data fields may be represented as 4-byte floating point, 2- and 1-byte scaled integers and 4-byte RGBA image pixels. The data is stored in network-byte-order (big-endian order).

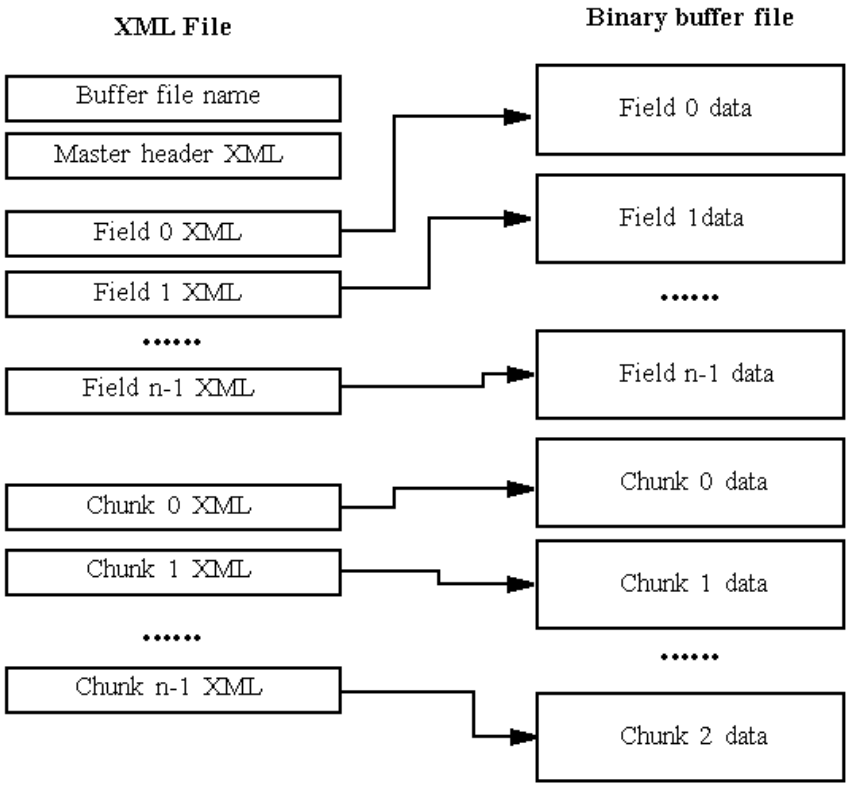
Internal compression is supported, with each field individually compressed using the gzip compression scheme, for speed of access.

The MDV format is extensible in that it provides space and access capabilities for optional generic “chunk” data defined by the MDV user. Chunk data allows MDV users to attach to the data set additional information that is not suitable for storage in the MDV headers or data fields. As examples, the user may wish to store the elevation angles from which a Cartesian radar grid was derived, or the navigation header for satellite data.

3 File format overview

3.1 File structure

The data set for each time is split into two files, (a) and XML meta-data file and (b) a binary data buffer file.



Included in the meta-data XML are byte offsets and lengths for the field data in the binary buffer file.

When a data set is read, the XML is read first, and the meta-data then identifies the data area in the buffer file.

3.2 Binary data types

The data fields in MDV use the following data types:

- unsigned scaled 1-byte integer
- unsigned scaled 2-byte integers
- unsigned 4-byte integers for representing RGBA image values

- IEEE 4-byte floating point values

3.3 Byte ordering

For data types with a length of more than 1 byte, the ordering of the data bytes is important in interpreting the data.

All MDV binary data is stored in so-called **big-endian** or **network-byte-order**. This is the native byte ordering for platforms such as SUN, and the opposite of that used by INTEL and AMD processors common to LINUX and Windows systems.

When reading MDV data on a little-endian platform, the multi-byte values will require byte-swapping.

3.4 File naming conventions

For meteorological data, one of the most important attributes is **time**. Each MDV file contains data for a single time.

MDV files are named according to the time of the data stored in the file. As a general rule, UTC times are used in all MDV data.

A number of times may be applicable:

- **valid time** - the time at which an observation was made. This is also referred to as 'observation time'.
- **generate time** - the time at which a model was run or a forecast was generated.
- **forecast time** - the time at which a forecast is valid.
- **lead time** - the time difference between the forecast time and generate time for a forecast.

MDV XML files are named in two ways, as follows.

By valid time:

```
data_dir/yyyymmdd/hhmmss.mdv.xml
data_dir/yyyymmdd/hhmmss.mdv.buf
```

By generate time and lead time:

```
data_dir/yyyymmdd/g_hhmmss/f_llllllll.mdv.xml
data_dir/yyyymmdd/g_hhmmss/f_llllllll.mdv.buf
```

Most data sets are stored using **valid time**. The files for a single day are stored in a subdirectory (`yyyymmdd`), named after the year, month and day. The files within the directory are named according to the hour, minute and second (`hhmmss`).

For forecast-style data, you can choose whether to use the **valid-time** naming convention above, or whether to use the more complicated **generate-time** and **lead-time** convention. The former is simpler, but can lead to over-writing of data sets if forecast results from different generate times have the same valid time. For example, for a model run at 0 UTC and 6 UTC, the 9-hour forecast from the 0 UTC run will be over-written by the 3-hour forecast from the 6 UTC run because they will both be valid at 09 UTC.

To avoid over-writing, use the second naming convention. There are 2 levels of sub-directory. The name of the upper one is based on the year, month and day of the **generate** time (yyyymmdd). The lower one is named g_hhmmss, based on the hour, minute and second of the **generate** time.The files themselves are named using an 8-digit lead time in seconds (f_llllllll). For example, the 6-hour (21600 second) forecast for the 9 UTC model run on 1 July 2005 will be named:

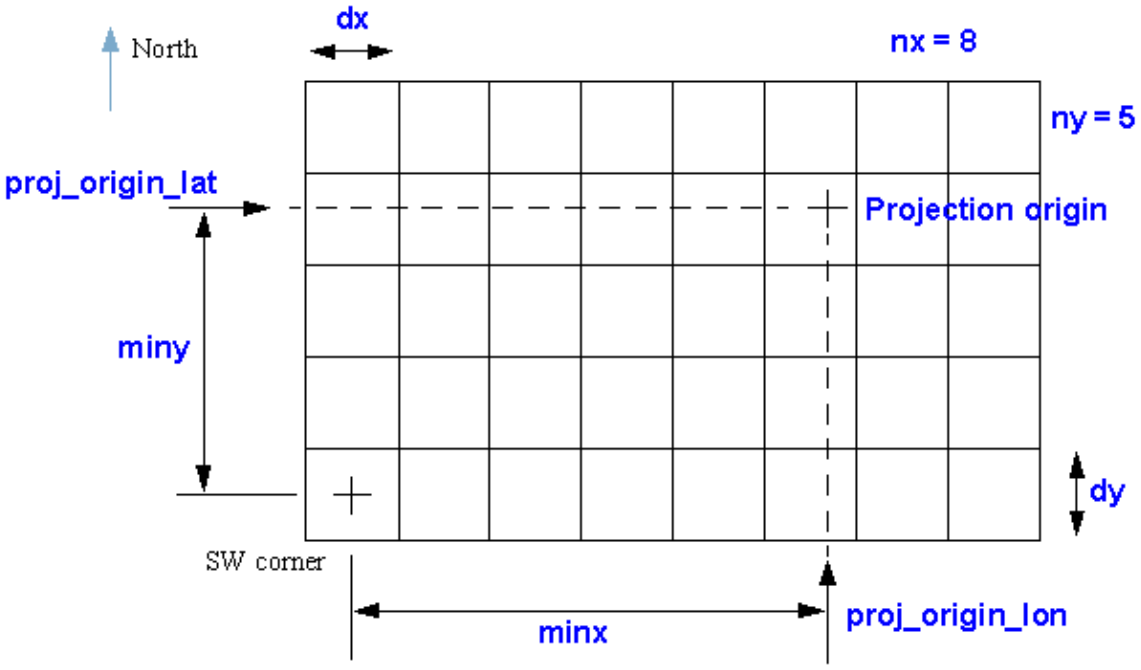
```
20050701/g_090000/f_00021600.mdv.xml
20050701/g_090000/f_00021600.mdv.buf
```

3.5 The X-Y grid

MDV data sets are 2-D or 3-D grids. The grid spacing is regular in X and Y, while in the vertical the spacing may be irregular.

Most data grids conform to a standard projection, such as a regular latitude-longitude grid (Simple Cylindrical) or Lambert Conformal.

The following figure shows how the grid geometry works.



For each grid cell, the location of the data is referenced to the center of the grid cell.

The (X,Y) location of a grid cell may be found as follows:

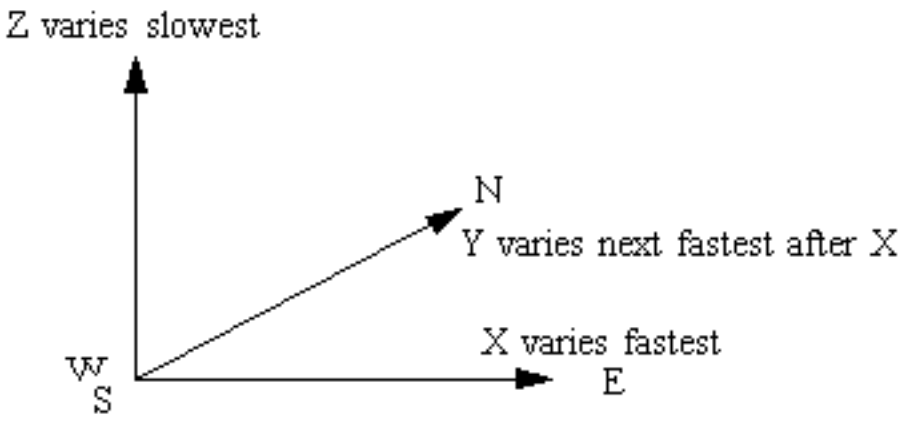
x = minx + ix * dx

y = miny + iy * dy

For most projections, the grid units are in km. For the `latlon` grid projection (see below), the units are in degrees. For radar projections, the units may also be in degrees.

3.6 Data storage order

The field data is stored in an array, packed in X-Y-Z ordering, meaning that X varies the fastest, Y next and Z slowest.



The first byte in the array is at the SW corner of the grid and at the lowest vertical level.

To start unpacking the data, begin at the SW corner of the grid and at the lowest vertical level.

Move first in the X dimension from West to East in the first row along the South edge of the grid, for the lowest vertical level, incrementing X as you go.

Once one row in X is completed, increment Y and go back to the West edge.

Then, repeat the process from West to East, this time in the second row.

Continue until the first vertical level is unpacked.

Then, the process repeats for each vlevel in turn, from the bottom to the top.

4 Meta-data details

4.1 XML schema

The MDV XML meta-data conforms to an XML schema which is published on the web at:

http://www.ral.ucar.edu/xml/schemas/mdv.1.0.xsd

The current version of the schema is included in the appendix. An simple example of an MDV XML file is also included in the appendix..

4.2 Buffer file name

The name of the buffer file is stored with the XML tag buf-file-name.

4.3 Master header meta-data

The master header element occurs only once. The following tags may be found in the master header. Refer to the schema for further details.

Note: all times are in UTC.

XML tag	Description
time-valid	Valid time of the data set. This is the **principal time** used for the data set, to name the files, retrieve data, etc. For observation data, this is normally set to the observation time. For model data, this is set to time-gen + forecast-lead-secs .
time-gen	Generate time for forecast data sets. Not used for non-forecast data, in which case value = 0.
forecast-lead-secs	Lead time of the forecast, relative to gen-time. Only applies to forecast data. Should be equal to time-valid minus time-gen. Not used for non-forecast data, in which case value = 0.
time-written	Time at which the file was written.
time-user	User-specific data. Not used internally by MDV. Normally 0.
time-begin	Start time of data set, if applicable. Often set equal to time-valid.
time-end	End time of data set, if applicable. Often set equal to time-valid.
time-expire	Not used. Included for backwards compatibility.
data-set-name	Data set name, in ASCII. For information only. 127 characters or less.
data-set-info	Data set information in ASCII. For information only. 511 characters or less.
data-set-source	Data set source, in ASCII. for information only. 127 characters or less.
sensor-lon	Longitude of the sensor, in degrees, if applicable. Otherwise 0. E longitudes are positive. An example would be the location of a radar.
sensor-lat	Latitude of the sensor, in degrees, if applicable. Otherwise 0. N latitude are positive.
sensor-alt	Altitude of the sensor, in km MSL, if applicable. Otherwise 0.
data-dimension	2 if all fields contain 2D data, 3 if any fields contain 3D data.

XML tag	Description
data-collection-type	Text string used as information on how the data was obtained or generated. This is for information purposes only. It is not used internally by MDV. Currently understood values are: • measured: data was measured by an instrument • extrapolated: data was created by a forecasting algorithm which uses extrapolation (nowcasting) • forecast: data was forecast by a model or algorithm • synthesis: data was synthesized • mixed: mixture of types • rgba-image: derived from an RGBA image • rgba-graphic: created as an RBGA graphic • climo-analysis: climatology of modeled data • climo-observed: climatology of observed data
vlevel-type	This is the vertical level type of the data in the fields. If the vlevel-type varies from field to field it should be set to variable. See vlevel-type in the field section for details.
native-vlevel-type	The native vertical data type of the data, before translation into MDV. See vlevel-type in the field section for details.
user-data	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-0	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-1	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-2	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-3	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-4	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-5	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-6	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-7	User-specific integer data. Not used internally by MDV. Normally 0.
user-float-0	User-specific floating pt data. Not used internally by MDV. Normally 0.

XML tag	Description
user-float-1	User-specific floating pt data. Not used internally by MDV. Normally 0.
user-float-2	User-specific floating pt data. Not used internally by MDV. Normally 0.
user-float-3	User-specific floating pt data. Not used internally by MDV. Normally 0.
user-float-4	User-specific floating pt data. Not used internally by MDV. Normally 0.
user-float-5	User-specific floating pt data. Not used internally by MDV. Normally 0.
field-grids-differ	false if all fields in the file have the same geometry, true if not.
n-fields	number of fields in the data set
n-chunks	number of chunks in the data set

4.4 Field meta-data

There is one field XML element for each data field in the file.

The following tags may be found in the field section. Refer to the schema for further details.

XML tag	Description
`field-name`	Field name in ASCII. 15 characters or less
`field-name-long`	Long field name in ASCII. 63 characters or less
`field-units`	Units for the field data. 15 characters or less.
`field-transform`	Transform for the field data. 15 characters or less. For informational purposes only. Example is 'dB' for power values in dBm.
`encoding-type`	Encoding type for the data in the field. MDV supports scaled 1-byte and 2-byte integers, 4-byte floating point and 4-byte RGBA image data. • `int8`: unsigned 8 bit scaled integer • `int16`: unsigned 16 bit scaled integer • `float32`: 32 bit IEEE floating point • `rgba32`: RGBA image (4 x 8 bits) - same as TIFF RGBA See scale and bias below for how to convert scaled integers into floats.
`byte-width`	Number of bytes per data point. 1 for int8, 2 for int16, 4 for float32 and rgba32.

XML tag	Description
field-name	Field name in ASCII. 15 characters or less
field-name-long	Long field name in ASCII. 63 characters or less
field-units	Units for the field data. 15 characters or less.
field-transform	Transform for the field data. 15 characters or less. For informational purposes only. Example is 'dB' for power values in dBm.
encoding-type	Encoding type for the data in the field. MDV supports scaled 1-byte and 2-byte integers, 4-byte floating point and 4-byte RGBA image data. • int8: unsigned 8 bit scaled integer • int16: unsigned 16 bit scaled integer • float32: 32 bit IEEE floating point • rgba32: RGBA image (4 x 8 bits) - same as TIFF RGBA See scale and bias below for how to convert scaled integers into floats.
byte-width	Number of bytes per data point. 1 for int8, 2 for int16, 4 for float32 and rgba32.

field-data-scale	scaling factor for converting between scaled integers and floating point numbers.
	
	`float-val = int-val * scale + bias`
	`int-val = floor((float-val – bias) / scale + 0.5)`

field-data-bias	bias for converting between scaled integers and floating point numbers. See field-data-scale.
compression-type	Internal compression type for the data in the fields. Currently supported types: • none: no compression • gzip: each field is individually compressed with gzip

XML tag	Description
`transform-type`	Internal compression type for the data in the fields. Currently supported types: • `none`: no compression • `log`: natural log transform. Useful for data with a large dynamic range, such as precipitation amount.
`scaling-type`	Internal scaling type for the data in the fields. This applies to scaled integers only. This field is for information only. It is not required by MDV. Currently supported types: • `none`: no scaling • `dynamic`: scale and bias computed automatically from the range of the data. • `integer`: scale and bias computed automatically from the range of the data, and then rounded so that values are integral. • `rounded`: scale and bias is computed automatically from the range of the data and then rounded to reasonable values • `specified`: scale and bias are set by the user..
`missing-data-value`	Value set in the field to indicate missing data. For scaled integer fields, it is important to note that the missing-data-value **applies to the scaled integer value**, not to the floating point number which it represents. So when testing for missing data in a scaled integer field, test the field value as it is stored.
`bad-data-value`	Same as missing-data-value. Maintained for backward compatibility.
`min-value`	Minimum floating-point value of the data in the field.
`max-value`	Maximum floating point value of the data in the field.
`data-dimension`	2 for 2D field., 3 for 3D field.
`dz-constant`	`true` if the spacing between the vertical levels is constant, `false` otherwise.

XML tag	Description
projection	Projection details – see separate table below.
xy-grid	X-Y grid details – see separate table below.
n-vlevels	Number of vertical levels in the field
vlevel-type	The vertical level type. This describes the vertical coordinate type for the field data. It is possible that the vlevel type will actually vary from one level to another, as in the case of a hybrid model. In that case variable is used here, and the vlevel type is added as an attribute to the level object. surface: earth surface field: 2D sigma-p: sigma pressure levels pressure: pressure levels, units = mb height-msl-km: constant altitude, units = km MSL sigma-z: model sigma Z levels eta: ETA model levels theta :isentropic surface, units = Kelvin mixed: any hybrid vertical grid, seldom used elevation-angles: radar elevation angles composite: i.e., max value at any height cross-section: cross sectional view of a set of planes satellite: satellite data flight-level: ICAO flight level (100's of ft) earth-conformal: conformal to the surface of the earth. For example, an image of the earth's surface. azimuth-angles: angles in a radar RHI – polar vertical section. tops-msl-km: radar echo top or cloud tops, in km MSL height-agl-ft: constant altitude above ground, units = ft variable: if variable types in fields
native-vlevel-type	The vertical level type of the native data from which this field was derived. See above.
vlevels	The vertical levels section. See separate table below.
vert-reference	Not used – backward compatibility.
data-offset-bytes	Offset, in bytes, of the start of the field data from the start of the binary buffer file.

XML tag	Description
data-length-bytes	Length, in bytes, of the field data in the binary buffer. If the field is compressed, this will be the length of the compressed buffer. If not compressed, this should be: grid-nx * grid-ny * n-vlevels * byte-width.
user-int-0	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-1	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-2	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-3	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-4	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-5	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-6	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-7	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-8	User-specific integer data. Not used internally by MDV. Normally 0.
user-int-9	User-specific integer data. Not used internally by MDV. Normally 0.
user-float-0	User-specific floating pt data. Not used internally by MDV. Normally 0.
user-float-1	User-specific floating pt data. Not used internally by MDV. Normally 0.
user-float-2	User-specific floating pt data. Not used internally by MDV. Normally 0.
user-float-3	User-specific floating pt data. Not used internally by MDV. Normally 0.
user-time-1	User-specific time data. Not used internally by MDV. Normally 0.
user-time-2	User-specific time data. Not used internally by MDV. Normally 0.
user-time-3	User-specific time data. Not used internally by MDV. Normally 0.
user-time-4	User-specific time data. Not used internally by MDV. Normally 0.
grib-code	If available, this is the GRIB code for this field.

4.5 Projection meta-data

The projection information occurs inside the field meta-data.

XML tag	Description
proj-type	Type of projection for this field. The projection types which are important for file-based grids are: • latlon: simple latitude-longitude grid • lambert-conformal (conic) • mercator • polar-stereograaphic • oblique-stereographic • flat (radar Cartesian coordinates) • polar-radar (radar polar coordinates)
origin-lat	Origin of the projection. Applies to all projections except latlon and lambert-conformal.
origin-lon	Origin of the projection. Applies to all projections. For the latlon projection, this is the mean longitude of the grid..
lat1	Standard parallel 1, lambert-conformal
lat2	Standard parallel 2, lambert-conformal
tangent-lon	lambert-conformal, oblique-stereographic
tangent-lat	oblique-stereographic
pole	polar-stereographic Values are "N" or "S"
central-scale	polar-stereographic
rotation	Rotation of the grid relative to true N. flat projection only. Used to align a radar grid with magnetic N.

4.6 XY grid meta-data

The XY grid information occurs inside the field meta-data.

XML tag	Description
nx	Number of grid cells in the X dimension
ny	Number of grid cells in the Y dimension
minx	x-coordinate of the center of the SW (lower left) grid cell (km, or deg for latlon projection)
miny	y-coordinate of the center of the SW (lower left) grid cell(km, or deg for latlon projection)
dx	grid resolution in the x dimension (km, or deg for latlon projection)
dy	grid resolution in the y dimention (km, or deg for latlon projection)

4.7 Vertical levels meta-data

The vertical level information occurs inside the field meta-data. There is a <vlevels> tag, within which exists an array of <level> tags, one for each vertical level in the grid.

The <level> element specifies the value of the vertical level. The units are dependent upon the vertical level type.

If the vertical levels all have the same type, the type is specified by the vlevel-type element which is part of the field element.

If the vertical level types differ for different levels, the vlevel-type element should be set to variable, and the type for each level is specified with the vtype attribute, which may be optionally included with each <level> element.

The following units are appropriate for each vlevel type:

Vertical level type	Units
surface	none
sigma-p	sigma-p (model)
pressure	mb
height-msl-km	km MSL
sigma-z	sigma-z (model)

Vertical level type	Units
eta:	eta (model)
theta	theta
elevation-angles:	deg
composite	none
satellite	none
flight-level	FL (100's of feet)
earth-conformal	none
azimuth-angles	deg
tops-msl-km	km
height-agl-ft	ft

4.8 Chunk meta-data

Chunks are data buffers the details of which are not necessarily understood by the MDV infrastructure. They are essentially 'user defined', and they can be used to store any type of data.

There is one chunk XML element for each chunk in the file.

Refer to the schema for further details.

XML tag	Description
chunk-id	Integer ID used for identifying the chunk type. Not understood by MDV
chunk-info	ASCII text which describes the chunk. 479 characters or less.
data-offset-bytes	Offset, in bytes, of the start of the chunk data from the start of the binary buffer file.
data-length-bytes	Length, in bytes, of the chunk data in the binary buffer.

5 Appendix

5.1 MDV XML schema

The MDV XML schema is published on the web at:

http://www.ral.ucar.edu/xml/schemas/mdv.1.0.xsd

The current version of the schema is included below:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" >

  <xs:element name="mdv">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="buf-file-name"/>
        <xs:element ref="master-header"/>
        <xs:choice maxOccurs="unbounded">
          <xs:element ref="chunk"/>
          <xs:element ref="field"/>
        </xs:choice>
      </xs:sequence>
      <xs:attribute ref="version"/>
    </xs:complexType>
  </xs:element>

  <xs:attribute name="version" type="xs:decimal"/>
  <xs:element name="buf-file-name" type="xs:NMTOKEN"/>

  <xs:element name="master-header">
    <xs:complexType>
      <xs:all>
        <xs:element ref="time-valid"/>
        <xs:element ref="time-gen" minOccurs="0"/>
        <xs:element ref="forecast-lead-secs" minOccurs="0"/>
        <xs:element ref="time-written"/>
        <xs:element ref="time-user" minOccurs="0"/>
        <xs:element ref="time-begin" minOccurs="0"/>
        <xs:element ref="time-end" minOccurs="0"/>
        <xs:element ref="time-expire" minOccurs="0"/>
        <xs:element ref="data-set-name"/>
        <xs:element ref="data-set-info"/>
        <xs:element ref="data-set-source"/>
        <xs:element ref="sensor-lon" minOccurs="0"/>
        <xs:element ref="sensor-lat" minOccurs="0"/>
        <xs:element ref="sensor-alt" minOccurs="0"/>
        <xs:element ref="data-dimension"/>
        <xs:element ref="data-collection-type"/>
        <xs:element ref="vlevel-type"/>
        <xs:element ref="native-vlevel-type"/>
        <xs:element ref="user-data" minOccurs="0"/>
        <xs:element ref="user-int-0" minOccurs="0"/>
        <xs:element ref="user-int-1" minOccurs="0"/>
        <xs:element ref="user-int-2" minOccurs="0"/>
        <xs:element ref="user-int-3" minOccurs="0"/>
        <xs:element ref="user-int-4" minOccurs="0"/>
        <xs:element ref="user-int-5" minOccurs="0"/>
        <xs:element ref="user-int-6" minOccurs="0"/>
        <xs:element ref="user-int-7" minOccurs="0"/>
        <xs:element ref="user-float-0" minOccurs="0"/>
        <xs:element ref="user-float-1" minOccurs="0"/>
        <xs:element ref="user-float-2" minOccurs="0"/>
        <xs:element ref="user-float-3" minOccurs="0"/>
```

```
        <xs:element ref="user-float-4" minOccurs="0"/>
        <xs:element ref="user-float-5" minOccurs="0"/>
        <xs:element ref="field-grids-differ"/>
        <xs:element ref="n-fields"/>
        <xs:element ref="n-chunks"/>
      </xs:all>
    </xs:complexType>
  </xs:element>

  <xs:element name="time-valid" type="xs:dateTime"/>
  <xs:element name="time-gen" type="xs:dateTime"/>
  <xs:element name="forecast-lead-secs" type="xs:integer"/>
  <xs:element name="time-written" type="xs:dateTime"/>
  <xs:element name="time-user" type="xs:dateTime"/>
  <xs:element name="time-begin" type="xs:dateTime"/>
  <xs:element name="time-end" type="xs:dateTime"/>
  <xs:element name="time-expire" type="xs:dateTime"/>
  <xs:element name="data-set-name" type="xs:string"/>
  <xs:element name="data-set-info" type="xs:string"/>
  <xs:element name="data-set-source" type="xs:string"/>
  <xs:element name="sensor-lon" type="xs:decimal"/>
  <xs:element name="sensor-lat" type="xs:decimal"/>
  <xs:element name="sensor-alt" type="xs:decimal"/>

  <xs:element name="data-collection-type" type="data-collection-enum"/>
  <xs:simpleType name = "data-collection-enum">
    <xs:annotation>
      <xs:documentation>
      The data-collection-type documents how the data was collected
      in the field.
      int8: unsigned 8-bit scaled integers
      int16: unsigned 16-bit scaled integers
      fl32: 32-bit IEEE floating point
      rgba32: RGBA image pixels
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base = "xs:string">
      <xs:enumeration value = "measured"/>
      <xs:enumeration value = "extrapolated"/>
      <xs:enumeration value = "forecast"/>
      <xs:enumeration value = "synthesis"/>
      <xs:enumeration value = "mixed"/>
      <xs:enumeration value = "rgba-image"/>
      <xs:enumeration value = "rgba-graphic"/>
      <xs:enumeration value = "climo-analysis"/>
      <xs:enumeration value = "climo-observed"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="user-data" type="xs:integer"/>
  <xs:element name="field-grids-differ" type="xs:boolean"/>
  <xs:element name="n-fields" type="xs:integer"/>
  <xs:element name="n-chunks" type="xs:integer"/>

  <xs:element name="field">
    <xs:complexType>
      <xs:all>
        <xs:element ref="field-name"/>
        <xs:element ref="field-name-long"/>
        <xs:element ref="field-units"/>
        <xs:element ref="field-transform"/>
        <xs:element ref="encoding-type"/>
        <xs:element ref="byte-width"/>
        <xs:element ref="field-data-scale"/>
        <xs:element ref="field-data-bias"/>
        <xs:element ref="compression-type"/>
        <xs:element ref="transform-type"/>
        <xs:element ref="scaling-type"/>
```

```
        <xs:element ref="missing-data-value"/>
        <xs:element ref="bad-data-value"/>
        <xs:element ref="min-value"/>
        <xs:element ref="max-value"/>
        <xs:element ref="data-dimension"/>
        <xs:element ref="dz-constant"/>
        <xs:element ref="projection"/>
        <xs:element ref="xy-grid"/>
        <xs:element ref="n-vlevels"/>
        <xs:element ref="vlevel-type"/>
        <xs:element ref="native-vlevel-type"/>
        <xs:element ref="vlevels"/>
        <xs:element ref="vert-reference" minOccurs="0"/>
        <xs:element ref="data-offset-bytes"/>
        <xs:element ref="data-length-bytes"/>
        <xs:element ref="user-int-0" minOccurs="0"/>
        <xs:element ref="user-int-1" minOccurs="0"/>
        <xs:element ref="user-int-2" minOccurs="0"/>
        <xs:element ref="user-int-3" minOccurs="0"/>
        <xs:element ref="user-int-4" minOccurs="0"/>
        <xs:element ref="user-int-5" minOccurs="0"/>
        <xs:element ref="user-int-6" minOccurs="0"/>
        <xs:element ref="user-int-7" minOccurs="0"/>
        <xs:element ref="user-int-8" minOccurs="0"/>
        <xs:element ref="user-int-9" minOccurs="0"/>
        <xs:element ref="user-float-0" minOccurs="0"/>
        <xs:element ref="user-float-1" minOccurs="0"/>
        <xs:element ref="user-float-2" minOccurs="0"/>
        <xs:element ref="user-float-3" minOccurs="0"/>
        <xs:element ref="user-time-1" minOccurs="0"/>
        <xs:element ref="user-time-2" minOccurs="0"/>
        <xs:element ref="user-time-3" minOccurs="0"/>
        <xs:element ref="user-time-4" minOccurs="0"/>
        <xs:element ref="grib-code" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>

  <xs:element name="field-name" type="xs:string"/>
  <xs:element name="field-name-long" type="xs:string"/>
  <xs:element name="field-units" type="xs:string"/>
  <xs:element name="field-transform" type="xs:string"/>

  <xs:element name="encoding-type" type="encoding-enum"/>
  <xs:simpleType name = "encoding-enum">
    <xs:annotation>
      <xs:documentation>
      The encoding type of the field data
      int8: unsigned 8-bit scaled integers
      int16: unsigned 16-bit scaled integers
      fl32: 32-bit IEEE floating point
      rgba32: RGBA image pixels
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base = "xs:string">
      <xs:enumeration value = "int8"/>
      <xs:enumeration value = "int16"/>
      <xs:enumeration value = "fl32"/>
      <xs:enumeration value = "rgba32"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="byte-width" type="xs:integer"/>
  <xs:element name="field-data-scale" type="xs:double"/>
  <xs:element name="field-data-bias" type="xs:decimal"/>

  <xs:element name="compression-type" type="compression-enum"/>
  <xs:simpleType name = "compression-enum">
```

```
      <xs:annotation>
        <xs:documentation>
        The compression type of the field data
        none: no compression
        gzip: each field is individually gzipped
        </xs:documentation>
      </xs:annotation>
      <xs:restriction base = "xs:string">
        <xs:enumeration value = "none"/>
        <xs:enumeration value = "gzip"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:element name="transform-type" type="transform-enum"/>
    <xs:simpleType name = "transform-enum">
      <xs:annotation>
        <xs:documentation>
        The transform applied to create the data.
        none: no transform
        log: natural log of the data
        </xs:documentation>
      </xs:annotation>
      <xs:restriction base = "xs:string">
        <xs:enumeration value = "none"/>
        <xs:enumeration value = "log"/>
        <xs:enumeration value = "point"/>
        <xs:enumeration value = "sum"/>
        <xs:enumeration value = "diff"/>
        <xs:enumeration value = "product"/>
        <xs:enumeration value = "max"/>
        <xs:enumeration value = "min"/>
        <xs:enumeration value = "mean"/>
        <xs:enumeration value = "median"/>
        <xs:enumeration value = "mode"/>
        <xs:enumeration value = "mid"/>
        <xs:enumeration value = "stddev"/>
        <xs:enumeration value = "variance"/>
        <xs:enumeration value = "covariance"/>
        <xs:enumeration value = "normalized"/>
        <xs:enumeration value = "unknown"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:element name="scaling-type" type="scaling-enum"/>
    <xs:simpleType name = "scaling-enum">
      <xs:annotation>
        <xs:documentation>
        The scaling type used to compute scaled integers
        none: no scaling - floats
        dynamic: scale and bias computed from dynamic range of the data
        rounded: version of dynamic, scale and bias computed by rounding
appropriately
        integral: version of dynamic, scale and bias are integer values
        specified: scale and bias specified by the caller
        </xs:documentation>
      </xs:annotation>
      <xs:restriction base = "xs:string">
        <xs:enumeration value = "none"/>
        <xs:enumeration value = "rounded"/>
        <xs:enumeration value = "dynamic"/>
        <xs:enumeration value = "integral"/>
        <xs:enumeration value = "specified"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:element name="missing-data-value" type="xs:decimal"/>
    <xs:element name="bad-data-value" type="xs:decimal"/>
    <xs:element name="min-value" type="xs:decimal"/>
```

```
<xs:element name="max-value" type="xs:decimal"/>
<xs:element name="dz-constant" type="xs:boolean"/>

<xs:element name="projection">
  <xs:annotation>
    <xs:documentation>
       origin-lat and origin-lon are required by all projections.
       Some parameters are specific to the projection.
       lambert: needs lat1 and lat2
       polar stereo: needs tangent-lon, pole and central-scale
       oblique-stereo: needs tangent-lat and tangent-lon
       flat: needs rotation
    </xs:documentation>
    </xs:annotation>
  <xs:complexType>
    <xs:all>
      <xs:element ref="proj-type"/>
      <xs:element ref="origin-lat"/>
      <xs:element ref="origin-lon"/>
      <xs:element ref="lat1" minOccurs="0"/>
      <xs:element ref="lat2" minOccurs="0"/>
      <xs:element ref="tangent-lat" minOccurs="0"/>
      <xs:element ref="tangent-lon" minOccurs="0"/>
      <xs:element ref="pole" minOccurs="0"/>
      <xs:element ref="central-scale" minOccurs="0"/>
      <xs:element ref="rotation" minOccurs="0"/>
    </xs:all>
  </xs:complexType>
</xs:element>

<xs:element name="proj-type" type="proj-enum"/>
<xs:simpleType name = "proj-enum">
  <xs:annotation>
    <xs:documentation>
    The projection type for the field
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base = "xs:string">
    <xs:enumeration value = "latlon"/>
    <xs:enumeration value = "lambert-conformal"/>
    <xs:enumeration value = "mercator"/>
    <xs:enumeration value = "polar-stereographic"/>
    <xs:enumeration value = "flat"/>
    <xs:enumeration value = "polar-radar"/>
    <xs:enumeration value = "vertical-section"/>
    <xs:enumeration value = "oblique-stereographic"/>
    <xs:enumeration value = "rhi-radar"/>
    <xs:enumeration value = "time-height"/>
    <xs:enumeration value = "unknown"/>
  </xs:restriction>
</xs:simpleType>

<xs:element name="origin-lat" type="xs:decimal"/>
<xs:element name="origin-lon" type="xs:decimal"/>
<xs:element name="lat1" type="xs:decimal"/>
<xs:element name="lat2" type="xs:decimal"/>
<xs:element name="tangent-lat" type="xs:decimal"/>
<xs:element name="tangent-lon" type="xs:decimal"/>

<xs:element name="pole" type="pole-enum"/>
<xs:simpleType name = "pole-enum">
  <xs:annotation>
    <xs:documentation>
    N or S pole for polar stereographic
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base = "xs:string">
    <xs:enumeration value = "N"/>
```

```
      <xs:enumeration value = "S"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="central-scale" type="xs:decimal"/>
  <xs:element name="rotation" type="xs:decimal"/>

  <xs:element name="xy-grid">
    <xs:complexType>
      <xs:all>
        <xs:element ref="nx"/>
        <xs:element ref="ny"/>
        <xs:element ref="minx"/>
        <xs:element ref="miny"/>
        <xs:element ref="dx"/>
        <xs:element ref="dy"/>
      </xs:all>
    </xs:complexType>
  </xs:element>

  <xs:element name="nx" type="xs:integer"/>
  <xs:element name="ny" type="xs:integer"/>
  <xs:element name="minx" type="xs:decimal"/>
  <xs:element name="miny" type="xs:decimal"/>
  <xs:element name="dx" type="xs:decimal"/>
  <xs:element name="dy" type="xs:decimal"/>

  <xs:element name="n-vlevels" type="xs:integer"/>
  <xs:element name="vlevels">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="122" ref="level"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="level">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:decimal">
          <xs:attribute name="vtype" type="vlevel-enum"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="vert-reference" type="xs:integer"/>
  <xs:element name="user-int-0" type="xs:integer"/>
  <xs:element name="user-int-1" type="xs:integer"/>
  <xs:element name="user-int-2" type="xs:integer"/>
  <xs:element name="user-int-3" type="xs:integer"/>
  <xs:element name="user-int-4" type="xs:integer"/>
  <xs:element name="user-int-5" type="xs:integer"/>
  <xs:element name="user-int-6" type="xs:integer"/>
  <xs:element name="user-int-7" type="xs:integer"/>
  <xs:element name="user-int-8" type="xs:integer"/>
  <xs:element name="user-int-9" type="xs:integer"/>
  <xs:element name="user-float-0" type="xs:float"/>
  <xs:element name="user-float-1" type="xs:float"/>
  <xs:element name="user-float-2" type="xs:float"/>
  <xs:element name="user-float-3" type="xs:float"/>
  <xs:element name="user-float-4" type="xs:float"/>
  <xs:element name="user-float-5" type="xs:float"/>
  <xs:element name="user-time-1" type="xs:dateTime"/>
  <xs:element name="user-time-2" type="xs:dateTime"/>
  <xs:element name="user-time-3" type="xs:dateTime"/>
  <xs:element name="user-time-4" type="xs:dateTime"/>
  <xs:element name="grib-code" type="xs:integer"/>
```

```
    <xs:element name="data-dimension" type="xs:integer"/>

    <xs:element name="vlevel-type" type="vlevel-enum"/>
    <xs:element name="native-vlevel-type" type="vlevel-enum"/>

    <xs:simpleType name = "vlevel-enum">
      <xs:annotation>
        <xs:documentation>
        The type of vertical level coordinate
        </xs:documentation>
      </xs:annotation>
      <xs:restriction base = "xs:string">
        <xs:enumeration value = "surface"/>
        <xs:enumeration value = "sigma-p"/>
        <xs:enumeration value = "pressure"/>
        <xs:enumeration value = "height-msl-km"/>
        <xs:enumeration value = "sigma-z"/>
        <xs:enumeration value = "eta"/>
        <xs:enumeration value = "theta"/>
        <xs:enumeration value = "mixed"/>
        <xs:enumeration value = "elevation-angles"/>
        <xs:enumeration value = "composite"/>
        <xs:enumeration value = "cross-section"/>
        <xs:enumeration value = "satellite"/>
        <xs:enumeration value = "variable-elevations"/>
        <xs:enumeration value = "field-specifc-variable-elevations"/>
        <xs:enumeration value = "flight-level"/>
        <xs:enumeration value = "earth-conformal"/>
        <xs:enumeration value = "azimuth-angles"/>
        <xs:enumeration value = "tops-msl-km"/>
        <xs:enumeration value = "height-agl-ft"/>
        <xs:enumeration value = "variable"/>
        <xs:enumeration value = "unknown"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:element name="data-offset-bytes" type="xs:integer"/>
    <xs:element name="data-length-bytes" type="xs:integer"/>

    <xs:element name="chunk">
      <xs:complexType>
        <xs:all>
          <xs:element ref="chunk-id"/>
          <xs:element ref="chunk-info"/>
          <xs:element ref="data-offset-bytes"/>
          <xs:element ref="data-length-bytes"/>
        </xs:all>
      </xs:complexType>
    </xs:element>

    <xs:element name="chunk-id" type="xs:integer"/>
    <xs:element name="chunk-info" type="xs:string"/>

  </xs:schema>
```

5.2 Example of a simple XML meta-data file

The following is a simple example of an MDV meta-data XML file.

The file contains a single field, radar reflectivity, named DBZ. This is observational data, so the forecast time fields are not applicable.

The data grid is on a simple LAT-LON projection. There are 17 vertical levels, with 1 km being the lowest and 17 km being the highest. The vertical spacing is regular in this case, but this is frequently not the case.

For completeness, all of the header fields are shown. Normally many of the fields with zero values would not appear in the XML since they are not applicable to the data set.

```
<?xml version="1.0" ?>
<mdv xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0"
xsi:schemaLocation="http://www.ral.ucar.edu/xml/schemas/mdv.1.0.xsd
http://www.ral.ucar.edu/xml/schemas/mdv.1.0.xsd">
  <buf-file-name>000000.mdv.buf</buf-file-name>
  <master-header>
    <time-valid>2008-01-04T00:00:00</time-valid>
    <time-gen>2008-01-04T00:00:06</time-gen>
    <forecast-lead-secs>0</forecast-lead-secs>
    <time-written>2008-01-24T17:23:36</time-written>
    <time-user>1970-01-01T00:00:00</time-user>
    <time-begin>2008-01-03T23:50:36</time-begin>
    <time-end>2008-01-03T23:54:59</time-end>
    <time-expire>2008-01-04T00:04:23</time-expire>
    <data-set-name>SAWS 3D Mosaic - include MZ</data-set-name>
    <data-set-info>Merged radar data</data-set-info>
    <data-set-source>Merged data created by MdvMerge2.</data-set-source>
    <sensor-lon>0.00000000</sensor-lon>
    <sensor-lat>0.00000000</sensor-lat>
    <sensor-alt>0.00000000</sensor-alt>
    <data-dimension>3</data-dimension>
    <data-collection-type>measured</data-collection-type>
    <vlevel-type>height-msl-km</vlevel-type>
    <native-vlevel-type>elevation-angles</native-vlevel-type>
    <user-data>0</user-data>
    <user-int-0>0</user-int-0>
    <user-int-1>0</user-int-1>
    <user-int-2>0</user-int-2>
    <user-int-3>0</user-int-3>
    <user-int-4>0</user-int-4>
    <user-int-5>0</user-int-5>
    <user-int-6>0</user-int-6>
    <user-int-7>0</user-int-7>
    <user-float-0>0</user-float-0>
    <user-float-1>0</user-float-1>
    <user-float-2>0</user-float-2>
    <user-float-3>0</user-float-3>
    <user-float-4>0</user-float-4>
    <user-float-5>0</user-float-5>
    <field-grids-differ>false</field-grids-differ>
    <n-fields>1</n-fields>
    <n-chunks>0</n-chunks>
  </master-header>
  <field>
    <field-name>DBZ</field-name>
    <field-name-long>DBZ</field-name-long>
```

```
    <field-units>dBZ</field-units>
    <field-transform>dBZ</field-transform>
    <encoding-type>int16</encoding-type>
    <byte-width>2</byte-width>
    <field-data-scale>0.00133588</field-data-scale>
    <field-data-bias>-31.5267</field-data-bias>
    <compression-type>none</compression-type>
    <transform-type>none</transform-type>
    <scaling-type>dynamic</scaling-type>
    <missing-data-value>0.00000000</missing-data-value>
    <bad-data-value>0.00000000</bad-data-value>
    <min-value>-31.50000000</min-value>
    <max-value>56.00000000</max-value>
    <data-dimension>3</data-dimension>
    <dz-constant>true</dz-constant>
    <projection>
      <proj-type>latlon</proj-type>
      <origin-lat>0.00000000</origin-lat>
      <origin-lon>0.00000000</origin-lon>
    </projection>
    <xy-grid>
      <nx>1380</nx>
      <ny>1200</ny>
      <minx>15.00000000</minx>
      <miny>-37.00000000</miny>
      <dx>0.01666666</dx>
      <dy>0.01666666</dy>
    </xy-grid>
    <n-vlevels>17</n-vlevels>
    <vlevel-type>height-msl-km</vlevel-type>
    <native-vlevel-type>elevation-angles</native-vlevel-type>
    <vlevels>
      <level>1</level>
      <level>2</level>
      <level>3</level>
      <level>4</level>
      <level>5</level>
      <level>6</level>
      <level>7</level>
      <level>8</level>
      <level>9</level>
      <level>10</level>
      <level>11</level>
      <level>12</level>
      <level>13</level>
      <level>14</level>
      <level>15</level>
      <level>16</level>
      <level>17</level>
    </vlevels>
    <vert-reference>0</vert-reference>
    <data-offset-bytes>0</data-offset-bytes>
    <data-length-bytes>56304000</data-length-bytes>
    <user-int-0>0</user-int-0>
    <user-int-1>0</user-int-1>
    <user-int-2>0</user-int-2>
    <user-int-3>0</user-int-3>
    <user-int-4>0</user-int-4>
    <user-int-5>0</user-int-5>
    <user-int-6>0</user-int-6>
    <user-int-7>0</user-int-7>
    <user-int-8>0</user-int-8>
    <user-int-9>0</user-int-9>
    <user-float-0>0</user-float-0>
    <user-float-1>0</user-float-1>
    <user-float-2>0</user-float-2>
    <user-float-3>0</user-float-3>
  <user-time-1>1970-01-01T00:00:00</user-time-1>
```

```
      <user-time-2>1970-01-01T00:00:00</user-time-2>
      <user-time-3>1970-01-01T00:00:00</user-time-3>
      <user-time-4>1970-01-01T00:00:00</user-time-4>
        <grib-code>0</grib-code>
      </field>
    </mdv>
```