SUN/102.4

Starlink Project
Starlink User Note 102.4

Malcolm J. Currie

2021 February 25

HDSTRACE — Listing HDS Data Files
1.2
User's Guide

Abstract

This document describes HDSTRACE, an application that lists the contents of an HDS file.

Contents

1 Introduction

One of the most popular features of the ADAM environment is the Hierarchical Data System (HDS) (described in SUN/92nd SG/4). HDS enables associated data items to be stored together in a single file and in a structured fashion. HDS is also highly flexible, allowing a myriad of ways to organise your data. So HDS is jolly useful, but it does have one drawback—you cannot just type or print an HDS file to see its contents—you must run a programme. Obviously, you (or your friendly programmer) could write some code to list a particular structure, but this would be inefficient given the number of structures that are already in use, let alone the ones to come. What is required is a single utility that lists the contents of an HDS data structure to your terminal and, optionally, to a file for printing or documentation. This is where HDSTRACE comes in handy.

HDSTRACE lists recursively the name, data type, and values of an HDS structure or object. The type is bracketed by <> to distinguish it from the name (*cf.* SGP/38). Indentation delineates structures. The format of the output is flexible; and more importantly, you can now always see the data values—something that was often impossible with original version. You control the formatting by specifying optional parameters on the command line, otherwise they have sensible defaults.

2 Running HDSTRACE

This document adheres to the normal style, where your input follows the $, % or > prompts.

2.1 UNIX

HDSTRACE runs as an ADAM task. ADAM must therefore be initiated directly or indirectly using the `ADAMSTART` or `ADAM` command before HDSTRACE can be run.

HDSTRACE may be run directly from the UNIX shell with the following command.

```
% hdstrace
```

You will be prompted for the HDS object or structure to be traced.

If you wish to trace several files or objects it is more efficient to run HDSTRACE from ICL. If you are already using ICL just enter

```
ICL> HDSTRACE
```

If you are in DCL, the following will start both ICL and HDSTRACE.

```
$ ADAM HDSTRACE
```

The command can be abbreviated to HDST and the old command `TRACE` will also work from both the shell and ICL.

2.2 UNIX

HDSTRACE may be also run directly from a UNIX shell,

```
% hdstrace
```

There is no initialisation procedure to run first, but the command may not be abbreviated.

2.3 Getting Help

The following command gives help on HDSTRACE.

```
ICL> HELP HDSTRACE
```

Notice that this only available for ICL since there is no global help system for Starlink applications. However, it is possible to access the full help on HDSTRACE from the UNIX shell by entering `??` in response to a prompt for a parameter.

```
% hdstrace
OBJECT - Object to be examined > ??
```

2.4 Parameters

All but one of the parameters used by HDSTRACE are normally defaulted. If you wish to be prompted for all or some of the defaulted values give the `PROMPT` keyword on the command line; you can terminate the prompting by entering a \ at any prompt and each remaining parameter will take its default value. Normally, this should not be necessary as most of the time the defaults will be satisfactory or just one or two need changing by specifying them on the command line. Prompting does have one advantage in that a `?` may be entered to elicit help on that parameter.

Parameters may be given by keyword, *e.g.*

```
% hdstrace object=myfile nlines=3
```

or by position, *e.g.*

```
% hdstrace myfile no 3
```

where the `no` is for the FULL parameter, or by a combination of both provided the positional parameters come before any keywords, *e.g.*

```
% hdstrace myfile nlines=3
```

Full details of the parameters, their defaults and their command-line positions are given in the Appendix.

2.4.1 Tailoring HDSTRACE

Beginners can skip over this section.

If you want to alter some of the defaults, or always make HDSTRACE prompt for a normally defaulted parameter, you can have your own version of the interface file. To achieve this enter the following from the UNIX shell.

```
% cd ~abc/xyz                     # Substitute the actual disk and directory
                                  # where you keep private versions of IFLs
% cp $HDSTRACE_DIR/hdstrace.ifl .
% vi trace.ifl                    # Make the changes you wish.
% setenv ADAM_IFL ~abc/xyz:$ADAM_IFL # Append the directory to the
                                  # interface-file search path.
```

SUN/115 tells you how to interpret the interface file; it describes the meanings and available options of the various keywords, and the use of the ADAM_IFL environment variable. The most likely things that you would wish to alter are the values of *default* and *vpath*.

SUN/144 has more details of the ADAM_IFL environmental variable.

The best way to describe the parameters is to show some examples.

3 Examples

In the following examples a a UNIX shell is used, but the commands would also work from ICL except where noted.

3.1 Getting Started

The first HDS file is quite simple. It is an old IMAGE-format file may still be used in KAPPA. Note that on UNIX the HDS file name is case sensitive.

```
% hdstrace beq

BEQ  <IMAGE>
  TITLE          <_CHAR*72>      'KAPPA - Glitch'
  DATA_ARRAY(109,64)  <_REAL>    124,92,98,156,171,166,171,181,190,180,
                                 ... 127.32,127.56,128.36,129.4,130.08,130.44
  DATA_MIN       <_REAL>         12
  DATA_MAX       <_REAL>         254

End of Trace.
```

The first line tells us that the structure's *name* is BEQ and its *type* is IMAGE. (Note that the structure's name is not necessarily the file name.) The <> delimiters are just a convention to differentiate name from type; they are not part of the type itself. The next few lines are indented. HDSTRACE uses indentation to indicate the position within the hierarchy. Here TITLE,

DATA_ARRAY *etc.* are at the top level of the hierarchy within the *container file*. (If they are at the top, why is BEQ negatively indented? BEQ applies to the file as a whole—a kind of zeroth level. An analogy is your login directory. It is at the top of your hierarchy of files in directories. However, it is a file itself stored on disc.) The second line shows that the TITLE is a character object of length 72, and has a *value* of `'KAPPA - Glitch'`. The third line shows an array. The dimensions follow the name thus DATA_ARRAY has 109 columns and 64 lines. Since there is insufficient room to list the values of 6976 elements, HDSTRACE lists the first few and last few values separated by an ellipsis to indicate that there are missing values.

We can look at a specific object or structure. For example to look at just the data array in beq enter:

```
% hdstrace beq.data_array

BEQ.DATA_ARRAY  <_REAL>
  DATA_ARRAY(109,64)  124,92,98,156,171,166,171,181,190,180,196,205,217,
                      ... 126.6,127.32,127.56,128.36,129.4,130.08,130.44

End of Trace.
```

The case of the object name you supply does not matter since in HDS all object names are in uppercase.

We can look at more of the DATA_ARRAY as follows.

```
% hdstrace beq.data_array nlines=3

BEQ  <IMAGE>
  TITLE          <_CHAR*72>      'KAPPA - Glitch'
  DATA_ARRAY(109,64)  <_REAL>    124,92,98,156,171,166,171,181,190,180,
   196,205,217,211,204,210,202,197,201,196,194,192,189,190,191,187,196,
   202,202,206,204,202,191,172,150,*,131,176,192,204,206,182,170,164,161,
   ... 127.16,126.2,126.36,126.6,127.32,127.56,128.36,129.4,130.08,130.44
  DATA_MIN       <_REAL>         12
  DATA_MAX       <_REAL>         254

End of Trace.
```

The `nlines=3` specifies that up to a maximum of three lines, excluding any extra line that shows the final few values, can be used to present the values. In order to present more values the continuation lines are indented one column right of the object's name. Notice `nlines=3` has no effect on the other objects in the structure because their values each fit onto a line; had there been other arrays these too may have up to three lines to list their values. The NLINES parameter normally defaults to one, hence we usually just see the initial values on a single line.

The observant reader will have noticed an asterisk replacing a numerical value in the third line. This is shorthand for a bad value, and it has the added advantage that it is more distinctive.

For inspection of two-dimensional arrays and subsets GAIA or LOOK in KAPPA may be more convenient.

If the array was much smaller we could display all the values on a single line.

```
% hdstrace SMALL.DATA_ARRAY

SMALL  <IMAGE>
  TITLE          <_CHAR*72>      'KAPPA - Pick2d'
  DATA_ARRAY(3,2)  <_REAL>       124,92,98,156,156,149

End of Trace.
```

3.2 Formatting

If you wish to use the output for some documentation you might like to align the trace into neat columns. In the above examples the type of the DATA_ARRAY is displaced because of the dimensions. The indentation of the type with respect to the start of the name, and the values with respect to the start of the type may be controlled by Parameters TYPIND and VALIND respectively. For example,

```
% hdstrace beq typind=20

BEQ  <IMAGE>
  TITLE               <_CHAR*72>      'KAPPA - Glitch'
  DATA_ARRAY(109,64)  <_REAL>         124,92,98,156,171,166,171,181,190,
                                      ... 127.56,128.36,129.4,130.08,130.44
  DATA_MIN            <_REAL>         12
  DATA_MAX            <_REAL>         254

End of Trace.
```

indents the type five characters more than the default.

There is another way to format the output, and that it to place the values on a new line rather than appending them to the name and type. This can help when dealing with moderate-sized arrays.

```
% hdstrace beq newline

BEQ  <IMAGE>
  TITLE          <_CHAR*72>
   'KAPPA - Glitch'
  DATA_ARRAY(109,64)  <_REAL>
   124,92,98,156,171,166,171,181,190,180,196,205,217,211,204,210,202,197,
   ... 127.16,126.2,126.36,126.6,127.32,127.56,128.36,129.4,130.08,130.44
  DATA_MIN       <_REAL>
   12
  DATA_MAX       <_REAL>
   254

End of Trace.
```

A record of the trace may be stored in an text file given by the Parameter LOGFILE. The width of the output defaults to the screen width and may be altered via Parameter WIDTH.

3.3 Structures

Let us move on to a different example HDS file to demonstrate some of the other parameters of HDSTRACE.

```
% hdstrace $ADAM_USER/GLOBAL valind=18

GLOBAL  <STRUC>
  LUT             <ADAM_PARNAME>     {structure}
     NAMEPTR         <_CHAR*132>        'xmascomet_lut'

  IMAGE_OVERLAY   <ADAM_PARNAME>     {structure}
     NAMEPTR         <_CHAR*132>        'ikonov'

  GRAPHICS_DEVICE  <ADAM_PARNAME>    {structure}
     NAMEPTR         <_CHAR*132>        'ikon'

  IMAGE_DISPLAY   <ADAM_PARNAME>     {structure}
     NAMEPTR         <_CHAR*132>        'ikon'

  DATA_ARRAY      <ADAM_PARNAME>     {structure}
     NAMEPTR         <_CHAR*132>        'cac1d'

  HDSOBJ          <_CHAR*132>        'AST_ROOT:<DATA.DEMO>PSS_DEMO'
  SST_SOURCE      <_CHAR*132>        'TRACE.FOR'

End of Trace.
```

This HDS file stores the values of global parameters shared by ADAM applications. LUT, GRAPHICS_DEVICE *etc.* are structures, and therefore instead of listing values, HDSTRACE writes the comment `{structure}`. Each of these structures contains one object, NAMEPTR, which is indented in the trace. HDSTRACE inserts a blank line after listing the contents at the end of a structure for clarity.

Structures can also be arrays, as in the following example of an NDF.

```
% hdstrace moimp

MOIMP  <NDF>
  DATA_ARRAY(512,512)  <_REAL>   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                                 ... 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
  TITLE           <_CHAR*16>     'created with pro'
  UNITS           <_CHAR*18>     'NORMALIZED       B'
  AXIS(2)         <AXIS>         {array of structures}

  Contents of AXIS(1)
     DATA_ARRAY(512)  <_REAL>       -3542.08,-3512.186,-3482.291,
                                    ... 11644.38,11674.27,11704.17,11734.06
     LABEL            <_CHAR*18>    'PIXELS           B'

  MORE            <EXT>          {structure}
     FITS(31)        <_CHAR*80>     'SIMPLE  =                    T / St...'
                                    ... 'HISTORY  ESO-DESCRIPTOR...','','END'
```

```
End of Trace.
```

AXIS is an array of structures. Therefore it has no values and HDSTRACE puts the comment `{array of structures}`. By default HDSTRACE only lists the contents of the first element of an array of structures. To obtain a full trace of arrays of structures you must set the Parameter FULL to be true.

```
% hdstrace MOIMP FULL

MOIMP  <NDF>
  DATA_ARRAY(512,512)  <_REAL>   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                                 ... 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
  TITLE          <_CHAR*16>      'created with pro'
  UNITS          <_CHAR*18>      'NORMALIZED       B'
  AXIS(2)        <AXIS>          {array of structures}

  Contents of AXIS(1)
     DATA_ARRAY(512)  <_REAL>       -3542.08,-3512.186,-3482.291,
                                    ... 11644.38,11674.27,11704.17,11734.06
     LABEL          <_CHAR*18>      'PIXELS           B'

  Contents of AXIS(2)
     DATA_ARRAY(512)  <_REAL>       11734.1,11704.21,11674.31,11644.42,
                                    ... -3482.252,-3512.146,-3542.041
     LABEL          <_CHAR*18>      'PIXELS           B'

  MORE           <EXT>           {structure}
     FITS(31)       <_CHAR*80>      'SIMPLE  =                    T / St...'
                                    ... 'HISTORY  ESO-DESCRIPTOR...','','END'

End of Trace.
```

3.4 Character Arrays

In MOIMP there is an array of characters—the NDF FITS extension. If HDSTRACE cannot accommodate a string in the space available it attempts to give the first few characters followed by an ellipsis to indicate only a part of the string is reported. Thus in the above trace only part of the first FITS header card is shown. Part of the anti-penultimate header is reported. The last two lines are blank and 'END' and are listed in full. These truncations are not very convenient. HDSTRACE provides a mechanism for placing each character-array element on a new line.

```
% hdstrace moimp.more.fits newline eachline

MOIMP.MORE.FITS  <_CHAR*80>
  FITS(31)
   'SIMPLE  =                   T / Standard FITS format', ...
   'END'

End of Trace.
```

To list the whole array specify `nlines=all` or give an NLINES with at least the number of array elements. Note the object need not necessarily be the array. Below we trace the MORE extension.[1]

```
% hdstrace moimp.more nlines=a newline eachline

MOIMP.MORE  <EXT>
  FITS(31)       <_CHAR*80>
   'SIMPLE  =                    T / Standard FITS format',
   'BITPIX  =                   32 / No. of bits per pixel',
   'NAXIS   =                    2 / No. of axes in image',
   'NAXIS1  =                  512 / No. of pixels',
   'NAXIS2  =                  512 / No. of pixels',
   'EXTEND  =                    T / FITS extension may be present',
   'BLOCKED =                    T / FITS file may be blocked',
   ' ',
   'BUNIT   = 'NORMALIZED       B  ' / Units of data values',
   'BSCALE  =   7.450580607332E-06 / Scaling factor: r = f*i + z',
   'BZERO   =   1.600000000000E+04 / Zero offset: r = f*i + z',
   ' ',
   'CRPIX1  =   1.000000000000E+00 / Reference pixel',
   'CRVAL1  =  -3.542080000000E+03 / Coordinate at reference pixel',
   'CDELT1  =   2.989460000000E+01 / Coordinate increment per pixel',
   'CTYPE1  = 'PIXELS           B  ' / Units of coordinate',
   'CRPIX2  =   1.000000000000E+00 / Reference pixel',
   'CRVAL2  =   1.173410000000E+04 / Coordinate at reference pixel',
   'CDELT2  =  -2.989460000000E+01 / Coordinate increment per pixel',
   'CTYPE2  = 'PIXELS           B  ' / Units of coordinate',
   ' ',
   'ORIGIN  = 'ESO-MIDAS'          / Written by MIDAS',
   'OBJECT  = 'created with pro'   / MIDAS desc.: IDENT(1)',
   ' ',
   'HISTORY  ESO-DESCRIPTORS START   ................',
   'HISTORY  'HEADER_VERS'    ,'C*1 '   ,    1,   13,'13A1',' ',' '',
   'HISTORY  06-APR-1990',
   'HISTORY',
   'HISTORY  ESO-DESCRIPTORS END     ................',
   ' ',
   'END'

End of Trace.
```

The `nlines=all` facility should be used with care; select an individual component or a structure that does not contain a large array. The output from a data array such as found in a typical NDF will be huge.

3.5 Current Value

If you wish to examine the same object repeatedly perhaps with other parameters changed you can use the ACCEPT keyword or its shorthand. Thus

[1] FITSLIST in KAPPA offers a more-convenient way of listing the FITS headers within NDFs.

```
% hdstrace full \\
```

would give a full trace of the last object traced.

A Reference Section

HDSTRACE
Examines the contents of a data-system object

Description:

Data files in ADAM are stored in an hierarchical format (HDS). This cannot be read by just typing the file at the terminal or spooling it to a printer—a special application is required. Now rather than writing separate code to read a variety of structures, this application is sufficiently general to examine almost all HDS structures or objects. The examination may also be written to an text file as well as being reported to the user.

The version number of the HDS data format used by the supplied file may also be displayed. See parameter VERSION.

For the specified ADAM data-system object (*X*) there are three cases which are handled:

1) *X* is a *primitive* object. The value, or the first and last few values of *X* are listed.

2) *X* is a *structure*. The contents of the structure are listed. If a component is encountered which is itself a structure then its contents are listed down to a level of six nested structures.

3) *X* is an *array of structures*. All elements will be listed if Parameter FULL is set to `TRUE`; only the first element will be listed when Parameter FULL is set to `FALSE` (default).

Listings are in the following order: name; dimensions, if any; type; and value or comment. Comments are enclosed in braces.

The values are normally listed at the end of each line, but may start on a new line. The maximum number of lines of data values may also be set. For all but the smallest arrays where the values of all elements can be displayed in the space provided, the last few values in the array as well as the first few are presented. The last few values appear on a new line, indented the same as the line above with the ellipsis notation to indicate any missing values. Note the number of elements shown depends on the number of characters that will fit on the line. The ellipsis notation is also used for long character values where there is only room to show the first and last few characters. Bad values appear as asterisks.

The exact layout may be adjusted and is controlled by four additional parameters: a) the indentation of the type string with respect to the beginning of the name string; b) indentation of the value(s) (if not on a new line) with respect to the beginning of the type string; and c) the width of the output. If the name and dimensions do not fit within the space given by parameters a) and b), the alignment will be lost because at least two spaces will separate the name from the type, or the type from the value(s). The fourth parameter defines how character arrays are arranged. The default is that character-array elements are concatenated to fill the available space delimited by commas. The alternative is to write the value of each element on a new line. This improves readability for long strings.

Usage:

```
hdstrace object [full] [nlines] [typind] [valind] [logfile] [eachline] [newline]
         [width] [widepage] [version] [sorted]
```

Parameters:

EACHLINE = _LOGICAL (Read)
If `true`, the elements of a character array will each appear on a separate line. Otherwise elements fill the available space and may span several lines, paragraph style. `[FALSE]`

FULL = _LOGICAL (Read)
If `true`, all the contents of an array of structures will be traced, otherwise only the first element is examined. `[FALSE]`

HDSVERSION = _INTEGER (Write)
An output parameter in which is placed the version number of the HDS data format used by the supplied file. See also parameter VERSION.

LOGFILE = FILENAME (Read)
The name of the text file to contain a log of the examination of the data object. Null (`!`) means do not create a log file. `[!]`

NEWLINE = _LOGICAL (Read)
True indicates that data values are to start on a new line below the name and type, and indented from the name. Otherwise the values are appended to the same line. `[FALSE]`

NLINES = LITERAL (Read)
The maximum number of lines in which data values of each primitive array component may be displayed, but excluding the continuation line used to show the last few values. Note that there may be several data values per line. There is no formatting of the values. If you require the whole of each array use NLINES = `"ALL"`. Beware this facility can result in a large report, so select just the array or arrays you wish to trace. `[1]`

OBJECT = UNIV (Read)
The name of the data-system object to be traced. This may be a whole structure if the name of the container file is given, or it may be an object within the container file, or even a sub-section of an array component.

SORTED = _LOGICAL (Read)
If `true`, list structure components in sorted order. `[FALSE]`

TYPIND = _INTEGER (Read)
Column indentation of the component's type with respect to the current indentation of the component's name. If the name plus dimensions cannot fit in the space provided alignment will be lost, since HDSTRACE insists that there be a gap of at least two columns. Note that HDS names can be up to 15 characters, and the dimension in the format (dim1,dim2,...) is abutted to the name. `[15]`

VALIND = _INTEGER (Read)
Column indentation of the component's value(s) with respect to the current indentation of the component's type provided NEWLINE is `false`. If, however, NEWLINE is `true`, VALIND is ignored and the value is indented by one column with respect to the component's name. If the type cannot fit in the space provided alignment will be lost, since HDSTRACE insists that there be a gap of at least two columns. HDS types can be up to 15 characters. `[15]`

VERSION = _LOGICAL (Read)
If true, the version number (an integer) of the HDS data format used by the supplied file is appended to the end of the trace. It is also stored in output parameter HDSVERSION. `[FALSE]`

WIDEPAGE = _LOGICAL (Read)
If true a 132-character-wide format is used to report the examination. Otherwise the format is 80 characters wide. It is only accessed if WIDTH is null. `[FALSE]`

WIDTH = _INTEGER (Read)
Maximum width of the the output in characters. The default is the screen width of a terminal (up to the maximum message buffer, currently 300 characters), and 80 for a file. `[]`

Notes:

This application allows far more flexibility in layout than earlier applications like LS and the original TRACE, though the order of the attributes of an object has been fixed and rearranged for standardisation, particularly for documentation purposes.