LA-UR-19-24891
Approved for public release; distribution is unlimited.

Title: Sesame ASCII File Format

Author(s): Young, Ginger Ann

Intended for: Distribute to scientists who need to create Sesame ASCII data.
Web

Issued: 2019-06-18 (rev.1)

LOS ALAMOS NATIONAL LABORATORY

Sesame ASCII File Format

Version 1.21

Ginger Young
6/13/2019

1. Introduction

1.1 Intended Audience

The intended audience for this document is a Sesame IO library developer or anyone trying to write or parse Sesame ASCII files with no regards to the meaning of the actual content. LA-UR-92-3407 “SESAME: The Los Alamos National Laboratory Equation Of States Database” is the current document for describing the data contained in Sesame Database.

1.2 Purpose

This document describes the physical format in Sesame’s ASCII files. It may have a little information about the data in these files. However, this data is only cursory.

Historically Sesame ASCII files are fixed format based on old 80-character punch cards. A stack of punch cards is called a card deck. Card decks were read into a computer using card readers and the first version(s) of FORTRAN had fixed file formats. Sesame ASCII files originate from this. Each ASCII File Format file represents a card deck.

1.3 Contact Information/SRS Team Members

The original developer for this library is Hilary Abhold, retired. Currently, Ginger Young is the developer.

2. Overall Description

2.1 Overview of Records

The Sesame ASCII format is unique because the position of a record and the record type describes what the record contains. The location of a record in the file actually defines a field in the description record.

Think of a static file as a description in time, where the first line (record) is written at the beginning instance of time, and the last line at the end of the time creating this file.

This means that the first line of a file read is in fact the first record in time. The placement of records in a file not only describes data, but the type and amount of data that follows. This is important to note since this an artifact of the data being originally on physical decks of cards. The parsing of the data is

based on those old card decks and may appear to be nonsense from our current perspective. Remember this while reading this document. Now back to describing records.

A description record is always placed before its comment or data record(s). A description record may have one or more comment or data records. Comment and data records have their own description record. Records are in an old fixed format restricted to an 80-column space, which means fields are not space delimited.

2.2 Record Types

Description record describes a defined number of sequential records below. Its fields are right justified and must be exactly 80 characters. A description record's Table ID and Num. Words indicate if comment or data records follow and how many records are expected. Only comment records or data records may follow a description record.

Comment records contain descriptive ASCII text about the material and are at most 80 characters long. All comment records except for the last comment record must have 80 characters. If you desire a comment with fewer than 80 characters and you want another comment after it, you must add spaces to create an 80-character record. Only the last comment record may have less than 80 characters.

Data records contain floating-point data and a 5-digit boolean mask. This data may be longer than 80 characters. There are two lengths of floating-point data supported: medium & short.

There are three placements for a record within a file: initial, middle, and end. Each material described in this file composes of an initial record and middle records. The file contains one end record which signifies an end of file.

2.2.1 Description Record

All description record's fields are right justified. This record looks like:

Field	File Number	Material ID	Table ID	Num. Words	r	Creation Date	Update Date	Version	Spaces	File Number
Number Chars	2	6	6	6	4	9	9	4	32	2
Indices	1-2	3-8	9-14	15-20	21-24	25-33	34-42	43-46	47-78	79-80

File Number: Please note that this field is at the beginning and end of each description record.

Contains 2 characters, the first is a space the second can be: 0, 1, or 2, where:

- ' 0' – Initial description for each material.
- ' 1' – All middle description records.
- ' 2' – End of file marker.

Each record has the above values in character indices: 1,2 and 79,80. For instance, the initial description record has a '0' at index 2 & 80.

Material ID: 6 characters (integer), a unique material identifier. This field is right justified and the value may have leading spaces and/or one to more leading zeros.

Table ID: 6 characters (integer), ASCII legacy Table IDs contains: 3 spaces and then 3 integers. The integer characters can be:

Table ID	Description
101	Comments – Must be created.
102 - 199	Comments
201	Atomic Number, Atomic Mass, Normal Density
301	Total EOS (304 + 305 + 306)
303	Ion EOS Plus Cold Curve (305 + 306)
304	Electron EOS
305	Ion EOS (Including Zero Point)
306	Cold Curve (No Zero Point)
401	Vaporization Table
411	Solid Melt Table
412	Liquid Melt Table
431	Shear Modulus Table
502	Opacity Grid Boundary: Calculated vs. Interpolated
502	Rosseland Mean Opacity (cm^2g^{-1})
503	Electron Conductive Opacity (cm^2g^{-1})

504	Mean Ion Charge[1] (free electrons per atom)
505	Planck Mean Opacity (cm^2g^{-1})
601	Mean Ion Charge[2] (Free electrons per atom)
602	Electrical Conductivity (sec^{-1})
603	Thermal Conductivity ($cm^{-1}sec^{-1}$)
604	Thermoelectric Coefficient ($cm^{-1}sec^{-1}$)
605	Electron Conductive Opacity[2] (cm^2g^{-1})

[1]Opacity Model (Hubbard-Lampe)

[2]Conductivity Model (Ziman)

Num. Words: 6 characters (integer), number of words in the following data. There can be 999,999 maximum words per data. This field is right justified with leading spaces.

r: 4 characters, indicator flag where ‘r’ stands for read. This flag is no longer used. It is always 3 spaces then the character r.

Creation Date: 9 characters (integer), format is **MMDDYY** and has three leading spaces.

Update Date: 9 characters (integer), format is **MMDDYY** and has three leading spaces.

Version: 4 characters (integer), right justified.

Spaces: 32 spaces

File Number: 2 characters, where the last character is ‘ 0’, ‘ 1’, ‘ 2’.

2.2.2 Comment Record

Comment records contain ASCII text and they are at most 80 characters long. All comment records must be 80 characters long, except for the last record. The last may be less than 80 characters. Any comment record can be padded by spaces, if the author so desires.

The description record for a comment table must have a Table ID of 101 to 199, and Num. Words is the number of bytes (characters) in all of the comment records. For instance, if there are 4 comment records of 80 characters apiece, Num. Words = 320.

2.2.3 Data Record

Data records contain floating-point data for the material. The data is in scientific notation. There are no spaces between each value, up to 5 values per line, and a 5-digit boolean mask at the end. The number of data records and floating-point values in each depends on how many values there are for the current description record. All floating-point values are left justified. The 5-digit boolean mask is always at the same index depending on floating-point length.

If a description record has 15 Num. Words, there are 3 data records. These 3 records have 5 floating-point values in each.

As another example, if the description record has 36 Num. Words, there are 8 data records. There are 7 records with 5 floating-point values, and then a data record with 1 floating-point value and spaces for 4 floating point values until the 5-digit boolean mask.

Note: data records always contain 5 floating-point values until the last data record. The last data record may contain 5 floating-point values or fewer. All data records end with a boolean mask.

There are two lengths of floating-point values: short and medium. Records containing short are 80 characters long, while medium are 115 characters long. Short and medium formats are not mixed; a file contains either one or the other.

Data is either **medium** or **short** in length, and each value looks like:

Type	Sign	Data	Max. Chars/Line
Medium	‘ ‘/’-‘	2.123456789012345E+01	5*22=110
Indices	1	23456789012345678 9012	
Short	‘ ‘/’-‘	2.12345678E-01	5*15=75

Medium floating-point values contain 22 characters per floating-point value. For each value the 1st character is either a space for a positive number or a minus sign for a negative number. The 2nd character is an integer. The 3rd a decimal followed by 15 integers. The 19th character is ‘E’. The 20th character denotes a positive or negative exponent. The 21st and 22nd digits are the exponents. The medium value above is +/-2.123456789012345 X 10^{1} or +/-21.23456789012345. Each record has at most 5 floating-point values and then a 5-digit boolean mask. The mask always starts at the index of 111.

Short floating-point values contain 15 characters. The 1st character is either a space for a positive number or a minus sign for a negative number. The 2nd character is an integer. The 3rd a decimal followed by 8 integers. The 12th character is ‘E’. The 13th character denotes a positive or negative exponent. The 14th and 15th digits are the exponents. The short value above is +/-2.12345678 X 10^{-1} or .212345678. Each record has at most 5 floating-point values and then a 5-digit boolean mask. This mask always starts at the index of 76.

Boolean Mask is a 5-digit value at the end of each data record. The digit may be a ‘1’ (true) or ‘0’ (false). The ancient purpose of this mask was a positional checksum for the floating-point values.

The value of the mask indicates whether or not a floating-point value is expected. All floating-point values are left justified in a record. The value of the mask can only be: 11111, 11110, 11100, 11000, or 10000. Where a ‘1’ indicates there is a floating-point value.

A boolean mask value of ‘11111’ means there are 5 floating-point values in that data record. While the boolean mask value of ‘11000’ means there are 2 floating-point values in that data record. Any data record with fewer then 5 floating-point values must be the last record for that description record.

2.3 Positional Records

Positional records are description records with the **File Number** indicating the position: ‘ 0’ the initial record, ‘ 1’ all of the middle records, and ‘ 2’ the end record. The end record only has data in **File Number** all of the other fields are spaces.

2.3.1 Initial Material Record

An initial record is always the first item of a Sesame ASCII file. Also it must be the initial record for each material. This record must start and end with the **File Number** of ‘ **0’**. That’s a space and zero. Since this is also a description record, it also contains information about the following comment or data records.

2.3.2 Middle Records

All of the middle records start and end with the **File Number** of ‘ **1’**. This is a space and one. Unlike initial and end records, a Sesame ASCII file may have multiple middle records. A middle record whose **Table ID** is describing data must have the data immediately after the record. The data must be the number of words long described in **Num. Words**.

2.3.3 End Record

An end record contains only the **File Number** of ‘ **2**’. This is a space and two. The area in between two file numbers in a record are filled with space. To be clear there are 76 spaces between the first ‘ 2’ and the ending ‘ 2’. This is an end of file indicator, which is required.

3. Examples

3.1 Medium

Here is an example of an ASCII file with **medium** sized words. This example contains two materials.

```
 0     2   101   184   r    61813    61813   1                                  0
material. omega-zirconium (z= 40.000000 a= 91.224000) /source. Scott Crockett an
d Carl Greeff /date. june 13 /refs.  /comp. omega-zirconium /codes. opensesame /
classification.  /
 1     2   102   224   r    61813    61813   1                                  1
Modified data corresponding to first and second density for firt two isotherms i
n order to validate fix for the EOSPAC6 inverse interpolation problem described
in the TeamForge artf38112: inverse interpolation problem.
 1     2   201     8   r    61813    61813   1                                  1
 4.000000000000000E+01 9.122400000000000E+01 6.591329000000000E+00 0.000000000000000E+00 0.000000000000000E+0011111
 4.938500000000000E+01-9.452800000000000E-02 0.000000000000000E+00                                              11100
 1     2   401     4   r    61813    61813   1                                  1
-7.748592047557000E-01-7.947264759670790E-02 0.000000000000000E+00-7.748639204858263E-02                       11110
 0  3031   101    10   r   122193    60794   1                                  0
12abcdefgh
 1  3031   103    10   r   122193    60794   1                                  1
098765abcd
 1  3031   201     5   r   122193    60794   1                                  1
 1.999999999999989E+01 4.007999999999970E+01 1.546999999999976E+00 0.000000000000000E+00 0.000000000000000E+0011111
 1  3031   301    55   r   122193    60794   1                                  1
 5.000000000000000E+00 3.000000000000000E+00 1.000000000000000E+01 2.000000000000000E+01 3.000000000000000E+0111111
 4.000000000000000E+01 5.000000000000000E+01 1.010000000000000E+01 1.020000000000000E+01 1.030000000000000E+0111111
 2.010000000000000E+01 2.020000000000000E+01 2.030000000000000E+01 2.040000000000000E+01 2.050000000000000E+0111111
 2.060000000000000E+01 2.070000000000000E+01 2.080000000000000E+01 2.090000000000000E+01 2.100000000000000E+0111111
 2.110000000000000E+01 2.120000000000000E+01 2.130000000000000E+01 2.140000000000000E+01 2.150000000000000E+0111111
 3.010000000000000E+01 3.020000000000000E+01 3.030000000000000E+01 3.040000000000000E+01 3.050000000000000E+0111111
 3.060000000000000E+01 3.070000000000000E+01 3.080000000000000E+01 3.090000000000000E+01 3.100000000000000E+0111111
 3.110000000000000E+01 3.120000000000000E+01 3.130000000000000E+01 3.140000000000000E+01 3.150000000000000E+0111111
-4.440000000000000E+01 4.020000000000000E+01-4.030000000000000E+01 4.040000000000000E+01-4.050000000000000E+0111111
 4.060000000000000E+01-4.070000000000000E+01 4.080000000000000E+01-4.090000000000000E+01 4.100000000000000E+0111111
-4.110000000000000E+01 4.120000000000000E+01-4.130000000000000E+01 4.140000000000000E+01-4.150000000000000E+0111111
 2                                                                             2
```

3.2 Short

Here is an example of an ASCII file with **short** sized words:

```
 0   031   101   184   r    61813    61813   1                                  0
material. omega-zirconium (z= 40.000000 a= 91.224000) /source. Scott Crockett an
d Carl Greeff /date. june 13 /refs.  /comp. omega-zirconium /codes. opensesame /
classification.  /
 1   031   102   225   r    61813    61813   1                                  1
Modified data corresponding to first and second density for first two isotherms
in order to validate fix for the EOSPAC6 inverse interpolation problem described
in the TeamForge artf38112: inverse interpolation problem.
 1   031   301     5   r    61813    61813   1                                  1
 4.00000000E+01-9.12240000E+01 6.59132900E+00 0.00000000E+00 0.00000000E+0011111
 2                                                                             2
```