DESIGN SPECIFICATION

Component name	Version
MVACDF	2.0-6
Author	**Date**
Bengt Olsson (translated by Mats Josefson)	1999-10-07

REVISION CONTROL

1999-12-14 /BOL	Added revision control
1999-12-22 /BOL	Version 2.0-5 Minor fix for memory problem in ReadColumnHeaders when the number of columns where very large.
2000-04-03 /BOL	Improved documentation concerning Column Headers usage.
2000-07-13 /BOL	Version 2.0-6 Added creation of default values for mandatory global attributes.

CONTENTS

INTRODUCTION

Design specification for NetCDF raw-data files containing multivariate data.

This document describes a multivariate raw data format that is intended for use in the analytical laboratory and for process analytical chemistry. Files created according to this specification is called MVACDF files.

This document is also a design specification of the programming interface that has been created for the NetCDF storage mechanism that should make it easier to create NetCDF files that adheres to the following specification.

NETCDF - SUMMARY

NetCDF is a format for storage of scientific data. The NetCDF format is designed for storage of multidimensional arrays in a flexible way. With a function library that can be used to create and manipulate NetCDF files, it is also relatively easy to use this format. This function library has been developed by Unidata Program Center, Boulder, Colorado, USA. The library has been implemented for a number of programming languages and computer platforms.

The following description of the NetCDF format is available on Unidata's web pages **http://www.unidata.ucar.edu/packages/netcdf/**. These pages also give a full description of the NetCDF format together with links to related information.

"NetCDF (network Common Data Form) is an interface for array-oriented data access and a library that provides an implementation of the interface. The NetCDF library also defines a machine-independent format for representing scientific data. Together, the interface, library, and format support the creation, access, and sharing of scientific data. The NetCDF software was developed at the Unidata Program Center in Boulder, Colorado. The freely available source can be obtained by anonymous FTP from ftp://ftp.unidata.ucar.edu/pub/netcdf/ or from other mirror sites."

MVACDF

MVACDF is using a subset of the functionality and the possibilities available in NetCDF and its programming interface. MVACDF also defines a number of conventions for information that should be contained in a MVACDF file and how this should be named.

An MVACDF file contains the following items, each item will be discussed in the following sections.

- Global Attributes
- Variable Descriptors
- Column Headers
- Observation (raw) Data
- Observation Descriptors
- Data Matrices
- Data Matrix Attributes

Global attributes

A MVACDF file contains a number of global attributes that describe the content of the file. The attributes are divided in the three groups: mandatory, system, and additional attributes as described in the tables below.

The mandatory and the system attributes has to be present in a MVACDF file. The additional attributes may be useful in certain cases and should be named as specified if present.

Mandatory Attributes

data_set_origin	Where the data have been produced
equipment_id	What equipment has been used for the experiment
equipment_manufacturer	The manufacturer of the equipment
equipment_type	The type of equipment e.g. model number
operator_name	The operator identity
experiment_date_time_stamp	Date and time for the start of the experiment

System Attributes

mva_template_revision	Version/revision level for MVACDF
netcdf_file_date_time_stamp	Date and time when this file was created
netcdf_revision	Version/revision level for netCDF (3.3.1)
raw_data_mva_format	What numeric format is used for the raw data (see section ‘Valid datatypes’ for details.
missing_value	-100000, or NaN

dectime_unit	Unit for decimal time (h)

Additional Attributes

experiment_title	The experiment title
experiment_type	The experiment type
dataset_owner	The owner of the data set
source_file_reference	File name of the original file (at conversion from another file format)
source_file_date_time_stamp	Source file creation date and time of the original file (if any).
source_file_format	The original file format
instrument_sw_version	The instrument software version used
instrument_os_version	The instrument operating system version
photometric_resolution	The photometric resolution of the instrument
spectral_pretreatment	The method(s) for spectral pretreatment
no_of_scans	The number of scans averaged for a stored spectrum
detector_type	The Detector type, e.g. InGaAs, PbS
database_id	A data base id for the raw data
sample_comments	Comments related to the data or the experiment

Observation (raw) data

Each observation consists of a vector or matrix of floating point values. Each vector or matrix is described by an index number and the date and time for the observation. Each observation may also be given a name.

MISSING VALUES

A value is treated as missing if it is given the value of the global attribute missing_value. A suggestion for the missing value is -100000 another suggestion is the code for IEEE floating point code for “not a number” e.g. NaN in Matlab.

Observation Descriptors

It is possible to store additional information about an observation in addition to the date/time, dectime and name of the observation. This is done by creating one or more observation descriptors. An observation descriptor can be thought of as an additional column of string values stored together with the observations. Each observation descriptor must have its own unique name within the file.

Variable Descriptors

The variables may be labeled by individual names, then one variable descriptor is used for each variable. For the case of sections of variables sharing the same scale and

coming from the same sensor or instrument (e.g. a spectrometer), one variable descriptor may be used to describe the entire section of variables with its scale (e.g. wavelength scale). A variable descriptor has the following parts:

Name	The name of the sensor that was used for data collection
Position	The position of the sensor in the equipment where the measurement took place
zUnit	The unit of the measured variable/variables
StartColumn	This descriptor describes the variables starting at StartColumn
EndColumn	This descriptor describes the variables ending at EndColumn
StartValue	The value of the x-scale at StartColumn (e.g. the first wavelength in a spectrum)
EndValue	The value of the x-scale at EndColumn (e.g. the last wavelength in a spectrum)
Step	The step size in the x-scale between two variables(e.g. digital wavelength resolution)
xUnit	The unit of the x-scale including StartValue, EndValue and Step

Column Headers
It is possible to use Column Headers instead of (or together with) variable descriptors. A column header is a text value which is associated with each column, i.e., one column header for each column of observation data.

Data Matrices

An optional feature of MVACDF is the possibility to store additional matrices of data in the MVACDF file. Such matrices could be used for multivariate modelling purposes etc. The matrices must be given their own unique names.

Data Matrix Attributes

It is possible to give each Data Matrix its own set of user defined attributes.

Valid Datatypes

Observation (raw) data and Data matrices can be of one of the following datatypes.

byte	8-bit signed or unsigned integers
short	16-bit signed integers
long	32-bit signed integers
float	32-bit IEEE floating-point
double	64-bit IEEE floating point

The datatype used in a specific MVACDF-file is defined at creation of the file and can not be changed thereafter.

Example

Below is an overview of the structure in a MVACDF file. The file contains data in 701 columns or variables. The first 700 variables are measured by the NIR sensor “NS-X” and the last variable contains the results from temperature measurements with the sensor “TEMPZ”.

Column Headers has also been stored to give each variable (column) a name.

The file contains one additional observation descriptor named ‘ObsDescrBatch’ which contains batch information for each observation.

There is also two matrices stored in the file ‘ModelMatrix1’ and ‘ModelMatrix2’.

Column Headers

1100	1102	.....	2484	2486	2488	2490	2492	2494	2496	2498	TEMP

Global attributes

operator_name="ANAXX"
equipment_id="Mixer XZ-12"
equipment_manufacturer="Acme Inc."
equipment_type="Blender"
experiment_datetime="1999-06-07 12:31:42"
data_set_origin="AstraZeneca R&D Molndal"
.
.
.

Var. Descr. 1

Name="NS-X"
Position="N1"
zUnit="log(1/R)"
StartCol=1
EndCol=700
StartValue=1100
EndValue=2498
Step=2
xUnit="um"

Var. Descr. 2

Name="TEMPZ"
Position="T1"
zUnit="K"
StartCol=701
EndCol=701
StartValue=
EndValue=
Step=
xUnit=

Idx	Date + Time	Name												
1	1999-06-07 10:10:10	QPX_A	12.2	12.1	.....	12.2	19.4	12.0	5.4	3.2	2.2	1.6	4.8	12.7
2	1999-06-07 10:10:10	QPX_B	12.3	11.9	.....	12.2	19.4	12.1	5.6	3.2	2.2	1.6	4.9	12.5
3	1999-06-07 10:10:10	QPX_C	12.2	12.1	.....	12.7	19.3	12.1	5.7	3.2	2.3	1.7	4.8	12.7
4	1999-06-07 10:10:10	QPX_D	12.6	12.1	.....	12.4	19.4	12.1	5.4	3.2	2.2	1.6	4.8	12.7
.														
.														

ObsDescrBatch

Idx	
1	Batch7
2	Batch8
3	Batch3
4	Batch2
.	
.	

ModelMatrix1

description=The first matrix used for the model <- User defined Data Matrix Attribute

0.01	0.32	0.13	12.5	0.13	1.74	0.01	1.74	0.32	0.13
0.01	1.74	12.5	1.74	0.01	0.32	0.01	0.13	1.74	0.32
0.32	0.13	0.01	12.5	0.32	0.13	12.5	0.32	0.01	12.5
1.74	12.5	0.32	1.74	12.5	0.01	1.74	12.5	0.13	1.74

ModelMatrix2

mean=12.5 <- User defined Data Matrix Attribute
stdev=7.3 <- User defined Data Matrix Attribute

12.5	1.74	0.01	0.32	0.01	0.13
0.01	12.5	0.32	0.13	12.5	0.32

PROGRAMMING INTERFACE

A programming interface was made to create and read files in the MVACDF format. This is implemented as an ActiveX server (COM, OLE-automation). This implementation can be called from all programming and other tools that supports OLE-automation (e.g. C, Delphi, Excel, Labview, Visual Basic, etc).

The interface is written as an OLE-dll (mvacdf.dll) with Microsoft Visual C++ and ATL 3.0 and call in its turn the NetCDF library for Win32 (netcdf.dll, written by Glenn Davis for netcdf version 3.3.1). The name of the OLE-object is “MVACDF.MVACDFFile”.

The functions in this interface can be divided in two groups: reading and writing of files. Below follows a description of the functions in these two groups. The output arguments are marked with an asterisk (*) in front of the argument name. The data types are given in parentesis after the argument name.

t[x]	8bit characters (x = string length, which is given a value between 5 and 255 at file creation time).
t[]	8bit characters (“unlimited” length)
n	32bit integer
n[7]	Array of 7 32bit numbers, year, month, day, hour, minutes, seconds and 1/10000 seconds.
f	64bit floating point number (IEEE)
d[]	Array of raw data or matrix data. See section ’Valid datatypes’ for a description.

Writing files

`Create(filename, nofCol, nofVarDescr, datatype, stringlen, nofRowsPerObs)`

Creates a new file with a number of columns and column descriptors.

filename(t[255])	The name of the file to be created. The file can not overwrite an existing file with te same name.
nofCol (n)	The total number of columns in the raw data matrix.
nofVarDescr (n)	The number of variable descriptors.
datatype (t[32])	The datatype of observation (raw) data and data in additional matrices. See section ’Valid datatypes’ for a description of valid types.
stringlen (n)	Maximum length of text string (valid values: 4<stringlen<256)
nofRowsPerObs (n)	The number of rows for each observation, i.e., nofRowsPerObs=1 equals a vector with nofCol columns. If nofRowsPerObs>1 the

	observations are matrices with nofRowsPerObs rows and nofCol columns.

`WriteGlobalAttr(data_set_origin, equipment_id, equipment_manufacturer, equipment_type, operator_name, experiment_datetime)`

Writes the global attributes that are mandatory for MVACDF files.

data_set_origin (t[])	Where the data have been produced
equipment_id (t[])	What equipment has been used for the experiment
equipment_manufacturer (t[])	The manufacturer of the equipment
equipment_type (t[])	The type of equipment e.g. model number
operator_name (t[])	The operator identity
experiment_datetime (t[])	Date and time for the start of the experiment

`InquireAddGlobalAttr(*attr[])`

Returns a list with attribute names for the “Additional Global Attributes” that are recomended in the MVACDF-standard.

`WriteAddGlobalAttr(name, value)`

Writes the additional file attributes and other undefined attributes that are needed. The name might contain characters A-Z, a-z, 0-9 and _ but should not begin with an _.

name (t[120])	Additional attribute name.
value (t[])	The value of the attribute

`WriteVariableDescription(idx, sensorName, sensorPosition, zUnit, startCol, endCol, startVal, endVal, step, xUnit)`

Fills in the information for a variable descriptor indexed by idx.

idx (n)	The number of the variable descriptor to be written (starts at, n=1).
sensorName (t[x])	The name of the sensor that was used for data collection
sensorPosition (t[x])	The position of the sensor in the equipment where the measurement took place
zUnit (t[x])	The unit of the measured variable/variables
startCol (n)	This descriptor describes the variables starting at StartCol.
endCol (n)	This descriptor describes the variables ending at EndCol
startVal (f)	The value of the x-scale at StartCol (e.g. the first wavelength in a spectrum)

endVal (f)	The value of the x-scale at EndCol (e.g. the last wavelength in a spectrum)
step (f)	The step size in the x-scale between two variables(e.g. digital wavelength resolution)
xUnit (t[x])	The unit of the x-scale including StartVal, EndVal and Step

`WriteColumnHeaders(headers[])`

Writes the column headers.

headers(t[])	An array of headers, one for each column of observation data.

`WriteObsData(name, time, dectime, observation)`

Writes raw data for one observation to the file.

name (t[x])	The name of the observation (object).
time (n[7])	A vector that describes the date and time for the observation.
dectime (f)	Time in decimal hours from the start of the experiment.
observation (d[][])	The actual data for the one observation (object) containing all variables for all variable descriptors.

`CreateAddObsDescr(name)`

Create an additional data descriptor for each observation. The name might contain characters A-Z, a-z, 0-9 and _ but should not begin with an _.

name (t[120])	The name of the descriptor.

`WriteAddObsDescr(descrname, idx, value)`

Write information to an additional observation descriptor.

descrname (t[120])	The name of the descriptor.
idx (n)	Index of the observation to describe. If idx=0 then the index of the last observation will be used.
value (t[x])	The describing information.

`WriteDataMatrix(name, matrixdata)`

Writes one data matrix to the file. The name might contain characters A-Z, a-z, 0-9 and _ but should not begin with an _.

name (t[120])	The name of the matrix.
matrixdata (d[][])	A one or two dimensional array of data contained by the matrix.

`WriteDataMatrixAttr(matrixname, attrname, attrvalue)`

Writes one Data Matrix Attribute to the file.

matrixname (t[120])	The name of the matrix.
attrname (t[120])	The name of the data matrix attribute.
attrvalue (t[])	The value of the attribute.

`Close()`

Closes the file.

Reading files

`Open(filename, *nofCol, *nofVarDescr, *nofObservations, *datatype, *stringlen, *nofRowsPerObs)`

Opens a file for reading.

filename (t[255])	The name of the file to be read.
nofCol (n)	The total number of columns or variables for the observations.
nofVarDescr (n)	The number of variable descriptors in the file.
nofObservations (n)	The number of observations in the file.
datatype (t[32])	The datatype of observation (raw) data and data in additional matrices. See section 'Valid datatypes' for a description of valid types.
stringlen (n)	Maximum length of text string (valid values: 4<stringlen<256)
nofRowsPerObs (n)	The number of rows for each observation, i.e., nofRowsPerObs=1 equals a vector with nofCol columns. If nofRowsPerObs>1 the observations are matrices with nofRowsPerObs rows and nofCol columns.

`ReadGlobalAttr(*data_set_origin, *equipment_id, *equipment_manufacturer, *equipment_type, *operator_name, *experiment_datetime)`

Reads the mandatory global attributes for a MVACDF file.

data_set_origin (t[])	Where the data have been produced
equipment_id (t[])	What equipment has been used for the experiment

equipment_manufacturer (t[])	The manufacturer of the equipment
equipment_type (t[])	The type of equipment e.g. model number
operator_name (t[])	The operator identity
experiment_datetime (t[])	Date and time for the start of the experiment

`ReadAddGlobalAttr(name, *value)`

Reads the additional file attributes and other undefined attributes that are needed.

name (t[120])	Additional attribute name
value (t[])	The value of the attribute

`ReadAllGlobalAttr(*name_and_value[][])`

Reads all global file attributes including the ones that have been added with WriteAddGlobalAttr.

name_and_value (t[][])	Attribute names and values are returned in a two-dimensional array of strings. Each row has two columns (name of attribute and the value of the attribute).

`ReadVariableDescription(idx, *Name, *Position, *zUnit, *startCol, *endCol, *startVal, *endVal, *step, *xUnit)`
Reads a variable descriptor.

idx (n)	The number of the variable descriptor to be written (starts at, n=1).
sensorName (t[x])	The name of the sensor that was used for data collection
sensorPosition (t[x])	The position of the sensor in the equipment where the measurement took place
zUnit (t[x])	The unit of the measured variable/variables
startCol (n)	This descriptor describes the variables starting at StartCol.
endCol (n)	This descriptor describes the variables ending at EndCol
startVal (f)	The value of the x-scale at StartCol (e.g. the first wavelength in a spectrum)
endVal (f)	The value of the x-scale at EndCol (e.g. the last wavelength in a spectrum)
step (f)	The step size in the x-scale between two variables(e.g. digital wavelength resolution)
xUnit (t[x])	The unit of the x-scale including StartVal, EndVal and Step

ReadColumnHeaders(*headers[])

Returns a list (array of strings) of column headers. If there is no column headers in the file then the column headers are derived from the variable descriptors. This function can be used for client programs wanting to view the observation data in some kind of spreadsheet format.

NOTE: If there is no column headers in the file column headers are derived from each variable descriptor as follows:

1. If the startVal, endVal and step values are set for the descriptor then calculate headers. Ex.: startVal=1100, endVal=2498 and step=2 results in headers 1100, 1102, etc. No decimals will be used in the headers.
2. If startVal, endVal and step are not set then the column header will be the variable descriptor name + variable no (if more than one variable in the descriptor).
3. If for some reason the above is not applicable then column headers will be C_1, C_2, C_3 etc. where the number is the column no.

ReadObsData(idx, *time, *dectime, *name, *observation)

Reads an observation (object) -vector at index idx including its time parameters and name.

idx (n)	The index of the observation (object) in the file
name (t[x])	The name of the observation (object).
time (n[7])	A vector that describes the date and time for the observation.
dectime (f)	Time in decimal hours from the start of the experiment.
observation (d[][])	The actual data for the one observation (object) containing all variables for all variable descriptors

ReadPartialObsData(col1, col2, idx1, idx2, *partobsdata)

Reads a user specified range of observation data the from the file. The is specified using observation indexes (row numbers) and columns. This function does only work if each observation is a vector (nofRowsPerObs=1) and not a matrix (nofRowsPerObs>1).

col1 (n)	The number of the first column that should be included.
col2 (n)	The number of the last column that should be included.
idx1 (n)	The index of the first observation (object) that should be included.
idx2 (n)	The index of the last observation (object) that should be included.
partobsdata (d[][])	The actual data for the specified range of observations and columns (one or two dimensional).

ReadObservationTime (firstidx, lastidx, *times)

Reads a user specified range of observation times from the file.

firstidx (n)	The index of the first observation time that should be included.
lastidx (n)	The index of the last observation time that should be included.
times (n[][])	A matrix with lastidx-firstidx+1 rows and 7 (year, month, day, hour, minute, second, 1/10000 s) columns.

`ReadObservationDecTime (firstidx, lastidx, *dectimes)`

Reads a user specified range of observation dectimes from the file.

firstidx (n)	The index of the first observation time that should be included.
lastidx (n)	The index of the last observation time that should be included.
dectimes (d[])	A vector with lastidx-firstidx+1 elements.

`InquireAddObsDescr(*descriptornames[])`

Returns a list (array of strings) of names of additional observation descriptors available in the file.

`InquireDataMatrix(*matrixnames[])`

Returns a list (array of strings) of names of data matrices available in the file

`ReadDataMatrix(name, *matrixdata)`

Reads one data matrix from the file. The name might contain characters A-Z, a-z, 0-9 and _ but should not begin with an _.

name (t[120])	The name of the matrix.
matrixdata (d[][])	A one or two dimensional array of data contained by the matrix.

`ReadDataMatrixAttr(matrixname, attrname, *attrvalue)`

Reads one data matrix attribute from the file.

matrixname (t[120])	The name of the matrix.
attrname (t[120])	The name of the data matrix attribute.
attrvalue (t[])	The value of the attribute.

`ReadAllDataMatrixAttr(matrixname, * name_and_value[][])`

Reads all data matrix attributes for one data matrix from the file.

matrixname (t[120])	The name of the matrix.
name_and_value[][]	Attribute names and values are returned in a two-dimensional array of strings. Each row has two columns (name of attribute and the value of the attribute).

`GetDataMatrixSize(matrixname, *rows, *cols)`

Returns the size of a matrix.

matrixname (t[120])	The name of the matrix.
rows	The number of rows in the matrix.
cols	The number of columns in the matrix.

`ReadAddObsDescr(descrname, idx, *value)`

Read information from an additional observation descriptor.

descrname (t[120])	The name of the descriptor.
idx (n)	Index of the observation to read describing information for.
value (t[x])	The describing information.

`Close()`

Closes the file.

Miscellaneous functions

`GetCurrentVersion(*version)`

Returns a string value representing the version of the MVACDF-dll (currently 2.0-0).

`GetFileInfo(filename, *ObsStart, *ObsEnd, *VarStart, *VarEnd, *nofVars, *nofObs, *AddInfo)`

This function can be used by client programs that wants to derive some simple information about the file without opening it. The function is particulary useful when handling large amounts of MVACDF-files with spectral data (with one variable descriptor) but it will not be useful for all applications (for example when using more than one variable descriptor).

filename (t[255])	The name of the file for which information is wanted.
ObsStart (t[x])	The date and time of the first observation.

ObsEnd (t[x])	The date and time of the last observation.
VarStart (t[x])	The wavelength (when applicable) for the first variable.
VarEnd (t[x])	The wavelength (when applicable) for the last variable.
nofVars (n)	Number of variables (columns)
nofObs (n)	Number of observations
AddInfo	Additional information derived from the file

`ObservationCount()`

This property can be used for checking how many observations that are currently in the file. The return value is a number.

`ReadOnly()`

This property can be set (=TRUE) before calling the *Open* function for opening the file in ReadOnly-mode. This is useful when two processes is accessing the same file (one writer and one reader).

MVACDF IN NETCDF TERMS

In this section the MVACDF file structure will be explained using NetCDF terms and the 'network data format language' (CDL) notation as defined in NetCDF. This explanation will be useful for example when writing programs in environments where the MVACDF.dll can not be used or when not wanting to use the MVACDF.dll for some other reason.

The following (all) the different MVACDF objects will be explained.

- Global Attributes
- Variable Descriptors
- Column Headers
- Observation (raw) Data
- Observation Descriptors
- Data Matrices
- Data Matrix Attributes

The different datatypes that may be used are described in a separate section. And also there is a specification of the differences between the structure of the MVACDF 1.0 and the MVACDF 2.0 standard.

Global Attributes

Global attributes are simply the same as global attributes in NetCDF with the limitation that only text attributes is allowed.

Variable Descriptors

A variable descriptor is described in NetCDF by 7 variables and two dimensions.

```
dimensions:
number_variable_descriptions = 2 ;
string_len = 200 ;

variables:
char vardescr_sensorname(number_variable_descriptions, string_len) ;
char vardescr_sensorposition(number_variable_descriptions, string_len) ;
char vardescr_zunit(number_variable_descriptions, string_len) ;
long vardescr_startcol(number_variable_descriptions) ;
long vardescr_endcol(number_variable_descriptions) ;
double vardescr_startval(number_variable_descriptions) ;
double vardescr_endval(number_variable_descriptions) ;
double vardescr_step(number_variable_descriptions) ;
```

Each variable corresponds to one of the arguments of the WriteVariableDescription function.

Column Headers

The column headers are described in NetCDF by one variable and two dimensions.

```
dimensions:
number_data_points = 700 ; // The number of columns
string_len = 200 ;

variables:
char column_headers(number_data_points, string_len);
```

Observation (raw) Data

An observation is described in NetCDF by 5 variables and 4 dimensions.

```
dimensions:
sequence_no = UNLIMITED ; // The number of observations
number_data_points = 700 ; // The number of columns
number_time_data = 7 ; // The number of values in a time "record"
string_len = 200 ; // Max length of the name of an observation

variables:
long sequence_no(sequence_no) ;
long time_of_measurement(sequence_no, number_time_data) ;
double dectime_of_measurement(sequence_no) ;
char name_of_measurement(sequence_no, string_len) ;
double observation(sequence_no, number_data_points) ;
```

If the observation data is matrices (pictures?) instead of vectors (spectra?) then there will be another dimension present in the file and the observation-variable will be dependent of this dimension also. (When using the MVACDF.dll to create an MVACDF-

file then this dimension will be present when nofRowsPerObs>1 in the call to the Create-function.)

```
dimensions:
number_rows_per_data_point = 500 ; // The number of rows for each observation

variables:
double observation(sequence_no, number_data_points,
number_rows_per_data_point);
```

NOTE: The datatype of the observation variable is not fixed to double. See 'Datatypes' section below for details.

Observation Descriptors

An observation descriptor is described in NetCDF by one variable and two dimensions. The name of the variable should be prefixed with "aod_" so that it can be recognised by MVACDF as an observation descriptor. The handling of the prefix is taken care of internally in the MVACDF.dll, and is thus automatically introduced when using WriteDataMatrixAttr.

```
dimensions:
sequence_no = UNLIMITED ;
string_len = 200 ; // Max length of the name of an observation

variables:
char aod_BatchNo (sequence_no, string_len) ;
```

Data Matrices

A data matrix is described in NetCDF by one variable and one or two dimensions. The name of the variable should be prefixed with "mtx_" so that it can be recognised by MVACDF as a data matrix. . The handling of the prefix is taken care of internally in the MVACDF.dll, and is thus automatically introduced when using WriteDataMatrix.

```
dimensions:
_40 = 40;
_65 = 65;

variables:
double mtx_ModelComponent (_40, _65)  ;
```

or when the matrix dimension is of the same size

```
dimensions:
_40 = 40;

variables:
double mtx_Model (_40, _40)  ;
```

or in the case of a vector

```
dimensions:
_40 = 40;

variables:
double mtx_OneVector (_40)  ;
```

NOTE: The datatype of a datamatrix is not fixed to double. See ‘Datatypes’ section below for details.

Data Matrix Attributes

Data matrix attributes are simply the same as variable attributes in NetCDF with the limitation that only text attributes is allowed.

Datatypes

The following data types are supported by MVACDF. The MVACDF-name of the datatype used in one file must be stored in a global attribute called ‘raw_data_mva_format’. Mixing of datatypes in one MVACDF-file is not allowed.

MVACDF-name	**Length and type**	**NetCDF-name**
byte	8-bit signed or unsigned integers	byte (unsigned char)
short	16-bit signed integers	short
long	32-bit signed integers	long
float	32-bit IEEE floating-point	float
double	64-bit IEEE floating point	double

Compatibility with MVACDF 1.0

The only backward compatibility issue with MVACDF 1.0 is the length of strings for names of observations and attribute descriptors. In MVACDF 1.0 this length was fixed to 32 characters and the dimension was called:

dimension:
_32_byte_string = 32;

In MVACDF 2.0 this dimension has been replaced with the

dimension:
string_len = 250;

dimension that has a variable length (min 5 and max 255 characters).
However the MVACDF.dll does handle this and it is possible to read MVACDF 1.0 files with the MVACDF.dll version 2.