HDF-EOS2 v3.0 Users Guide

EED2-170-001

HDF-EOS Library User's Guide for EOSDIS Evolution and Development-2 (EED-2) Contract, Volume 1: Overview and Examples

Technical Paper

July 2021

Prepared Under Contract # NNG15HZ39C

Raytheon Company
Riverdale, Maryland

This page intentionally left blank.

Preface

This document is a Users Guide for HDF-EOS (Hierarchical Data Format - Earth Observing System) library tools. HDF is the scientific data format standard selected by NASA as the baseline standard for EOS. This Users Guide accompanies Version 3.0 software, which is available to the user community on the Earthdata Wiki page: https://wiki.earthdata.nasa.gov/display/DAS/Toolkit+Downloads . This library is aimed at EOS data producers and consumers, who will develop their data into increasingly higher order products. These products range from calibrated Level 1 to Level 4 model data. The primary use of the HDF-EOS library will be to create structures for associating geolocation data with their associated science data. This association is specified by producers through use of the supplied library. Most EOS data products which have been identified, fall into categories of grid, point or swath structures, which are implemented in the current version of the library. Services based on geolocation information will be built on HDF-EOS structures. Producers of products not covered by these structures, e.g. non-geolocated data, can use the standard HDF libraries.

In the ECS (EOS Core System) production system, the HDF-EOS library will be used in conjunction with SDP (Science Data Processing) Toolkit software. The primary tools used in conjunction with HDF-EOS library will be those for metadata handling, process control and status message handling. Metadata tools will be used to write ECS inventory and granule specific metadata into HDF-EOS files, while the process control tools will be used to access physical file handles used by the HDF tools. (*SDP Toolkit Users Guide for the EOSDIS Evolution and Development-2 Contract, December 2017, EED2-333-001).*

HDF-EOS is an extension of The HDF Group (THG) HDF and uses HDF library calls as an underlying basis. Version 4.2.13 or 4.2.15 of HDF can be used. The library tools are written in the C language and a FORTRAN interface is provided. The current version contains software for creating, accessing, and manipulating Grid, and Swath structures. This document includes overviews of the interfaces, and code examples. EOSView, the HDF-EOS viewing tool, uses an older version of the library.

An email address has been provided for user help:

sdps-support@earthdata.nasa.gov

Any questions should be addressed to:

Data Management Office
Raytheon Company
EOSDIS Evolution and Development-2 (EED-2) Contract
5700 Rivertech Court
Riverdale, MD 20737

Abstract

This document will serve as the user's guide to the HDF-EOS file access library. HDF refers to the scientific data format standard selected by NASA as the baseline standard for EOS, and HDF-EOS refers to EOS conventions for using HDF. This document will provide information on the use of the three interfaces included in HDF-EOS – Point, Swath, and Grid – including overviews of the interfaces, and code examples. This document should be suitable for use by data producers and data users alike.

Keywords: HDF-EOS, Metadata, Standard Data Format, Standard Data Product, Disk Format, Point, Grid, Swath, Projection, Array, Browse

This page intentionally left blank.

Contents

Preface

Abstract

1. Introduction

2. Related Documentation

3. Overview of HDF-EOS

4. Point Data

5. Swath Data

6. Grid Data

List of Figures

List of Tables

Appendix A. Installation and Maintenance

Abbreviations and Acronyms

This page intentionally left blank.

1. Introduction

1.1 Identification

The *HDF-EOS User's Guide* was prepared under the EOSDIS Evolution and Development-2 Contract (NNG15HZ39C).

1.2 Scope

This document is intended for use by anyone who wishes to write software to create or read EOS data products. Users of this document will likely include EOS instrument team science software developers and data product designers, DAAC personnel, and end users of EOS data products such as scientists and researchers.

1.3 Purpose and Objectives

This document will serve as a user's guide for the HDF-EOS file access library developed for ECS. Upon reading this document, the reader should have a thorough understanding of each data model and corresponding programming interface provided as part of HDF-EOS. Specifically, this user's guide contains an overview of each data model, a complete function-by-function reference for each software interface, and sample programs illustrating the basic features of each interface.

The reader should note that this paper will not discuss the HDF structures underlying HDF-EOS nor the specific conventions employed. For more information on HDF, its design philosophy, and its logical and physical formats, the reader is referred to The HDF Group (THG) (formerly NCSA) documentation listed in Section 2.2 Applicable Documents. For more information on the conventions employed by HDF-EOS, the reader is referred to the various design White Papers listed in Section 2.2.

Important Note:

The FORTRAN-literate reader is cautioned that dimension ordering is row-major in C (last dimension varying fastest), whereas FORTRAN uses column-major ordering (first dimension varying fastest). Therefore, FORTRAN programmers should take care to use dimensions in the reverse order to that shown in the text of this document. (FORTRAN code examples are correct as written.)

1.4 Status and Schedule

January 1996, Prototype Library Available

January 1996 Version 1 API Available

March 1996, Version 1 API Frozen

April 1996 - Delivery of HDF-EOS Users Guide and Beta Software

June 1996 - Delivery of Version 1 HDF-EOS, Release A EOSView Available, Beta 1.9

November 1996 - Delivery of Version 1.5 of the HDF-EOS Library. Release A of EOSView, Beta 2.1 Available, Delivery of SDP Toolkit Version 5.1.1

May 1997 - Delivery of Version 2.0 of the HDF-EOS Library. Release B.0 of EOSView, Beta 2.3 Available, Delivery of SDP Toolkit Version 5.2

October 1997 - Delivery of Version 2.1 of the HDF-EOS Library. Release B.0 of EOSView, Delivery of SDP Toolkit Version 5.2.1

March 1998 - Delivery of Version 2.2 of the HDF-EOS Library. Release B.0 of EOSView, Delivery of SDP Toolkit Version 5.2.2

October 1998 - Delivery of Version 2.3 of the HDF-EOS Library. Release B.0 of EOSView, Delivery of SDP Toolkit Version 5.2.3

January 1999 - Delivery of Version 2.4 of the HDF-EOS Library. Release B.0 of EOSView, Delivery of SDP Toolkit Version 5.2.4

June 1999 - Delivery of Version 2.5 of the HDF-EOS Library. Release B.0 of EOSView, Delivery of SDP Toolkit Version 5.2.5

December 1999 - Delivery of Version 2.6 of the HDF-EOS Library. Release B.0 of EOSView, Delivery of SDP Toolkit Version 5.2.6

October 2000 - Delivery of Version 2.7 of the HDF-EOS Library. Release B.0 of EOSView, Delivery of SDP Toolkit Version 5.2.7

March 2002 - Delivery of Version 2.8 of the HDF-EOS Library. Release B.0 of EOSView, Delivery of SDP Toolkit Version 5.2.8

April 2003 - Delivery of Version 2.9 of the HDF-EOS Library. Delivery of SDP Toolkit Version 5.2.9

October 2003 - Delivery of Version 2.10 of the HDF-EOS Library. Delivery of SDP Toolkit Version 5.2.10

May 2004 - Delivery of Version 2.11 of the HDF-EOS Library. Release of HDFView_EOS, Delivery of SDP Toolkit Version 5.2.11

August 2004 - Delivery of Version 2.12 of the HDF-EOS Library. Delivery of SDP Toolkit Version 5.2.12

April 2005 - Delivery of Version 2.13 of the HDF-EOS Library. Delivery of SDP Toolkit Version 5.2.13

March 2006 - Delivery of Version 2.14 of the HDF-EOS Library. Delivery of SDP Toolkit Version 5.2.14

February 2008 - Delivery of Version 2.15 of the HDF-EOS Library. Delivery of SDP Toolkit Version 5.2.15

July 2009 - Delivery of Version 2.16 of the HDF-EOS Library. Delivery of SDP Toolkit Version 5.2.16

August 2010 - Delivery of Version 2.17 of the HDF-EOS Library. Delivery of SDP Toolkit Version 5.2.17

January 2012 - Delivery of Version 2.18 of the HDF-EOS Library. Delivery of SDP Toolkit Version 5.2.18

March 2014 - Delivery of Version 2.19 of the HDF-EOS Library. Delivery of SDP Toolkit Version 5.2.19

December 2017 - Delivery of Version 2.20 of the HDF-EOS Library. Delivery of SDP Toolkit Version 5.2.20

July 2021 – Delivery of Version 3.0 of the HDF-EOS Library.

1.5 Document Organization

This document is organized as follows:

- Section 1 - Introduction - Presents Scope and Purpose of this document

- Section 2 - Related Documentation

Section 3 - Overview of HDF-EOS - Background and design features of the libr ar

- y.

- Section 4 - Point Data - Design features and listing of the HDF-EOS Point Library.

Section 5 - Swath Data - Design features and listing of the HDF-EOS Swath Libr ar

- y.

- Section 6 - Grid Data - Design features and listing of the HDF-EOS Grid Library.

- Appendix A - Installation Instructions, Test Drivers, User Feedback

- Acronyms

The accompanying Function Reference Guide is organized as follows:

- Section 1 - Introduction

Section 2 - Function Reference - Specification of the HDF-EOS, Swath and

- Grid
- APIs
- Acronyms

This page intentionally left blank.

2. Related Documentation

2.1 Parent Documents

The following documents are the parents from which this document's scope and content derive:

456-TP-013	The HDF-EOS Design Document for the ECS Project
170-WP-002	Thoughts on HDF-EOS Metadata, A White Paper for the ECS Project
170-WP-003	The HDF-EOS Swath Concept, A White Paper for the ECS Project
170-WP-011	The HDF-EOS Grid Concept, A White Paper for the ECS Project

170-WP-012	The HDF-EOS Point Concept, A White Paper for the ECS Project

2.2 Related Documents

The following documents are referenced within this technical paper, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this document.

EED2-333-001	Release 9 SDP Toolkit Users Guide for the EED Contract
163-WP-001	An ECS Data Provider's Guide to Metadata in Release A, A White Paper for the ECS Project
175-WP-001	HDF-EOS Primer for Version 1 EOSDIS, A White Paper for the ECS Project

none	Getting Started with HDF, Version 3.2, University of Illinois, May 1993
none	NCSA HDF Reference Manual, Version 3.3, University of Illinois, February 1994
none	NCSA HDF Specification and Developer’s Guide, Version 3.2, University of Illinois, September 1993
none	NCSA HDF User’s Guide, Version 4.0, University of Illinois, February 1996
none	NCSA HDF User’s Guide, Version 3.3, University of Illinois, March 1994
none	An Album of Map Projections, USGS Professional Paper 1453, Snyder and Voxland, 1989
none	Map Projections - A Working Manual, USGS Professional Paper 1395, Snyder, 1987

The WMO Format for the Storage of Weather Product Information and the Exchange of Weather Product Messages in Gridded Binary Form, John D. Stackpole, Office Note 388, GRIB Edition 1, U.S. Dept. of Commerce, NOAA, National Weather Service National Meteorological Center, Automation Division, Section 1, pp. 9-12, July 1,
none 1994.

The Michigan Earth Grid: Description, Registration Method for SSM/I Data, and Derivative Map Projection, John F. Galntowicz, Anthony W. England, The University of Michigan, Radiation Laboratory, Ann Arbor, Michigan,
none Feb. 1991.

none	Selection of a Map Grid for Data Analysis and Archival, William B. Rossow, and Leonid Garder, American Meteorological Society Notes, pp. 1253-1257, Aug. 1984.
none	Level-3 SeaWiFS Data Products: Spatial and Temporal Binning Algorithms, Janet W. Campbell, John M. Blaisdell, and Michael Darzi, NASA Technical Memorandum 104566, GSFC, Volume 32, Appendix A, Jan. 13, 1995.

3. Overview of HDF-EOS

3.1 Background

The Hierarchical Data Format (HDF) has been selected by the EOSDIS Project as the format of choice for standard product distribution. HDF is a function library that was originally developed by The HDF Group (THG) to provide a portable storage mechanism for supercomputer simulation results. Although this user's guide does not attempt to explain the inner workings of THG HDF, a cursory knowledge of HDF may help the reader to understand the basic workings of HDF-EOS.

HDF files consist of a directory and a collection of data objects. Every data object has a directory entry, containing a pointer to the data object location, and information defining the datatype (much more information about HDF can be found in the HDF documentation referenced in Section 2.2 of this Guide). Many of the HDF defined datatypes map well to EOS datatypes. Examples include raster images, multi-dimensional arrays, and text blocks. There are other EOS datatypes, however, that do not map directly to HDF datatypes, particularly in the case of geolocated datatypes. Examples include projected grids, satellite swaths, and field campaign or point data. Therefore, some additions to traditional HDF are required to fully support these datatypes.

To bridge the gap between the needs of EOS data products and the capabilities of HDF, three new EOS specific datatypes – *point* , *swath* , and *grid* – have been defined within the HDF framework. Each of these new datatypes is constructed using conventions for combining standard HDF datatypes and is supported by a special application programming interface (API) which aids the data product user or producer in the application of the conventions. The APIs allow data products to be created and manipulated in ways appropriate to each datatype, without regard to the actual HDF objects and conventions underlying them.

The sum of these new APIs comprises the HDF-EOS library. The *Point* interface is designed to support data that has associated geolocation information but is not organized in any well defined spatial or temporal way. The *Swath* interface is tailored to support time-ordered data such as satellite swaths (which consist of a time-ordered series of scanlines), or profilers (which consist of a time-ordered series of profiles). The *Grid* interface is designed to support data that has been stored in a rectilinear array based on a well defined and explicitly supported projection.

3.2 Design Philosophy

Since the HDF-EOS library is intended to support end users of EOS data as well as EOS data producers, it is essential that HDF-EOS be available separately from other ECS software. For this reason, HDF-EOS does not rely on any other ECS software, including the SDP Toolkit. It is treated as an extension to the HDF library and, as such, it follows the general design philosophy and coding style of HDF. For more information on the design of HDF, please refer to the appropriate HDF documentation listed in Section 2.2.

3.3 Packaging

Because of the functional overlap of HDF, HDF-EOS, and the SDP Toolkit, it is important to understand what each one contains and how they are related. The HDF Group's HDF is a subroutine library freely available as source code from the HDF Group (hdfgroup.org). The basic HDF library has its own documentation and comes with a selection of simple utilities.

HDF-EOS is a higher-level library available from the ECS project as an add-on to the basic HDF library. It requires The HDF Group's HDF for successful compiling and linking and will be widely available (at no charge) to all interested parties.

The SDP Toolkit is a large, complex library of functions for use by EOS data producers. It presents a standard interface to Distributed Active Archive Center (DAAC) services for data processing, job scheduling, and error handling. While the SDP Toolkit may be available to individual researchers, it is unlikely to be of great use outside of EOS DAACs and Science Computing Facilities (SCF). The Toolkit distribution includes source code for both HDF and HDF-EOS.

EOS instrument data producers will use the SDP Toolkit in conjunction with the HDF-EOS and HDF libraries. Of primary importance will be process control and metadata handling tools. The former will be used to access physical file handles required by the HDF library. The SDP Toolkit uses logical file handles to access data, while HDF (HDF-EOS) requires physical handles. Users will be required to make one additional call, using the SDP toolkit to access the physical handles. Please refer to the SDP Toolkit Users Guide for the EED Contract, December 2017, 333-EED2-001, Revision 01, Section 6.2.1.2 for an example). Section 7 of this document gives examples of HDF-EOS usage in conjunction with the SDP Toolkit.

Metadata tools will be used to access and write inventory and granule specific metadata into their designated HDF structures. Please refer to Section 6.2.1.4 of the SDP Toolkit Users Guide.

We make an important distinction between granule metadata and the structural metadata referred to in the software description below. Structural metadata specifies the internal HDF-EOS file structure and the relationship between geolocation data and the data itself. Structural metadata is created and then accessed by calling the HDF-EOS functions. Granule metadata will be used by ECS to perform archival services on the data. A copy will be attached to HDF-EOS files by SDP toolkit calls and another copy is placed in the ECS archives. The two sets of metadata are not dynamically linked. However, the data producer should use consistent naming conventions when writing granule metadata when calling the HDF-EOS API. Please refer to the examples in Section 7, below.

THG HDF libraries, on which HDF-EOS is based, is installed automatically with the SDP Toolkit installation script. Please refer to The SDP Toolkit Users Guide for the EED Contract, Section 5 for information pertaining installation and maintenance of the SDP Toolkit.

Note that a subsetted version of the SDP Toolkit is also available. This is the MTD Toolkit, which contains time and date conversion and metadata tool only. The MTD Toolkit is intended to be used by data producers who will produce products outside ECS but will archive the data within ECS.

This page intentionally left blank.

4. Point Data

4.1 Introduction

This section will describe the routines available for storing and retrieving HDF-EOS *Point Data* . A Point Data set is made up of a series of data records taken at [possibly] irregular time intervals and at scattered geographic locations. Point Data is the most loosely organized form of geolocated data supported by HDF-EOS. Simply put, each data record consists of a set of one or more data values representing, in some sense, the state of a point in time and/or space.

Figure 4-1 shows an example of a simple point data set. In this example, each star on the map represents a reporting station. Each record in the data table contains the location of the point on the Earth and the measurements of the temperature and dew point at that location. This sort of point data set might represent a snapshot in time of a network of stationary weather reporting facilities.

Lat	Lon	Temp(C)	Dewpt(C)
61.12	-149.48	15.00	5.00
45.31	-122.41	17.00	5.00
38.50	-77.00	24.00	7.00
38.39	-90.15	27.00	11.00
30.00	-90.05	22.00	7.00
37.45	-122.26	25.00	10.00
18.00	-76.45	27.00	4.00
43.40	-79.23	30.00	14.00
34.03	-118.14	25.00	4.00
32.45	-96.48	32.00	8.00
33.30	-112.00	30.00	10.00
42.15	-71.07	28.00	7.00
35.05	-106.40	30.00	9.00
34.12	-77.56	28.00	9.00
46.32	-87.25	30.00	8.00
47.36	-122.20	32.00	15.00
39.44	-104.59	31.00	16.00
21.25	-78.00	28.00	7.00
44.58	-93.15	32.00	13.00
41.49	-87.37	28.00	9.00
25.45	-80.11	19.00	3.00

Figure 4-1. A Simple Point Data Example

A more realistic example might record the changes in the parameters over time by including multiple values of the parameters for each location. In this case, the identity and location of the reporting stations would remain constant, while the values of the measured parameters would vary. This sort of setup naturally leads to a hierarchical table arrangement where a second table is used to record the static information about each reporting station, thereby removing the redundant information that would be required by a single “flat” table and acting as an index for quick access to the main data table. Such an arrangement is depicted in Figure 4-2.

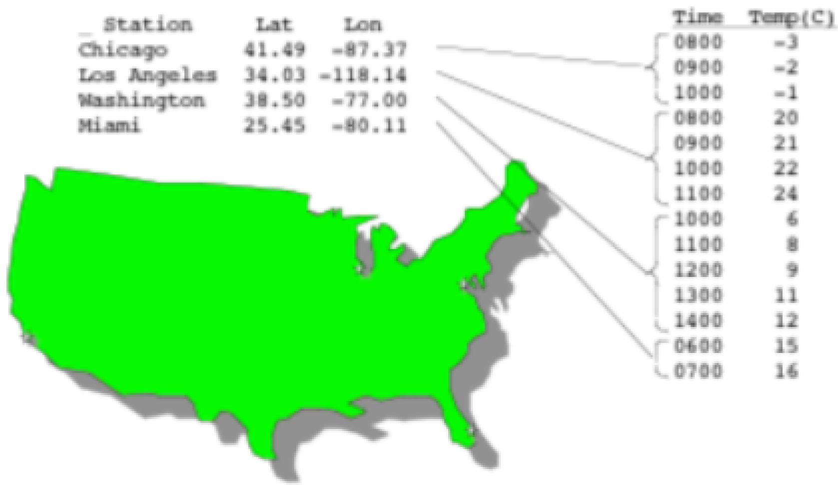
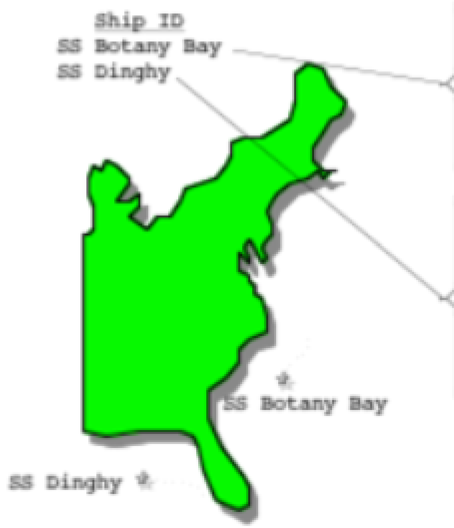


Figure 4-2. Recording Points Over Time

An even more complex point data set may represent data taken at various times aboard a moving ship. Here, the only thing that remains constant is the identity of the reporting ship. Its location varies with each data reading and is therefore treated similarly to the data. Although this example seems more complicated than the static example cited above, its implementation is nearly identical. Figure 4-3 shows the tables resulting from this example. Note that the station location information has been moved from the static table to the data table.



Lat	Lon	Temp(C)	Salinity(PPM)
34.12	-74.56	18	146
33.24	-74.62	18	158
32.45	-75.65	19	151
32.03	-76.02	19	160
30.94	-76.48	20	162
30.04	-76.89	20	165
29.88	-76.94	21	159
25.51	-84.67	26	167
25.97	-85.06	26	173
26.22	-85.65	25	168
26.53	-86.14	26	162
26.58	-86.74	27	144
26.62	-87.45	27	156
26.87	-87.97	25	160
27.09	-88.68	25	166

Figure 4-3. Point Data from a Moving Platform

In fact, the hierarchical arrangement of the tables in the last two examples can be expanded upon to include up to seven indexing levels (a total of eight levels, including the bottom level data table). A normal data access on a multi-level hierarchical point data set would involve starting at the top (first) level and following successive pointers down the structure until the desired information is found. As each level is traversed, more and more specific information is gained about the data.

In rare cases, it may be advantageous to access a point data set from the bottom up. The point data model implemented in HDF-EOS provides for backward (or upward) pointers which facilitate bottom-up access.

4.2 Applicability

The Point data model is very flexible and can be used for data at almost any level of processing. It is expected that point structure will be used for data for which there is no spatial or temporal organization, although lack of those characteristics do not preclude the use of a point structure. For example, profile data, which is accumulated in sparsely located spatial averages, may be most useful in a point structure.

4.3 The Point Data Interface

The PT interface consists of routines for storing, retrieving, and manipulating data in point data sets.

4.3.1 PT API Routines

All C routine names in the point data interface have the prefix “PT” and the equivalent FORTRAN routine names are prefixed by “pt.” The PT routines are classified into the following categories:

Access routines initialize and terminate access to the PT interface and point datasets (inc luding opening and closing files).

- *Definition* routines allow the user to set key features of a point data set.

- *Basic I/O* routines read and write data and metadata to a point data set.

Index I/O routines read and write information which links two tables in a point dat

a
se
• t.

- *Inquiry* routines return information about data contained in a point data set.

- *Subset* routines allow reading of data from a specified geographic region.

The PT function calls are listed in Table 4-1 and are described in detail in the Software Reference Guide that accompanies this document. The page number column in the following table refers to the Software Reference Guide.

Table 4-1. Summary of the Point Interface

Category	Routine Name		Description	Page
	C	FORTRAN		Nos.
Access	PTopen	ptopen	creates a new file or opens an existing one	2-28
	PTcreate	ptcreate	creates a new point data set and returns a handle	2-6
	PTattach	ptattach	attaches to an existing point data set	2-2
	PTdetach	ptdetach	releases a point data set and frees memory	2-14
	PTclose	ptclose	closes the HDF-EOS file and deactivates the point interface	2-5
Definition	PTdeflevel	ptdeflev	defines a level within the point data set	2-8
	PTdeflinkage	ptdeflink	defines link field to use between two levels	2-10
	PTdefvrtregion	ptdefvrtreg	defines a vertical subset region	2-12
Basic I/O	PTwritelevel	ptwrlev	writes (appends) full records to a level	2-40
	PTreadlevel	ptrdlev	reads data from the specified fields and records of a level	2-32
	PTupdatelevel	ptuplev	updates the specified fields and records of a level	2-37
	PTwriteattr	ptwrattr	creates or updates an attribute of the point data set	2-39
	PTreadattr	ptrdattr	reads existing attribute of point data set	2-31
	PTnlevels	ptnlevs	returns the number of levels in a point data set	2-26
	PTnrecs	ptnrecs	returns the number of records in a level	2-27
	PTnfields	ptnflds	returns number of fields defined in a level	2-25
	PTlevelinfo	ptnlevinfo	returns information about a given level	2-24
	PTlevelindx	ptlevidx	returns index number for a named level	2-23
Inquiry	PTbcklinkinfo	ptblinkinfo	returns link field to previous level	2-4
	PTfwdlinkinfo	ptflinkinfo	returns link field to following level	2-17
	PTgetlevelname	ptgetlevname	returns level name given level number	2-18
	PTsizeof	ptsizeof	returns size in bytes for specified fields in a point	2-36
	PTattrinfo	ptattrinfo	returns information about point attributes	2-3
	PTinqattrs	ptinqattrs	retrieves number and names of attributes defined	2-21
	PTinqpoint	ptinqpoint	retrieves number and names of points in file	2-22
Utility	PTgetrecnums	ptgetrecnums	returns corresponding record numbers in a related level	2-19
Subset	PTdefboxregion	ptdefboxreg	define region of interest by latitude/longitude	2-7
	PTregioninfo	ptreginfo	returns information about defined region	2-34
	PTregionrecs	ptregrecs	returns # of records and record #s within region	2-35
	PTextractregion	ptextreg	read a region of interest from a set of fields in a single level	2-16
	PTdeftimeperiod	ptdeftmeper	define time period of interest	2-11
	PTperiodinfo	ptperinfo	returns information about defined time period	2-29
	PTperiodrecs	ptperrecs	returns # of records and record #s within time period	2-30
	PTextractperiod	ptextper	read a time period from a set of fields in a single level	2-15

4.3.2 File Identifiers

As with all HDF-EOS interfaces, file identifiers in the PT interface are 32-bit values, each uniquely identifying one open data file. They are not interchangeable with other file identifiers created with other interfaces.

4.3.3 Point Identifiers

Before a point data set is accessed, it is identified by a name, which is assigned to it upon its creation. The name is used to obtain a *point identifier* . After a point data set has been opened for access, it is uniquely identified by its point identifier.

4.4 Programming Model

The programming model for accessing a point data set through the PT interface is as follows:

1. Open the file and initialize the PT interface by obtaining a file id from a file name.
2. Open OR create a point data set by obtaining a point id from a point name.
3. Perform desired operations on the data set.
4. Close the point data set by disposing of the point id.
5. Terminate point access to the file by disposing of the file id.

To access a single point data set that already exists in an HDF-EOS file, the calling program must contain the following sequence of C calls:

```
file_id = PTopen(filename, access_mode);
pt_id = PTattach(file_id, point_name);
<Optional operations>
status = PTdetach(pt_id);
status = PTclose(file_id);
```

To access several files at the same time, a calling program must obtain a separate id for each file to be opened. Similarly, to access more than one point data set, a calling program must obtain a separate point id for each data set. For example, to open two data sets stored in two files, a program would execute the following series of C function calls:

```
file_id_1 = PTopen(filename_1, access_mode);
pt_id_1 = PTattach(file_id_1, point_name_1);
file_id_2 = PTopen(filename_2, access_mode);
pt_id_2 = PTattach(file_id_2, point_name_2);
<Optional operations>
status = PTdetach(pt_id_1);
status = PTclose(file_id_1);
status = PTdetach(pt_id_2);
status = PTclose(file_id_2);
```

Because each file and point data set is assigned its own identifier, the order in which files and data sets are accessed is very flexible. However, it is very important that the calling program individually discard each identifier before terminating. Failure to do so can result in empty or, even worse, invalid files being produced.

5. Swath Data

I
5.1 ntroduction

The Swath concept for HDF-EOS is based on a typical satellite swath, where an instrument takes a series of scans perpendicular to the ground track of the satellite as it moves along that ground track. Figure 5-1 below shows this traditional view of a swath.

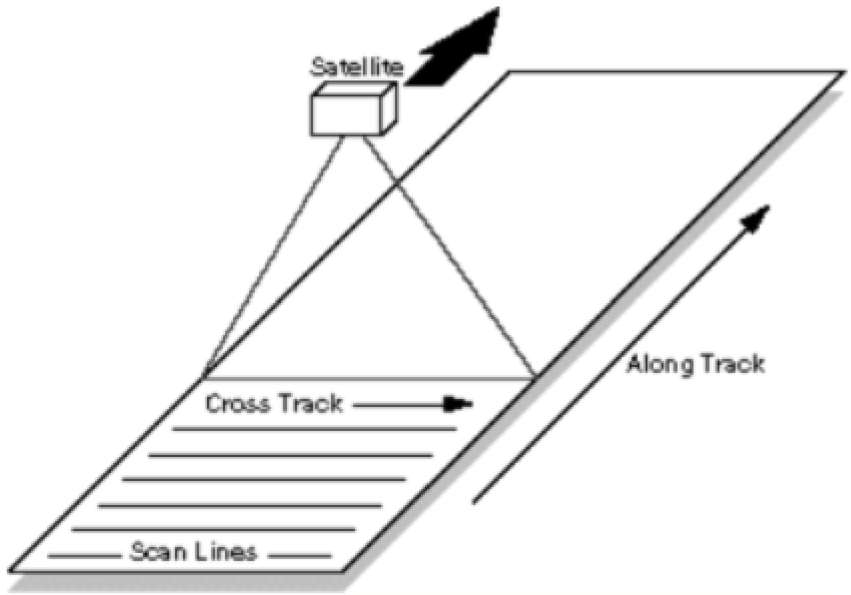


Figure 5-1. A Typical Satellite Swath: Scanning Instrument

Another type of data that the Swath is equally well suited to arise from a sensor that measures a vertical profile, instead of scanning across the ground track. The resulting data resembles a standard Swath tipped up on its edge. Figure 5-2 shows how such a Swath might look.

In fact, the two approaches shown in Figures 5-1 and 5-2 can be combined to manage a profiling instrument that scans across the ground track. The result would be a three dimensional array of measurements where two of the dimensions correspond to the standard scanning dimensions (along the ground track and across the ground track), and the third dimension represents a height above the Earth or a range from the sensor. The "horizontal" dimensions can be handled as normal geographic dimensions, while the third dimension can be handled as a special "vertical" dimension.

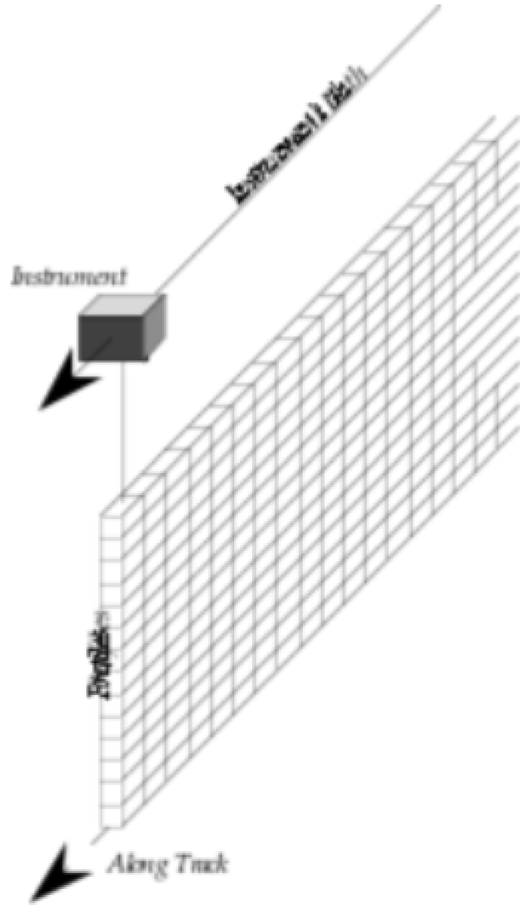


Figure 5-2. A Swath Derived from a Profiling Instrument

A standard Swath is made up of four primary parts: data fields, geolocation fields, dimensions, and dimension maps. An optional fifth part called an index can be added to support certain kinds of access to Swath data. Each of the parts of a Swath is described in detail in the following subsections.

5.1.1 Data Fields

Data fields are the main part of a Swath from a science perspective. Data fields usually contain the raw data (often as *counts*) taken by the sensor or parameters derived from that data on a value-for-value basis. All the other parts of the Swath exist to provide information about the data fields or to support particular types of access to them. Data fields typically are two-dimensional arrays, but can have as few as one dimension or as many as eight, in the current library implementation. They can have any valid C data type.

5.1.2 Geolocation Fields

Geolocation fields allow the Swath to be accurately tied to particular points on the Earth's surface. To do this, the Swath interface requires the presence of at least a time field ("Time") or a latitude/longitude field pair ("Latitude" [1] and "Longitude"). Geolocation fields must be either one- or two-dimensional and can have any data type.

In addition to the Geolocation fields “Latitude”, “Longitude”, and “Time”, one can define other geolocation fields related to the third or higher dims in the datafields. For example, if the third dim in a datafield is altitude, then on can define a geofield called “Altitude” and map the third dimension in the data field to the dimension of this field like the first 2 dimensions in the swath.

5.1.3 Dimensions

Dimensions define the axes of the data and geolocation fields by giving them names and sizes. In using the library, dimensions must be defined before they can be used to describe data or geolocation fields.

Every axis of every data or geolocation field, then, must have a dimension associated with it. However, there is no requirement that they all be unique. In other words, different data and geolocation fields may share the same named dimension. In fact, sharing dimension names allows the Swath interface to make some assumptions about the data and geolocation fields involved which can reduce the complexity of the file and simplify the program creating or reading the file.

5.1.4 Dimension Maps

Dimension maps are the glue that holds the Swath together. They define the relationship between data fields and geolocation fields by defining, one-by-one, the relationship of each dimension of each geolocation field with the corresponding dimension in each data field. In cases where a data field and a geolocation field share a named dimension, no explicit dimension map is needed. In cases where a data field has more dimensions than the geolocation fields, the “extra” dimensions are left unmapped.

In many cases, the size of a geolocation dimension will be different from the size of the corresponding data dimension. To take care of such occurrences, there are two pieces of information that must be supplied when defining a dimension map: the *offset* and the *increment* . The offset tells how far along a data dimension you must travel to find the first point to have a corresponding entry along the geolocation dimension. The increment tells how many points to travel along the data dimension before the next point is found for which there is a corresponding entry along the geolocation dimension. Figure 5-3 depicts a dimension map.

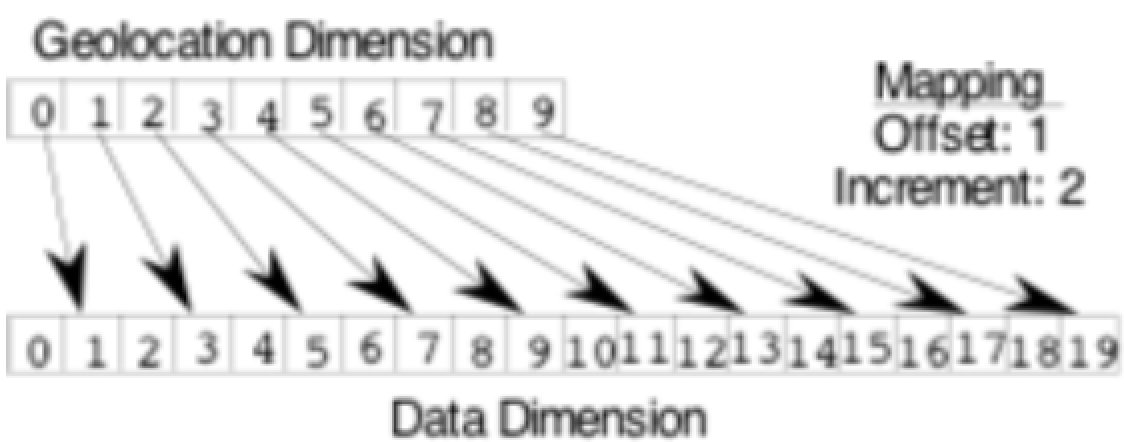


Figure 5-3. A “Normal” Dimension Map

The “data skipping” method described above works quite well if there are fewer regularly spaced geolocation points than data points along a particular pair of mapped dimensions of a Swath. It is conceivable, however, that the reverse is true – that there are more regularly spaced geolocation points than data points. In that event, both the offset and increment should be expressed as negative values to indicate the reversed relationship. The result is shown in Figure 5-4. Note that in the reversed relationship, the offset and increment are applied to the geolocation dimension rather than the data dimension.

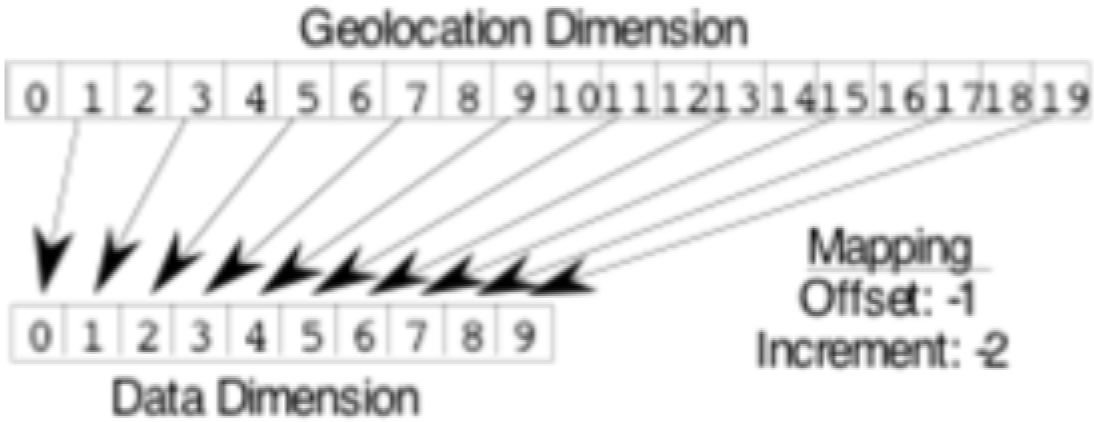


Figure 5-4. A “Backwards” Dimension Map

5.1.5 Index

The index was designed specifically for Landsat 7 data products. These products require geolocation information that does not repeat at regular intervals throughout the Swath. The index allows the Swath to be broken into unequal length *scenes* which can be individually geolocated.

For this version of the HDF-EOS library, there is no particular content required for the index. It is quite likely that a later version of the library will impose content requirements on the index in an effort to standardize its use.

5.2 Applicability

The Swath data model is most useful for satellite [or similar] data at a low level of processing. The Swath model is best suited to data at EOS processing levels 1A, 1B, and 2. Swath structures are for data storage by MODIS, MISR, MOPITT instrument teams on EOS-Terra and AIRS in EOS-AQUA.

5.3 The Swath Data Interface

The SW interface consists of routines for storing, retrieving, and manipulating data in swath data sets.

5.3.1 SW API Routines

All C routine names in the swath data interface have the prefix “SW” and the equivalent FORTRAN routine names are prefixed by “sw.” The SW routines are classified into the following categories:

- *Access routines* initialize and terminate access to the SW interface and swath data sets (including opening and closing files).

- *Definition* routines allow the user to set key features of a swath dataset.

- *Basic I/O* routines read and write data and metadata to a swath dataset.

- *Inquiry* routines return information about data contained in a swath dataset.

- *Subset* routines allow reading of data from a specified geographic region.

The SW function calls are listed in Table 5-1 and are described in detail in the Software Reference Guide that accompanies this document. The page number column in the following table refers to the Software Reference Guide.

Table 5-1. Summary of the Swath Interface (1 of 2)

Category	Routine Name C	Routine Name FORTRAN	Description	Page Nos.
Access	SWopen	swopen	opens or creates HDF file in order to create, read, or write a swath	2-87
	SWcreate	swcreate	creates a swath within the file	2-45
	SWattach	swattach	attaches to an existing swath within the file	2-41
	SWdetach	swdetach	detaches from swath interface	2-63
	SWclose	swclose	closes file	2-43
Definition	SWdefdim	swdefdim	defines a new dimension within the swath	2-52
	SWdefdimmap	swdefmap	defines the mapping between the geolocation and data dimensions	2-53
	SWdefidxmap	swdefimap	defines a non-regular mapping between the geolocation and data dimension	2-57
	SWdefgeofield	swdefgfld	defines a new geolocation field within the swath	2-55
	SWdefdatafield	swdefdfld	defines a new data field within the swath	2-50
	SWdefcomp	swdefcomp	defines a field compression scheme	2-48
	SWwritegeometa	swwrgmeta	writes field metadata for an existing swath geolocation field	2-108
	SWwritedatameta	swwrdmeta	writes field metadata for an existing swath data field	2-105
Basic I/O	SWwritefield	swwrfld	writes data to a swath field	2-106
	SWreadfield	swrdfld	reads data from a swath field.	2-91
	SWwriteattr	swwrattr	writes/updates attribute in a swath	2-104
	SWreadattr	swrdattr	reads attribute from a swath	2-90
	SWsetfillvalue	swsetfill	sets fill value for the specified field	2-101
	SWgetfillvalue	swgetfill	retrieves fill value for the specified field	2-75
Inquiry	SWinqdims	swinqdims	retrieves information about dimensions defined in swath	2-80
	SWinqmaps	swinqmaps	retrieves information about the geolocation relations defined	2-83
	SWinqidxmaps	swinqimaps	retrieves information about the indexed geolocation/data mappings defined	2-82
	SWinqgeofields	swinqgflds	retrieves information about the geolocation fields defined	2-81
	SWinqdatafields	swinqdflds	retrieves information about the data fields defined	2-79
	SWinqattrs	swinqattrs	retrieves number and names of attributes defined	2-78
	SWnentries	swnentries	returns number of entries and descriptive string buffer size for a specified entity	2-86
	SWdiminfo	swdiminfo	retrieve size of specified dimension	2-64
	SWmapinfo	swmapinfo	retrieve offset and increment of specified geolocation mapping	2-85
	SWidxmapinfo	swimapinfo	retrieve offset and increment of specified geolocation mapping	2-76
	SWindexinfo	swidxinfo	Retrieve the indices information about a subsetted region	2-77
	SWattrinfo	swattrinfo	returns information about swath attributes	2-42
	SWfieldinfo	swfldinfo	retrieve information about a specific geolocation or data field	2-68
	SWcompinfo	swcompinfo	retrieve compression information about a field	2-44
	SWinqswath	swinqswath	retrieves number and names of swaths in file	2-84
	SWregionindex	swregidx	returns information about the swath region ID	2-93

Table 5-1. Summary of the Swath Interface (2 of 2)

Category	Routine Name C	Routine Name FORTRAN	Description	Page Nos.
Subset	SWupdateidxmap	swupimap	update map index for a specified region	2-102
	SWgeomapinfo	swgmapinfo	retrieves type of dimension mapping when first dimension is geodim	2-70
	SWdefboxregion	swdefboxreg	define region of interest by latitude/longitude	2-46
	SWregioninfo	swreginfo	returns information about defined region	2-95
	SWextractregion	swextreg	read a region of interest from a field	2-67

	SWdeftimeperiod	swdeftmeper	define a time period of interest	2-58
	SWperiodinfo	swperinfo	returns information about a defined time period	2-88
	SWextractperiod	swextper	extract a defined time period	2-66
	SWdefvrtregion	swdefvrtreg	define a region of interest by vertical field	2-60
	SWdupregion	swdupreg	duplicate a region or time period	2-65
Dimension Scale	SWsetdimscale	swsetdimscale	sets dimension scale for a given dimension	2-97
	SWdefdimscale	swdefdimscale	sets dimension scale for a given dimension that is used in all fields defined in the swath	2-97
	SWgetdimscale	swgetdimscale	gets dimension scale for a given dimension	2-71
	SWsetdimstrs	swsetdimstrs	sets the label, unit, and format strings for a given dimension	2-99
	SWdefdimstrs	swdefdimstrs	sets the label, unit, and format strings for a given dimension that is used in all fields defined in the swath	2-99
	SWgetdimstrs	swgetdimstrs	gets the label, unit, and format strings for a given dimension	2-73

5.3.2 File Identifiers

As with all HDF-EOS interfaces, file identifiers in the SW interface are 32-bit values, each uniquely identifying one open data file. They are not interchangeable with other file identifiers created with other interfaces.

5.3.3 Swath Identifiers

Before a swath data set is accessed, it is identified by a name which is assigned to it upon its creation. The name is used to obtain a *swath identifier* . After a swath data set has been opened for access, it is uniquely identified by its swath identifier.

5.4 Programming Model

The programming model for accessing a swath data set through the SW interface is as follows:

1. Open the file and initialize the SW interface by obtaining a file id from a file name.
2. Open OR create a swath data set by obtaining a swath id from a swath name.
3. Perform desired operations on the data set.
4. Close the swath data set by disposing of the swath id.
5. Terminate swath access to the file by disposing of the file id.

To access a single swath data set that already exists in an HDF-EOS file, the calling program must contain the following sequence of C calls:

```
file_id = SWopen(filename, access_mode);
sw_id = SWattach(file_id, swath_name);
<Optional operations>
status = SWdetach(sw_id);
status = SWclose(file_id);
```

To access several files at the same time, a calling program must obtain a separate id for each file to be opened. Similarly, to access more than one swath data set, a calling program must obtain a separate swath id for each data set. For example, to open two data sets stored in two files, a program would execute the following series of C function calls:

```
file_id_1 = SWopen(filename_1, access_mode);
sw_id_1 = SWattach(file_id_1, swath_name_1);
file_id_2 = SWopen(filename_2, access_mode);
```

```
sw_id_2 = SWattach(file_id_2, swath_name_2);
<Optional operations>
status = SWdetach(sw_id_1);
status = SWclose(file_id_1);
status = SWdetach(sw_id_2);
status = SWclose(file_id_2);
```

Because each file and swath data set is assigned its own identifier, the order in which files and data sets are accessed is very flexible. However, it is very important that the calling program individually discard each identifier before terminating. Failure to do so can result in empty or, even worse, invalid files being produced.

6. Grid Data

6.1 Introduction

This section will describe the routines available for storing and retrieving HDF-EOS *Grid Data* . A Grid data set is similar to a swath in that it contains a series of data fields of two or more dimensions. The main difference between a Grid and a Swath is in the character of their geolocation information.

As described in Section 4, swaths carry geolocation information as a series of individually located points (tie points or ground control points). Grids, though, carry their geolocation in a much more compact form. A grid merely contains a set of projection equations (or references to them) along with their relevant parameters. Together, these relatively few pieces of information define the location of all points in the grid. The equations and parameters can then be used to compute the latitude and longitude for any point in the grid.

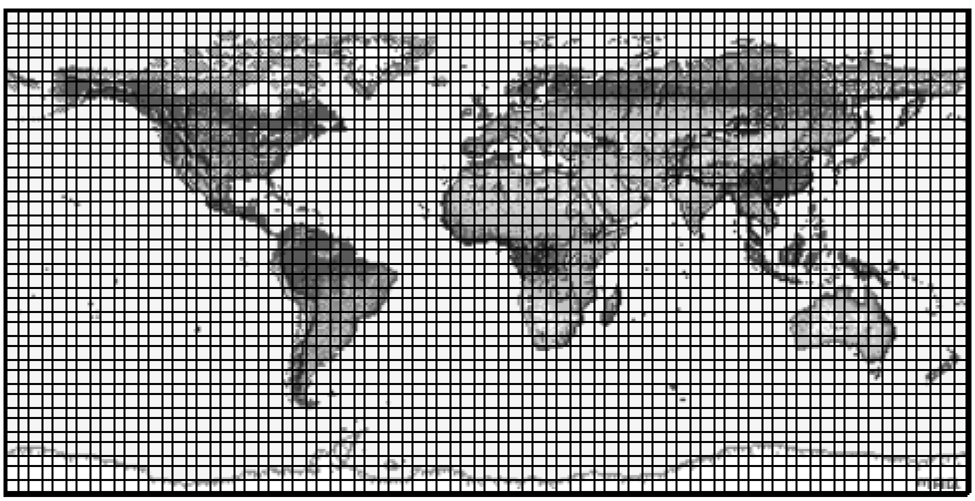


Figure 6-1. A Data Field in a Mercator-Projected Grid

In loose terms, each data field constitutes a map in a given standard projection. Although there may be many independent Grids in a single HDF-EOS file, within each Grid only one projection may be chosen for application to all data fields. Figures 6-1 and 6-2 show how a single data field may look in a Grid using two common projections.

There are three important features of a Grid data set: the data fields, the dimensions, and the projection. Each of these is discussed in detail in the following subsections.

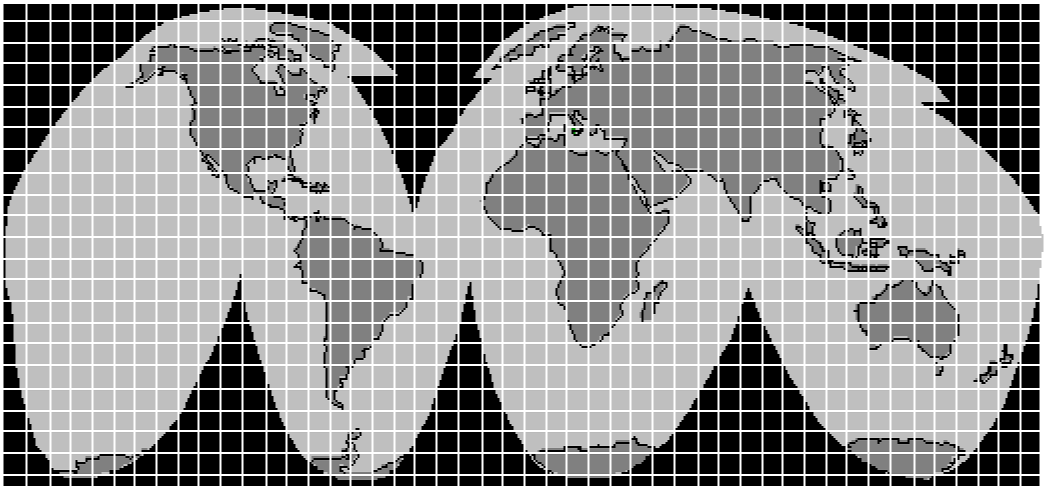


Figure 6-2. A Data Field in an Interrupted Goode's Homolosine-Projected Grid

6.1.1 Data Fields

The data fields are, of course, the most important part of the Grid. Data fields in a Grid data set are rectilinear arrays of two or more dimensions. Most commonly, they are simply two-dimensional rectangular arrays. Generally, each field contains data of similar scientific nature which must share the same data type. The data fields are related to each other by common geolocation. That is, a single set of geolocation information is used for all data fields within one Grid data set.

6.1.2 Dimensions

Dimensions are used to relate data fields to each other and to the geolocation information. To be interpreted properly, each data field must make use of the two predefined dimensions: "XDim" and "YDim". These two dimensions are defined when the grid is created and are used to refer to the X and Y dimensions of the chosen projection (see 5.1.3 below). Although there is a limit of eight dimensions a data field in a Grid data set my have, it is not likely that many fields will need more than three: the predefined dimensions "XDim" and "YDim" and a third dimension for depth or height.

6.1.3 Projections

The projection is really the heart of the Grid. Without the use of a projection, the Grid would not be substantially different from a Swath. The projection provides a convenient way to encode geolocation information as a set of mathematical equations which are capable of transforming Earth coordinates (latitude and longitude) to X-Y coordinates on a sheet of paper.

The choice of a projection to be used for a Grid is a critical decision for a data product designer. There are a large number of projections that have been used throughout history. In fact, some projections date back to ancient Greece. For the purposes of this release of HDF-EOS, however, only 14 families of projections are supported: Geographic, Universal Transverse Mercator, Albers Conical Equal Area, Lambert Conformal, Mercator, Polar Stereographic, Polyconic, Transverse Mercator, Lambert Azimuthal Equal Area, Hotin Obligue Metcator, Space Oblique, Interrupted Goode's Homolosine, Integerized Sinusoidal, and Cylindrical Equal area. These projections coincide with those supported by the SDP Toolkit for ECS Release B.

The producer's choice of a projection should be governed by knowledge of the specific properties of each projection and a thorough understanding of the requirements of the data set's users. Two excellent resources for information on projections and their properties are the USGS Professional Papers cited in Section 2.2 "Related Documents."

This release of HDF-EOS assumes that the data producer will use to create the data the General Coordinate Transformation Package (GCTP), a library of projection software available from the U.S. Geological Survey. This manual will not attempt to explain the use of GCTP. Little documentation accompanies the GCTP source code. For the purposes of this Grid interface, the data are assumed to have already been projected. The Grid interface allows the data producer to specify the exact GCTP parameters used to perform the projection and will provide for basic subsetting of the data fields by latitude/longitude bounding box.

See section below for further details on the usage of the GCTP package.

6.2 Applicability

The Grid data model is intended for data processed at a high level. It is most applicable to data at EOS processing levels 3 and 4.

As an example, the ASTER & MODIS teams on EOS-Terra uses grid structures to store data.

6.3 The Grid Data Interface

The GD interface consists of routines for storing, retrieving, and manipulating data in grid data sets.

6.3.1 GD API Routines

All C routine names in the grid data interface have the prefix "GD" and the equivalent FORTRAN routine names are prefixed by "gd." The GD routines are classified into the following categories:

Access routines initialize and terminate access to the GD interface and grid datasets

- (including opening and closing files).

- *Definition* routines allow the user to set key features of a grid dataset.

- *Basic I/O* routines read and write data and metadata to a grid dataset.

- *Inquiry* routines return information about data contained in a grid data set.

- *Subset* routines allow reading of data from a specified geographic region.

The GD function calls are listed in Table 6-1 and are described in detail in the Software Reference Guide that accompanies this document. The page number column in the following table refers to the Software Reference Guide.

Table 6-1. Summary of the Grid Interface (1 of 2)

Category	Routine Name		Description	Page Nos.
	C	FORTRAN		
Access	GDopen	gdopen	creates a new file or opens an existing one	2-157
	GDcreate	gdcreate	creates a new grid in the file	2-115
	GDattach	gdattach	attaches to a grid	2-109
	GDdetach	gddetach	detaches from grid interface	2-134
	GDclose	gdclose	closes file	2-113
Definition	GDdeforigin	gddeforigin	defines origin of grid pixels	2-124
	GDdefdim	gddefdim	defines dimensions for a grid	2-121
	GDdefproj	gddefproj	defines projection of grid	2-126
	GDdefpixreg	gddefpixreg	defines pixel registration within grid cell	2-125
	GDdeffield	gddeffld	defines data fields to be stored in a grid	2-122
	GDdefcomp	gddefcomp	defines a field compression scheme	2-119
	GDblkSOMoffset	none	This is a special function for SOM MISR data. Write block SOM offset values.	2-111
	GDsettilecomp	none	This routine was added as a fix to a bug in HDF-EOS. The current method of implementation didn't allow the user to have a field with fill values and use tiling and compression. This function allows the user to access all of these features.	2-173
Basic I/O	GDwritefieldmeta	gdwrmeta	writes metadata for field already existing in file	2-178
	GDwritefield	gdwrfld	writes data to a grid field	2-176
	GDreadfield	gdrdfld	reads data from a grid field	2-162
	GDwriteattr	gdwrattr	writes/updates attribute in a grid	2-175
	GDreadattr	gdrdattr	reads attribute from a grid	2-161
	GDsetfillvalue	gdsetfill	sets fill value for the specified field	2-171
	GDgetfillvalue	gdgetfill	retrieves fill value for the specified field	2-144
Inquiry	GDinqdims	gdinqdims	retrieves information about dimensions defined in grid	2-151
	GDinqfields	gdinqdflds	retrieves information about the data fields defined in grid	2-152
	GDinqattrs	gdinqattrs	retrieves number and names of attributes defined	2-150
	GDnentries	gdnentries	returns number of entries and descriptive string buffer size for a specified entity	2-156
	GDgridinfo	gdgridinfo	returns dimensions of grid and X-Y coordinates of corners	2-149
	GDprojinfo	gdprojinfo	returns all GCTP projection information	2-160
	GDdiminfo	gddiminfo	retrieves size of specified dimension	2-135
	GDcompinfo	gdcompinfo	retrieve compression information about a field	2-114
	GDfieldinfo	gdfldinfo	retrieves information about a specific geolocation or data field in the grid	2-138
	GDinqgrid	gdinqgrid	retrieves number and names of grids in file	2-153
	GDattrinfo	gdattrinfo	returns information about grid attributes	2-110
	GDorigininfo	gdorginfo	return information about origin of grid pixels	2-158
	GDpixreginfo	gdpreginfo	return pixel registration information for given grid	2-159
Subset	GDdefboxregion	gddefboxreg	define region of interest by latitude/longitude	2-118
	GDregioninfo	gdreginfo	returns information about a defined region	2-165
	GDextractregion	gdextrreg	read a region of interest from a field	2-137
	GDdeftimeperiod	gddeftmeper	define a time period of interest	2-130

Table 6-1. Summary of the Grid Interface (2 of 2)

Category	Routine Name		Description	Page Nos.
	C	FORTRAN		
Subset	GDdefvrtregion	gddefvrtreg	define a region of interest by vertical field	2-132
	GDgetpixels	gdgetpix	get row/columns for lon/lat pairs	2-145
	GDgetpixvalues	gdgetpixval	get field values for specified pixels	2-147
	GDinterpolate	gdinterpolate	perform bilinear interpolation on a grid field	2-154

	GDdupregion	gddupreg	duplicate a region or time period	2-136
Tiling	GDdeftile	gddeftle	define a tiling scheme	2-128
	GDtileinfo	gdtleinfo	returns information about tiling for a field	2-174
	GDsettilecache	gdsettleche	set tiling cache parameters	2-172
	GDreadtile	gdrdtle	read data from a single tile	2-164
	GDwritetile	gdwrtile	write data to a single tile	2-179
Utility	GDij2ll	Gdij2ll	convert (i,j) coordinates to (lon,lat) for a grid	2-184
	GDll2ij	Gdll2ij	convert (lon,lat) coordinates to (i,j) for a grid	2-187
	GDrs2ll	gdrs2ll	convert (r,s) coordinates to (lon,lat) for EASE grid	2-190
Dimension Scale	GDsetdimscale	gdsetdimscale	sets dimension scale for a given dimension	2-167
	GDdefdimscale	gddefdimscale	sets dimension scale for a given dimension used in all fields defined in the grid	2-167
	GDgetdimscale	gdgetdimscale	gets dimension scale for a given dimension	2-140
	GDsetdimstrs	gdsetdimstrs	sets the label, unit, and format strings for a given dimension	2-169
	GDdefdimstrs	gddefdimstrs	sets the label, unit, and format strings for a given dimension used in all fields defined in the grid	2-169
	GDgetdimstrs	gdgetdimstrs	gets the label, unit, and format strings for a given dimension	2-142

6.3.2 File Identifiers

As with all HDF-EOS interfaces, file identifiers in the GD interface are 32-bit values, each uniquely identifying one open data file. They are not interchangeable with other file identifiers created with other interfaces.

6.3.3 Grid Identifiers

Before a grid data set is accessed, it is identified by a name which is assigned to it upon its creation. The name is used to obtain a *grid identifier* . After a grid data set has been opened for access, it is uniquely identified by its grid identifier.

6.4 Programming Model

The programming model for accessing a grid data set through the GD interface is as follows:

1. Open the file and initialize the GD interface by obtaining a file id from a file name.
2. Open OR create a grid data set by obtaining a grid id from a grid name.
3. Perform desired operations on the data set.
4. Close the grid data set by disposing of the grid id.
5. Terminate grid access to the file by disposing of the file id.

To access a single grid data set that already exists in an HDF-EOS file, the calling program must contain the following sequence of C calls:

```
file_id = GDopen(filename, access_mode);
gd_id = GDattach(file_id, grid_name);
<Optional operations>
status = GDdetach(gd_id);
status = GDclose(file_id);
```

To access several files at the same time, a calling program must obtain a separate id for each file to be opened. Similarly, to access more than one grid data set, a calling program must obtain a separate grid id for each data set. For example, to open two data sets stored in two files, a program would execute the following series of C function calls:

```
file_id_1 = GDopen(filename_1, access_mode);
gd_id_1 = GDattach(file_id_1, grid_name_1);
file_id_2 = GDopen(filename_2, access_mode);
gd_id_2 = GDattach(file_id_2, grid_name_2);
<Optional operations>
status = GDdetach(gd_id_1);
status = GDclose(file_id_1);
status = GDdetach(gd_id_2);
status = GDclose(file_id_2);
```

Because each file and grid data set is assigned its own identifier, the order in which files and data sets are accessed is very flexible. However, it is very important that the calling program individually discard each identifier before terminating. Failure to do so can result in empty or, even worse, invalid files being produced.

6.5 GCTP Usage

The HDF-EOS Grid API uses the U.S. Geological Survey General Cartographic Transformation Package (GCTP) to define and subset grid structures. This section describes codes used by the package.

6.5.1 GCTP Projection Codes

The following GCTP projections are supported for HDFEOS. The projection codes are used in the grid API described in Section 6 below:

GCTP_GEO	(0) Geographic
GCTP_UTM	(1) Universal Transverse Mercator

GCTP_SPCS	(2) State Plane Coordinate System
GCTP_ALBERS	(3) Albers Conical Equal_Area
GCTP_LAMCC	(4) Lambert Conformal Conic
GCTP_MERCAT	(5) Mercator
GCTP_PS	(6) Polar Stereographic
GCTP_POLYC	(7) Polyconic
GCTP_TM	(9) Transverse Mercator

GCTP_LAMAZ	(11) Lambert Azimuthal Equal Area
GCTP_SNSOID	(16) Sinusoidal
GCTP_HOM	(20) Hotine Oblique Mercator
GCTP_SOM	(22) Space Oblique Mercator

GCTP_GOOD	(24) Interrupted Goode Homolosine
GCTP_ISINUS1	(31) Integerized Sinusoidal Projection*
GCTP_ISINUS	(99) Integerized Sinusoidal Projection*

GCTP_CEA	(97)	Cylindrical Equal-Area (for EASE grid with corners in meters)**
GCTP_BCEA	(98)	Cylindrical Equal-Area (for EASE grid with grid Corners in packed degrees, DMS)**

* The Integerized Sinusoidal Projection was not part of the original GCTP package. It has been added by ECS. See *Level-3 SeaWiFS Data Products: Spatial and Temporal Binning Algorithms* . Additional references are provided in Section 2.

** The Cylindrical Equal-Area Projection was not part of the original GCTP package. It has been added by ECS. See Notes for section 6.5.4.

In the new GCTP package the Integerized Sinusoidal Projection is included as the 31st projection. The Code 31 was added to HDFEOS for users who wish to use 31 instead of 99 for Integerized Sinusoidal Projection.

Note that other projections supported by GCTP will be adapted for next HDF-EOS Version as new user requirements are surfaced. For further details on the GCTP projection package, please refer to Section 6.3.5 and Appendix G of the SDP Toolkit Users Guide for the EED Contract, December, 2017, (333-EED2-001, Revision 01).

6.5.2 UTM Zone Codes

The Universal Transverse Mercator (UTM) Coordinate System uses zone codes instead of specific projection parameters. The table that follows lists UTM zone codes as used by GCTP Projection Transformation Package. C.M. is Central Meridian

Zone	C.M.	Range	Zone	C.M.	Range
01	177W	180W-174W	31	003E	000E-006E
02	171W	174W-168W	32	009E	006E-012E
03	165W	168W-162W	33	015E	012E-018E
04	159W	162W-156W	34	021E	018E-024E
05	153W	156W-150W	35	027E	024E-030E
06	147W	150W-144W	36	033E	030E-036E
07	141W	144W-138W	37	039E	036E-042E
08	135W	138W-132W	38	045E	042E-048E
09	129W	132W-126W	39	051E	048E-054E
10	123W	126W-120W	40	057E	054E-060E
11	117W	120W-114W	41	063E	060E-066E
12	111W	114W-108W	42	069E	066E-072E
Zone	C.M.	Range	Zone	C.M.	Range
13	105W	108W-102W	43	075E	072E-078E
14	099W	102W-096W	44	081E	078E-084E
15	093W	096W-090W	45	087E	084E-090E
16	087W	090W-084W	46	093E	090E-096E
17	081W	084W-078W	47	099E	096E-102E
18	075W	078W-072W	48	105E	102E-108E
19	069W	072W-066W	49	111E	108E-114E
20	063W	066W-060W	50	117E	114E-120E
21	057W	060W-054W	51	123E	120E-126E
22	051W	054W-048W	52	129E	126E-132E
23	045W	048W-042W	53	135E	132E-138E
24	039W	042W-036W	54	141E	138E-144E
25	033W	036W-030W	55	147E	144E-150E
26	027W	030W-024W	56	153E	150E-156E
27	021W	024W-018W	57	159E	156E-162E
28	015W	018W-012W	58	165E	162E-168E
29	009W	012W-006W	59	171E	168E-174E
30	003W	006W-000E	60	177E	174E-180W

6.5.3 GCTP Spheroid Codes

Clarke 1866 (default) (0)

Clarke 1880 (1)

Bessel (2)

International 1967 (3)

International 1909 (4)

WGS 72 (5)

Everest (6)

WGS 66 (7)

GRS 1980 (8)

Airy (9)

Modified Airy (10)

Modified Everest (11)

WGS 84 (12)

Southeast Asia (13)

Australian National (14)

Krassovsky (15)

Hough (16)

Mercury 1960 (17)

Modified Mercury 1968 (18)

Sphereof Radius 6370997m (19)

Sphereof Radius 6371228m (20)

Sphereof Radius 6371007.181m (21)

6.5.4 Projection Parameters

Table 6-2. Projection Transformation Package Projection Parameters (1 of 2)

	Array Element							
Code & Projection Id	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
0 Geographic								
1 U T M	Lon/Z	Lat/Z						
2 PGSd_SPCS			Spheroid	Zone				
3 Albers Conical Equal_Area	Smajor	Sminor	STDPR1	STDPR2	CentMer	OriginLat	FE	FN
4 Lambert Conformal C	Smajor	Sminor	STDPR1	STDPR2	CentMer	OriginLat	FE	FN
5 Mercator	Smajor	Sminor			CentMer	TrueScale	FE	FN
6 Polar Stereographic	Smajor	Sminor			LongPol	TrueScale	FE	FN
7 Polyconic	Smajor	Sminor			CentMer	OriginLat	FE	FN
9 Transverse Mercator	Smajor	Sminor	Factor		CentMer	OriginLat	FE	FN
11 Lambert Azimuthal**	Smajor	Sminor			CentLon	CenterLat	FE	FN
Lambert Azimuthal	Sphere				CentLon	CenterLat	FE	FN
16 PGSd_SNSOID**	Smajor	Sminor			CentMer			
PGSd_SNSOID	Sphere				CentMer			
20 Hotin Oblique Merc A	Smajor	Sminor	Factor			OriginLat	FE	FN
20 Hotin Oblique Merc B	Smajor	Sminor	Factor	AziAng	AzmthPt	OriginLat	FE	FN
22 Space Oblique Merc A	Smajor	Sminor		IncAng	AscLong		FE	FN
22 Space Oblique Merc B	Smajor	Sminor	Satnum	Path			FE	FN
24 Interrupted Goode	Sphere							
97 CEA utilized by EASE grid (see Notes)	Smajor	Sminor			CentMer	TrueScale	FE	FN
98 BCEA utilized by EASE grid (see Notes)	Smajor	Sminor			CentMer	TrueScale	FE	FN

** Lambert Azimuthal and Sinusoidal supports both spherical and WGS84 ellipsoidal Earth model

Table 6-3. Projection Transformation Package Projection Parameters Elements

	Array Element				
Code & Projection Id	**9**	**10**	**11**	**12**	**13**
0 Geographic					
1 U T M					
2 PGSd_SPCS					
3 Albers Conical Equal_Area					
4 Lambert Conformal C					
5 Mercator					
6 Polar Stereographic					
7 Polyconic					
9 Transverse Mercator					
11 Lambert Azimuthal					
Lambert Azimuthal					
16 PGSd_SNSOID					
PGSd_SNSOID					
20 Hotin Oblique Merc A	Long1	Lat1	Long2	Lat2	zero
20 Hotin Oblique Merc B					one
22 Space Oblique Merc A	PSRev	SRat	PFlag	HDF-EOS Para	zero
22 Space Oblique Merc B				HDF-EOS Para	one
24 Interrupted Goode					
31 & 99 Integerized Sinusoidal	NZone		RFlag		
97 CEA utilized by EASE grid (see Notes)					
98 BCEA utilized by EASE grid (see Notes)					

Where,

Lon/Z	Longitude of any point in the UTM zone or zero. If zero, a zone code must be specified.
Lat/Z	Latitude of any point in the UTM zone or zero. If zero, a zone code must be specified.

Smajor	Semi-major axis of ellipsoid. If zero, Clarke 1866 in meters is assumed. It is recommended that explicit value, rather than zero, is used for Smajor.

Eccentricity squared of the ellipsoid if less than one, if zero, a spherical form is assumed, or if greater than one, the semi-minor axis of ellipsoid.

It should be noted that a negative sphere code should be used in order to have user specified Smajor and Sminor be accepted by GCTP, otherwise

def
ault
elli
psoi
d
Sm
ajor
and
Smi
nor
will
be
use
Sminor d.

Sphere	Radius of reference sphere. If zero, 6370997 meters is used. It is recommended that explicit value, rather than zero, is used for Sphere.

STDPR1	Latitude of the first standard parallel
STDPR2	Latitude of the second standard parallel

CentMer	Longitude of the central meridian
OriginLat	Latitude of the projection origin

FE	False easting in the same units as the semi-major axis
FN	False northing in the same units as the semi-major axis
TrueScale	Latitude of true scale
LongPol	Longitude down below pole of map

Factor	Scale factor at central meridian (Transverse Mercator) or center of projection (Hotine Oblique Mercator)
CentLon	Longitude of center of projection
CenterLat	Latitude of center of projection

Long1	Long itude o f first point o n cente r line (Hoti n e Obli qu e Merc ator, form at A)
Long2	Long itude o f seco n d point o n cente r line (Hoti n e Obli qu e Merc ator, frmt A)

Lat1	Latitude of first point on center line (Hotine Oblique Mercator, format A)
Lat2	Latitude of second point on center line (Hotine Oblique Mercator, format A)

AziAng	Azimuth angle east of north of center line (Hotine Oblique Mercator, frmt B)

Longitude of point on central meridian where azimuth occurs (Hotine Oblique Mercator, fo

AzmthPt	rmat B)
IncAng	Inclination of orbit at ascending node, counter-clockwise from equator (SOM, format A)

AscLong	Longitude of ascending orbit at equator (SOM, format A)
PSRev	Period of satellite revolution in minutes (SOM, format A)

SRat	Satellite ratio to specify the start and end point of x,y values on earth surface (SOM, format A -- for Landsat use 0.5201613)
PFlag	End of path flag for Landsat: 0 = start of path, 1 = end of path (SOM, frmt A)

Satnum	Landsat Satellite Number (SOM, format B)
	Landsat Path Number (Use WRS-1 for Landsat 1, 2 and 3 and WRS-2 for Landsat 4 and 5.)
Path	(SOM, format B)

Nzone	Number of equally spaced latitudinal zones (rows); must be two or larger and even

Right justify columns flag is used to indicate what to do in zones with an odd number of columns. If it has a value of 0 or 1, it indicates the extra column is on the right (zero) left (one) of the projection Y-axis. If the flag is set to 2 (two),

Rflag	the number of columns are calculated so there are always an even number of columns in each zone.

Notes:

- Array elements 14 and 15 are set to zero.
- All array elements with blank fields are set to zero.

All angles (latitudes, longitudes, azimuths, etc.) are entered in packed degrees/ minutes/ seconds (DDDMMMSSS.SS) format.

The following notes apply to the Space Oblique Mercator A projection:

- A portion of Landsat rows 1 and 2 may also be seen as parts of rows 246 or 247. To place these locations at rows 246 or 247, set the end of path flag (parameter 11) to 1--end of path. This flag defaults to zero.
- When Landsat-1,2,3 orbits are being used, use the following values for the specified parameters:

	0990
	0503
Parameter 4	1.2

Parameter 5	128.87 degrees - (360/251 * path number) in packed DMS format
Parameter 9	103.2669323
Parameter 10	0.5201613

- When Landsat-4,5 orbits are being used, use the following values for the specified parameters:

Parameter 4	098012000.0
Parameter 5	129.30 degrees - (360/233 * path number) in packed DMS format
Parameter 9	98.884119
Parameter 10	0.5201613

The following notes apply for **BCEA and CEA projections,** and **EASE grid**:

HDFEOS 2.7 and 2.8: Behrmann Cylindrical Equal-Area (BCEA) projection was used for 25 km global EASE grid. For this projection, the Earth radius is set to 6371228.0m and latitude of true scale is 30 degrees. For 25 km, global EASE grid the following apply:

Grid Dimensions:

Width 1383

Height 586

Map Origin:

Column (r0) 691.0

Row (S0) 292.5

Latitude 0.0

Longitude 0.0

Grid Extent:

Minimum Latitude 86.72 S

Maximum Latitude 86.72 N

Minimum Longitude 180.00 W

Maximum Longitude 180.00E

Actual grid cell size 25.067525km

Grid coordinates (r,s) start in the upper left corner at cell (0.0), with r increasing to the right and s increasing downward.

HDFEOS 2.8.1 and later: Although the projection code and name (tag) kept the same, BCEA projection was generalized to accept Latitude of True Scales other than 30 degrees, Central Meridian other than zero, and ellipsoid earth model besides the spherical one with user supplied radius. This generalization along with the removal of hard coded grid parameters will allow users not only subsetting, but also creating other grids besides the 25-km global EASE grid and having freedom to use different appropriate projection parameters. With the current version one can create the above mentioned 25-km global EASE grid of previous versions using:

Grid Dimensions:

Width 1383

Height 586

Grid Extent:

UpLeft
Latitud
e 86.72

LowRi
ght
Latitud
e
-86.72

UpLeft
Longit
ude
-180.00

LowRight Longitude 180.00

Projection Parameters:

1) 6371.2280/25.067525 = 254.16263

2) 6371.2280/25.067525 = 254.16263

5) 0.0

6) 30000000.0

7) 691.0

8) –292.5

Also one may create **12.5 km global EASE grid** using:

Grid Dimensions:

Width 2766

Height 1171

Grid Extent:

UpLeft
Latitud
e 85.95

LowRi
ght
Latitud
e
–85.95

UpLeft
Longit
ude
–179.93

LowRight Longitude 180.07

Projection Parameters:

1) 6371.2280/(25.067525/2) = 508.325253

2) 6371.2280/(25.067525/2) = 508.325253

5) 0.0

6) 30000000.0

7) 1382.0

8) –585.0

Any other grids (normalized pixel or not) with generalized BCEA projection can be created using appropriate grid corners, dimension sizes, and projection parameters. Please note that like other projections Semi-major and Semi-minor axes will default to Clarke 1866 values (in meters) if they are set to zero.

HDFEOS 2.10 and later: A new projection CEA (97) was added to GCTP. This projection is the same as the generalized BCEA, except that the EASE grid produced will have its corners in meters rather than packed degrees, which is the case with EASE grid produced by BCEA.

HDFEOS 2.19 and later : The Lambert Azimuthal Equal area projection was generalized to support WGS84 ellipsoidal Earth model in addition to the spherical model that was supported before. This generalization was needed to support EASE GRID 2.0 used for SMAP products.

HDFEOS 2.20 and later : The Sinusoidal projection was generalized to support WGS84 ellipsoidal Earth model in addition to the spherical model that was supported before. This generalization was needed to support EASE GRID 2.0 used for SMAP products and for Datum/Ellipsoid conversions in HDF-EOS to Geotiff conversion Tool.

This page intentionally left blank.

Appendix A. Installation and Maintenance

A.1 Installation Procedures

A.1.1 Preliminary Step

Before installing HDFEOS, you must already have installed THG HDF, Version 4.2.13 or 4.2.15 on your host. Please see the SDP Toolkit Users Guide for the EED-2 Contract, Section 5 for instructions on installing both the Toolkit and HDF. See also: https://support.hdfgroup.org/ftp/HDF/releases/ for instructions on how to access HDF libraries.

A.1.2 Unpacking the Distribution File

1) Select a location for the HDFEOS directory tree. Installing HDFEOS alone requires a disk partition with at least 45 Mb of free space.

2) Copy the file hdf-eos2-3.0-src.tar.gz to the target directory by typing the command:

```
cp hdf-eos 2-3.0-src.tar.gz <target-dir>
```

where <target-dir> is the full pathname of your target directory.

3) Set your default directory to the target directory by typing the command:

c d <target-dir>

4) Uncompress this file and extract the contents by typing the command:

tar -zxvf hdf-eos2-3.0-src.tar.gz

This will create a subdirectory of the current directory called 'hdfeos2-3.0'. This is the top-level HDFEOS directory, which contains the full HDFEOS directory structure.

A.1.3 Starting the Installation Procedure

HDF-EOS is built and installed using autotools as outlined in section A.1.3.1.

A.1.3.1 Installation Using Installation Scripts

<u>Building/Installing HDF-EOS2 using Autoconf/Automake</u>

The HDF-EOS2 library requires HDF4 library version 4.2.13 or later. You can download HDF4

from the HDF Group:

https://confluence.hdfgroup.org/display/support/Download+HDF4

or, depending upon your system, you may be able to install it from a package. The HDF4 library may depend upon other libraries, such as szip, jpeg, and z (compress). The HDF-EOS2 configure script will do its best to determine these dependencies and find the appropriate libraries, but it may occasionally need help. Please refer to the ‘Troubleshooting’ section in the README document in the ‘docs’ directory in the HDF-EOS source directory tree.

After downloading and unpacking the tar file, 'cd' into the hdfeos directory and type

```
./configure
```

This will run a series of tests to determine the necessary library dependencies, and configure the makefiles accordingly. The configure script will output a summary of the configuration options at completion. These can be useful for troubleshooting.

By default, the HDF-EOS2 libraries and include headers will be installed in /usr/local. This can be changed by using the --prefix option when running configure, e.g.

```
./configure --prefix=/custom/install/location
```

A number of other options may be provided to the configure script. You can get details of these options using the command:

```
./configure --help
```

To build HDF-EOS2, simply type

```
make
```

from the hdfeos directory.

The HDF-EOS2 library comes with some tests. You can run these using the command

```
make check
```

from the hdfeos directory. These tests are primarily for internal testing and are not part of the official delivery. However, they may provide useful examples. USE AT YOUR OWN RISK.

To install HDF-EOS2, simply type

```
make install
```

from the hdfeos directory. You need to make sure you have the necessary permissions to install the files in the chosen location.

<u>Build HDF-EOS2 using Visual Studio</u>

For building on Windows, the HDF-EOS2 comes with a Visual Studio 2019 project that can be used to build native Windows versions of the libraries. This combines both the GCTP and HDF-EOS2 code into a single library. To build on Windows, after starting up Visual Studio load the HDF-EOS2 project file in the vs2019/HDF-EOS2 directory. Then select Build -> Build HDF-EOS2 from the menu, or type <ctrl> B.

Fortran API

The HDF-EOS2 library can build a FORTRAN API to provide access to HDF-EOS2 functions from FORTRAN programs. By default, the FORTRAN API is disabled. To enable the FORTRAN API on Autoconf systems, use the

    --enable-fortran

option. The configure script will attempt to determine the appropriate API naming convention used by the fortran compiler automatically. You can still build the FORTRAN API on Autotools systems even if you do not have access to a FORTRAN compiler. To do this, do not use the --enable-fortran option, but instead use the options listed below for enabling the API on windows. e.g.

./configure CPPFLAGS="-DFORTRAN_API -DFORTRAN_API_UPPERCASE -DFORTRAN_API_SUFFIX=_"

For building the FORTRAN API on Windows using Visual Studio, you will need to configure some additional compiler options. These are:

FORTRAN_API Just add this as a preprocessor definition (it does not need a specific value)

FORTRAN_API_UPPERCASE Add this as a preprocessor definition if your FORTRAN compiler uses upper-case for external functions names (e.g. ifort)

FORTRAN_API_LOWERCASE Add this as a preprocessor definition if your FORTRAN compiler uses lower-case for external functions names.

FORTRAN_API_PREFIX Add this as a preprocessor definition if your FORTRAN compiler adds a prefix of any sort to external function names. The value of this definition should be the character prefix (e. g. _)

FORTRAN_API_SUFFIX Add this as a preprocessor definition if your FORTRAN compiler adds a suffix of any sort to external function names. The value of this definition should be the character suffix (e. g. _)

The HDF-EOS2 Visual Studio project file is preconfigure with FORTRAN_API and FORTRAN_API_UPPERCASE, to suit the Intel ifort compiler. Note that a FORTRAN compiler is not actually needed in order to generate the FORTRAN API.

A.1.3.2

For issues with configuring or building HDF-EOS, please refer to the 'Troubleshooting' section in the README document in the 'docs' directory in the HDF-EOS source directory tree.

A.2 User Feedback Mechanism

The mechanism for handling user feedback, documentation and software discrepancies, and bug reports follows:

1) The following accounts at the ECS Riverdale facility have been set up for user response:

sdps-support@earthdata.nasa.gov

2) Users will e–mail problem reports and comments to the above account. Responses will be prioritized

3) In order to help expedite responses, we request the following information be supplied with problem re

Name:

Date:

EOS Affiliation (DAAC, Instrument, Earth Science Data and Information System (ESDIS), etc.):

Phone No.:

Development Environment:

Computing Platform:

Operating System:

Compiler and Compiler Flags:

Tool Name:

Problem Description:

(Please include exact inputs to and outputs from the toolkit call, including error code returned by the function, plus exact error messa ge returned where applicable.)

Suggested Resolution (include code fixes or workarounds if applicable):

4) In addition to the above email address, a list server has also been set up for users. The address is: SDI

Abbreviations and Acronyms

AI&T	algorithm integration & test
AIRS	Atmospheric Infrared Sounder
API	application program interface
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer
CCSDS	Consultative Committee on Space Data Systems

CDRL	Contract Data Requirements List
CDS	CCSDS day segmented time code
CERES	Clouds and Earth Radiant Energy System
CM	configuration management
COTS	commercial off–the–shelf software
CUC	constant and unit conversions
CUC	CCSDS unsegmented time code

DAAC	distributed active archive center
DBMS	database management system
DCE	distributed computing environment
DCW	Digital Chart of the World
DEM	digital elevation model
DTM	digital terrain model
ECR	Earth centered rotating
ECS	EOSDIS Core System
EDC	Earth Resources Observation Systems (EROS) Data Center

EDHS	ECS Data Handling System
EDOS	EOSDIS Data and Operations System
EOS	Earth Observing System
EOSAM	EOS AM Project (morning spacecraft series)
EOSDIS	Earth Observing System Data and Information System

EOSPM	EOS PM Project (afternoon spacecraft series)
ESDIS	Earth Science Data and Information System (GSFC Code 505)
FDF	flight dynamics facility
FOV	field of view
ftp	file transfer protocol
GCT	geo–coordinate transformation
GCTP	general cartographic transformation package
GD	grid

GPS	Global Positioning System
GSFC	Goddard Space Flight Center
HDF	hierarchical data format
HEG	HDF-EOS to GeoTIFF Conversion Tool
HITC	Hughes Information Technology Corporation
http	hypertext transport protocol
I&T	integration & test
ICD	interface control document
IDL	interactive data language
IP	Internet protocol

IWG	Investigator Working Group
JPL	Jet Propulsion Laboratory
LaRC	Langley Research Center
LIS	Lightening Imaging Sensor
M&O	maintenance and operations
MCF	metadata configuration file
MET	metadata
MODIS	Moderate–Resolution Imaging Spectroradiometer
MSFC	Marshall Space Flight Center

NASA	National Aeronautics and Space Administration
NCSA	National Center for Supercomputer Applications
netCDF	network common data format
NGDC	National Geophysical Data Center
NMC	National Meteorological Center (NOAA)
ODL	object description language
PC	process control

PCF	process control file
PDPS	planning & data production system
PGE	product generation executive (formerly product generation executable)
POSIX	Portable Operating System Interface for Computer Environments
PT	point
QA	quality assurance
RDBMS	relational database management system

RPC	remote procedure call
RRDB	recommended requirements database
SCF	Science Computing Facility
SDP	science data production
SDPF	science data processing facility
SGI	Silicon Graphics Incorporated
SMF	status message file
SMAP	Soil Moisture Active Passive
SMP	Symmetric Multi–Processing
SOM	Space Oblique Mercator

SPSO	Science Processing Support Office
SSM/I	Special Sensor for Microwave /Imaging
SW	swath
TAI	International Atomic Time
TBD	to be determined
TDRSS	Tracking and Data Relay Satellite System
THG	The HDF Group
TRMM	Tropical Rainfall Measuring Mission (joint US – Japan)

UARS	Upper Atmosphere Research Satellite
UCAR	University Corporation for Atmospheric Research
URL	universal reference locator
USNO	United States Naval Observatory
UT	universal time
UTC	Coordinated Universal Time
UTCF	universal time correlation factor
UTM	universal transverse mercator
VPF	vector product format

	World
	Wide
WWW	Web

---

[1] “Co-latitude” may be substituted for “Latitude.”