National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, MD

GEOMS idlcr8hdf User's Guide

Authors
Ian Boyd, University of Massachusetts
Christian Retscher, USRA, NASA/GSFC

NASA, BC Scientific Consulting GESTAR Task 114

Original: June 17, 2005
Revised: September 25, 2013
Version: 4.0

Document Profile Information

Title:	GEOMS idlcr8hdf User's Guide
Issue	v4.0
Issue date:	September 25, 2013
Status:	Draft
Audience:	Data providers
Reference(s):	B.R. Bojkov, De Mazière, M. and R. Koopman, Generic metadata guidelines on atmospheric and oceanographic datasets for the ENVISAT Calibration and Validation Project, Version 01R001, April 23, 2002. C. Retscher et al., Generic Earth Observation Metadata Standard (GEOMS), Version 1.0, March 21, 2011. HDF 4.2r3 User's Guide, ftp://ftp.hdfgroup.org/HDF/Documentation/HDF4.2r3, The HDF Group, January 2008. ISO, "Representation of Dates and Times", ISO 8601:1988, International Organization for Standardization (ISO), Geneva, Switzerland, (1988). IDL Scientific Data Formats, http://www.ittvis.com/portals/0/pdfs/idl/sdf.pdf, ITT Visual Information Solutions, May 2009.
Filename:	idlcr8hdf-v4.0-readme.pdf
Distribution:	Public
Author(s):	Christian Retscher (UMBC, NASA/GSFC, USA, Christian.Retscher@nasa.gov) Ian Boyd (NIWA, New Zealand, i.boyd@niwa.com)
Contributor(s):	Bojan Bojkov (ESA/ESRIN, Italy, Bojan.Bojkov@esa.int)

Table of Contents

1 Overview

NASA has developed the idlcr8hdf.pro and idlcr8hdf.sav programs to support users writing data to files that conform to Generic Earth Observation Metadata Standard (GEOMS) metadata and data structure requirements for the exchange of data (*Retscher et al.*, 2011). Groups using this standard include the Aura Validation Data Center (AVDC), Network for Detection of Atmospheric Composition Change (NDACC), and ESA Earth Observation missions supported by the ENVISTA Validation Data Centre (EVDC).

idlcr8hdf writes scientific measurements to HDF4 or HDF5 files. It requires, as input, a Table Attribute Values (TAV) file, and either a Metadata file and Data file(s), or the Metadata Global Attributes in an array, and the Data and Metadata in a heap structure using pointers. It writes the inputs to an HDF file as multi-dimensional (maximum of 8 dimensions) Scientific Data Sets. It uses predefined HDF routines for manipulating HDF files, and has been tested on IDL versions 6.4 through to 8.2. The program requires IDL version 6.0 or later to run, and version 6.2 or later if creating HDF5 files. Due to differences in the HDF versions, the HDF4 library should be HDF4.2r3 or later, and the HDF5 library should be 1.6.1 or later[1]. Files containing these libraries, and instructions for installation, are available for download from http://avdc.gsfc.nasa.gov or by contacting Ian Boyd or Ghassan Taha.

Inputs to idlcr8hdf can either be by IDL command line, shell script or batch file (licensed or 'DEMO' version of IDL (IDL LIC)) or by DIALOG_BOX prompts (IDL LIC, or IDL Virtual Machine (IDL VM)). For users who do not have an IDL license, IDL VM can be downloaded free from http://www.exelisvis.com. The resulting download is the full version of IDL. The user can install IDL VM only, or the full version (including IDL VM). The full version can only be used in DEMO mode without a license. IDL VM can be used freely without a license and executes the binary version of the code (idlcr8hdf.sav). Refer to Chapter **4** for details.

idlcr8hdf is also a core program for other GEOMS supported programs, including geoms_harmon, geoms_qa, and cds_convert. These programs, respectively, convert HDF files, created using obsolete standards, to current GEOMS standards; perform QA on GEOMS compliant HDF files; and convert measurements written in formats supported by other datacenters (e.g. WOUDC, NASA/Ames (NDACC), GlobWave) to GEOMS compliant HDF.

2 I want to run this program NOW

There are several ways to run this program. However, if you have previously used the EVDC's asc2hdf program or are otherwise familiar with the format of the Metadata and Data files required for input to this program, then idlcr8hdf can be run in its simplest form, as follows:

[1] To determine the HDF4 library used on your version of IDL call HDF_LIB_INFO, Version=ver. For HDF5 call res=H5_GET_LIBVERSION() & print, res.

- Download the latest Table Attribute Values (TAV) file.
- At the IDL command line type .r idlcr8hdf or, if using IDL VM in a windows environment, drag and drop the idlcr8hdf.sav icon or file onto the idlrt.exe icon, or open IDL VM, and select idlcr8hdf.sav. For Mac or Unix type idl –vm = /your/path/to/idlcr8hdf.sav at the Xterm shell prompt. If the path and filename are omitted (typing only idl –vm), a file selection dialog box will open allowing you to select the file.
- At the Introduction window, select Check Boxes as required, and click on the HDF4 or HDF5 button.
- At the first File Selection Box select the Metadata File for input.
- At the second File Selection Box select the ASCII Data File(s) for input.
- At the third File Selection Box select the TAV file.
- At the Directory Selection Box select the Output Directory, to which any log and HDF files will be written.

If all inputs are valid the program will perform integrity checks and, if successful, create the HDF file(s). If the Popup option is chosen, a log window shows progress through the operation. The program will stop once all input data files have been checked, or an error is detected that compromises the ability of the program to do further checks. To close the program, select the Finish button. If the Popup option is not chosen, a DIALOG_BOX will open at the end of the processing, indicating completion of the program.

An important design feature of idlcr8hdf is that it can generate multiple HDF files with a single call (important if using IDL VM). In this case the inputs are a **single** metadata template file; multiple associated ASCII data files created with an external program; the Table Attribute Values file; and the output directory.

This is possible as most attribute values in a metadata file do not change from measurement to measurement – the ones that do can be determined by the program. Therefore, as long as these values are left empty the program will calculate them.

There are ten metadata global and variable attribute labels where the values can be determined directly by idlcr8hdf. These are:

- DATA_START_DATE (determined from the data)
- DATA_STOP_DATE (determined from the data)
- DATA_TEMPLATE (determined from the DATA_SOURCE value)
- FILE_NAME (determined from metadata values)
- FILE_GENERATION_DATE (determined by the program)
- FILE_META_VERSION (determined by the program)
- VAR_SIZE (determined from the size of the corresponding dataset, or from the VAR_SIZE(s) of the corresponding VAR_DEPEND variable(s))
- VAR_SI_CONVERSION (linked to the VAR_UNITS values)

- VAR_VALID_MIN/VAR_VALID_MAX (for datasets that have VAR_UNITS=MJD2K only – the program will determine these values based on the dataset entries).

Once the metadata template file is configured correctly for a particular instrument/species/location/PI, it should generally not need changing from measurement set to measurement set, allowing generation of more than one HDF file from a single call to idlcr8hdf.

To read about other options for running this program, please refer to Chapters **3** and **4**. A discussion of the Metadata and Data formats is included in Chapter **6**. **Please take note of the different conventions between IDL and HDF for the ordering of multi-dimensional datasets, and how this is treated by the program, in Section 6.2**. The user should also refer to the GEOMS document for a detailed description of the GEOMS reporting standards.

3 Running idlcr8hdf.pro on IDL LIC

3.1 Run-time Options

idlcr8hdf.pro can be run as a standalone program or called by another program, by compiling the program (or including the directory containing the program in the !PATH environment) and calling the main procedure, idlcr8hdf, at the IDL prompt or from another program. The full command line, including parameters is:

idlcr8hdf [,GA,SDS,TAV,ODIR[,RETERR][,/H5][,/AVK][,/Log][,/Popup] [,/QA],[,/NoHDF]]

Possible run-time options are:

- Calling the procedure with no parameters (idlcr8hdf) – An 'Introduction' pop-up box will be displayed. The user has the option of continuing with DIALOG_BOX prompts, or stopping the program. This is equivalent to running the program on IDL VM (see Chapter **4**).
- Session Memory Option – Inputs are a string array containing the Global Attributes, a heap structure containing the Variable Attributes and Data, the TAV filename, and the output directory for files created by the program. If wanting to use pop-up DIALOG_BOX prompts to choose the TAV filename and output directory, then do not include 'TAV' and 'ODIR' in the command line, e.g. idlcr8hdf, GA, SDS_Structure.
- Input File Option – Inputs are Metadata, Data and TAV filenames as well as an output directory. The data file input can be a string array containing the names of multiple data files, or a scalar string giving the name of a file spec (e.g. 'input*.data'), or a scalar string giving the name of a single file (e.g. idlcr8hdf, 'Metafile', ['Datafile_1','Datafile_2'], 'TAVfile', 'Odir' or idlcr8hdf,'','','','' etc). If no input parameters are provided then pop-up DIALOG_BOXES will prompt for the inputs (using the IDL function DIALOG_PICKFILE), otherwise the program tests

for the existence of the files and the directory, and if any of the inputs are not valid, opens DIALOG_BOXES to prompt for new inputs.

If intending to use the DIALOG_PICKFILE function to select input files, the user can set the 'PATH', 'FILE' and 'FILTER' function parameters to reflect the directory and filename structure on his/her computer. This function is used in the idlcr8hdf procedure.

The output HDF file will be named based on the contents of the Metadata file, and placed in the output directory. If an HDF file of the same name already exists, it will be overwritten without any warnings. All integrity checks made by the program are done before HDF file creation.

If input is in the form of an array and a heap structure, the user may want to free up memory, used by the SDS heap structure from previous calls to the program, by calling the PTR_FREE procedure in his/her own program (e.g. PTR_FREE, SDS.Data, SDS.VA). Input in this form can only contain information for one HDF file.

The program automatically checks for the type of input (session memory or files), by testing the SDS parameter. If it is a structure then it is assumed input is from session memory. If it is a scalar string or string array, then it is assumed that input is from files. If idlcr8hdf is called with no parameters, it is assumed that inputs will be in the form of files if the user continues beyond the 'Introduction' dialog box.

If compiling idlcr8hdf on IDL6.1 or lower, the module AVDC_HDF5_WRITE may show one or more compilation errors. This is to be expected as HDF5 write routines were introduced in IDL from version 6.2. The program will still run successfully, but the option to write HDF5 files will not be available. A program generated error message will be displayed if idlcr8hdf is called with the /H5 option from the command line. If the 'Introduction' display window is opened, the HDF5 Widget Button will be insensitive.

3.2 Command Line Input Parameters

3.2.1 **GA**

3.2.1.1 Session Memory Option

A one-dimensional string array, of size NG_ATTS, containing the Global Attributes (where NG_ATTS is the number of Global Attributes). Each entry to the array consists of one Global Attribute, defined through a metadata statement, in the form:

label = value

where label is the name of the attribute, and value is the corresponding value of that attribute. See Section **6.1**, or the GEOMS document, for more information on the composition of the Global Attributes.

3.2.1.2 Input File Option

This is a file containing the Metadata (both the Global and Variable Attributes). Each Metadata Attribute is defined through a Metadata statement in the file. A Metadata statement has the form:

label = value

where label is the name of the attribute, and value is the corresponding value of that attribute. See Section **6.1**, or the GEOMS document, for more information on the Metadata file.

3.2.2 **SDS**

3.2.2.1 Session Memory Option

This is a structure, using pointers, containing the user's scientific metadata and data. It contains N_SDS x NV_ATTS pointer arguments, where N_SDS is the number of datasets making up a single data measurement, and NV_ATTS is the number of variable attributes for each dataset.

The heap structure sent to the procedure is expected to consist of at least three storage structures named sds.data, sds.va_l, and sds.va_v, where sds is the heap structure name; .data holds the datasets; .va_l holds the Variable Attribute Labels; and .va_v holds the Variable Attribute Values.

The datasets can either be in the form of formatted strings, or as the type given by the corresponding VAR_DATA_TYPE value for that dataset. The individual datasets can either have dimensions corresponding to the VAR_SIZE values, or a single dimension containing the total number of data values. For example, if the VAR_SIZE value for a dataset is 10;10, then the corresponding structure can either be represented as a two dimensional 10 x 10 array, or a single dimension of size 100. The dataset is expected to be in the '0' NV_ATTS index position i.e. sds[N_SDS,0].data.

The variable attributes must be separated into labels and values. This is necessary so that numeric variable attribute values can be written to the structure in the correct data type although, as for the datasets, numeric attributes may also be written to the structure as formatted strings. Refer to section **6.1**, or the GEOMS document, for more information on the composition of the Variable Attributes.

The heap structure must contain the Data and the Variable Attributes in the same order. For example, the Data in the structure *sds[12,0].data must correspond to the Variable Attribute information in the structures *sds[12,*].va_l and *sds[12,*].va_v.

More information on the connection between the data, as shown in the heap structure, and the indices of a multi-dimensional variable, is given in Section **6.2**.

3.2.2.2 Input File Option

This can be a single filename containing the user's scientific data (e.g. measurements or model data), or a set of filenames described by a single file spec (e.g. input*.data), or a set of single filenames saved in a string array. The file(s) could be the output from an Excel spreadsheet, or from another program, which has converted the user's original data to an ASCII text format.

Each line has one value only. The value is either a header value or a data value. The header value is a text string containing the name of a variable corresponding to a VAR_NAME attribute value in the metadata file (without any quotes around it). See Section **6.2** for more information on the composition of the data file.

3.2.3 TAV

The Table Attribute Values (TAV) file is an ASCII file containing tables of all legal values for a set of metadata attributes. The file is organized as a set of tables, each starting with the name of the table followed by a set of lines, each beginning with the equal character (=).

This program identifies the version number of the TAV file and performs integrity (consistency) checks on the contents of the metadata file based on the values given in the file.

Non-comment lines in the file can be divided into two categories:

- A label line containing the name of a table
- A legal value line containing an equal character (=) followed by an allowed value of the corresponding metadata attribute (label)

An example of a label line and a set of corresponding legal value lines could be:

```
! This is a comment
VAR_DATA_TYPE
=REAL;32-bit floating point numbers
=DOUBLE;64-bit floating point numbers
=BYTE;8-bit unsigned integers
=SHORT;16-bit signed integers
=INTEGER;32-bit signed integers
=STRING;character string
# Another comment
```

If an attribute label is in the TAV file, only values listed in the legal values list are allowed. On the other hand, if an attribute label is not in the TAV file, any value is allowed. According to this definition the metadata statement, VAR_DATA_TYPE = REAL is allowed, but not, for example, VAR_DATA_TYPE = FLOAT.

The legal value line after the '=' character may be either empty or may consist of only white space characters (blank or horizontal tab characters). In this case any empty string, or a line consisting of only white space characters, will be a legal value for the corresponding attribute (label) in the metadata file.

3.2.4 ODIR

This is the directory to which any files created by idlcr8hdf will be written. The files include the HDF file(s) generated by the program as well as the file idlcr8hdf.log if the Log keyword is included. If the input Metadata and Data are in the form of files, shortcut values can be used if the desired output directory is the same as the directory containing the Metadata file (use 'M') or the Data file (use 'D') e.g. idlcr8hdf,<Metadata file>,<Data file>,<TAV file>,'M'.

3.2.5 RetErr

This is an optional string argument. When idlcr8hdf detects an error that forces it to discontinue, it will typically stop all operations and return to the IDL prompt. By including this argument when idlcr8hdf is called by another program, an error message will be returned to the calling program allowing operations in that program to continue. Note that the Pop-up option will be deselected if present together with this argument. This option is not available when using the graphical user interface.

3.2.6 /H5 Keyword

Instead of creating a standard format AVDC HDF4 file, a compatible HDF5 file is created. It can be used, for example, together with idlcr8ascii to convert AVDC HDF4 files to HDF5 files. Global Attributes and Datasets (with associated Variable Attributes) are written to the 'root' group. This option is only available on IDLv6.2 or greater.

3.2.7 /AVK Keyword

In the event that there are multi-dimensional Averaging Kernel data present in the measurement, the program will append a sentence to the relevant VAR_NOTES in the Metadata indicating the correct array order, if this option is chosen. The sentence reads: 'First three AVK values for the lowest altitude level are: x.xx x.xx x.xx'. If the dataset consists of more than one set of Averaging Kernels, then the sentence reads: 'First three AVK values for the lowest altitude level for the first measurement are: x.xx x.xx x.xx'. This option may be useful, for example, when using a metadata template file as input together with more than one data file. The program assumes that the first three values for the lowest altitude level would be equivalent to the first three values if the multi-dimensional array were written in a single column. It is up to the user to ensure correct array order of the Averaging Kernels in the input data.

3.2.8 /Log Keyword

This option will cause the program to write input/output information to the file idlcr8hdf.log. If this file does not exist in the output directory then it will be created, otherwise the information will be appended to the file. The information is the same as that written to the IDLDE output log window and the pop-up display window (if this option is chosen), and also includes start and stop date and time of the program.

idlcr8hdf is set-up to display a run-time log in different ways, as follows:

- Default log output is to the IDL DE output log window if idlcr8hdf is called from another program within IDL, or from the IDL Command Input Line.
- There is no default log output if idlcr8hdf.sav is opened in IDL VM. The only way to monitor progress during run-time is to choose the Popup option.
- If the Log option is selected, output will also be to the file idlcr8hdf.log.
- If the Popup option is selected, output will also be displayed in a Pop-up window.

3.2.9 /Popup Keyword

This option will cause the program to write input/output information to a Pop-up window. Select Finish to end the program. Note that if idlcr8hdf will be called multiple times by another program in a single runtime, then choosing this option may cause the program to stop after the first call.

3.2.10 /QA Keyword

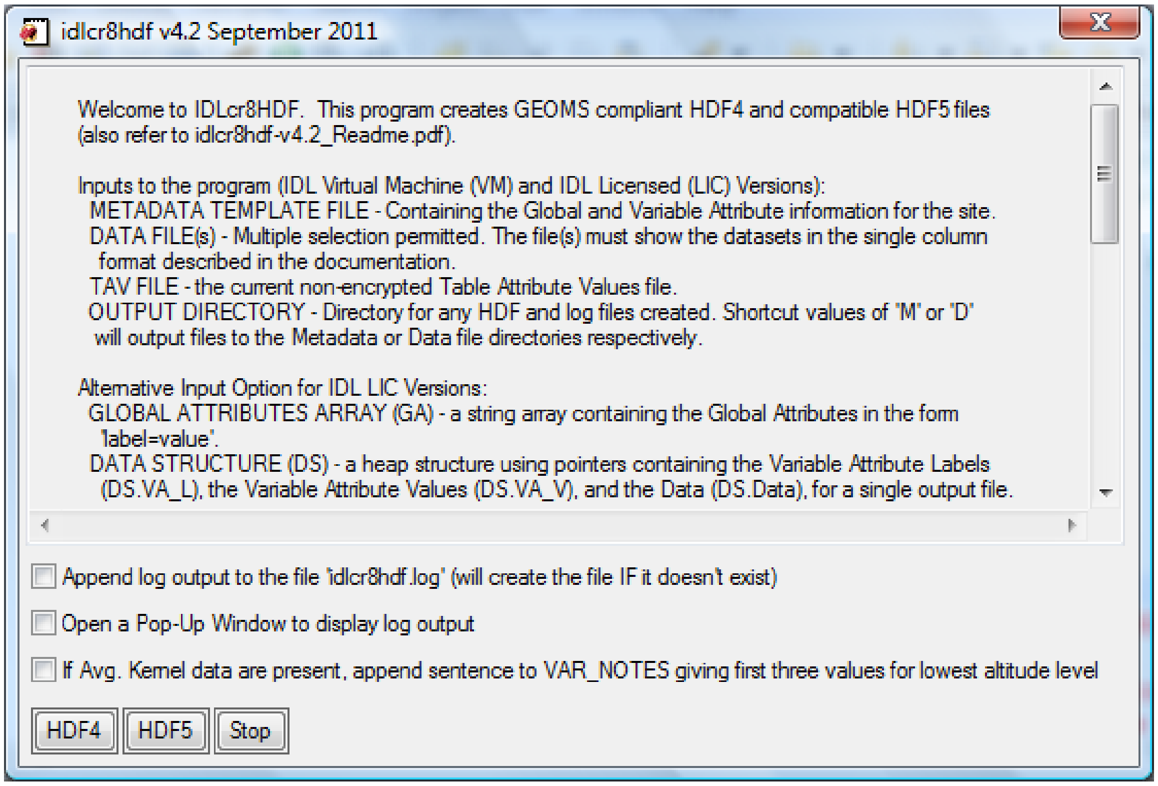
This option will perform Quality Assurance on an HDF file passed to the program using the session memory option. It is used when idlcr8hdf is called by the geoms_qa program. However another option could be to first run idlcr8ascii with the ga and sds arguments, then call idlcr8hdf in session memory mode with these same arguments. A log-file is automatically generated and is called idlcr8qa.log instead of idlcr8hdf.log. This option is not available when using the graphical user interface.

3.2.11 /NoHDF Keyword

This option will perform all the integrity checks on the input data but does not generate an HDF file. The log-file idlcr8hdf.log is automatically generated. This option is not available when using the graphical user interface.

4 Running idlcr8hdf.sav on IDL VM or IDL LIC

The file idlcr8hdf.sav is the binary version of idlcr8hdf.pro and can be run on IDL VM or IDL LIC. To run it on IDL LIC in the manner described in Chapter 3, the routines need to be restored (restore, 'idlcr8hdf.sav'), and the main procedure called (idlcr8hdf). To run on IDL VM in Windows, or if using IDL VM v7.0 or greater: Drag and drop the idlcr8hdf.sav icon or file onto the idlrt.exe icon, or open IDL VM, and select idlcr8hdf.sav. In Mac OS X or Unix the following command at the Xterm shell prompt can be executed: idl -vm=/your/path/to/idlcr8hdf.sav. If the path and filename are omitted (typing only idl –vm), IDL will open a file selection dialog allowing you to select the file. The following 'Introduction' window will be displayed:



Welcome to IDLcr8HDF. This program creates GEOMS compliant HDF4 and compatible HDF5 files (also refer to idlcr8hdf-v4.2_Readme.pdf).

Inputs to the program (IDL Virtual Machine (VM) and IDL Licensed (LIC) Versions):
METADATA TEMPLATE FILE - Containing the Global and Variable Attribute information for the site.
DATA FILE(s) - Multiple selection permitted. The file(s) must show the datasets in the single column format described in the documentation.
TAV FILE - the current non-encrypted Table Attribute Values file.
OUTPUT DIRECTORY - Directory for any HDF and log files created. Shortcut values of 'M' or 'D' will output files to the Metadata or Data file directories respectively.

Alternative Input Option for IDL LIC Versions:
GLOBAL ATTRIBUTES ARRAY (GA) - a string array containing the Global Attributes in the form 'label=value'.
DATA STRUCTURE (DS) - a heap structure using pointers containing the Variable Attribute Labels (DS.VA_L), the Variable Attribute Values (DS.VA_V), and the Data (DS.Data), for a single output file.

☐ Append log output to the file 'idlcr8hdf.log' (will create the file IF it doesn't exist)

☐ Open a Pop-Up Window to display log output

☐ If Avg. Kernel data are present, append sentence to VAR_NOTES giving first three values for lowest altitude level

Output options are selected by a combination of exclusive and non-exclusive buttons as follows:

- Log Output Option – This is equivalent to choosing the /Log option at the command line (refer to section 3.2.8). This option is non-exclusive, meaning other options are still available if this is selected.
- Popup Output Option – This is equivalent to choosing the /Popup option at the command line (refer to section 3.2.9). This option is non-exclusive.
- Avg. Kernel Option – This is equivalent to choosing the /AVK option at the command line (refer to section 3.2.7). This option is non-exclusive.
- HDF4 – output will be HDF4. This button is exclusive, meaning no other options are available once it is selected.
- HDF5 – output will be HDF5. This button is exclusive. It will be shaded if this option is not available. This is equivalent to choosing the /H5 option at the command line (refer to section 3.2.6).
- Stop or X – the Program will close.

Once the HDF4 or HDF5 option is selected, the program will prompt for input file and directory selection by Dialog Boxes. Inputs are: The Metadata Template file (refer to section 3.2.1.2 and 6.1), the Data file(s) (refer to section 3.2.2.2 and 6.2), the TAV file (refer to section 3.2.3), and the Output Directory (refer to section 3.2.4).

If all inputs are valid the program will perform integrity checks and, if successful, create the HDF file(s). If the Popup option is chosen, a log window, like the one below, shows progress through the operation. The program will stop once all input data files have been checked, or an error is detected that compromises the ability of the program to do further checks. To close the program, select the Finish button. If the Popup option is not chosen, a DIALOG_BOX will open at the end of the processing, indicating completion of the program.

5 Event Log Display Options

An event log is generated by the program, showing:

- Start and Finish times of the program (for the default and file output options)
- Input files called and output files created
- Error and Information messages

Displaying the log is dependent both on how idlcr8hdf is called, as well as the options chosen by the user, as follows:

- Default log output is to the IDL DE output log window if idlcr8hdf is called from another program within IDL, or from the IDL Command Input Line.
- There is no default log output if idlcr8hdf.sav is opened in IDL VM. The only way to monitor progress during run-time is to choose the Popup option.
- If the Log option is selected, output will also be to the file idlcr8hdf.log (refer to section **3.2.8**).
- If the Popup option is selected, output will also be displayed in a Pop-up window (refer to section **3.2.9**).

6 The Metadata and Data

The metadata and data representation follows the GEOMS file reporting guidelines outlined by Retscher et al., 2011.

6.1 Metadata

The purpose of the metadata is to describe the users actual scientific data (e.g. measurements) using a set of descriptors or attributes. Each such attribute describes a certain part of the scientific data. Examples of attributes could be the name of the owner of the data, or the type of instrument that was used, or the measurement unit etc.

Each metadata attribute is defined through a metadata statement in the file. A metadata statement has the following form:

```
label = value
```

where label is a text string containing the name of the attribute, and value is a text string with the corresponding value of that attribute. An example of a metadata statement could be:

```
PI_NAME = Retscher; Christian
```

The semicolon character is used as a delimiter in the value text strings. Each item between the semicolons in a value is called a sub-value. The number of sub-values for a metadata attribute value is defined for each attribute (refer to the metadata guidelines for a detailed description of each attribute).

For many metadata attributes the concept of sub-values on the right hand side of the metadata statement is important since individual values will be checked for their legality against specified tables of allowed values. Tables of such legal values are defined in the TAV file. For a description of this file, see Section **3.2.3**.

Some metadata attributes describe date and time values. These are:

- DATA_START_DATE
- DATA_STOP_DATE
- FILE_GENERATION_DATE

generally, and possibly:

- VAR_VALID_MIN
- VAR_VALID_MAX

The date and time value for these attributes can either be in the ISO8601 format (YYYYMMDDThhmmssZ) or in the MJD2K (Modified Julian Date 2000) format on input to the program. The MJD2K value is a double precision value counting the time (in fractional number of days) from 1 January 2000 at 00:00:00.0. Thus 20000101T000000Z is 0.0 MJD2000, while 20000101T120000 is 0.5 MJD2000. Date and time attribute values in the resulting HDF file are stored using MJD2K, apart from DATA_START_DATE, DATA_STOP_DATE, and FILE_GENERATION_DATE, which are stored using ISO8601. If using ISO8601 format on input, the program will automatically convert the date and time values to MJD2K where required, and vice versa. The HDF filename uses the ISO8601 format for the part that describes the data start and stop dates.

There are ten metadata attribute labels where the values can be determined directly by the idlcr8hdf code. It is therefore up to the user whether to include them in the input Metadata (although the attribute label and the '=' sign are still required).
These attribute labels are:

- DATA_START_DATE (determined from the data)
- DATA_STOP_DATE (determined from the data)
- DATA_TEMPLATE (determined from the DATA_SOURCE value)
- FILE_NAME (determined from metadata values)
- FILE_GENERATION_DATE (determined by the program)
- FILE_META_VERSION (determined by the program)

- VAR_SIZE (determined from the size of the corresponding dataset, or from the VAR_SIZE(s) of the corresponding VAR_DEPEND variable(s))
- VAR_SI_CONVERSION (linked to the VAR_UNITS values)
- VAR_VALID_MIN/VAR_VALID_MAX (for datasets that have VAR_UNITS=MJD2K only – the program will determine these values based on the dataset entries).

Most of the attribute values do not change from measurement to measurement, which makes it possible to use a single metadata template as input with multiple data files. Exceptions to this could be the DATETIME and related variables, e.g. XX_XX_START.TIME, XX_XX_STOP.TIME, where the VAR_VALID_MIN/MAX values could change from measurement to measurement. If any of the minimum or maximum metadata attribute values are missing, or the value is not valid, then the program will determine these values based on the dataset entries which have VAR_UNITS=MJD2K. Note that if valid values for these attributes are present in the metadata file, they will not be replaced.

6.2 Data

The data file is a plain ASCII file. As with the Metadata file, comment lines starting with an exclamation character (!) or a hash-mark character (#), and blank lines will be skipped over during input. Although an error will occur, and the program will stop, if a comment or blank line is inserted in the middle of a set of data values. **Applying offsets to, or scaling the data, is not permitted.**

The data file is organized as follows:

```
<Header 1>
<Data value   1 for header 1>
<Data value   2 for header 1>
…
<Data value m1 for header 1>
<Header 2>
<Data value   1 for header 2>
<Data value   2 for header 2>
…
<Data value m2 for header 2>
…
<Header n>
<Data value    1 for header n>
<Data value    2 for header n>
…
<Data value mn for header n>
```

Each line has one value only. The value is either a header value or a data value. The header value is a text string containing the name of a variable corresponding to a VAR_NAME attribute value in the metadata file (without any quotes around it). For both the header line and on each of the data lines, any leading or trailing white-space characters will be stripped off the values. Thus, they do not need to start in position 1 of the lines.

A data file containing data for the variables:

- LONGITUDE
- LATITUDE
- ALTITUDE
- DATETIME
- TEMPERATURE

may look like this (example):

```
LONGITUDE
11.0
11.1
LATITUDE
60.0
61.0
62.0
ALTITUDE
9.6
12.4
18.9
23.5
DATETIME
0.0
1.0
2.0
3.0
4.0
TEMPERATURE
-2.7
-2.9
-3.4
-1.9
-0.7
…
-8.9
```

If TEMPERATURE has been made dependent upon the first four variables in the metadata file (i.e.):

```
VAR_SIZE = 2;3;4;5
VAR_DEPEND = LONGITUDE;LATITUDE;ALTITUDE;DATETIME
```

then it will contain a total of 2*3*4*5=120 values.

The connection between the data lines in the data and the indices of a multi-dimensional variable is always defined in IDL by letting the first index vary fastest, the second index next fastest, and so on.

To use the example above again, the connection between the data values after the header line for TEMPERATURE and the indices of the corresponding 4-dimensional variable TEMPERATURE is then the following:

```
…
1,1,1,1
2,1,1,1
1,2,1,1
2,2,1,1
1,3,1,1
2,3,1,1
1,1,2,1
2,1,2,1
…
2,3,4,1
1,1,1,2
2,1,1,2
…
2,3,4,5
```

HDF4 supports multidimensional variables with up to 8 dimensions.

Please note that dimension ordering is defined in HDF in the opposite way to IDL, i.e. For HDF, first index varies slowest in a multi-dimensional array. The IDL HDF library routines automatically transpose the dataset arrays and the idlcr8hdf code also then reverses the VAR_DEPEND and VAR_SIZE values so that they continue to match the array ordering in the HDF file. It is therefore important that the VAR_DEPEND and VAR_SIZE values are always written in the input Metadata file or structure so that they correspond to the IDL convention of ordering the data.

The GEOMS Standard supports six different variable data types: BYTE, SHORT, INTEGER, REAL, DOUBLE, and STRING. BYTE is used for 8-bit unsigned integer data (0-255); SHORT is used for 16-bit integer data (integer*2); INTEGER is used for 32-bit integer data (integer*4); REAL is used for 32-bit single precision real data (real*4); DOUBLE is used for 64-bit double precision real data (real*8); STRING is used for Characters (HDF4) or Strings (HDF5).

7 The HDF Output File

The program writes out an HDF file where all legal metadata attributes and their corresponding values are stored as HDF-attributes together with the SDS datasets from the data file. All attributes and their values are stored in the HDF file using text strings of variable length except for VAR_VALID_MIN, VAR_VALID_MAX and VAR_FILL_VALUE. The values for these attributes are saved in the same data type as the data i.e. as defined by VAR_DATA_TYPE.

In addition to the user-defined attributes, the program also outputs to the HDF4 file a set of predefined HDF attributes, which NCSA (National Center for Supercomputing Applications) recommends should be part of any HDF file. Some of these predefined attributes are shown in Table 1. Note that, for a particular dataset, if VAR_DATA_TYPE=STRING then no predefined attributes will be written to the file. The predefined attributes are not included in HDF5 files.

Table 1: Predefined attributes and corresponding user-defined attributes.

Predefined attribute	Corresponding user-defined attribute
units	VAR_UNITS
valid_range	VAR_VALID_MIN,VAR_VALID_MAX
_FillValue	VAR_FILL_VALUE

All predefined attributes and attribute names, which start with VAR (for variable), are defined as being dataset attributes (local attributes) belonging to one of the SDS's in the HDF file. All other attributes are defined in the HDF file as global attributes.

8 Troubleshooting

- **idlcr8hdf is set-up to be called repeatedly by another program during a single run-time event but stops after the first call** – This can occur when the /Popup option is included in the idlcr8hdf call. Remove this option or, alternatively, rewrite the parent program so that it only performs a single call to idlcr8hdf e.g. by generating an array containing all the input data files before calling the program.
- **idlcr8hdf generates an error message and stops, but the problem based on the reported message cannot be identified** – This could be due to a problem not currently checked for in the code, but which indirectly causes an unrelated error to be reported, or that the explanation accompanying the error is not sufficient to help isolate the problem. Please report the error to Ian Boyd and Ghassan Taha, together with IDL version and OS details, and example input, if possible.
- **idlcr8hdf generates a run-time error without an associated idlcr8hdf generated error message** – A problem with the metadata and/or data inputs has been found which is sufficiently obscure that it is not accounted for in any of the existing QA integrity checks. Please report this error to Ian Boyd and Ghassan Taha, together with IDL version and OS details, and example input, if possible.
- **IDL generates compilation errors when compiling idlcr8hdf** – If the compilation errors are restricted to the AVDC_HDF5_WRITE procedure; the IDL version being used is earlier than IDL6.2. This will not affect the running of the program, except that the option to create HDF5 files will not be available. If the compilation errors occur in other procedures then the IDL version may predate

IDL6.0. One solution could be to run idlcr8hdf.sav using the most recent version of IDL VM (which is free). Otherwise, please report the compilation error to Ian Boyd and Ghassan Taha, together with the IDL version.

1 Appendix – Procedures

The program is a standard IDL program consisting of 24 procedures and 2 functions. All the user data is read from external files or session memory.

A brief description of the procedures and functions is given below:

1.1 *idlcr8hdf*

The main procedure called at run-time or by another external program. This procedure checks that the input is valid, calls the procedures that read the input and carry out the integrity checks, and finally calls the procedures that create the output HDF file.

1.2 *idlcr8hdf_Common*

This procedure defines the common blocks used by the program.

1.3 *Intro_Event*

Called by the Intro procedure. This procedure controls what happens when a Widget Button is selected in the Introduction display window.

1.4 *Intro*

Displays the Introduction display window, and allows the user to continue running the program after choosing from the available options.

1.5 *idlcr8hdf_Event*

Called by the Stop_With_Error procedure and the idlcr8hdf procedure. This procedure closes the Log display window and ends the program when the Finish Widget Button is selected.

1.6 *Stop_With_Error*

Called when an Integrity Check determines there is an error which prevents the program from being able to successfully continue. This procedure displays the error according to the options chosen by the user and the method in which the program was called, and stops the program in a ‘clean’ manner (closes any open files etc).

1.7 *Infotxt_Output*

Called when reporting information or error messages encountered when performing checks on the input Data and Metadata and creating the HDF file. If an error message is generated in this context the program can continue to run but the HDF file will not be created. The messages are displayed according to the options chosen by the user and the method in which the program was called.

1.8 JDF_2_Datetime

A function that returns a floating point array in the form [YYYY,MM,DD,hh,mm,ss.sss], or a string in the form YYYYMMDDThhmmssZ, or a string in the form YYYYMMDDThhmmss.sssZ. Input can either be the Julian Day, or the date in MJD2000 format.

1.9 Julian_Date

A function that returns the Julian Day, or the date in MJD2000 format. Input can either be a numeric array in the form [YYYY,MM,DD[,hh[,mm[,ss]]]] or a string in the form YYYYMMDDThhmmssZ.

1.10 Var_Units_Test

Called by Check_Metafile. Checks that the values given for VAR_UNITS are valid (equivalent to a base unit or a unit prefix and base unit) and independently calculates the VAR_SI_CONVERSION. If the VAR_SI_CONVERSION value is not present in the metadata, it automatically adds it, otherwise it checks that it matches the metadata value and, if not, it will replace it and report to the output log.

1.11 Read_Tablefile

Identifies the version number of the TAV file. Also reads the FIELD/LABEL categories, the number of entries under each FIELD, and the FIELD entries into an NF x FCNT+2 string array called tab_arr (where NF is the number of FIELD categories, and FCNT is the maximum number of entries under the FIELDS). The additional two elements of the second dimension contain the name of the field and the number of values in the field.

1.12 Test_File_Input

Called by Read_Metadata. If a free text attribute is in the form of a filename, then checks that the file exists, and that the file size is less than or equal to the maximum permitted attribute length. Note: Filename entries are not permissible in the GEOMS Standard.

1.13 GEOMS_Rule_Changes

This procedure performs checks on the Metadata for old or redundant rules, labels, and/or values. Depending on how idlcr8hdf is called it will update, replace or remove Metadata Attributes and report as required.

1.14 Pre_Defined_Att_Checks

This procedure performs checks on the HDF4 pre-defined attributes, as well as check that the numeric variable attribute values are of the same data type as that given by the corresponding VAR_DATA_TYPE value (if input is via session memory).

1.15 Read_Metadata

Puts the metadata contents into a string array called meta_arr. It ensures uniform input (e.g. making the ATTRIBUTE_NAME uppercase and deleting unwanted spaces etc), checks that the attribute value character length falls within allowable limits. It can also check for filenames for the Free Text Attributes (not allowed in the GEOMS Standard).

1.16 Extract_And_Test

Called by Check_Metafile. Extracts the relevant portions of the attribute values from the metadata file and the TAV file and compares them.

1.17 Check_Metadata

Checks that metadata values for attribute names matching those in the TAV file are legal (i.e. included in the file). In some cases it will also make corrections to the metadata entry e.g. match VAR_SI_CONVERSION values to the VAR_UNITS value.

1.18 Extract_Data

Called by Set_Up_Structure and Read_Data. If called by Set_Up_Structure, the VAR_SIZE of a variable is determined by counting the number of values in a dataset. If called by Read_Data, the procedure extracts a specified dataset from the input data.

1.19 Set_Up_Structure

Performs additional checks on the metadata (for attributes not in the TAV file), and sets up a storage structure for the data based on the metadata values.

1.20 Check_String_Datatype

Called by Read_Data if the VAR_DATA_TYPE is string. Does the following:

- Checks that VAR_UNITS, VAR_VALID_MIN/MAX, and VAR_FILL_VALUE values are <empty>.
- All data value entries are assigned the string length of the entry with the longest string length. The data values are returned in the string data type.

1.21 Check_Min_Max_Fill

Called by Read_Data if the VAR_DATA_TYPE is numeric. Does the following:

- Checks that the data values fall within VAR_VALID_MIN and VAR_VALID_MAX values (except for fill values).
- Checks that VAR_FILL_VALUE, VAR_VALID_MIN, VAR_VALID_MAX, and data values fall within the range of the given data type.
- Returns the data in an array of type determined by the VAR_DATA_TYPE attribute, and to the precision given in the VIS_FORMAT attribute.

1.22 Read_Data

This procedure reads the input data, and after carrying out integrity checks, writes the datasets to a data structure for writing to the HDF file. This procedure also fills in missing VAR_VALID_MIN/MAX values for datasets with VAR_UNITS=MJD2K and, if requested, adds a sentence to VAR_NOTES giving array order for Averaging Kernel datasets.

1.23 Find_HDF_Filename

Does the following:

- Compares the DATA_START_DATE (if present) with the first entry under DATETIME[.START] in the data file. If necessary creates/changes the DATA_START_DATE to match the first entry, and puts it in ISO8601 format.
- Compares the DATA_STOP_DATE (if present) with the last entry under DATETIME[.STOP] in the data file. If necessary creates/changes the DATA_STOP_DATE to match the last entry, and puts it in ISO8601 format.
- Compares the filename created from the DATASET_ATTRIBUTES with that under the FILE_NAME entry (if present). If necessary creates/changes the FILE_NAME entry to match the created filename.
- If necessary creates/changes the FILE_GENERATION_DATE to ISO8601 format.

1.24 MakeAn

Called by the two HDF write procedures in the event that a free text attribute entry is a filename. This procedure reads the contents of the file, and writes it to the HDF file as an SDS attribute. It can also be set up to write the file contents to the HDF file as a File or Data annotation (HDF_AN). Note: this feature is not implemented in the GEOMS Standard.

1.25 AVDC_HDF5_Write

Called by AVDC_HDF_Write if the HDF5 option is chosen. Writes the Global and Variable Attributes and Data to an HDF5 file.

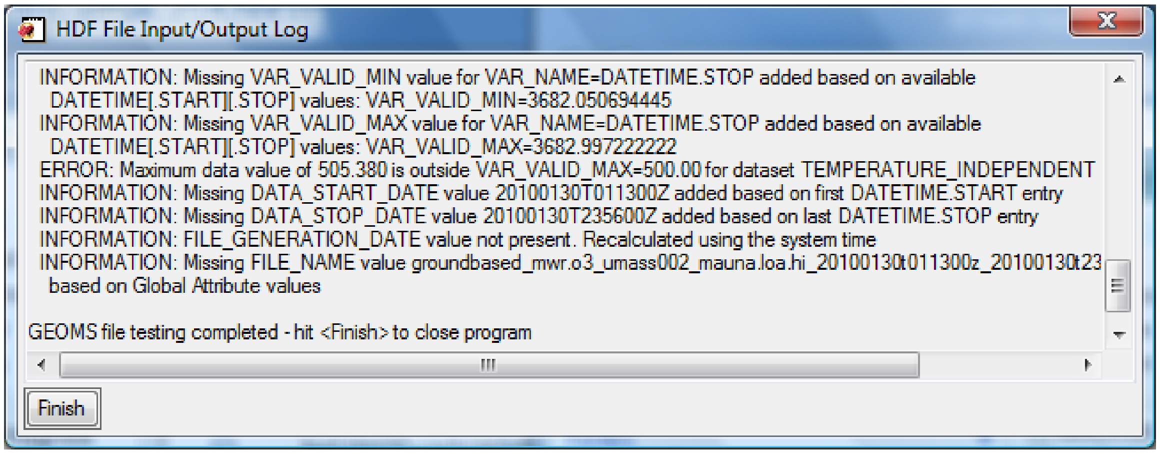
1.26 AVDC_HDF_Write

Writes the Global and Variable Attributes and Data as a Scientific Dataset to an HDF4 file according to GEOMS guidelines, or calls AVDC_HDF5_Write if this option is chosen.

2 Appendix – Error and Information Messages

Many of the procedures include integrity checks on the contents of the metadata and data. If an error is detected, depending on the type of error, the program may immediately stop and return to the IDL command line or calling program, or it may be able to continue with the checks. In both cases, an HDF file will not be generated. An error message will be appended to the output log as well as the pop-up logging window or a separate pop-up box. The message may include the name of the procedure where the error was detected, a description of the error and the affected portion of the input file.

In the case of information messages, the program continues to run, but a message appears in the log display window and/or output log. A message is generated when the program is reporting relevant or useful information, or is able to complete or correct metadata entries. For example:



```
INFORMATION: Missing VAR_VALID_MIN value for VAR_NAME=DATETIME.STOP added based on available
  DATETIME[.START][.STOP] values: VAR_VALID_MIN=3682.050694445
INFORMATION: Missing VAR_VALID_MAX value for VAR_NAME=DATETIME.STOP added based on available
  DATETIME[.START][.STOP] values: VAR_VALID_MAX=3682.997222222
ERROR: Maximum data value of 505.380 is outside VAR_VALID_MAX=500.00 for dataset TEMPERATURE_INDEPENDENT
INFORMATION: Missing DATA_START_DATE value 20100130T011300Z added based on first DATETIME.START entry
INFORMATION: Missing DATA_STOP_DATE value 20100130T235600Z added based on last DATETIME.STOP entry
INFORMATION: FILE_GENERATION_DATE value not present. Recalculated using the system time
INFORMATION: Missing FILE_NAME value groundbased_mwr.o3_umass002_mauna.loa.hi_20100130t011300z_20100130t23
  based on Global Attribute values

GEOMS file testing completed - hit <Finish> to close program
```

In this case, idlcr8hdf has automatically determined missing VAR_VALID_MIN and VAR_VALID_MAX values for the DATETIME.STOP dataset, as well as DATA_START_DATE, DATA_STOP_DATE, FILE_GENERATION_DATE, and FILE_NAME Global Attribute values. It has also determined that a value in the TEMPERATURE_INDEPENDENT dataset exceeds the VAR_VALID_MAX value given for this attribute, and generated an error. The HDF file has, therefore, not been generated.

All checks are carried out before the HDF file is created and opened. The checks are the same as those used by the datacenters when performing QA operations on uploaded HDF files. Note that, if idlcr8hdf were called in QA mode, all the above INFORMATION messages would be returned as errors. For example, the first message would read:

ERROR: Missing VAR_VALID_MIN value for VAR_NAME=DATETIME.STOP

Below is a list of the possible errors and information messages, with further explanation. The list is sorted by procedure. Note that when a '|' is present in the message, this denotes a demarcation between the full message and the abbreviated message that is generated when running in QA mode. Likewise if two options are given inside square brackets, the second option refers to the message given in QA mode. *Italicized* words are replaced by metadata names or values in the actual message.

2.1 idlcr8hdf

Errors that will stop the program:

- **Metadata, Data Table Attribute Values files and/or Output Directory selection not valid**. Checks are made on the Table Attribute Values file and the output directory and, if input is from files, the metadata and data inputs. If any of the inputs are not valid (e.g. are not found, or no input), then this message will be generated.
- **Global Attributes Array is not of type string, or is not one dimensional**. If input is from session memory then checks that the Global Attribute array is set-up as required by the program.

- ***SDS heap/SDS.Data/SDS.VA* structure is not valid**. If input is from session memory then uses the N_Tags function to check for the existence of structure tag expressions, uses the Tag_Names function to check that tags names VA_L, VA_V and DATA are present, checks that the structure has 2 dimensions, and uses the PTR_Valid function to verify the validity of the pointer arguments to the heap structure.
- **Array sizes of the heap structure do not match**. If input is from session memory then checks that the VA_L and VA_V array sizes match.
- **IDLvX.X does not support HDF5 Write Routines**. The IDL version must be 6.2 or greater if the /H5 option is chosen.
- **IDL requires session memory input to perform the QA function**. To perform QA, idlcr8hdf must be called via geoms_qa or after reading the contents of the HDF file into session memory using idlcr8ascii.

Other messages:

- **/POPUP keyword cannot be used together with the 'RetErr' argument. The request for a POPUP window has been ignored**.
- **Argument 'RetErr' must be of type string. idlcr8hdf will stop normally if an error is encountered**.

2.2 Read_Tablefile

Errors that will stop the program:

- **Table Attribute Values file version not identified with the search criteria used by this program**. The program does a search on the text '! Version' to find the file version of the TAV file. This is used as an entry under the FILE_META_VERSION global attribute.
- **Envisat table.dat ASC2HDF program version not found with the search criteria used by this program**. If the input is an Envisat style table.dat file then the program does a search on the text 'ASC2HDF' to identify the ASC2HDF program version.
- **First Envisat table.dat field value not found with the search criteria used by this program**. If the input is an Envisat style table.dat file then the program does a search for the first Label in the file.
- **Table Attribute Values file version is invalid**. The program verifies that the format of the TAV file version number is either ddRddd or ddRdddd, where dd and ddd(d) are numbers, separated by the letter R.

Other messages:

- **EVDC *or* GEOMS Reporting Guidelines Apply**. This standard message is generated depending on whether an AVDC TAV file or EVDC table.dat file is read in.
- **Old version of the Table Attribute Values file used as input**. If the input is an AVDC style TAV file, the version of the file must be 04 or greater (i.e. 04Rxxx).

2.3 Test_File_Input (not part of GEOMS Standard)

- **Syntax of Filename entry incorrect**. When listing a filename as a value for a Free Data Attribute, it needs to be in the form FILE("filename").
- **Filename entry not found or not usable**. The procedure checks that the file exists and can be read.
- **File size is too large**. The procedure checks that the file size is within the allowable limit given in the code (currently set to 4096 bytes).

2.4 GEOMS_Rule_Changes

Other messages:

- ***Attribute Value* not a GEOMS Global/Variable Attribute|. Removed from the metadata saved to the HDF file**. An obsolete EVDC or AVDC attribute is included in the Metadata.
- **Dataset name *Old Dataset Name* [renamed/expected to be] DATETIME.START/ STOP/ INTEGRATION**. Previous standards used START/ STOP/ INTEGRATION.TIME as a variable descriptor.
- **Double underscore present in Variable Name| replaced with a single underscore**. Previous standards indicated a missing variable mode with a double underscore when a variable descriptor was included in the Variable Name.
- ***Var_Depend value* should be self-referencing for axis variable VAR_NAME=*value*|. VAR_DEPEND value changed in metadata**. Previously VAR_DEPEND had a value of INDEPENDENT for axis-variables.
- **VAR_UNITS=MJD2000 not GEOMS compliant|. Changed to VAR_UNITS=MJD2K**.
- **DATA_FILE_VERSION=*value* must be in the form 'nnn'[|. Renamed from *xyz* to *nnn*]**. Previous AVDC standards allowed for this value to be a user-defined processing version. If it is possible to correct the value, the extended message will be included.
- **Dataset name *Old Dataset Name* [renamed/expected to be] *New Dataset Name***. The routine includes a set of obsolete Variable Name compositions and lists the GEOMS Standard equivalent.
- **DATA_SOURCE value *Old DATA_SOURCE value* [renamed/expected to be] *New DATA_SOURCE value***. Obsolete DATA_SOURCE values are updated with GEOMS Standard values.
- **FILE_ACCESS value *Old FILE_ACCESS value* [renamed/expected to be] *New FILE_ACCESS value***. Obsolete FILE_ACCESS values are updated with GEOMS Standard values.
- **VAR_UNITS=DIMENSIONLESS/NONE [renamed/expected to be] VAR_UNITS=1/<empty> based on VAR_DATE_TYPE=*value***. Dimensionless and None are no longer valid VAR_UNITS values.
- **Required Metadata Template: DATA_TEMPLATE field is not present in the TAV file. Checks cannot be performed**. idlcr8hdf expects to see this field in the TAV file to be able to perform checks on the metadata entry.

- **DATA_TEMPLATE value [renamed/expected to be]** ***DATA_TEMPLATE value*** **based on DATA_SOURCE value**.
- **Required Metadata Template:** ***DATA_TEMPLATE value***. A standard entry is included in the log to allow for Template Checking of the HDF file.
- **Required Metadata Template: DATA_TEMPLATE filename does not correspond with any TAV entries**.
- **Required Metadata Template: DATA_TEMPLATE value not required based on selection criteria**. The contents of the HDF file do not need to conform to a template.

2.5 Pre_Defined_Att_Checks

Other messages:

- **VAR_VALID_MIN/MAX or VAR_FILL_VALUE data types do not match VAR_DATA_TYPE=*value* for Dataset *value*|. Data type will be changed to match VAR_DATA_TYPE in HDF file**. This message is reported when input is via session memory. The values for these attributes must be written to the HDF file in the reported VAR_DATA_TYPE.
- **HDF4 pre-defined attribute *value* data type does not match VAR_DATA_TYPE=*value* for Dataset *value*|. Data type will be changed to match VAR_DATA_TYPE in HDF file**. As above except applying to the HDF4 pre-defined attributes valid_range and _FillValue.
- **HDF4 pre-defined attribute units=*value* does not match VAR_UNITS=*value* for Dataset *value*|. HDF4 units will be changed to match VAR_UNITS value in HDF file**.
- **HDF4 pre-defined attribute *label* value does not equal *Variable Attribute Label* value for Dataset *value*|. HDF4 *label* will be changed to match *Variable Attribute Label* value in HDF file**. This applies to a mismatch between the VAR_VALID_MIN/MAX or VAR_FILL_VALUE values and the HDF4 pre-defined equivalents.
- **HDF4 pre-defined attribute *label* [will not be written to/present in] the HDF file**. If creating an HDF file, any pre-defined attributes other than those mentioned in Chapter 7 will not be written to the file. For QA these attributes will be ignored.
- **Datasets that have been scaled or had an offset applied are not permitted**.

2.6 Read_Metadata

Error messages that will stop the program:

- **No valid metadata in the input.**
- ***Attribute Label* value contains too many characters**. The procedure checks the number of characters in the attribute value against a limit given in the code (currently 65535 characters).
- **No or invalid DATA_VARIABLES and VAR_NAME attributes present in the metadata**. Program cannot recognize or find the DATA_VARIABLES list or the VAR_NAME attributes.

- **DATA_VARIABLE or VAR_NAME value appears more than once**.
- **VAR_NAME does not match corresponding DATA_VARIABLES value**. The program expects that the order of the DATA_VARIABLES list matches the order that VAR_NAMES are given in the metadata.
- **Number of DATA_VARIABLES values does not match the number of VAR_NAME occurrences**.
- **Global Attribute label repeated in the metadata**.
- **Number of *Variable Attribute* occurrences does not match the number of datasets**. Reported if the number of mandatory variable attributes does not match the number of Data Variables.

Other messages:

- **Metadata Attribute Labels [made/must be] UPPERCASE**.
- **FILE_META_VERSION updated with corrected tool name and TAV file information**. If the FILE_META_VERSION does not match the program name or the input TAV file version then it will be updated.
- **Unknown Attribute Label *Label Name*| ignored**. The program cannot identify the Attribute Label as either a Global or Variable Attribute.
- **Missing DATA_VARIABLES Global Attribute| added to metadata based on VAR_NAME values**.
- **Missing DATA_VARIABLE attribute values| appended to the metadata based on VAR_NAME values**.
- **Missing VAR_NAME value for *Variable Name*|. *Value* appended to label based on DATA_VARIABLES value**.
- **Missing VAR_NAME attribute labels and values| added to metadata based on DATA_VARIABLES values**.
- **Missing [mandatory] Global Attribute *Attribute Name*| added**. A Global Attribute can be added if the value can be determined by the program or it does not require a value.
- **Global Attribute Label not present in the metadata**. The Global Attribute cannot be automatically added in this case as the value is unknown.
- **Non-standard Global Attribute *Attribute Name*| added**.

2.7 Extract_And_Test

Error messages that will stop the program:

- **Number of (sub-)values for this attribute should be nn**. The number of values or sub-values being tested against the TAV file values is not correct.
- ***Attribute Value* doesn't match any Table Attribute Values under FIELD: *TAV Field Name***. The code is unable to find a match between a metadata value (or set of sub-values), and the list of possible values given in the TAV file.

Other messages:

- ***Attribute Value* doesn't match any Table Attribute Values under ORIGINATOR field**. While still generating an error, in this case the program can still continue running without causing subsequent errors.

2.8 Check_Metadata

Error messages that will stop the program:

- **No or Invalid value for this Attribute**. Relevant for metadata attribute values listed in the TAV file. This error will occur if no value exists or if there is more than one equal sign in the value.
- **Number of (sub-)values for this Attribute should be nn**. The number of values or sub-values being tested against the TAV file values is not correct.
- **AFFILIATION value doesn't match NAME value**. If the ORIGINATOR field is present in the TAV file, the program checks that the AFFILIATION given in the metadata matches up to an AFFILIATION next to the NAME in the TAV file.
- ***AFFILIATION* or *DATA_SOURCE_02* field not found in the input Table Attribute Values file**. Given the inputs, the Program was expecting to find one of these fields in the TAV file (AFFILIATION for AVDC style and DATA_SOURCE_02 for Envisat style).

Other messages:

- **Provisional make-up of entry required in the TAV file in the ORIGINATOR field, based on the HDF file values. Note that values may be missing or invalid, or may not conform to GEOMS Standard: *New Originator Entry***. This is a 'Debug' Statement which helps the Datacenter determine possible new entries to the TAV file.
- ***Originator Value* [replaced with/should be] *Corrected Originator Value* based on the TAV entry**. This message is generated if there is a case mismatch between the metadata and TAV entries.
- ***Originator* values do not match those in the Table Attribute Values file|, and replaced**. Affiliation, Address, or e-mail originator values in the metadata do not correspond to the TAV entry based on the Name value.
- **VAR_SI_CONVERSION=*value* [replaced with/should be] *New VAR_SI_CONVERSION value***. The VAR_UNITS_TEST routine calculates the VAR_SI_CONVERSION value based on the VAR_UNITS entry, and this is compared against the value in the metadata, if present.

2.9 Extract_Data

Error messages that will stop the program:

- **EOF encountered but not expected**. If input is from files then this could occur when searching for the correct variable name header, or when attempting to determine the VAR_SIZE of a variable, or when reading in the data and encountering the EOF when expecting more data values.
- **Invalid data point found in the dataset at line nn**. If input is from files then a non-data point was found during the block read of the dataset.
- **Two instances of this VAR_NAME present in the Data File**. The VAR_NAME label is present more than once in the input data file.
- **Too many data values present in the dataset or invalid data line entry following the dataset**. After a block read of the dataset, either the next line is another data point or an invalid comment or dataset label.

- **Number of data values does not match product of VAR_SIZE**. If data input is in the form of a structure, then the number of data points in the selected structure is not equal to the product of the VAR_SIZE.
- **ISO8601 format not valid**. Occurs when trying to convert a Datetime value, which the program thinks is in ISO8601 format, to MJD2000.
- **Unable to determine correct number of data values due to VAR_SIZE calculation error**. Occurs when the VAR_SIZE values are being determined by the program, and it is unable to do so (for example, because of differences in the number of values in dependent and independent datasets).

Other messages:

- **Dataset values [changed from ISO8601 to MJD2K format/cannot be in ISO8601 format] for VAR_NAME=*value***. All dataset values for variable attributes with VAR_UNITS=MJD2K must be written to the HDF file in MJD2K format.

2.10 Set_Up_Structure

Error messages that will stop the program:

- ***Var_Name* value does not match the equivalent VAR_NAME in DATA_VARIABLES**. The procedure checks that the variable name given in the Dataset Attributes corresponds to a variable name given as a value in the Global Attribute DATA_VARIABLES.
- **Number of dimensions too high to make HDF file (max. allowed 8)**. IDL/HDF4 can only handle arrays up to eight dimensions.
- **VAR_DEPEND attribute value missing or does not match the number of VAR_SIZE values**. Checks that the number of values in VAR_SIZE and VAR_DEPEND match.
- **VAR_DEPEND value not CONSTANT, INDEPENDENT nor self-referencing, and doesn't match a previous VAR_NAME**. If the listed variable is dependent on another variable (e.g. DATETIME), then the VAR_DEPEND attributes must be listed before the dependent variable attributes.
- **VAR_DEPEND variable name(s) must have VAR_DEPEND value of 'CONSTANT or be self-referencing**. The given variable name(s) must, in turn, have a VAR_DEPEND attribute value of CONSTANT or be self-referencing.
- **VAR_DEPEND variable name must have matching VAR_SIZE value**. The given variable name(s) must have the same number of data points (as indicated by VAR_SIZE), as the variable name being checked.
- **Data type not BYTE, SHORT, INTEGER, REAL, DOUBLE, or STRING**. The VAR_DATA_TYPE cannot be identified as one of the six allowable types.
- **Unexpected number of ATTRIBUTE values extracted**. Checks that there is the correct number of values needed to obtain the array size information.
- **VAR_UNITS must be MJD2K for VAR_NAME=*value***. The VAR_UNITS value is not equal to MJD2K for a variable related to DATETIME.

- **VAR_FILL_VALUE not a valid value**. The program checks that the VAR_FILL_VALUE for attributes which have VAR_UNITS=MJD2K, is a valid number if it is in ISO8601 format.
- **VAR_DEPEND value(s) not valid**. The program checks for valid accompanying measurements when the VAR_NAME is one of ALTITUDE/ ALTITUDE.GPH/PRESSURE.BOUNDARIES, or if the second VAR_DEPEND value is INDEPENDENT.
- **Second VAR_SIZE value should be '2' for this VAR_NAME**. This check is made when the VAR_NAME is one of ALTITUDE/ALTITUDE.GPH/ PRESSURE.BOUNDARIES.
- **Axis variable cannot be dependent on another variable**.
- ***Variable Name* is not an axis variable so VAR_DEPEND value cannot be self-referencing**.
- **Type conversion error. Entry is not a valid number**. The program cannot convert the attribute value from a string to its numeric value.

Other messages:

- **64-bit LONG Data Type is not currently supported|. Will attempt to write data in 32-bit INTEGER Data Type**. The previous AVDC and EVDC standards defined LONG as a 32-bit integer type. This is now defined as INTEGER in the GEOMS Standard.
- ***VAR_DEPEND entries* not valid for *Variable Name* [in IDL/Fortran Dimension Ordering**. This may occur if the VAR_DEPEND values are in the incorrect order or are not valid when the Variable Name includes .BOUNDARIES.
- ***VAR_DEPEND value* should be self-referencing for axis variable VAR_NAME=*value*|. VAR_DEPEND value changed in metadata**.
- **VAR_DEPEND=*value* variable name(s) for *VAR_NAME value* must have VAR_DEPEND value of CONSTANT or be self-referencing**.
- **Unable to determine missing VAR_SIZE value(s) for *VAR_NAME value* based on *VAR_SIZE of the VAR_DEPEND values* and the total number of dataset values**.
- **VAR_SIZE values for *VAR_NAME value* [added:/not recorded. Calculated as] VAR_SIZE=*value(s)***.
- **Dataset must only have a single value for VAR_DEPEND=CONSTANT**.
- **VAR_UNITS=*value* must be MJD2K for *VAR_NAME value*|. Value changed in metadata**.
- ***Numeric Attribute* for VAR_NAME=*value* [replaced with MJD2K formatted value/should be in MJD2K format]**. Occurs if VAR_FILL_VALUE or VAR_VALID_MIN/MAX values for a DATETIME attribute have been written to the metadata in ISO8601 format.

2.11 Check_String_Datatype

Other messages:

- **Attribute *Metadata Attribute* in Dataset *Variable Name* not valid for VAR_DATA_TYPE=STRING|. Has been corrected to *New Metadata Attribute***. When the data type for a dataset is String, then many of the Metadata attributes have <empty> values, which the program can automatically correct and report on, as required.

2.12 Check_Min_Max_Fill

Error messages that will stop the program:

- ***Data Value* not valid, or does not match VAR_DATA_TYPE**. Where ***Data Value*** could be VAR_VALID_MIN, VAR_VALID_MAX, VAR_FILL_VALUE or a Data Value. The program could not convert the value to a valid number given the constraints of the data type.
- ***Data Value* outside the data type range**. Where ***Data Value*** could be VAR_VALID_MIN, VAR_VALID_MAX, VAR_FILL_VALUE or a Data Value. For example, if a data value is 40000, but VAR_DATA_TYPE=SHORT (maximum allowable value of 32767).
- **Type conversion error. Entry is not a valid value**. Cannot convert a data value to a valid number.

Other messages:

- **Fill value for *Variable Name* falls within the data range and is defined as a 'Default' value**. GEOMS rules allows the VAR_FILL_VALUE to fall within the defined data range, in which case it is defined as a Default value rather than a Missing or Fill value.
- **Minimum data value of *value* is outside VAR_VALID_MIN=*value* for dataset *Dataset Name***.
- **Maximum data value of *value* is outside VAR_VALID_MAX=*value* for dataset *Dataset Name***.
- **DATETIME contains fill value(s)**. This is typically not permitted as DATETIME is an axis variable.
- **DATETIME values not in chronological order**. If more than one DATETIME value is present, the program checks that the dataset is in chronological order.

2.13 Read_Data

Other messages:

- **DATETIME.START=*value* is greater than DATETIME=*value***.
- **DATETIME=*value* is greater than DATETIME.STOP=*value***.
- **DATETIME.START=*value* is greater than DATETIME.STOP=*value***.
- **Missing *VAR_VALID_MIN/MAX value* for *Metadata Attribute*| added based on available DATETIME[.START][.STOP] values**. The program can automatically complete VAR_VALID_MIN or MAX entries for DATETIME and related attributes.

2.14 Find_HDF_Filename

Error messages that will stop the program:

- **Incorrect number of sub-values under DATA_nnn**. Checks that the number of sub-values, corresponding to the attributes DATA_DISCIPLINE/_SOURCE/ _LOCATION/_FILE_VERSION, is correct.
- **ISO8601 format not valid**. Occurs when trying to convert a Datetime value, which the program thinks is in ISO8601 format, to MJD2K.
- **Type conversion error. MJD2K entry is not valid**. Occurs when trying to convert a Datetime value, which the program thinks is in MJD2K format, from a string to a double precision value.

Other messages:

- ***Datetime Related Global Attribute* [converted to/is not in] ISO8601 format**.
- **Date in DATA_START/STOP_DATE does not match first/last DATETIME[.START][.STOP] entry|. DATA_START/STOP_DATE changed to match entry**. The program compares the earliest and latest DATETIME value, from the data, to the DATA_START_DATE and DATA_STOP_DATE (if present). If they are not the same then they are replaced with the respective values.
- **Missing DATA_START/STOP_DATE value [*value* added/should be *valu*e] based on DATETIME[.START][.STOP] entry**. The code will automatically complete these values if they are missing.
- **FILE_GENERATION_DATE value not present|. Recalculated using the system time**.
- **FILE_GENERATION_DATE value not in ISO8601 format|. Recalculated using the system time**.
- **Calculated DATA_STOP_DATE=*value* is later than the FILE_GENERATION_DATE**.
- **Filename determined from DATASET ATTRIBUTES does not match FILE_NAME entry under FILE ATTRIBUTES**. The program independently determines the filename from the DATASET ATTRIBUTES. If this is different from the FILE_NAME value (if present), then the filename determined by the program is used.
- **Missing FILE_NAME value [added/should be] based on Global Attribute values**.

2.15 AVDC_HDF_Write

Other messages:

- ***HDF Version* truncates the dataset name if its length is greater than 63-characters. If possible update the HDF4 library to HDF4.2R2 or newer| (refer to program documentation for procedure to do this). The Archive Data Center may also be set up to update the file on submission**.

3 Appendix – Acronyms

ASCII	American Standard Code for Information Interchange
AVDC	Aura Validation Data Center
AVK	Averaging Kernel
DHF	The NDACC's Data Handling Facility
ENVISAT	European Space Agency satellite
ESA	European Space Agency
EVDC	Envisat Validation Data Center
GEOMS	Generic Earth Observation Metadata Standard
HDF	Hierarchical Data Format
IDL	ITT, Interactive Data Language
ISO	International Organization for Standardization
ITT	ITT Visual Information Solutions (developers of IDL)
MJD2K	Modified Julian Date 2000
NCSA	National Center for Supercomputing Applications
NDACC	Network for the Detection of Atmospheric Composition Change (ex-NDSC)
SDS	Scientific Data Set data object in HDF
TAV	Table Attribute Values file used to create HDF

4 Appendix – Version History

4.1 v1.0 – Initial version release June, 2005

4.2 v1.1 – Bug fixes and improvements (release August, 2005):

- Change procedure heading format and order to allow library calls.
- Make VAR_SI_CONVERSION values match the VAR_DATA_TYPE, and add fix for VAR_SI_CONVERSION calculation rules e.g. if VAR_UNITS=cm m-3 then VAR_SI_CONVERSION=0;0.01;m-2.
- Additional checks of VAR_DEPEND values to reduce possibility of errors when creating the HDF file.
- Check that VAR_UNITS=MJD2000 for VAR_NAME=DATETIME.
- Improve checks and error messages when reading in data from a file.
- Add fix for converting a scalar data structure value to an array.
- Check that the first DATETIME value is the lowest when VAR_SIZE is greater than 1.
- Fix method that the program uses to check for integer overflow e.g. where VAR_DATA_TYPE=INTEGER but a data value is greater than 32767.
- Add fix for the situation where only fill values are present in a dataset.
- Change the format that DATA_START_DATE and FILE_GENERATION_DATE values are saved in the HDF file from MJD2000 to ISO8601.
- Replace JULDAY function calls (i.e. JULDAY(1,1,2000,0,0,0)) with the actual value (2451544.5D).
- Fix to write the DATA_FILE_VERSION value to the filename in its entirety.

- Add fix to avoid rounding errors when checking and writing double precision values.

4.3 v1.11 – Bug fixes and improvements (release November, 2005)

- Change order of the routines to avoid problems during compilation.
- Write VAR_VALID_MIN, VAR_VALID_MAX, VAR_FILL_VALUE, VIS_SCALE_MIN, VIS_SCALE_MAX and Data Values to file using the precision and format given by VIS_FORMAT.
- Fix problems with detection of data values outside VAR_VALID_MIN or VAR_VALID_MAX, or outside the range given by VIS_FORMAT, and improve corresponding error messages.

4.4 v2.0 – New Release (release October, 2006)

- Introduce option to write HDF5 versions of the AVDC HDF4 files (needs IDL6.2 or greater).
- Make the code suitable for running on a licensed version of IDL (using the .pro and .sav versions of the code) and on IDL Virtual Machine (using the .sav version of the code).
- Change the command line keyword options. Remove the /Help (window now opened if there are no command line parameters) and /File options (now automatically identifies whether input is from session memory or files), and add the /H5 (write HDF5), /AVK (for Averaging Kernel Datasets add sentence to VAR_NOTES indicating averaging kernel order), /Log (append input/output information to a log file), and /Popup options (append input/output, error and warning information to a pop-up display window).
- If input is from files, the program can now handle multiple data files (either as a string array or as a file spec), together with a Metadata template file and the TAV file.
- Input/output information as well as errors and warnings included in the IDL DE output log window.
- Program can fill in missing VAR_VALID_MIN/MAX and VIS_SCALE_MIN/MAX attribute values for datasets where VAR_UNITS=MJD2000.
- Add checks to DATETIME dataset – that the entries are in chronological order, if the dataset contains more than one value, and that the dataset does not contain Fill Values.
- Help window becomes an Introduction window, and the user has the option of continuing to run the program with file inputs.
- Input data now written to a structure instead of arrays, after the integrity checks have been performed and the datasets have been converted to the given data type and precision.
- The HDF4 file now not closed between writing the global attributes and the variable attributes to the file. This may result in a smaller file size.

- Fixes a problem with the calculation of VAR_SI_CONVERSION, when there is more than one VAR_UNITS sub-value with a unit prefix. The portion of the code that performs these checks also made clearer.
- Permits additional Global Attributes. If the attribute label DATA_TYPE is detected it is automatically changed to DATA_LEVEL.
- Permits the additional Variable Attribute VAR_MONOTONE.
- Input Variable and Visualization attributes can be in any order – they will be put in a standard order when writing to the HDF file.
- Fix to ensure that VAR_NAME, DATA_VARIABLES, and VAR_DEPEND values are case sensitive.
- Fixes a problem, in a multi-dimension array, when the last dimension is 1, e.g. VAR_SIZE=41;1 (previously the array size would be automatically reduced by IDL).

4.5 v3.0 – New Release (release March, 2008)

- Full and standardized descriptions of each Procedure and Function added.
- Formatting of the code standardized (Procedure/Function names, and IDL keywords capitalized; Variable and array names in lowercase; square brackets used for arrays).
- Function JULIAN_DATE: Removed from the main set of routines, and made stand-alone.
- Function JDF_2_DATETIME: Ensure input Julian Day or MJD2000 value is converted to a double precision value. Removed from the main set of routines, and made stand-alone.
- Procedure INTRO: Change the order of the option buttons, so that Continue and Stop appear at the bottom of the window.
- Procedure READ_TABLEFILE: Added code to differentiate between the AVDC TAV and original Envisat table.dat file. The tab_type flag is used to differentiate between the two formats. Note that some of the table.dat labels are renamed to match the equivalent TAV file labels for compatibility when testing Metadata entries – refer to EnviName/AVDCName arrays; Convert the Meta Version to ddRddd format if necessary, and generate a warning if it is different to the format in the TAV file; Check that the version of the TAV file is 03 or greater (i.e. 03Rxxx).
- Procedure READ_METADATA: If an AVDC style TAV file has been read in, VAR_MONOTONE will not be included as an attribute if it is present in the input Metadata (Warning written to the log if this occurs); If an AVDC style TAV file has been read in, DATA_TYPE will be renamed DATA_LEVEL (and a Warning written to the log), and vice versa if an original Envisat style table.dat fle is read in; A Warning also written to the log if extra Global Attributes are included in the Metadata; Checks added for missing or repeated Global Attributes; Global Attributes can now be listed in any order in the Metadata but will be written to the HDF file in a standard order.

- Procedure VAR_UNITS_TEST: var_unit_arr and unit_pre_arr added which hold, respectively, the valid VAR_UNITS and corresponding VAR_SI_CONVERSION values, and the set of UNIT_PREFIXs (previously values from the AVDC TAV file have been used). This has been done so that the routine can be stand-alone (i.e. not dependent on also having to read in a TAV file). The input parameters have been changed to reflect this: bu and up arrays (previously containing the VAR_UNIT and UNIT_PREFIX values from the TAV file), and nbu and nup (the number of elements in the bu and up arrays) are no longer used. The integer flag tab_type has been added to account for the different handling of some of the VAR_UNIT and VAR_SI_CONVERSION values between AVDC and original Envisat. The STOP_WITH_ERROR routine is no longer called in the event of an error, but an Error State string is returned instead. Bug fixed when testing for a UNIT_PREFIX – previously only the first character of the VAR_UNIT value was checked for a possible UNIT_PREFIX, thus 'deka' (da) was excluded. Bug Fixed when the calculated Power Value of the last base SI unit shown in VAR_SI_CONVERSION is '1', this is now set so that the power value does not appear. Removed from the main set of routines, and made stand-alone.
- Procedure CHECK_METADATA: Sections included to handle checks when an original Envisat table.dat file is used, including: Handling different table.dat format of ORIGINATOR attributes; using DATA_SOURCE_02 instead of AFFILIATION for the DATA_SOURCE checks. The section which puts TAV file VAR_UNITS and UNIT_PREFIX values into arrays for the VAR_UNITS_TEST call is no longer needed and is removed.
- Procedure EXTRACT_DATA: The datasets can now be in any order, rather than the same order as that given by the DATA_VARIABLES attribute; The variable dtest is used as an input flag to indicate whether the procedure call is from SET_UP_STRUCTURE [-1] or READ_DATA [1]. If the call is from SET_UP_STRUCTURE then only the number of data lines (ndl) is determined by the routine; an error is generated in the case where the same VAR_NAME is repeated in the data file; Error messages clarified when an incorrect number of data values are found in the input data file.
- Procedure SET_UP_STRUCTURE: Can calculate the VAR_SIZE value from the input data or from the VAR_SIZE value(s) of the VAR_DEPEND variable name(s), if they are not present in the metadata. Section added to perform checks on the new variables, ALTITUDE/ALTITUDE.GPH/PRESSURE.BOUNDARIES; Additional checks added for VIS_PLOT_TYPE, VIS_SCALE_TYPE, VIS_SCALE_MIN, and VIS_SCALE_MAX values.
- Procedure IDLCR8HDF: Remove HELP,/TRACEBACK call, which identified how the program was called. This was used to stop output to the IDLDE output window in the event that IDL VM was used, but it is not required as IDL VM ignores these print calls; Add DIALOG_BOX to show completion of the program if program inputs are via the INTRO box, and logging window option isn't requested.

4.6 v3.01 – Bug Fix (release April, 2008)

- Fixed bug in the check_min_max_fill procedure so that unwanted precision from all data and relevant metadata values is removed before doing QA checks, to avoid unnecessary error calls when comparing values.

4.7 v3.02 – Improvement (release April, 2008)

- Change made so that any Free Attribute Metadata values that include semi-colons do not have white space removed (i.e. the semi-colons are not defined as separating sub-values).

4.8 v3.03 – Change in Process (release May, 2008)

- Removed the portion of the code that creates the DIMENSIONLIST attribute in an HDF5 file containing the object reference pointers to the variables listed in the VAR_DEPEND values. Note that the concept of including the DIMENSIONLIST attribute was developed by the NCSA as a means to deal with the HDF4 and HDF5 formats in a consistent way (it is applied in the h4toh5.exe code). In HDF5 it appears to be performing the function equivalent to the HDF4 HDF_SD_DIMSET command. For the purposes of the AVDC, however, it is not considered a requirement to have this additional attribute.

4.9 v3.04 – Bug Fix (release June, 2008)

- Fixed bug in the check_min_max_fill procedure that could generate an error message with incorrect information. This occurred when a data value exceeded either the specified VAR_DATA_MIN or VAR_DATA_MAX values, and there were also fill values present in the data, in which case the index value of the out-of-range data point could be determined incorrectly.

4.10 v3.05 – Bug Fix (release October, 2008)

- Changed dataset count variables from INT16 to INT32, to allow handling of datasets that contain more than 32767 values. Account for modified rules for *.BOUNDARIES variable names.

4.11 v3.06 – Change in Process and Improvements (release March, 2009)

- Stop VAR_DEPEND values being saved as Dimension Variables (HDF4).
- Check for SDS names greater than 63-characters together with HDF4 library of 4.2r1 or lower (if so, provide information message to user).
- Change 'WARNING' messages TO 'INFORMATION'.
- Add procedure to handle 'INFORMATION' text output.
- Add 'INFORMATION' text regarding type of Table Attribute Values File found.
- Add Output Directory argument as input (HDF and log files written to this directory).

Note that due to use of the FILE_DIRNAME and FILE_BASENAME functions in the code, IDL v6.0 or greater is now required to run idlcr8hdf.pro and idlcr8hdf.sav.

4.12 v3.07 – Improvement (release June, 2009)

- Check for DATA_STOP_DATE as a Global Attribute (standard for IR template) and calculates the value if required (as per DATA_START_DATE).

4.13 v3.08 – Bug Fixes and Improvements (release December, 2009)

- Allow for VAR_DATA_TYPE=STRING.
- Replace EVDC attribute DATA_TYPE with DATA_LEVEL and remove instances of VAR_MONOTONE, if found.
- Allow VIS_SCALE_MIN/MAX values to be determined by the program for VAR_UNITS=MJD2000.
- Allow for '__' in the composition of DATA_VARIABLES (i.e. no Variable Mode).
- Modifications to the VAR_UNITS_TEST procedure including making VAR_SI_CONVERSION values consistent between EVDC and AVDC, bug fix when there are more than two instances of the same SI unit in VAR_SI_CONVERSION, and putting the third field of the calculated VAR_SI_CONVERSION value in order of power.
- Change the maximum allowable characters in a Metadata attribute line from 4096 to 65535.
- Assign directory path according to the most recent file selection when using DIALOG_BOXES to select inputs (initial path is the IDL working directory).

4.14 v4.2 – New Release (release September, 2011)

While the front-end of the program looks similar to previous versions, there has been a significant rewrite of much of the code. idlcr8hdf now supports the GEOMS Standard, and many of the modifications to the code reflect the changes in this standard compared to the previous AVDC and EVDC standards. Also incorporated into the code are routines that facilitate users moving from the old standards to the new standard, by automatically updating historical metadata inputs to the new standard.

The functionality of idlcr8hdf has also been expanded. It is the basis of the software, used by datacenters supporting the GEOMS Standard, for performing QA on HDF files (geoms_qa). It is used in the harmonization of historical HDF files to the GEOMS standard (geoms_harmon), and it is part of the Conversion Data Suite (cds_convert), which converts data in different formats (including WOUDC, NDACC, Globwave) to GEOMS compliant HDF.

5 Appendix – Testing

This program was written and tested on a Windows platform using ITT IDL versions 6.4 and 8.2, and NCSA HDF libraries v4.2r3 and v5-1.6.3. It has also been tested on Apple Macintosh OS X and Windows platforms using ITT IDL version 7.1.

6 Appendix – Calling idlcr8hdf sample code and script

The following sample code and shell script can be used to call idlcr8hdf in full license or DEMO mode. DEMO mode runs for 7 minutes and does not require a license.

Matlab:

```
if HDF_5
	cstring = sprintf(['idl <<EOF \n idlcr8hdf,'...
		'"%s", "%s", "%s", ' ...
		'"%s", %s, %s, %s\nEOF'], ...
		'data.meta', 'data.data', ...
		'tableattrvalue_04R005_idl.dat', ...
		hdf_dir, '/AVK', '/H5');
else
	cstring = sprintf(['idl <<EOF \n idlcr8hdf,'...
		'"%s", "%s", "%s", ' ...
		'"%s", %s, %s\nEOF'], ...
		'data.meta', 'data.data', ...
		'tableattrvalue_04R007_idl.dat', ...
		hdf_dir, '/AVK');
end
unix(cstring);
```

bash shell:

```
/usr/local/idl70/bin/idl << EOF
idlcr8hdf, 'data.meta','data.data','tableattrvalue_04R007_idl.dat','.',/AVK,/H5
EOF
```

7 Appendix – Contact Information

For comments and to report bugs, please contact:

Ghassan Taha (AVDC Project Coordinator)
NASA Goddard Space Flight Center, Code 613.3
Greenbelt, MD 20771, USA
E-mail: ghassan.taha@nasa.gov
Phone: (+01) 301-614-5979
Fax: (+01) 301-614-5903

and

Ian Boyd
Department of Astronomy
619 Lederle Graduate Research Centre, University of Massachusetts
710 North Pleasant St.
Amherst, MA 01003, USA
E-mail: iboyd@astro.umass.edu