Cluster Exchange Format - Data File Syntax

CSDS Archive Task Group
A. Allen, W. Baumjohan, A. Fazakerley, M. Hapgood, & C. Harvey,

Document Status Sheet			
1. Document Title: **Cluster Exchange File Syntax**			
2. Document Reference Number: **DS-QMW-TN-0010**			
3. Issue	4. Revision	5. Date	6. Reason for Change
1	0	13 April 2000	Draft
1	1	4 May 2000	Preliminary version for SWT
1	2	25 May 2001	Early operations trial format
1	3	5 Nov 2001	following ISTP input on array handling
1	4	16 Apr 2002	

Contents

1 Introduction 1

2 File Format 1
2.1 Time Series Data . . . 2
2.1.1 Sub-millisecond Timing . . . 2
2.2 Data Series . . . 2

3 File Headers 2
3.1 Metadata Syntax . . . 3
3.2 File Metadata . . . 3
3.2.1 Optional File Metadata . . . 3
3.2.2 Global attributes . . . 4
3.3 Variable Metadata . . . 5
3.3.1 Mandatory Variable Metadata . . . 5
3.3.2 Extra Metadata for Dimension variables . . . 9
3.3.3 Optional Variable Metadata . . . 9
3.4 Variable Metadata Summary . . . 10
3.5 Data Records . . . 10

4 Sample CEF 11
4.1 Full example of CEF file . . . 11
4.2 Minimal Example of CEF file . . . 15

1 Introduction

A single ascii format data file syntax has been recommended by the CSDS Archive Task Group for the exchange of science data between instrument teams. This format is intended as an exchange format to allow translation between the several native data formats used by science tools and data bases within the Cluster community. Adopting a single common format that can be written and read by all teams allows exchange of data between any of the other formats and storage systems.

The file syntax and minimum header content is specified to describe the products sufficiently for science use. The open format allows for inclusion of any extra metadata deemed desirable by the generating team.

This format is not *a priori* intended for archival or storage, but will ease the process of formatting data products into a final repository. This does not affect the instrument team's own plans for storing of raw (or decommutated) data as long term archive, but serves to facilitate the storage of commonly used science products in a robust and easily accessible form.

This format is not intended to replace the normal data exchange channels used by each instrument group - such as ISDAT and IDFS. It is intended to facilitate delivery of data products to workers outside the normal community of an instrument or consortium and to other instrument teams operating with different science and database software.

Software already exists that can translate between the CSDS standard CDF files and CEF files. The utility **Qtran** can be obtained from

`http://www.space-plasma.qmw.ac.uk/QSAS/qtran_welcome.html`

The same utility is able to read other record oriented ascii data and may help with translation from ascii data dumps into the recommended exchange format.

This document provides the syntax for these ”Cluster Exchange Format” (CEF) files.

2 File Format

All data files are record oriented and homogeneous - they have a sequence of records each with the same variables in the same order. Variables that are multi-dimensional take the natural C ordering, and one entry is required for each element. In time series data the records are ordered on the monotonic increasing time variable. Each record is ended by a **record_delimiter** which by default is a new line character, (\n).

Entries in a record are comma separated and white space is ignored.

The record delimiter and all carriage return (\r), new line (\n) and white space characters are deleted from data records on read. Thus data may safely be formatted with white space and end of line markers for readability. This also allows for easier exchange between platforms where the end of line marker is variously \r\n under DOS, \n under Unix and \r under MacOS. All these end of line markers are removed together with white space (including tab characters, \t).

A header provides sufficient metadata to describe the data and its formatting, and is specified in detail below. The header may be attached (preceding the data records), or detached (in a separate file).

A detached header may provide formatting information for a series of files which all have the same record structure.

If the header is attached, the data records must be immediately preceded by a line:
`Start_data = xxx`
where `xxx` is an integer specifying the number of records to follow. If the number of data records is unknown, this value is set to zero.

An exclamation mark is used as a comment marker, and all input to the right of this marker up to the newline character is ignored on input.

Blank lines (where the first character is the newline character, the end of record character, or contain only white space characters) are ignored.

It is recommended that files in this syntax have names with the extension **.cef** to assist identification. This does not form part of the syntax. Similarly a detached header should carry the extension **.ceh** to distinguish it from accompanying data files.

2.1 Time Series Data

Time series data have a time entry associated with each record. Each other entry in the record is associated with that time. If present the time tag must be the first entry in a record.

Time is represented as a text string in the ISO format adopted by CSDS. This is of the form:
*yyyy-mm-dd*T*HH*:*MM*:*ss.www*Z
it e.g. “1996-01-30T13:30:00.000Z” for January 30 1996 at 1.30 pm.

2.1.1 Sub-millisecond Timing

Users should be aware that some software, for example utilities for handling CDF data files, may be unable to handle timing accuracy greater than milliseconds. This results from the physical limitation of the CDF epoch data type and associated software.

The ISO format permits any number of digits after the decimal point in the seconds field. For example “1996-01-30T13:30:00.000000Z” is a valid time string to microsecond accuracy. The trailing ‘Z’ is retained as an end delimiter for this purpose.

This issue will require further discussion.

2.2 Data Series

Data series have no time entry in a record. If more than one data series is written to the same file then each data series must contain the same number of records and any one-to-one dependency (or otherwise) between variables in a record must be explicitly explained within the file metadata.

3 File Headers

The file header contains formatting information about the file, and metadata that describes the data. A header may apply to a family of data files that contain data in the same format, such as would result from dividing data into files covering distinct time intervals.

3.1 Metadata Syntax

All header entries take the format
`parameter = value`
where “`parameter`” and “`value`” are strings of printable characters. The equals sign may be embedded in space characters, and white space is stripped before the “parameter” token. White space is valid within the “`value`” string which is terminated by the newline character or comment marker, ‘`!`’. Leading and trailing white space is removed from the “`value`” token. In the case that the “`value`” token contains only white space then software should return a single space character as the token. If multiple values are to be specified for a single “parameter” then they must be comma separated.

Metadata is divided into global information that applies to the whole file and information that describes the variables and their formatting within a record.

There is no preferred order for parameters within the metadata, with the exception of variable metadata blocks that must appear in the same order as the variables appear in the data records. However, if present, the file metadata described below should appear before global attributes or variable blocks as they affect the reading of subsequent input.

3.2 File Metadata

3.2.1 Optional File Metadata

The file parameters below are optional and are provided for information purposes or to over-ride the default values described below.

File_name This specifies the name of the file. It should not include the path, but does include the file type extension. If this is a detached header it is the name and extension of the header. Good practice would suggest that the header and the data files to which it applies share the same file name stem up to the non-unique part of the name (usually an event code or date).

File_type This specifies the format type of the file. The value should be ‘d’ for delimited. When provided it can be used by general translation utilities to distinguish it from other flat file types that can be inter-converted with these files.

Data_delimiter It is useful to specify the delimiter to be used to separate data entries. This delimiter is the same for all variables (but may differ from that used in the header for attributes). It is a comma for CEF files. It has syntax
`Data_delimiter = ,`
Its inclusion assists general software that translates to other flat file formats to interpret the CEF syntax files.

Attribute_delimiter This specifies the character to be used in the header itself to separate each value in multi-value metadata. For CEF files a comma is used since space characters are frequently valid within metadata strings. Its inclusion assists general software that translates to other flat file formats to interpret the CEF syntax files.

End_of_record_marker This specifies the character to be used to indicate the end of each record. The default value is the new line character (\n). Setting a different character allows long records to be split across lines with new line characters for human readability and for software systems with maximum line lengths. On read all new line and carriage return characters together with white space (including tabs) and the end or record marker are stripped after the record is read. These characters and non-printing characters are thus not allowed values for character data types.

Start_meta This parameter starts a block of metadata supplying the entries associated with a global attribute (in cdf terminology). This block is closed by an **End_meta** parameter. They are described more fully below. The value associated with these parameters is the name of the global attribute. No defaults are provided, and no global attributes are required.

3.2.2 Global attributes

Global attributes are used by the cdf file syntax to provide informational metadata associated with the whole file. They are allowed here for completeness and as a means of attaching information that may be carried along with the data.

In contrast, commented lines may be stripped from the exchange files when reprocessed.

The Global attribute block starts with a line

`Start_meta = name`

and ends with the line

`End_meta = name`

where the value “name” is the name of the global attribute. The name is *not* restricted to the Global Attributes used by CSDS for the Prime and Summary Parameter files. Specification of a named metadata block implies creation of metadata with that name.

The global metadata block contains the following entries:

Number_of_entries The value associated with this parameter is the number of attribute entries provided for this attribute. It must be followed by the same number of “Entry” parameters within in the attribute block.

Entry The value associated with this entry is the next entry in the global attribute. Space characters are valid within a text entry. This parameter may be repeated, with each successive value providing the next entry in the attribute.

Value_type The ‘Value_type’ of a global attribute may be used to convert the ascii text entry in each ‘Entry’ into the appropriate type of the value in the data structure. The default value for each global metadata block independently is text. This parameter allows changing of type for each subsequent ‘Entry’ within a block until reset with another ‘Value_type’ parameter, or the end of the attribute block is reached. Allowed values are
epoch (a double CDF_epoch value)
float
double
char
byte

ISO standard date/time strings should be represented as character data types.

3.3 Variable Metadata

Blocks of information describing the variables start with a line
`Start_variable = name`
and end with the line
`End_variable = name`
where the value "name" is the name to be used for the variable. These blocks are required for variable description and header files are not valid without one describing each variable. Variables *must* appear in the same order within a record that the variable entries occur within the header.

3.3.1 Mandatory Variable Metadata

These metadata provide formatting information specific to the named variable. There is no preferred order for parameters within a variable metadata block.

Each block of variable metadata takes the form,

```
Start_variable = name
parameter = value
   ⋮       ⋮
End_variable = name
```

Where data in science units are provided it is highly desirable to provide metadata sufficient to describe that data. The following metadata are required whenever meaningful.

Value_type This identifies the data type and is necessary for conversion from the ascii entry.
Allowed values are
epoch (a double CDF_epoch value)
float
double
char
byte

Sizes This is essential for any variable that has more than one element, such as arrays and vectors. The value string must comprise as many ('Attribute_delimiter' separated) integer values as there are dimensions in the variable (in the CDF sense) with the number giving the size of the array in that dimension. Thus an 8 by 54 array would have the entry
`Sizes = 8,54`.
It is not required for scalars.

Time_format In time series data the time variable must have this parameter to identify the time format used. For CEF files this must take the value **ISO**. It is not used for non-time variable types.

Data The CDF concept of a variable that is fixed for all records is supported for cef files. Data for these 'non-record-varying' variables must be supplied within the header variable metadata segment, and no entry is then allowed in the data records. The presence of a parameter 'Data' will be taken to indicate that this is a non-record-varying variable. The value(s) associated with this parameter are the data for that variable. These are particularly useful for label variables. They are separated by the 'Attribute_delimiter' as they are specified within the metadata segment. Multiple Data lines may be used to accomodate large data arrays, and the input will be concatenated assuming the natural C ordering - last index varies fastest.

UNITS Text string with units as they would appear on plots where appropriate.

Frame Required only for vectors and tensors or components thereof; it is not required for scalars, general arrays and character strings. Text string in syntax
type>frame of reference _ representation
For example 'V_e_xyz_gse' has 'Frame' 'vector>gse_xyz' and identifies the nature of the variable, and the co-ordinate system to be in cartesian representation and frame of reference to be gse. The *type* of quantity may be one of the following:

vector

tensor

component [1]

or optional types (scalar, array, ...)

The system is one of

xyz

rtp

rlp

rt

xy

others to be user defined, or "na" if the representation field is not applicable

These must be used in conjunction with the SI_conversion to determine whether angles are in degrees or radians. The frame may be any of the standard space physics co-ordinate frames such as gse or gsm etc, together with instrument acronym for the instruments own reference frame for instrument frame and 'sc' for spacecraft frame. The frame specified is that underlying the data representation, so for a component it is also necessary to specify which component(s) are supplied using the associated 'Component_desc'. Similarly, any processing which renders the quantity not a true vector must be specified elsewhere and the type part of the Frame specified as 'component'.

Component_desc Required only for variables whose 'Frame' attribute takes the type 'component'. It is needed to specify which component or reduced quantity from a vector or tensor the variable represents. For example

Frame = component_gse_rtp
Component_desc = *p>azimuthal angle phi*

---

[1] This is an extension of the CSDS cdf file standard for 'Frame'

SI_conversion Required for all data in science units. Text string of the form

number>SI unit

where *number* is the conversion factor to SI units. It is the factor that the variable must be multiplied by in order to turn it into SI units. The string *SI unit* is the standard unit that it converts to. For example the magnetic field for FGM may be in **nT**, and to convert to Tesla the value of "SI_conversion" should be '1.0e-9>T'. For multi-dimensional units the grammar will be of a standard form: distinct unit dimensions will be separated by space characters and powers (signed) will be preceded by the carat, $\wedge$. Non-dimensional qualifiers, which do not appear in the SI units list, are to be enclosed in braces '()'. For example, 'm s$\wedge-1$' or '(number electrons) m$\wedge-3$'. Similarly '(percent)' and '(ratio)' would provide user information on dimensionless quantities. Non-integer powers are permitted, e.g. 'Hz$\wedge-0.5$'. SI units should be one of:

s second
kg kilogram
m metre
Hz hertz
A ampere
K kelvin
J joule
V volt
T tesla
Pa pascal
C coulomb
henry [needed for mu_o]
farad [needed for eps_o]
W watt
N newton
ohm
mho
rad radian
sr steradian
degree [alternative angle measure, not SI, but convenient and often used].

Depend_i This identifies the name of another variable within the file that contains information identifying the ith dimension in an array variable. They are not required for vectors, tensors, non-ordered arrays such as Status, or the Depend_i variables themselves. There must be as many of these parameters as there are entries in the **Sizes** value. Thus a 3-D array would require three parameters, Depend_1, Depend_2 and Depend_3. The variables identified may be either non-record-varying or supplied for each record to allow for changing bin boundaries such as variable energy sweeps. Each of the identified dimension variables must have a defining variable block in the header as normal. Depend variables should be 1-D arrays of the same size as the dimension described. Thus a Depend variable describing the N azimuthal angular bins for a data array might have the form Dimension_phi[N] giving the the start of each bin (where bins touch, so that over a full azimuthal sweep the end of the last bin corresponds to the start of the first bin).

Alternatively it might give the centre of each bin. [2]

Where bins overlap, are disjointed or non-uniform in width, other Depend variables may be required to provide complete information. There is no fixed naming scheme for further Depend variables, but “Depend_width_i”, “Depend_start_i”, and “Depend_end_i” are suggested as optional identifiers pointing to variables holding bin widths, start values and end values respectively. By default, the Depend_i variable will be used for labelling bins in plotting software.

The syntax does not permit the use of N+1 elements to describe a dimension of size N, even though this would be more efficient for uniform touching bins (i.e. just specifying the N+1 bin boundaries).

Bin_description This describes relationships between the Depend variables and the parent variable, such as the method required to construct the volume of the bin. No syntax is prescribed for the parameter which should be text.

For some data it will be necessary to specify different area and volume factors depending on the slice to be taken through the data. Added Bin_description parameters may be specified by adding _A, _B, _C *etc*. Numeric extensions are not used to avoid confusion with the array dimension.

For example, an array of phase space densities calculated at energies E_i and polar angles θ_j. A suitable Bin_description parameter and described Depend variables could take the value

Bin_description = bin area A[j] is (cos(theta[j]+Wth[j]) - cos(theta[j]))
Bin_description_A = energy bin size is d[i] is (EW[i])
Bin_description_B = bin volume is V[i][j] is (EW[i])(cos(theta[j]+Wth[j]) -cos(theta[j]))
Depend_1 = E
Depend_width_1 = WE
Depend_2 = theta
Depend_width_2 = Wth

In this example the energy dimension is tagged by an array giving the start energies for each energy bin, and another for the bin width. The Bin_description then describes constructing the volume element of the [i][j] element in the data array. Thus, if the data array is in partial densities, then the phase space density is found by dividing the data dn[i][j] by V[i][j].

For fixed bin widths of 100 eV, say, this could have been
Bin_description = Energy bin centre is E[i]
Bin_description_A = Energy bin width is 100 eV
Bin_description_B = bin volume is 100 eV x (cos(theta[j]+Wth[j]) - cos(theta[j]))
Depend_1 = E
Depend_2 = theta
Depend_width_2 = Wth

---

[2] This extends the Depend_0 and Depend_i of the ISTP standard.

Further discussion of Science Data array handling can be found in ”QSAS Array Handling”, S.J.Schwartz, CU-QMW-TN-0015.

3.3.2 Extra Metadata for Dimension variables

Depend variables require the **Sizes** and **Data_type** parameters to be set, and may take the **UNITS**, **SI_conversion** and **Data** parameters in the same way as a data variable. It is, however, also advisable to supply extra information to describe the interaction of the Depend variable with the parent variable.

Bin_location This specifies how the Data in the Depend variable are applied to the variable. The syntax takes a real value between 0.0 and 1.0. The value 0.0 indicates the start of the bin, while 1.0 indicates the end, 0.5 the centre value. For logarithmic scales this is determined as the fraction of (log(end) - log(start)) from log(start) to log(value).

Scaling This takes a text value of linear, logarithmic or inhomogeneous. [3]

Depend variables *do not*, themselves, take Depend variables to describe the arrays.
For example, an array whose first dimension energy in 7 bins might have
Bin_location = 0.5
Scaling = linear
Data = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7
UNITS = keV

3.3.3 Optional Variable Metadata

All CSDS variable attributes may be specified within the variable metadata block, but are not required. The following subset of the standard CSDS attributes are currently used by science tools.

FIELDNAM

CATDESC

SCALMIN

SCALMAX

FILLVAL

Other descriptive metadata in the same syntax may be supplied as deemed appropriate by the teams to describe the data adequately for scientific use.

---

[3] It is distinct from the ‘Scale_type’ attribute attached to the parent variable in the ISTP standard

3.4 Variable Metadata Summary

	time	vector tensor	scalar	array	Dimension
Value_type	Required	Required	Required	Required	Required
Sizes	No	Required	No	Required	Required
Time_format	Required	No	No	No	No
Data	Optional	Optional	Optional	Optional	Optional
UNITS	Required	Required	Required	Required	Required
Frame	No	Required	No	No	Possible
Component_desc	No	Required	No	No	No
SI_conversion	Required	Required	Required	Required	Required
Depend_i	No	No	No	Required	No
Bin_description	No	No	No	Required	No
Bin_location	No	No	No	No	Required
Scaling	No	No	No	No	Required
FIELDNAM	Optional	Optional	Optional	Optional	Optional
CATDESC	Optional	Optional	Optional	Optional	Optional
SCALMIN	Optional	Optional	Optional	Optional	Optional
SCALMAX	Optional	Optional	Optional	Optional	Optional
FILLVAL	Optional	Optional	Optional	Optional	Optional

3.5 Data Records

Data is record oriented with an end of record marker (by default the newline character) denoting the end of each record. The entries for each variable in a record must appear in the order specified by the header descriptors. Each entry within a record must be separated from its neighbours by a comma (”,”). Delimiters must be separated by valid input. Missing entries must be padded by a fill value, and the ‘FILLVAL’ metadata value may be used to identify the fill value used. The value of this fill is left to the discretion of the team generating the data.

Each variable may be multi-dimensional with the number of entries per variable being the product of the dimensionalities. Thus a velocity vector has three entries and a two by two array has four entries, while a sixteen by eight by twenty four array has 3072 entries. It must be recognised that some data products may result in extremely long record lines. The rigid record structure to the data is intended to allow simple generic software to handle all instrument files.

The ordering for arrays within a record is the natural ‘C’ ordering, that is, the last index varies the fastest.

Setting an ‘End_of_record’ character other than the default allows long records to be split across lines with new line characters for human readability. This also allows for software systems with maximum line lengths. On read all new line and carriage return characters together with white space (including tabs) and the end or record marker are deleted. These characters and non-printing characters are thus not allowed values for character data types.

4 Sample CEF

Attached is a sample file in CEF syntax. It includes most CSDS metadata to show the full use of the syntax. A minimal version showing the essential inputs for the same file follows.

4.1 Full example of CEF file

```
!------------------- CEF ASCII File --------------------|
! Free width data entries separated by delimiters       |
!                                                       |
! ASCII Format                                          |
! Native C ordering, last index varies fastest          |
! Blank lines are ignored                               |
! "!" escapes rest of line as comment                   |
!-------------------------------------------------------|
!
File_name = SC_RR_INS_YYYYMMDD_Extn_V01.cef

Attribute_delimiter = ,
File_type = d
Data_delimiter = ,
End_of_record_marker = $

!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!                                  Global Metadata    !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Start_meta = Logical_file_id
Number_of_entries = 1
Entry = SC_RR_INS_YYYYMMDD_Extn_V01.cef
End_meta = Logical_file_id
!
Start_meta = Project
Number_of_entries = 1
Entry = PROJ>LONG PROJECT NAME
End_meta = Project
!
Start_meta = Discipline
Number_of_entries = 1
Entry = SPACE PHYSICS> MAGNETOSPHERIC PHYSICS
End_meta = Discipline
!
Start_meta = Source_name
Number_of_entries = 1
Entry = SC_RR_INS_YYYYMMDD_Extn_V01.cdf
```

```
End_meta = Source_name
!
Start_meta = Data_type
Number_of_entries = 1
Entry = RES>RESOLUTION
End_meta = Data_type
!
Start_meta = Descriptor
Number_of_entries = 1
Entry = INS>LONG INSTRUMENT NAME
End_meta = Descriptor
!
Start_meta = Data_version
Number_of_entries = 1
Entry = 01
End_meta = Data_version
!
Start_meta = Generation_date
Number_of_entries = 1
Entry = YYYY-MM-DDTHH:MM:SS.SSSZ
End_meta = Generation_date
!
Start_meta = Caveats
Number_of_entries = 0
Entry = Dummy header only
End_meta = Caveats
!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!                                         Variables    !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Start_variable = epoch
! scalar, no Size entry
Value_type = epoch
Time_format = ISO
End_variable = epoch

Start_variable = VECTOR_B_FIELD
Sizes =  3
Value_type = float
FIELDNAM = Magnetic field
Frame = vector>gse_xyz
SI_conversion = 1.0e-9>T
UNITS = nT
FILLVAL = -1.0E-10
End_variable = VECTOR_B_FIELD
```

```
Start_variable = B_N_SIGMA
! scalar, no Size entry
Value_type = float
FIELDNAM = Normalised delta B / B
Frame = scalar>na
SI_conversion = 1.0>(ratio)
UNITS =        ! no units - this is contracted to single space on read
FILLVAL = 1.0E-10
End_variable = B_N_SIGMA

Start_variable = He_psd
Sizes =  5, 6
Value_type = float
FIELDNAM = Spin Ave Helium Phase Space Density
Frame = array>na
SI_conversion = (number)
UNITS = counts
FILLVAL = -1.0E-10
Bin_description = bin area A[j] is (cos(theta[j]+30) - cos(theta[j]))
Bin_description_A = energy bin size is d[i] is 1.0e3 eV
Depend_1 = Dimension_E
Depend_2 = Dimension_th
End_variable = He_psd

Start_variable = Dimension_E
Sizes =  5
Value_type = float
Frame = component>cis_rtp
Component_desc = E>energy
FIELDNAM = Energy bin edges
SI_conversion = 1.602e-19>J
UNITS = eV
Bin_location = 0.0
Scaling = linear
Data = 0.0,1.0e3,2.0e3,3.0e3,4.0e3
End_variable = Dimension_E

Start_variable = Dimension_th
Sizes =  6
Value_type = float
Frame = component>cis_rtp
Component_desc = t>polar angle theta
FIELDNAM = Polar bin edges
SI_conversion = 1>degree
UNITS = deg
```

```
Bin_location = 0.0
Scaling = linear
Data = 0.0,30.0,60.0,90.0,120.0,150.0
End_variable = Dimension_th

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!                                        Data         !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!epoch,SAMPLE_VECTOR_B_FIELD,SAMPLE_SCALAR_B
!
! Specifying 00 data records means "unknown" and defaults to all data

! White space between entries (for readability) is safe but not needed
! Time field must be in form e.g. 1995-01-23 02:33:17.235

Start_data = 00
1995-01-23T02:33:17.235Z, 2.7453, -0.15678, 77.456, 2.3475,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T02:33:21.124Z, 2.7453, -0.15678, 72.156, 2.1175,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T02:33:25.921Z, 2.729, -0.15678, 77.456, 3.2128,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T02:33:30.012Z, 2.7453, -0.12343, 72.156, 1e-10,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T02:33:34.235Z, 2.1268, -0.11253, 83.501, 1.1194,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
```

```
1995-01-23T17:44:46.845Z, 12.341, 5.2345, 83.247, 2.1563,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T17:44:50.334Z, 12.341, 5.2345, 83.247, 2.1563,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T17:44:55.112Z, 12.341, 5.2345, 83.247, 2.1563,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T17:44:59.345Z, 12.341, 5.2345, 83.247, 2.1563,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T17:45:03.749Z, 12.341, 5.2345, 83.247, 2.1563,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T17:45:08.153Z, 12.341, 5.2345, 83.247, 2.1563,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
```

4.2 Minimal Example of CEF file

The following file contents describe exactly the same data as the preceding example with the minimum specification required. Some optional blank lines have been retained for readability.

```
Start_variable = epoch
Value_type = epoch
Time_format = ISO
```

```
End_variable = epoch

Start_variable = VECTOR_B_FIELD
Value_type = float
Sizes =  3
Frame = vector>gse_xyz
SI_conversion = 1.0e-9>T
UNITS = nT
End_variable = VECTOR_B_FIELD

Start_variable = B_N_SIGMA
Value_type = float
End_variable = B_N_SIGMA

Start_variable = He_psd
Sizes =  5, 6
Value_type = float
FIELDNAM = Spin Ave Helium Phase Space Density
Frame = array>na
SI_conversion = (number)
UNITS = counts
FILLVAL = -1.0E-10
Bin_description = bin area A[j] is (cos(theta[j]+30) - cos(theta[j]))
Bin_description_A = energy bin size is d[i] is 1.0e3 eV
Depend_1 = Dimension_E
Depend_2 = Dimension_th
End_variable = He_psd

Start_variable = Dimension_E
Sizes =  5
Value_type = float
Frame = component>cis_rtp
Component_desc = E>energy
FIELDNAM = Energy bin edges
SI_conversion = 1.602e-19>J
UNITS = eV
Bin_location = 0.0
Scaling = linear
Data = 0.0,1.0e3,2.0e3,3.0e3,4.0e3
End_variable = Dimension_E

Start_variable = Dimension_th
Sizes =  6
Value_type = float
Frame = component>cis_rtp
Component_desc = t>polar angle theta
```

```
FIELDNAM = Polar bin edges
SI_conversion = 1>degree
UNITS = deg
Bin_location = 0.0
Scaling = linear
Data = 0.0,30.0,60.0,90.0,120.0,150.0
End_variable = Dimension_th

Start_data = 00
1995-01-23T02:33:17.235Z, 2.7453, -0.15678, 77.456, 2.3475,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T02:33:21.124Z, 2.7453, -0.15678, 72.156, 2.1175,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T02:33:25.921Z, 2.729, -0.15678, 77.456, 3.2128,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T02:33:30.012Z, 2.7453, -0.12343, 72.156, 1e-10,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T02:33:34.235Z, 2.1268, -0.11253, 83.501, 1.1194,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T17:44:46.845Z, 12.341, 5.2345, 83.247, 2.1563,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T17:44:50.334Z, 12.341, 5.2345, 83.247, 2.1563,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T17:44:55.112Z, 12.341, 5.2345, 83.247, 2.1563,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T17:44:59.345Z, 12.341, 5.2345, 83.247, 2.1563,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T17:45:03.749Z, 12.341, 5.2345, 83.247, 2.1563,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
1995-01-23T17:45:08.153Z, 12.341, 5.2345, 83.247, 2.1563,
        12.341, 5.245, 83.247, 2.156, 12.341, 5.235,
        13.442, 6.554, 60.244, 9.156, 15.341, 4.245,
        22.341, 5.245, 80.247, 10.163, 16.341, 6.345,
        11.366, 6.235, 73.247, 3.153, 18.341, 7.245,
        10.551, 8.234, 65.247, 2.563, 20.341, 9.235   $
```