

DS-QMW-TN-0010

Date: 21 Sept 2004

Issue: 2

Rev. : 0.3

Page : i

Cluster Exchange Format - Data File Syntax

CSDS Archive Task Group

A. Allen, S.J.Schwartz, C. Harvey, C. Perry, C. Huc, P. Robert

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	Extra detail on array handling
1	5	13 Apr 2004	Modified for Cluster Active Archive
2	0.0	16 Apr 2004	Modified at CAA meeting Toulouse
2	0.1	25 May 2004	Clarification of some issues
2	0.2	12 Aug 2004	Modification of tensor handling
2	0.3	21 Sept 2004	Updated after prototype CAA coding

Contents

1	Introduction	1
2	File Format	1
2.1	Time Series Data	3
2.1.1	Sub-millisecond Timing	4
2.2	Data Series	4
2.3	Metadata Syntax	4
2.4	File Metadata	5
2.5	Including External Files	5
2.6	Global Metadata	6
2.7	Variable Metadata	7
2.7.1	Mandatory Variable Metadata	7
2.7.2	Extra Metadata for Depend variables	13
2.7.3	Optional Variable Metadata	13
2.8	Variable Metadata Summary	14
2.9	Data Records	14
3	Sample CEF	16
3.1	Full example of CEF 2 file	16
3.2	Minimal Example of CEF file	21
A	Main Changes from CEF version 1	24

1 Introduction

A single ascii format data file syntax was recommended by the CSDS Archive Task Group for the exchange of science data between instrument teams. This format was 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.

This format was adopted by the Cluster Active Archive design team as the basis for the format to be used for delivery of science data to the CAA, and as a possible format for delivery to science users. As a consequence configuration control of this document has been transferred to the data Formats working group of the CAA, and the specification has been made more rigorous.

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

This format serves to facilitate the storage of commonly used science products in a robust and easily accessible form. It 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 of the instrument team with different science and database software. It will also facilitate delivery of science products to future scientists without access to specific software in use at the time of data archival.

Software that can translate between the CSDS standard CDF files and CEF files is maintained by the Queen Mary, University of London, Space Physics group. The utility, called **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. The syntax specified in version 2.0 and above of this document is not completely backwardly compatible with earlier versions used for Cluster data exchange. Earlier versions of this syntax will be referred to as cef 1, and the syntax specified for the active archive will be designated cef 2.

2 File Format

Files in this syntax must have names with the extension **.cef** to assist identification.

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.

A header provides sufficient metadata to describe the data and its formatting, and is specified in detail below. The header must be attached and precede the data records. Apart from quoted text data, and variable names, header information is **CASE INSENSITIVE**, so that, for example, entries ‘Depend_0’, ‘DEPEND_0’, and ‘DepEnD_0’ are equivalent.

Item	Ref.	Rule	Comments
CEF file extension	This section	.cef is written in lower case on UNIX and Linux systems	Be careful when copying files from systems which do not distinguish case (PC, Mac, VMS).
Header information	This section	Case insensitive, apart from the exceptions listed below.	For example, entries 'Depend_0', 'DEPEND_0', and 'DepEnD_0' are equivalent.
Text strings	This section	Anything written between quotes is case sensitive.	CAA will suppress case sensitivity when searching for keywords.
Time stamps	Sect. 2.1	The characters 'T' and 'Z' of the ISO time code are case insensitive.	
FILE_TYPE_VERSION	Sect. 2.4	CEF should be in upper case.	"CEF-2.0" will be parsed.
File names	Sect. 2.5	Case sensitive	To be compatible with the extension .cef, lower case is recommended.
Global attribute names	sect. 2.6	Case sensitive	Exception to the general rule.
Variable names	Sect. 2.7	Case sensitive	Exception to the general rule
Enumerated metadata		Case sensitive (data may be parsed)	The values proposed in the CAA Metadata Dictionary, CAA-CDPP-TN-0002, are to be used.

Table 1: Case sensitivity : the general rule is that anything not included in quotes is case insensitive. This table shows exceptions to this rule.

Lines of metadata may be continued using ‘\’ as a continuation marker following one of the commas separating a list of values.

The data records must be immediately preceded by a line which identifies the end of the data. This take the form

```
DATA_UNTIL = yyy
```

where yyy is a quoted string that will be found at the beginning of a line following the last data record in the file, or takes the value EOF (not in quotes) if the data is to be read until the end of file is reached. Spaces, tabs and non-printing characters are not permitted within the string yyy.

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). The **record delimiter** may not be the comment marker (exclamation mark, '!') or ampersand ('&') or a non-printing character. Entries in a record are comma separated and white space surrounding delimiters is ignored.

The record delimiter and all carriage return (\r), new line (\n) and white space characters at the start and end of records and surrounding delimiters are to be 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).

Note that in order to protect white space and commas in text data this data type is enclosed within double quotes " ". Quoted text is used when the text itself is to be used as the value of a parameter. When the text refers to an object, such as a variable name or an entry in a list of reserved names it will not be quoted. Variable names are case sensitive, but other values which refer to lists of reserved values such as data types, will be case insensitive. Such case insensitive parameters and values are shown throughout this document in uppercase characters.

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

2.1 Time Series Data

Time series data must 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. By default, time tags are centred within the sampling interval, and the spacing between timetags is the same as that sampling interval. For other behaviour, use Delta_PLUS and Delta_MINUS metadata in **seconds** to describe the sampling or integration interval corresponding to the data and the location of the time tag within that interval. The use of a second variable holding an absolute time (e.g., to hold the stop time) is possible, and other variables in a record may also themselves be time variables.

Time is represented as a text string in the strict ISO format adopted by CSDS. This is of the form:

```
yyyy-mm-ddTHH:MM:ss.wwwZ
```

it e.g. 1996-01-30T13:30:00.000Z for January 30 1996 at 1.30 pm.

2.1.1 Sub-millisecond Timing

This 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 format is itself robust for arbitrary timing accuracy, but software developers must take care to ensure that internal formats retain the requisite accuracy.

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. Note also that even at millisecond accuracy conversion between the cdf native epoch stored as a double and an ISO time string can result in a 1 msec error.

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.

2.3 Metadata Syntax

Three forms of metadata are recognised. *File metadata* is used to describe or override the default syntax for the data file itself. *Global metadata* has the same purpose as global attributes in a cdf file and describes the file contents collectively. *Variable metadata* is equivalent to the cdf variable attributes and describes each data product individually.

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. Leading and trailing white space is removed from the *value* token before evaluating it. Where the *value* token is text it is enclosed in double quotes, in which case white space is valid within the *value* string.

The *value* token can take three forms, a single data value (which may be quoted text), a comma delimited sequence of data values or the name of a data variable in the same file carrying the requisite information. The simplest of these options that is sufficient to describe the quantity (or quantities) should always be used. A metadata variable should only be used if the metadata itself requires further metadata to describe it, or its value is record varying.

Multiple *value*'s may be continued onto more than one line by using '\ ' as a continuation marker after a comma in a separated list of values. All text beyond the '\ ' is ignored and the next line is concatenated. Continuation markers may NOT be used within a quoted text string as they will be treated verbatim as part of the string.

Any file metadata should appear before global or variable metadata blocks as they affect the reading of subsequent input. Variable metadata blocks must appear in the same order as the variables appear in the data records. There is no preferred order for the global metadata blocks, nor parameters within any metadata block.

2.4 File Metadata

The parameters below describe the file itself.

FILE_NAME Required. This specifies the name of the file. It should not include the path, but does include the file type extension.

FILE_FORMAT_VERSION Required. CEF Specification Version, as a text string in syntax “CEF-n.m” to identify the cef version. In this case “CEF-2.0”.

END_OF_RECORD_MARKER Required if data records are broken across lines. This specifies the character to be used to indicate the end of each data 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 of record marker are stripped after the record is read. Neither the exclamation mark nor ampersand may be used as end of record marker.

2.5 Including External Files

Metadata of any kind can be included from external files by using an *INCLUDE* statement with the following format.

```
INCLUDE = "filename"
```

filename gives the name of the file to be included. It will need to be located by any reading software, but the path information should not be given inside the cef file. The contents of included file must conform to cef syntax and will be included as though they were pasted directly into the parent file in place of the *INCLUDE* statement.

When using included files, the start and end parameters for a variable block or metadata block must be within the same file. For example, **START_META** and **END_META** must be in the same physical file, although some or all of the content within this block may be included ...

```
START_META = TEXT  
INCLUDE = "textEntries.txt"  
END_META = TEXT
```

where the file `textEntries` contains

```
Number_of_entries = 3  
Entry = PEACE Sweep Data generated by QPEACE Software  
Entry = S.J.Schwartz@qmul.ac.uk; http://www.space-plasma.qmul.ac.uk  
Entry = Qpeace Software V2.5 18 March 2004
```

is equivalent to ...

```
INCLUDE = "textBlock.txt"
```

where the file `textBlock.txt` contains ...

```
START_META = TEXT
Number_of_entries = 3
Entry = PEACE Sweep Data generated by QPEACE Software
Entry = S.J.Schwartz@qmul.ac.uk; http://www.space-plasma.qmul.ac.uk
Entry = Qpeace Software V2.5 18 March 2004
END_META = TEXT
```

However, the construct ...

```
START_META = TEXT
INCLUDE = "textPart.txt"
```

where the file `textPart` contains ...

```
Number_of_entries = 3
Entry = PEACE Sweep Data generated by QPEACE Software
Entry = S.J.Schwartz@qmul.ac.uk; http://www.space-plasma.qmul.ac.uk
Entry = Qpeace Software V2.5 18 March 2004
END_META = TEXT
```

is **not** permitted as it splits the start and end parameters for a block.

2.6 Global Metadata

Global attributes are used to provide informational metadata associated with all the variables in the file, and as a means of attaching information that may be carried along with the data.

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 by this syntax. The CAA requires metadata as specified in the CAA Metadata Dictionary, *CAA-CDPP-TN-0002*.

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. This name is case insensitive.

The global metadata block contains the following entries:

ENTRY The value associated with this entry is the next entry in the global attribute. Space characters are valid within a text entry, and text entries must be enclosed in double quotes. 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

ISO_TIME
FLOAT
DOUBLE
INT
CHAR
BYTE

ISO standard date/time strings are used in the same format as used in data records.

2.7 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. The variable name is not in quotes. These blocks are required for variable description and headers are not valid without one describing each variable. Variable entries *must* appear within the header in the same order as the variable themselves within each data record. Variable names are case sensitive, and are an exception to the usual rule of case insensitivity.

2.7.1 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 required that meta-

data sufficient to describe that data for science use is provided. 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

ISO_TIME
FLOAT
DOUBLE

INT
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 comma-separated integer values as there are array indices in the variable with the number giving the size of the array over that index. Thus an 8 by 54 array would have the entry

SIZES = 8,54

It is not required for scalars.

DATA The 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 meta-data 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 comma separated. For array data elements will appear in the natural C ordering - last index varies fastest, and data lines for arrays may be continued using '\ ' as a continuation marker following one of the commas separating the list of values.

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. The text string has 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

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

The system is one of

xyz Cartesian

rtp (r, θ, ϕ) spherical polars - $\theta = 0$ along pole

rlp (r, λ, ϕ) spherical equatorial polars - latitude = λ

rpz (r, ϕ, z) cylindrical polars

rt plane polars

xy plane cartesians

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

The frame is one of

gse Geocentric Solar Ecliptic

gsm Geocentric Solar Magnetic
geoc Geographic (geocentric)
sm Solar Magnetic
gei Geocentric Inertial
magd Geomagnetic (dipole)
hae Heliospheric Aries Ecliptic
hee Heliospheric Earth Ecliptic
heeq Heliospheric Earth Equatorial
gseq Geocentric Solar Equatorial
sc Spacecraft frame
... other frames may be specified, e.g., instrument, ...

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 coordinate frames or those listed in the Metadata Dictionary, CAA-CDPP-TN-0002. The frame specified is that underlying the data representation.

The `FRAME` metadata is retained for compatibility with legacy code, and is more rigorously handled by the metadata `Tensor_FRAME`, `Tensor_RANK` and `REPRESENTATIONi` below.

Tensor_FRAME This is a rigorous generalisation of the `FRAME` attribute for CAA, and allowed values are enumerated in the Metadata Dictionary, CAA-CDPP-TN-0002.

REPRESENTATION_i This is a rigorous generalisation of the `FRAME` attribute for CAA, and allowed values are enumerated in the Metadata Dictionary, CAA-CDPP-TN-0002. It is only used for tensor types (vectors and tensors). There are as many of these attributes as there are indices *i* in the tensor. A vector takes only `REPRESENTATION_1`. This should have the same number of entries as the dimension of index *i*. For example, a cartesian pressure tensor takes

`REPRESENTATION_1 = "x", "y", "z"`

`REPRESENTATION_2 = "x", "y", "z"`

indicating that the components are

`"Pxx", "Pxy", "Pxz", "Pyx", "Pyy", "Pyz", "Pzx", "Pzy", "Pzz"`

A complete tensor in real 3-space will have three components in each index. An incomplete tensor may be specified where the dimension of one or more indices are less than three. For example a vector measured in the xy-plane will have

`REPRESENTATION_1 = "x", "y"`

and this in turn implies it may be rotated in the xy-plane but not around any other axis. This will often duplicate `LABELi`, and is provided distinctly as it identifies `Tensor` type data explicitly, and this has science processing connotations - ability to rotate, for example.

Tensor_RANK This is the number of indices for a tensor data type. Vectors have rank 1.

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 compound 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 caret, \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

H henry [needed for *mu_o*]

F 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].

FIELDNAM Name for the field, generally not as short as LABLAXIS

LABLAXIS Short name suitable for labelling the data. The units need not be included as they are supplied separately in **UNITS**. In the case of vectors and arrays this will be the common stem applicable for all dimensions, for example, a cartesian velocity might have LABLAXIS "V" with the component specified by LABEL_1 (see below).

DEPEND_i and LABEL_i These contain information identifying the i^{th} dimension in an array variable. Either one or the other of these attributes must exist for each index of an array, but never both.

The LABEL_i attribute is used when it is sufficient to provide a text label for each entry for this index. For example a vector such as a velocity in cartesian gse coordinates will take a LABEL_1 attribute such as

LABEL_1 = "x", "y", "z"

and the stem "V" is provided by the LABLAXIS (see above). Similarly, a pressure tensor in cartesians would have

LABEL_1 = "x", "y", "z"

LABEL_2 = "x", "y", "z"

and stem "P" provided by LABLAXIS. This permits construction of labels for individual components from the stem plus appropriate entry under LABEL_i.

By contrast, DEPEND_i will normally point to another variable as it is used when arrays need descriptions that are more complex than labels. Vectors, tensors, and non-ordered arrays, such as Status have LABEL_i attributes rather than DEPEND_i. Quantities such as power spectra and particle spectra will require DEPEND_i attributes to describe numerical quantities such as bin boundaries and other metadata describing the physical quantities associated with that array index (see below). DEPEND_i attributes will in general take a LABLAXIS attribute themselves to provide a label stem for the numerical bin descriptions, but they do not normally require LABEL_i or DEPEND_i attributes 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 (or equivalently LABEL_i for one or more of the indices). 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 for that array index. Thus for an array with the first index varying over azimuthal angular bins, a Depend_1 variable describing these N azimuthal angular bins might point to a variable Dimension_phi of size N, giving the centre of each bin.

Where DEPEND_i is providing information on binning (e.g. angle, energy and frequency ranges) then DEPEND_i will itself also have a DELTA_PLUS and a DELTA_MINUS attribute providing bin edge offsets. The lower bin boundary is given by the appropriate element in DEPEND_i - DELTA_MINUS and the upper boundary by DEPEND_i + DELTA_PLUS. The bin width is given by DELTA_PLUS + DELTA_MINUS.

If binning is regular so that all bins have the same width then a single value may be provided for the DELTA attributes, otherwise they must each have the same number of entries, N, as the dimension of index i. Where bins are open ended, e.g. "above 100MeV", then either DEPEND_i and DELTA_PLUS(MINUS) should represent an estimate of the instrument responsiveness to give an effective location and width, or they should fit a logical progression from lower bins. Descriptive attributes should be provided to explain this to the user.

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).

Complex data values are represented by an extra dimension taking a LABEL_i indicating either "Re" or "Im". If this is the last index then the real and imaginary parts of a value are successive entries in the record.

DELTA_PLUS and **DELTA_MINUS** `DEPEND.i` variables, including the time tags which are `DEPEND_0` for the data, may have these. They describe the range over which the data are integrated, representative, *etc.*, and locate the position of the time tag or value within this range. In the case of time, **DELTA_PLUS** and **DELTA_MINUS** are the number of seconds corresponding to the sampling interval or other representative time interval, given in seconds as a `float`. Alternatively, **DELTA_PLUS** and **DELTA_MINUS** for time variables could themselves be variables, e.g., if they are record-varying, in which case they require their own metadata. For an array of energies used as dependencies for array data, **DELTA_PLUS** and **DELTA_MINUS** used together with the energy value provide a complete description of the energy bin over which the measurement was made or is representative.

It is also advisable to supply extra information to describe the relationships between the `DEPEND` variables with the parent variable, such as the method required to construct the volume of the bin. No syntax is prescribed for these parameters which should be text, but examples of possible descriptors are listed below.

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 parameters may be specified to cover any integration or processing factors appropriate.

For example, consider a 2-D array of phase space densities calculated at centre energies E_i and starting polar angles θ_j and over a constant azimuthal angular range of 15° . A suitable description parameter and described `DEPEND` variables could take the value

```
DEPEND_1 = E
and the variable E block would contain metadata
DELTA_PLUS = WE
DELTA_MINUS = WE
```

where `WE` is another variable (or a single constant entry or a comma separated list of the right length) and

```
DEPEND_2 = theta
and the variable theta block would contain metadata
DELTA_PLUS = Wth
DELTA_MINUS = 0
```

Other attributes may be defined to show the use of these form factors in calculating quantities associated with the binning, such as

```
Theta_Factor = "TFactor[j] is cos(theta[j])-cos(theta[j]+Wth[j])"
Energy_Factor = "EFactor[i] for volume is E[i]^2 *2*WE[i] + (2/3)WE[i]^3"
Volume_Factor = "V[i][j] = EFactor[i]*TFactor[j]* pi * 15/180"
```

In this example the energy dimension is tagged by an array giving the mid energies for each energy bin, and another for the bin half-width. The extra attributes then describe 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 $V[i][j]$. (In practice, the partial density would probably be an integration in velocity and require converting from energy to velocity, with a different $V[i][j]$).

For fixed bin widths of 100 eV, say, the subscript would be removed from the $WE[i]$ in the above, and a separate WE variable would not be used.

2.7.2 Extra Metadata for Depend variables

Depend variables require the **SIZES** and **VALUE_TYPE** parameters to be set, and should take the **UNITS**, **SI_CONVERSION** and **DATA** parameters in the same way as a data variable. They should take a **LABLAXIS** attribute to provide labelling information specific to that index. A **DELTA_PLUS** and **DELTA_MINUS** attribute is also required to establish bin boundaries for **DEPEND_i**.

Depend variables *do not*, themselves, take Depend variables to describe the arrays.

For example, an array whose first dimension is energy in 7 bins might have

```
DEPEND_1 = EnergyBins  
and the associated variable, EnergyBins takes the form  
START_VARIABLE = EnergyBins  
DELTA_PLUS = 0.05  
DELTA_MINUS = 0.05  
SCALING = "linear"  
DATA = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7  
UNITS = "keV"  
LABLAXIS = "E"  
SI_CONVERSION = "1.602e-16>J"  
END_VARIABLE = EnergyBins
```

2.7.3 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 and recommended.

CATDESC The ISTEP Catdesc as used by, e.g., CSDS

SCALMIN Hint to plotting software giving the minimum for a plot scale

SCALMAX Hint to plotting software giving the maximum for a plot scale

FILLVAL Value to be treated as not physically meaningful, indicating a value which is missing or unavailable.

SCALETYP Takes a text value of linear, logarithmic or inhomogeneous as plotting hint

SIG_DIGITS Gives the number of significant digits in the data measurement.

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

2.8 Variable Metadata Summary

	time	vector tensor	scalar	array	DEPEND_i
VALUE_TYPE	Required	Required	Required	Required	Required
SIZES	No	Required	No	Required	Required
UNITS	No	Required	Required	Required	Required
LABLAXIS	Required	Required	Required	Required	Required
FRAME	No	Required	No	No	No
TENSOR_FRAME	No	Required	No	No	No
TENSOR_RANK	No	Required	No	No	No
REPRESENTATION_i	No	Required	No	No	No
COMPONENT_DESC	No	Required	No	No	No
SI_CONVERSION	No	Required	Required	Required	Required
DEPEND_0	No	Possible	Possible	Possible	Possible ¹
DEPEND_i	No	No	No	Required ²	No
LABEL_i	No	Required	No	Required ³	No
DELTA_PLUS	Required (secs)	No	No	No	Required
DELTA_MINUS	Required (secs)	No	No	No	Required
SCALETYP	Optional	Optional	Optional	Optional	Optional
FIELDNAM	Required	Required	Required	Required	Required
CATDESC	Optional	Optional	Optional	Optional	Optional
SCALMIN	Optional	Optional	Optional	Optional	Optional
SCALMAX	Optional	Optional	Optional	Optional	Optional
FILLVAL	Optional	Optional	Optional	Optional	Optional
DATA	Optional	Optional	Optional	Optional	Optional

2.9 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 should 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.

¹Required for DEPEND_i and LABEL_i if record-varying

²Not required if LABEL_i is present

³Not required if DEPEND_i is present

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. The END_OF_RECORD character cannot be either the exclamation mark or the ampersand.

On read all new line and carriage return characters together with white space (including tabs) and the end of record marker are deleted. These characters and non-printing characters are thus not allowed values for character data types.

3 Sample CEF

Below is a sample file in CEF syntax. A minimal version showing the essential inputs for the same data is shown in section 3.2.

3.1 Full example of CEF 2 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 Metadata          !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

FILE_NAME = "SC_RR_INS_YYYYMMDD_Extn_V01.cef"
FILE_FORMAT_VERSION = "CEF-2.0" !this is case insensitive
END_OF_RECORD_MARKER = "$"
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!                               Global Metadata        !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! This metadata is CSDS and ISTEP compliant
! A CAA metadata dictionary compliant version must be produced
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

START_META = Logical_file_id
    ENTRY = "SC_RR_INS_YYYYMMDD_Extn_V01.cef"
END_META = Logical_file_id
!
START_META = Project
    ENTRY = "PROJ>LONG PROJECT NAME"
END_META = Project
!
START_META = Discipline
    ENTRY = "SPACE PHYSICS> MAGNETOSPHERIC PHYSICS"
END_META = Discipline
!
START_META = Source_name
    ENTRY = "SC_RR_INS_YYYYMMDD_Extn_V01.cdf"
END_META = Source_name
```

```
!  
START_META = Data_type  
    ENTRY = "RES>RESOLUTION"  
END_META = Data_type  
!  
START_META = Descriptor  
    ENTRY = "INS>LONG INSTRUMENT NAME"  
END_META = Descriptor  
!  
START_META = Data_version  
    ENTRY = "01"  
END_META = Data_version  
!  
START_META = Generation_date  
    VALUE_TYPE = ISO_TIME  
    ENTRY = 1904-01-23T12:13:14.5678Z  
END_META = Generation_date  
!  
START_META = Caveats  
    ENTRY = "Dummy header only"  
END_META = Caveats  
!  
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  
!                               Variables                               !  
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  
START_VARIABLE = time_tags  
    ! scalar, no Size entry  
    VALUE_TYPE      = ISO_TIME  
    UNITS           = "s"  
    DELTA_PLUS      = 2.0  
    DELTA_MINUS     = 2.0  
    LABLAXIS        = "UT"  
    FIELDNAM        = "Universal Time"  
END_VARIABLE       = time_tags  
  
START_VARIABLE = vector_B_field  
    SIZES           = 3  
    VALUE_TYPE      = FLOAT  
    FIELDNAM        = "Magnetic field"  
    SI_CONVERSION   = "1.0e-9>T"  
    ! if this variable had Frame = "vector>gse_rtp" then we'd have  
    ! SI_conversion = "1.0e-9>T", "1>degree", "1>degree"  
    ! UNITS         = "nT", "deg", "deg"  
    UNITS           = "nT"  
    FILLVAL         = -1.0E-10  
    LABLAXIS        = "B"
```

```

LABEL_1      = "x", "y", "z"
FRAME        = "vector>gse_xyz"
REPRESENTATION_1 = "x", "y", "z"
TENSOR_RANK  = 1
TENSOR_FRAME = "gse"
DEPEND_0     = time_tags
END_VARIABLE = vector_B_field

START_VARIABLE = B_n_sigma
! scalar, no Size entry
VALUE_TYPE    = FLOAT
FIELDNAM      = "Normalised delta B / B"
LABLAXIS     = "dB/B"
FRAME        = "scalar>na"
SI_CONVERSION = "1.0>(ratio)"
UNITS        = "unitless"
FILLVAL      = -1.0E-10
DEPEND_0     = time_tags
END_VARIABLE = B_n_sigma

START_VARIABLE = He_psd
SIZES          = 5, 6
VALUE_TYPE    = FLOAT
FIELDNAM      = "Spin Ave Helium Partial Density"
LABLAXIS     = "N_He"
FRAME        = "array>na"
SI_CONVERSION = "1.e6>(number) m^-3"
UNITS        = "/cc"
FILLVAL      = -1.0E-10
Theta_Factor = "TFactor[j] is cos(theta[j])-cos(theta[j]+Wth[j])"
Energy_Factor = "EFactor[i] for volume is E[i]^2 *2*WE[i] + (2/3)WE[i]^3"
Volume_Factor = "V[i][j] = EFactor[i]*TFactor[j]* pi * 15/180"
Bin_volume_use = "DivideN_He by V[i][j] to get phase space density"
DEPEND_0     = time_tags
DEPEND_1     = Dimension_E
DEPEND_2     = Dimension_th
END_VARIABLE = He_psd

START_VARIABLE = Dimension_E
SIZES          = 5
VALUE_TYPE    = FLOAT
FIELDNAM      = "Energy bin edges"
LABLAXIS     = "E"
SI_CONVERSION = "1.602e-19>J"
UNITS        = "eV"
DELTA_PLUS   = 1.0e3
```

```
DELTA_MINUS      = 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
FIELDNAM         = "Polar bin edges"
LABLAXIS         = "Theta"
SI_CONVERSION    = "1>degree"
UNITS            = "deg"
DELTA_PLUS       = 30.0
DELTA_MINUS      = 0.0
SCALING          = LINEAR
DATA             = 0.0,30.0,60.0,90.0,120.0,150.0
END_VARIABLE     = Dimension_th
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!                                     Data                                !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

! White space between entries (for readability) is safe but not needed
! Time field must be in form e.g. 1995-01-23T02:33:17.235Z

```
DATA_UNTIL = "End_of_file"
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 \$

End_of_file

3.2 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.

```
FILE_NAME = "SC_RR_INS_YYYYMMDD_Extn_V01.cef"
FILE_FORMAT_VERSION = "CEF-2.0"
END_OF_RECORD_MARKER = "$"
START_VARIABLE=time_tags
  VALUE_TYPE=ISO_TIME
  DELTA_PLUS=2.0
  DELTA_MINUS=2.0
  UNITS="s"
  LABLAXIS="UT"
  FIELDNAM="Universal Time"
END_VARIABLE=time_tags
START_VARIABLE=vector_B_field
  SIZES=3
  VALUE_TYPE=FLOAT
  FIELDNAM="Magnetic field"
  FRAME="vector>gse_xyz"
  REPRESENTATION_1="x","y","z"
  TENSOR_RANK=1
  TENSOR_FRAME="gse"
  SI_CONVERSION="1.0e-9>T"
  UNITS="nT"
  LABLAXIS="B"
  LABEL_1="x","y","z"
  DEPEND_0=time_tags
END_VARIABLE=vector_B_field
START_VARIABLE=B_n_sigma
  VALUE_TYPE=FLOAT
  FIELDNAM="Normalised delta B / B"
  LABLAXIS="dB/B"
  SI_CONVERSION="1.0>(ratio)"
  UNITS="unitless"
  DEPEND_0=time_tags
END_VARIABLE=B_n_sigma
START_VARIABLE=He_psd
  SIZES=5,6
  VALUE_TYPE=FLOAT
  FIELDNAM="Spin Ave Helium Partial Density"
  LABLAXIS="N_He"
```

```
FRAME="array>na"
SI_CONVERSION="1.e6>(number) m^-3"
UNITS="/cc"
DEPEND_0=time_tags
DEPEND_1=Dimension_E
DEPEND_2=Dimension_th
END_VARIABLE=He_psd
START_VARIABLE=Dimension_E
SIZES=5
VALUE_TYPE=FLOAT
FIELDNAM="Energy bin edges"
LABLAXIS="E"
SI_CONVERSION="1.602e-19>J"
UNITS="eV"
DELTA_PLUS=1.0e3
DELTA_MINUS=0.0
DATA=0.0,1.0e3,2.0e3,3.0e3,4.0e3
END_VARIABLE=Dimension_E
START_VARIABLE=Dimension_th
SIZES=6
VALUE_TYPE=FLOAT
FIELDNAM="Polar bin edges"
LABLAXIS="Theta"
SI_CONVERSION="1>degree"
UNITS="deg"
DELTA_PLUS=30.0
DELTA_MINUS=0.0
DATA=0.0,30.0,60.0,90.0,120.0,150.0
END_VARIABLE=Dimension_th
DATA_UNTIL=EOF
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 \$

A Main Changes from CEF version 1

Summarised below are the main changes from the previous cef specification (which we shall denote CEF-1.0) and CEF-2.0. Software tools should consult the File Metadata parameter `FILE_FORMAT_VERSION` which is required from CEF-2.0 onwards.

- Files require a File Metadata entry `FILE_FORMAT_VERSION` to specify the version of CEF to which it complies. Note that CEF-2.x is NOT backward compatible with CEF-1.0.
- Generally the specification is more rigid/prescriptive
- Times can be given to arbitrary accuracy (though this does not guarantee that software tools will be able to retain arbitrary accuracy without due care and attention).
- INT data type has been added
- The time data type has been changed to `ISO_TIME`
- For time `DELTA_PLUS` and `DELTA_MINUS` variables are measured in seconds, not milliseconds.
- Text metadata and data are enclosed in double quotation marks.
- Within metadata comma-separated lists line continuation markers (`'\'`) have been introduced
- All keywords and enumerated types have been defined as case insensitive (and have been formatted in this document in capitals).
- Metadata names have been modified and added to be compatible with the Cluster Active Archive Metadata Dictionary. This activity is ongoing.
- Array-handling metadata syntax (e.g., `LABLAXIS`, `LABEL.i`, `DEPEND.i`, `DELTA_PLUS`, `DELTA_MINUS`, and others) has been modified.
- All metadata can take one of: a single value, a comma-separated list of values, or the name of a variable.
- Headers are required to be attached, but an `INCLUDE` parameter has been introduced to read in metadata from another file.
- `Start_data` has been replaced by `DATA_UNTIL` with an optional text value which corresponds to the text starting the last line of the file.