HDF Configuration Record Definition

Version 2.0

Technical Document

Revised in October 1999

Prepared Under Subcontract #301684

Hughes Information Technology Systems

(U. of Illinois Ref. No. 96-NASA SBC-F-0162)

Albert Cheng

Michael Folk

Paul Harten

Raymond Lu

William Whitehouse

National Center for Supercomputing Applications

University of Illinois at Urbana-Champaign

PART I. INTRODUCTION

1. Introduction

1.1 Purpose

This document defines the standard of the HDF Configuration Record (HCR). It describes the syntax and definitions of the HCR script language. It also provides examples to illustrate the usage of the HCR. Readers would be able to compose an HCR to create HDF-EOS or HDF files.

The readers are assumed to have a good knowledge of the concepts of the HDF-EOS objects [EOS96-2, EOS96-3, EOS96-4, EOS96-5] and the HDF-EOS library [EOS96-1, EOS97-1, EOS97-2] and HDF objects [HDF].

1.2 The Need for a Configuration Record

End-users may wish to create an HDF-EOS or HDF file using the HDF-EOS or HDF library interface which supports grid, swath, and point objects in HDF-EOS or objects in HDF. In order to simplify this task, the HDF Configuration Record (HCR) provides a high-level description of the configuration of objects in an HDF-EOS or HDF file and the conceptual relationships among them. Additional software tools can then be used to automatically create a *skeleton* file based on the contents of the HCR.

1.3 Organization of this Document

There are totally four parts in this paper. Part I is introduction. Part II talks about the definition of HDF-EOS objects. Part III talks about the definition of HDF objects. Part IV are related documents.

In the sequence of chapters, Chapter 2 gives an overview of the HCR file and the Object Description Language (ODL). Chapter 3 describes the definition of the Swath object. Chapter 4 describes the definition of the Grid object. Chapter 5 describes the definition of the Point object. Chapter 6 shows an example HCR file for HDF-EOS. Chapter 7 talks about the definition of SDS object. Chapter 8 talks about GR object. Chapter 9 describes Vdata object. Chapter 10 talks about Vgroup object. Chapter 11 talks about Palette object. Chapter 12 handles Annotation object. While Chapter 13 are related documents.

1.4 Version History

1.4.1 Version 2.0

Six HDF4 Objects(SDS, GR, Pallete, Vdata, Vgroup and Annotation) are added to the tools of *hcr2hdf* and *hdf2hcr*. They may also be added to *hcrhdfdiff* and *hcr_edit* later.

1.4.2 Version 1.2

Added Swath Field Storage Definition, Grid Field Storage Definition, three kinds of storage definitions--Merge, Compression, and Tile.

1.4.3 Version 1.1

Initial version.

2. Overview of the HCR

An HCR is a block of ASCII text composed of Object Description Language (ODL [PDS95-12]) style statements describing HDF-EOS objects contained in a corresponding HDF-EOS file.

2.1 Overview of ODL statements

ODL statements are in the form of

parameter = value

Where *parameter* is an identifier or a keyword and *value* is any elementary value such as an integer, a real number, a character string, a list of values, a set of values or another identifier. E.g.,

```
COORDINATES = (45.0, -87.75)
DATATYPE = FLOAT32
OBJECT = Swath
```

HCR uses only a subset of ODL statements, namely, the *Assignment Statement,* the *Object Statement* and the *End Statement.* (The *Group Statement* and *Pointer Statement* are not used by HCR.)

2.1.1 Assignment Statement

The Assignment Statement is the most common type of statement and is used to specify the value for an attribute of an object. It has the form as *attribute = Value.*

2.1.2 Object Statement

The *Object Statement* has the keyword **OBJECT** as its parameter. Its value should be an identifier. A matching *End_Object Statement* that has **END_OBJECT** as its parameter, should contain the same identifier. (ODL does not require a value for the End_Object Statement but HCR recommends its use.) All statements between these two matching Object and End_Object statements are grouped as one concept.

2.1.3 End Statement

The *End Statement* consists of only the keyword **END**. It signifies the end of ODL statements input.

2.1.4 Comment

Comments are enclosed in a pair of '/*' and '*/' similar to the C language. But ODL also ignores any data on the same line after a comment. Comments are not allowed to be embedded in other statements. It is best to keep all comments on their own lines.

2.1.5 Continuation

An ODL statement may run across multiple input lines with some restrictions like not breaking a keyword. The following two statements are equivalent.

```
      COORDINATES = (45.0, -87.75)
      COORDINATES = (45.0,
                     -87.75)
```

2.1.6 Case sensitivity

ODL is case insensitive in that all statements are interpreted as if they are coded in upper case. One exception is the quoted text strings, which are characters enclosed in a pair of quotation marks ("). Note that characters enclosed in a pair of apostrophes is called a Symbol String and is case insensitive. For examples, the following first three values are all equivalent but the last one is different.

```
       ShortRange
      'ShortRange'
      'SHORTRANGE'
      "ShortRange"
```

2.1.7 Example

The following is a simple example of ODL statements.

```
/* Project XYZ */
/* First version defined on June 10th, 1996 */
OBJECT = SWATH
      NAME = SCAN1
      OBJECT = Dimension
            NAME = GeoTrack
            Size = 1200
      END_OBJECT = Dimension
      OBJECT = Dimension
            NAME = GeoCrossTrack
            Size = 205
      END_OBJECT = Dimension
      OBJECT = Dimension
            NAME = DataX
            Size = 2410
      END_OBJECT = Dimension
END_OBJECT = SWATH
END
```

PART II. HCR FOR HDF-EOS

3. Swath Object

3.1 Swath Object Definition

```
<Swath Object> ::=
     OBJECT = Swath
          NAME = <SwathName>
          <Swath Dimension Definition>
          <Swath Geofield Definition>
          <Swath Datafield Definition>
          [<Swath Dimension Mapping Definition>]
     END_OBJECT = Swath
```

Additional requirements:

<SwathName> can be any legal quoted name but must be unique among all HCR object names.

3.2 Swath Dimension Definition

```
<Swath Dimension Definition> ::=
     <Dimension Definition>*

<Dimension Definition> ::=
     OBJECT = Dimension
          NAME = <DimensionName>
          Size = <DimensionSize>
     END_OBJECT = Dimension
```

Additional requirements:

<DimensionName> can be any legal quoted name but must be unique within <Swath Object>.

<DimensionSize> can be any non-negative integer representing the size of the dimension defined. A value zero or the keyword SD_UNLIMITED represents an unlimited dimension as defined in HDF.

3.3 Swath Datafield Definition

```
<Swath Datafield Definition> ::=
     <Datafield Definition>*

<Datafield Definition> ::=
```

```
OBJECT = DataField
     NAME = <DatafieldName>
     DataType = <DataType>
     DimList = (<DimName1>, <DimName2>, ...)
     [<Swath Field Storage Definition>]
END_OBJECT = DataField
```

Additional requirements:

<DatafieldName> can be any legal quoted name but must be unique within <Swath Object>.

<DataType> can be any legal datatype as defined in HDF.

<DimName> must be the name of a dimension defined in <Swath Dimension Definition>.

3.3.1 Swath Field Storage Definition

The Swath Field Storage Definition contains one of Field Merge Definition or Field Compression Definition. See their definitions under Grid Field Storage Definition.

```
<Swath Field Storage Definition> ::=
     <Field Merge Definition> |
     <Field Compression Definition>
```

3.4 Swath Geofield Definition

```
<Swath Geofield Definition> ::=
     <Geofield Definition>*

<Geofield Definition> ::=
     OBJECT = GeoField
          NAME = <GeofieldName>
          DataType = <DataType>
          DimList = (<DimName>, <DimName>, ...)
          [<Swath Field Storage Definition>]
     END_OBJECT = GeoField
```

Additional requirements:

<GeofieldName> can be any legal quoted name but must be unique within <Swath Object>.

<DataType> can be any legal datatype as defined in HDF.

<DimName> must be the name of a dimension defined in <Swath Dimension Definition>.

<Swath Field Storage Definition> is described under Swath Datafield Definition.

3.5 Reserved Field Names

HDF-EOS library version 1 reserves the following field names. If the fields are used in a swath definition, they must be defined with the following datatypes.

Keyword	Datatype	Comments
Latitude	FLOAT32 or FLOAT64	floating point latitude
Longitude	FLOAT32 or FLOAT64	floating point longitude
CoLatitude	FLOAT32 or FLOAT64	floating point colatitude
Time	FLOAT32 or FLOAT64	TAI93 time in float

3.6 Swath Dimension Mapping Definition

```
<Swath Dimension Mapping Definition> ::=
          { <Regular Dimension Mapping Definition> |
            <Index Dimension Mapping Definition> }*
```

The Mapping Definitions can be in any order.

3.6.1 Regular Dimension Mapping Definition

```
<Regular Dimension Mapping Definition> ::=
     OBJECT = DimensionMap
          GeoDimension = <GeoDimName>
          DataDimension = <DataDimName>
          Offset = <Offset>
          Increment = <Increment>
     END_OBJECT = DimensionMap
```

Additional requirements:

<GeoDimName> must be a Dimension name on which a Geofield is defined.

<DataDimName> must be a Dimension name on which a Datafield is defined.

<Increment> can be any non-zero integer value.

<Offset> can be any positive integer value.

3.6.2 Implicit Mapping Name

HDF-EOS library implicitly defines the name of a regular dimension map as GeoDimension and DataDimension joined by a slash.

3.6.3 Index Dimension Mapping Definition

The index dimension mapping is not defined yet and is not supported by this version of HCR.

3.7 Swath Object Example

```
OBJECT = Swath                        /* Defining a swath object */
      Name = "Swath 1"
      OBJECT = Dimension              /* Dimension definitions */
            Name = "GeoTrack"
            Size = 20
      END_OBJECT = Dimension
      OBJECT = Dimension
            Name = "GeoXtrack"
            Size = 10
      END_OBJECT = Dimension
      OBJECT = Dimension
            Name = "Res2tr"
            Size = 40
      END_OBJECT = Dimension
      OBJECT = Dimension
            Name = "Res2xtr"
            Size = 20
      END_OBJECT = Dimension
      OBJECT = Dimension
            Name = "Bands"
            Size = 15
      END_OBJECT = Dimension
      OBJECT = Dimension
            Name = "IndxTrack"
            Size = 12
      END_OBJECT = Dimension
      OBJECT = Dimension
            Name = "Unlim"
            Size = 0
      END_OBJECT = Dimension
      OBJECT = DimensionMap           /* Dimension mapping definitions */
            GeoDimension = "GeoTrack"
            DataDimension = "Res2tr"
            Offset = 0
            Increment = 2
      END_OBJECT = DimensionMap
      OBJECT = DimensionMap
            GeoDimension = "GeoXtrack"
            DataDimension = "Res2xtr"
            Offset = 1
            Increment = 2
      END_OBJECT = DimensionMap
      OBJECT = GeoField               /* Geofield Definitions */
            Name = "Time"
            DataType = DFNT_FLOAT64
            DimList = ("GeoTrack")
      END_OBJECT = GeoField
      OBJECT = GeoField
            Name = "Longitude"
            DataType = DFNT_FLOAT32
            DimList = ("GeoTrack","GeoXtrack")
      END_OBJECT = GeoField
      OBJECT = GeoField
            Name = "Latitude"
            DataType = DFNT_FLOAT32
            DimList = ("GeoTrack","GeoXtrack")
      END_OBJECT = GeoField
      OBJECT = DataField              /* Datafield Definitions */
            Name = "Density"
            DataType = DFNT_FLOAT32
            DimList = ("GeoTrack")
      END_OBJECT = DataField
      OBJECT = DataField
```

```
            Name = "Temperature"
            DataType = DFNT_FLOAT32
            DimList = ("GeoTrack","GeoXtrack")
            Merge = HDFE_AUTOMERGE
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "DewPoint"
            DataType = DFNT_FLOAT32
            DimList = ("GeoTrack","GeoXtrack")
            Merge = HDFE_AUTOMERGE
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Pressure"
            DataType = DFNT_FLOAT64
            DimList = ("Res2tr","Res2xtr")
            CompressionType = HDFE_COMP_DEFLATE
            CompressionParameters = (9)
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Spectra"
            DataType = DFNT_FLOAT64
            DimList = ("Bands","Res2tr","Res2xtr")
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Count"
            DataType = DFNT_INT16
            DimList = ("Unlim")
      END_OBJECT = DataField
END_OBJECT = Swath
END
```

4. Grid Object

4.1 Grid Object Definition

```
<Grid Object> ::=
     OBJECT = Grid
          NAME = <GridName>
          <Grid Parameters Definition>
          <Grid Dimension Definition>
          <Grid Datafield Definition>
     END_OBJECT = Grid
```

Additional requirements:
<GridName> can be any legal quoted name but must be unique among all HCR Object names.

4.2 Grid Parameters Definition

The Grid parameters definition contains the definitions as follows.

YDim	Grid Y Dimension size
XDim	Grid X Dimension size
UpperLeftPoint	Grid Upper-left-point
LowerRightPoint	Grid Lower-right-point
Projection	Grid Projection Type as used in GCTP
ProjectionParameters	GCTP parameters for the projection
SphereCode	Optional Sphere Code for the projection. Default value is 0.
ZoneCode	Zone Code for UTM projection only. Default value is 0.
PixelRegistration	Defines pixel origin. Possible Registration Codes are: **HDFE_CENTER** (default) **HDFE_CORNER**
OriginType	Defines location of first datapoint. Possible Origin Codes are: **HDFE_GD_UL** (default) **HDFE_GD_UR** **HDFE_GD_LL** **HDFE_GD_LR**

Users should consult the HDF-EOS library User Guide for the projection types supported and the GCTP documents for the appropriate values for the projection parameters and the two codes. Currently supported projection types are as follows.

ProjCode	Projection Type

GCTP_GEO	Geographic
GCTP_UTM	Universal Transverse Mercator
GCTP_PS	Polar Stereographic
GCTP_SOM	Space Oblique Mercator
GCTP_GOOD	Interrupted Goodes Homolosine

```
<Grid Parameters Definition> ::=
     YDim = <XYdimSize>
     XDim = <XYdimSize>
     UpperLeftPoint = (<Pt>)
     LowerRightPoint = (<Pt>)
     Projection = <Projcode>
     ProjectionParameters = (<ProjParamList>)
     SphereCode = <Spherecode>
     ZoneCode = <Zonecode>
     PixelRegistration = <PixelReg>
     OriginType = <OriginCode>
```

Additional requirements:
All definitions can be in any order.
<XYdimSize> can be any positive integer representing the size of the dimension defined.
<PT> is a pair of float64 numbers separated by a comma.
<Projcode> is a projection code supported by HDF-EOS library.
<ProjParamList> is a list of 15 float64 numbers separated by commas.
<Spherecode> is an integer representing the sphere code.
<Zonecode> is an integer representing the zone code.
<PixelReg> is a pixel registration code supported by HDF-EOS library.
<OriginCode> is a legal origin code supported by HDF-EOS library.

4.3 Grid Dimension Definition

```
<Grid Dimension Definition> ::=
     <Dimension Definition>*

<Dimension Definition> ::=
     OBJECT = Dimension
          NAME = <DimensionName>
          Size = <DimensionSize>
     END_OBJECT = Dimension
```

Additional requirements:
<DimensionName> can be any legal quoted name but must be unique within <Grid Object>.

<DimensionSize> can be any non-negative integer representing the size of the dimension defined. A value zero or the keyword SD_UNLIMITED represents an unlimited dimension as defined in HDF.

4.4 Grid Datafield Definition

```
<Grid Datafield Definition> ::=
     <Datafield Definition>*

<Datafield Definition> ::=
     OBJECT = DataField
          NAME = <DatafieldName>
          DataType = <DataType>
          DimList = (<DimName1>, <DimName2>, ...)
          [<Grid Field Storage Definition>]
     END_OBJECT = DataField
```

Additional requirements:

<DatafieldName> can be any legal quoted name but must be unique within <Grid Object>.

<DataType> can be any legal datatype as defined in HDF.

<*DimName*> can be **XDim**, **YDim** or any dimension defined in <Grid Dimension Definition>. If **XDim** is used, it must be immediately preceded by **YDim**.

4.4.1 Grid Field Storage Definition

```
<Grid Field Storage Definition> ::=
     <Field Merge Definition> |
     <Field Compression Definition> |
     <Field Tile Definition>
```

4.4.1.1 Field Merge Definition

```
<Field Merge Definition> ::=
     Merge = <MergeCode>
```

Additional requirements:

<MergeCode> can be one of **HDFE_NOMERGE** (default) or **HDFE_AUTOMERGE.**

4.4.1.2 Field Compression Definition

```
<Field Compression Definition> ::=
     CompressionType = <CompressionCode>
```

CompressionParameters = (*<CompParamList>***)**

Additional requirements:

<CompressionCode> can be one of **HDFE_COMP_RLE, HDFE_COMP_SKPHUFF, HDFE_COMP_DEFLATE** or **HDFE_COMP_NONE** (default).

<CompParamList> is a list of integers separated by commas. Its requirements and ranges depend on *<CompressionCode>*. See the HDF-EOS document for more details.

4.4.1.3 Field Tile Definition

<Field Tile Definition> ::=
TileDimList = (*<TileSize1>, <TileSize2>, ...***)**

Additional requirements:

<TileSize> are positive integers defining the sizes of the tile. The number of members in the TileDimList must equal to that of the DimList in the same field definition.

4.4.2 Reserved Field Names

HDF-EOS library version 1 reserves Time as a special field name. If it is used in a grid definition, it must be defined with the following datatype.

Keyword	**Datatype**	**Comments**
Time	FLOAT32 or FLOAT64	TAI93 time in float

4.5 Grid Object Example

```
                              /* Defining the structure of two */
                              /* grid objects */

OBJECT = Grid                       /* First grid object */
      Name = "UTMGrid"              /* Grid parameters */
      XDim = 120
      YDim = 200
      UpperLeftPoint = (210584.500410,3322395.954450)
      LowerRightPoint = (813931.109590,2214162.532780)
      Projection = GCTP_UTM
      ZoneCode = 40
      SphereCode = 0
      OBJECT = Dimension            /* Dimension definitions */
            Name = "Time"
            Size = 10
      END_OBJECT = Dimension
      OBJECT = DataField            /* Datafield definitions */
            Name = "Pollution"
            DataType = DFNT_FLOAT32
            DimList = ("Time","YDim","XDim")
```

```
            TileDimList = (2, 50, 60)
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Vegetation"
            DataType = DFNT_FLOAT32
            DimList = ("YDim","XDim")
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Extern"
            DataType = DFNT_FLOAT32
            DimList = ("YDim","XDim")
      END_OBJECT = DataField
END_OBJECT = Grid

OBJECT = Grid                        /* Second grid object */
      Name = "PolarGrid"             /* Grid parameters */
      XDim = 100
      YDim = 100
      UpperLeftPoint = (0.0, 30000000.0)
      LowerRightPoint = (15000000.0, 20000000.0)
      Projection = GCTP_PS
      ProjectionParameters = (0.,0.,0.,0.,0.,9.0E7,0.,0.,0.,0.,0.,0.,0.,0.)
      SphereCode = 3
      OriginType = HDFE_GD_LR
      OBJECT = Dimension             /* Dimension definitions */
            Name = "Bands"
            Size = 3
      END_OBJECT = Dimension
      OBJECT = DataField             /* Datafield definitions */
            Name = "Temperature"
            DataType = DFNT_FLOAT32
            DimList = ("YDim","XDim")
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Pressure"
            DataType = DFNT_FLOAT32
            DimList = ("YDim","XDim")
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Soil Dryness"
            DataType = DFNT_FLOAT32
            DimList = ("YDim","XDim")
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Spectra"
            DataType = DFNT_FLOAT64
            DimList = ("Bands","YDim","XDim")
      END_OBJECT = DataField
END_OBJECT = Grid
END
```

5. Point Object

5.1 Point Object Definition

```
<Point Object> ::=
     OBJECT = Point
          NAME = <PointName>
          <Point Table Definition>
          [<Point LevelLink Definition>]
     END_OBJECT = Point
```

Additional requirements:

<PointName> can be any legal quoted name but must be unique among all HCR Object names.

5.2 Point Table Definition

```
<Point Table Definition> ::=
     <Table Definition>*

<Table Definition> ::=
     OBJECT = Level
          NAME = <TableName>
          <Point Field Definition>*
     END_OBJECT = Level
```

Additional requirements:

<TableName> can be any legal quoted name but must be unique within <Point Object>.

5.3 Point Field Definition

```
<Point Field Definition> ::=
     OBJECT = PointField
          NAME = <FieldName>
          DataType = <DataType>
          Order = <Order>
     END_OBJECT = PointField
```

Additional requirements:

<FieldName> can be any legal quoted name but must be unique within <Point Table Definition>.

<DataType> can be any legal datatype as defined in HDF.

<Order> is any positive integer. It is optional and the default value is 1. The concept of *Order* is defined in HDF Vdata.

5.3.1 Reserved Field Names

HDF-EOS library version 1 reserves the following field names. If the fields are used in a point definition, they must be defined with the following datatypes.

Keyword	Datatype	Comments
Latitude	FLOAT32 or FLOAT64	floating point latitude
Longitude	FLOAT32 or FLOAT64	floating point longitude
CoLatitude	FLOAT32 or FLOAT64	floating point colatitude
Time	FLOAT32 or FLOAT64	TAI93 time in float

5.4 Point LevelLink Definition

There are two kinds of linking definitions, representing two kinds of Point objects, namely Simple Point object and Linked Field Point object.[1] A point object is a Simple Point object unless defined otherwise by the Point LevelLink Definition.

```
<Point LevelLink Definition> ::=
     <Linked Field Definition>*

<Linked Field Definition> ::=
     OBJECT = LevelLink
          Parent = <ParentTableName>
          Child  = <ChildTableName>
          LinkField = <FieldName>
     END_OBJECT = LevelLink
```

Additional requirements:
<ParentTableName> is the name of a table defined in <Point Table Definition>.
<ChildTableName> is the name of a table defined in <Point Table Definition>.
<FieldName> is a field name defined in both Parent and Child tables.

5.4.1 Point LevelLink Name

HDF-EOS library implicitly defines the name of a Linked Field Definition as Parent table name and Child table name joined by a slash.

5.5 Point Object Example

```
                              /* Defining the structure of three */
                              /* point objects */
OBJECT = Point                      /* A simple point object */
      Name = "SimplePoint"
      OBJECT = Level                /* Level table definitions */
            Name = "Sensor"
```

[1] Indexed field point objects are not supported.

```
            OBJECT = PointField
                  Name = "Time"
                  DataType = DFNT_FLOAT64
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Concentration"
                  DataType = DFNT_FLOAT32
                  Order = 4
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Species"
                  DataType = DFNT_CHAR8
                  Order = 4
            END_OBJECT = PointField
      END_OBJECT = Level
END_OBJECT = Point

OBJECT = Point                        /* A linked field point object */
      Name = "FixedBuoyPoint"
      OBJECT = Level                  /* 1st table definition */
            Name = "DescLoc"
            OBJECT = PointField
                  Name = "Label"
                  DataType = DFNT_CHAR8
                  Order = 8
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Longitude"
                  DataType = DFNT_FLOAT64
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Latitude"
                  DataType = DFNT_FLOAT64
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "DeployDate"
                  DataType = DFNT_INT32
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "ID"
                  DataType = DFNT_CHAR8
                  Order = 1
            END_OBJECT = PointField
      END_OBJECT = Level

      OBJECT = Level                  /* 2nd table definition */
            Name = "Observations"
            OBJECT = PointField
                  Name = "Time"
                  DataType = DFNT_FLOAT64
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Rainfall"
                  DataType = DFNT_FLOAT32
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Temperature"
                  DataType = DFNT_FLOAT32
                  Order = 1
```

```
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "ID"
                  DataType = DFNT_CHAR8
                  Order = 1
            END_OBJECT = PointField
      END_OBJECT = Level

      OBJECT = LevelLink            /* Level link definitions */
            Parent = "DescLoc"
            Child = "Observations"
            LinkField = "ID"
      END_OBJECT = LevelLink
END_OBJECT = Point

OBJECT = Point                      /* Another linked field object */
      Name = "FloatBuoyPoint"
      OBJECT = Level                /* 1st table definition */
            Name = "ClusterGroup"
            OBJECT = PointField
                  Name = "TeamCode"
                  DataType = DFNT_INT32
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "DeployDate"
                  DataType = DFNT_INT32
                  Order = 1
            END_OBJECT = PointField
      END_OBJECT = Level
      OBJECT = Level                /* 2nd table definition */
            Name = "Description"
            OBJECT = PointField
                  Name = "Label"
                  DataType = DFNT_CHAR8
                  Order = 8
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "DeployDate"
                  DataType = DFNT_INT32
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Weight"
                  DataType = DFNT_INT16
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "ID"
                  DataType = DFNT_CHAR8
                  Order = 1
            END_OBJECT = PointField
      END_OBJECT = Level
      OBJECT = Level                /* 3rd table definition */
            Name = "Measurements"
            OBJECT = PointField
                  Name = "Time"
                  DataType = DFNT_FLOAT64
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Longitude"
                  DataType = DFNT_FLOAT64
                  Order = 1
            END_OBJECT = PointField
```

```
            OBJECT = PointField
                  Name = "Latitude"
                  DataType = DFNT_FLOAT64
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Rainfall"
                  DataType = DFNT_FLOAT32
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Temperature"
                  DataType = DFNT_FLOAT32
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "ID"
                  DataType = DFNT_CHAR8
                  Order = 1
            END_OBJECT = PointField
      END_OBJECT = Level
      OBJECT = LevelLink            /* Level link definition */
            Parent = "ClusterGroup"
            Child = "Description"
            LinkField = "DeployDate"
      END_OBJECT = LevelLink
      OBJECT = LevelLink            /* Level link definition */
            Parent = "Description"
            Child = "Measurements"
            LinkField = "ID"
      END_OBJECT = LevelLink
END_OBJECT = Point
END
```

6. Example HCR File For HDF-EOS

Below is example of HCR file containing all HDF-EOS objects.

```
OBJECT = Swath                        /* Defining a swath object */
      Name = "Swath 1"
      OBJECT = Dimension              /* Dimension definitions */
            Name = "GeoTrack"
            Size = 20
      END_OBJECT = Dimension
      OBJECT = Dimension
            Name = "GeoXtrack"
            Size = 10
      END_OBJECT = Dimension
      OBJECT = Dimension
            Name = "Res2tr"
            Size = 40
      END_OBJECT = Dimension
      OBJECT = Dimension
            Name = "Res2xtr"
            Size = 20
      END_OBJECT = Dimension
      OBJECT = Dimension
            Name = "Bands"
            Size = 15
      END_OBJECT = Dimension
      OBJECT = Dimension
            Name = "IndxTrack"
            Size = 12
      END_OBJECT = Dimension
      OBJECT = Dimension
            Name = "Unlim"
            Size = 0
      END_OBJECT = Dimension
      OBJECT = DimensionMap           /* Dimension mapping definitions */
            GeoDimension = "GeoTrack"
            DataDimension = "Res2tr"
            Offset = 0
            Increment = 2
      END_OBJECT = DimensionMap
      OBJECT = DimensionMap
            GeoDimension = "GeoXtrack"
            DataDimension = "Res2xtr"
            Offset = 1
            Increment = 2
      END_OBJECT = DimensionMap
      OBJECT = GeoField               /* Geofield Definitions */
            Name = "Time"
            DataType = DFNT_FLOAT64
            DimList = ("GeoTrack")
      END_OBJECT = GeoField
      OBJECT = GeoField
            Name = "Longitude"
            DataType = DFNT_FLOAT32
            DimList = ("GeoTrack","GeoXtrack")
      END_OBJECT = GeoField
      OBJECT = GeoField
            Name = "Latitude"
            DataType = DFNT_FLOAT32
            DimList = ("GeoTrack","GeoXtrack")
      END_OBJECT = GeoField
```

```
      OBJECT = DataField            /* Datafield Definitions */
            Name = "Density"
            DataType = DFNT_FLOAT32
            DimList = ("GeoTrack")
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Temperature"
            DataType = DFNT_FLOAT32
            DimList = ("GeoTrack","GeoXtrack")
            Merge = HDFE_AUTOMERGE
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "DewPoint"
            DataType = DFNT_FLOAT32
            DimList = ("GeoTrack","GeoXtrack")
            Merge = HDFE_AUTOMERGE
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Pressure"
            DataType = DFNT_FLOAT64
            DimList = ("Res2tr","Res2xtr")
            CompressionType = HDFE_COMP_DEFLATE
            CompressionParameters = (9)
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Spectra"
            DataType = DFNT_FLOAT64
            DimList = ("Bands","Res2tr","Res2xtr")
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Count"
            DataType = DFNT_INT16
            DimList = ("Unlim")
      END_OBJECT = DataField
END_OBJECT = Swath

                              /* Defining the structure of two */
                              /* grid objects */

OBJECT = Grid                       /* First grid object */
      Name = "UTMGrid"              /* Grid parameters */
      XDim = 120
      YDim = 200
      UpperLeftPoint = (210584.500410,3322395.954450)
      LowerRightPoint = (813931.109590,2214162.532780)
      Projection = GCTP_UTM
      ZoneCode = 40
      SphereCode = 0
      OBJECT = Dimension            /* Dimension definitions */
            Name = "Time"
            Size = 10
      END_OBJECT = Dimension
      OBJECT = DataField            /* Datafield definitions */
            Name = "Pollution"
            DataType = DFNT_FLOAT32
            DimList = ("Time","YDim","XDim")
            TileDimList = (2, 50, 60)
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Vegetation"
            DataType = DFNT_FLOAT32
            DimList = ("YDim","XDim")
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Extern"
```

```
            DataType = DFNT_FLOAT32
            DimList = ("YDim","XDim")
      END_OBJECT = DataField
END_OBJECT = Grid

OBJECT = Grid                        /* Second grid object */
      Name = "PolarGrid"             /* Grid parameters */
      XDim = 100
      YDim = 100
      UpperLeftPoint = (0.0, 30000000.0)
      LowerRightPoint = (15000000.0, 20000000.0)
      Projection = GCTP_PS
      ProjectionParameters = (0.,0.,0.,0.,0.,9.0E7,0.,0.,0.,0.,0.,0.,0.,0.,0.)
      SphereCode = 3
      OriginType = HDFE_GD_LR
      OBJECT = Dimension             /* Dimension definitions */
            Name = "Bands"
            Size = 3
      END_OBJECT = Dimension
      OBJECT = DataField             /* Datafield definitions */
            Name = "Temperature"
            DataType = DFNT_FLOAT32
            DimList = ("YDim","XDim")
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Pressure"
            DataType = DFNT_FLOAT32
            DimList = ("YDim","XDim")
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Soil Dryness"
            DataType = DFNT_FLOAT32
            DimList = ("YDim","XDim")
      END_OBJECT = DataField
      OBJECT = DataField
            Name = "Spectra"
            DataType = DFNT_FLOAT64
            DimList = ("Bands","YDim","XDim")
      END_OBJECT = DataField
END_OBJECT = Grid

                              /* Defining the structure of three */
                              /* point objects */
OBJECT = Point                      /* A simple point object */
      Name = "SimplePoint"
      OBJECT = Level                /* Level table definitions */
            Name = "Sensor"
            OBJECT = PointField
                  Name = "Time"
                  DataType = DFNT_FLOAT64
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Concentration"
                  DataType = DFNT_FLOAT32
                  Order = 4
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Species"
                  DataType = DFNT_CHAR8
                  Order = 4
            END_OBJECT = PointField
      END_OBJECT = Level
END_OBJECT = Point
```

```
OBJECT = Point                        /* A linked field point object */
      Name = "FixedBuoyPoint"
      OBJECT = Level                  /* 1st table definition */
            Name = "DescLoc"
            OBJECT = PointField
                  Name = "Label"
                  DataType = DFNT_CHAR8
                  Order = 8
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Longitude"
                  DataType = DFNT_FLOAT64
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Latitude"
                  DataType = DFNT_FLOAT64
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "DeployDate"
                  DataType = DFNT_INT32
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "ID"
                  DataType = DFNT_CHAR8
                  Order = 1
            END_OBJECT = PointField
      END_OBJECT = Level

      OBJECT = Level                  /* 2nd table definition */
            Name = "Observations"
            OBJECT = PointField
                  Name = "Time"
                  DataType = DFNT_FLOAT64
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Rainfall"
                  DataType = DFNT_FLOAT32
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "Temperature"
                  DataType = DFNT_FLOAT32
                  Order = 1
            END_OBJECT = PointField
            OBJECT = PointField
                  Name = "ID"
                  DataType = DFNT_CHAR8
                  Order = 1
            END_OBJECT = PointField
      END_OBJECT = Level

      OBJECT = LevelLink              /* Level link definitions */
            Parent = "DescLoc"
            Child = "Observations"
            LinkField = "ID"
      END_OBJECT = LevelLink
END_OBJECT = Point

OBJECT = Point                        /* Another linked field object */
      Name = "FloatBuoyPoint"
      OBJECT = Level                  /* 1st table definition */
```

```
      Name = "ClusterGroup"
      OBJECT = PointField
            Name = "TeamCode"
            DataType = DFNT_INT32
            Order = 1
      END_OBJECT = PointField
      OBJECT = PointField
            Name = "DeployDate"
            DataType = DFNT_INT32
            Order = 1
      END_OBJECT = PointField
END_OBJECT = Level
OBJECT = Level                /* 2nd table definition */
      Name = "Description"
      OBJECT = PointField
            Name = "Label"
            DataType = DFNT_CHAR8
            Order = 8
      END_OBJECT = PointField
      OBJECT = PointField
            Name = "DeployDate"
            DataType = DFNT_INT32
            Order = 1
      END_OBJECT = PointField
      OBJECT = PointField
            Name = "Weight"
            DataType = DFNT_INT16
            Order = 1
      END_OBJECT = PointField
      OBJECT = PointField
            Name = "ID"
            DataType = DFNT_CHAR8
            Order = 1
      END_OBJECT = PointField
END_OBJECT = Level
OBJECT = Level                /* 3rd table definition */
      Name = "Measurements"
      OBJECT = PointField
            Name = "Time"
            DataType = DFNT_FLOAT64
            Order = 1
      END_OBJECT = PointField
      OBJECT = PointField
            Name = "Longitude"
            DataType = DFNT_FLOAT64
            Order = 1
      END_OBJECT = PointField
      OBJECT = PointField
            Name = "Latitude"
            DataType = DFNT_FLOAT64
            Order = 1
      END_OBJECT = PointField
      OBJECT = PointField
            Name = "Rainfall"
            DataType = DFNT_FLOAT32
            Order = 1
      END_OBJECT = PointField
      OBJECT = PointField
            Name = "Temperature"
            DataType = DFNT_FLOAT32
            Order = 1
      END_OBJECT = PointField
      OBJECT = PointField
            Name = "ID"
            DataType = DFNT_CHAR8
```

```
                  Order = 1
            END_OBJECT = PointField
      END_OBJECT = Level
      OBJECT = LevelLink            /* Level link definition */
            Parent = "ClusterGroup"
            Child = "Description"
            LinkField = "DeployDate"
      END_OBJECT = LevelLink
      OBJECT = LevelLink            /* Level link definition */
            Parent = "Description"
            Child = "Measurements"
            LinkField = "ID"
      END_OBJECT = LevelLink
END_OBJECT = Point
END
```

PART III. HCR FOR HDF

7. SDS Object

SDS Object refers to *Scientific Data Set* in HDF.

7.1 SDS Object Definition

```
<SDS Object> ::=
    OBJECT = SDS
        [<SDS Dimension With Name Definition>]*
        <SDS Array Definition>*
        [<File Attribute Definition>]*
    END_OBJECT = SDS
```

7.2 Object of SDS Dimension With Name

SDS Dimension With Name is defined as an object so that different SDS Array objects can share them. This design can enhance reusability.

7.2.1 SDS Dimension With Name Object Definition

```
<SDS Dimension With Name Object> ::=
    OBJECT = SDSDimensionWithName
        NAME = <DimensionName>
        SIZE = <DimensionSize>
        [<Dimension Predefined Attribute Definition>]
        [<User-defined Dimension Attribute Definition>]
        [<Dimension Scale Definition>]
    END_OBJECT = SDSDimensionWithName
```

Additional requirements:
<DimensionName> can be any legal quoted name but must be unique among the same type of objects.
<DimensionSize> is a positive integer.

7.2.2 Dimension Predefined Attribute Definition

```
<Dimension Predefined Attribute Definition> ::=
    OBJECT = Predefined_Dimension_Attribute
        [LABEL = <NameString>]
        [UNIT = <UnitString>]
        [FORMAT   = <FormatString>]
    END_OBJECT = Predefined_Dimension_Attribute
```

7.2.3 User-defined Dimension Attribute Definition

```
<User-defined Dimension Attribute Definition> ::=
```

```
    <User-defined Attribute Definition>*

<User-defined Attribute Definition> ::=
    OBJECT = User_Defined_Attribute
        NAME = <AttributeName>
        DATATYPE = <AttributeType>
        N_VALUES = <AttributeCount>
        DATA = <AttributeData>
    END_OBJECT = User_Defined_Attribute

<AttributeData> ::=
     <AttributeDataString> | (<data1>, <data2>,…,<dataN>)
```

Additional requirements:

<AttributeName> can be any legal quoted string.

<AttributeType> can be DFNT_FLOAT32, DFNT_FLOAT64, DFNT_INT8, DFNT_INT16, DFNT_INT32, DFNT_UINT8, DFNT_UINT16, DFNT_UINT32, or DFNT_CHAR8.

<AttributeCount> is the number of attribute data elements.

<AttributeDataString> can be any legally quoted string(like "Seconds").

(<data1>, <data2>,…, <dataN>) can be any legal HDF integer or real number, each datum is separated by a comma(like (0, 1, 2, 3, 4)).

7.2.4 File Attribute Definition

We discuss file attribute here although it does not belong to the scope of dimension. File Attribute has the same definition with User-defined Attribute in SDS dimension with name. Please refer to 7.2.3.

```
<File Attribute Definition> ::=
     <User-defined Attribute Definition>
```

7.2.5 Dimension Scale Definition

```
<Dimension Scale Definition> ::=
    OBJECT = SDSDimensionScale
        N_VALUES = <ScaleValueNum>
        DATATYPE = <ScaleDataType>
        DATA     = <ScaleData>
    END_OBJECT = SDSDimensionScale

<ScaleData> ::=
     <ScaleDataString> | (<data1>, <data2>,…, <dataN>)
```

Additional requirements:

<ScaleValueNum> is the number of data elements.

<ScaleDataType> can be DFNT_FLOAT32, DFNT_FLOAT64,

DFNT_INT8, DFNT_INT16, DFNT_INT32, DFNT_UINT8, DFNT_UINT16, DFNT_UINT32 or DFNT_CHAR8.

<ScaleDataString> can be any legally quoted string(like "Seconds").

(<data1>, <data2>,…, <dataN>) can be any legal HDF integer or real number, each datum is separated by a comma(like (0, 1, 2, 3, 4)).

7.3 SDS Array Definition

```
<SDS Array Object> ::=
    OBJECT = SDSArray
        NAME = <SDSArrayName>
        DATATYPE = <DataType>
        DIMENSIONRANK = <DimRank>
        DIMENSIONSIZE = (<DimSize1>,<DimSize2>,…,<DimSizeN>)
        DIMENSIONLIST = (<DimName1>,<DimName2>,…,<DimSizeN>)
        [<SDS Dimension Without Name Definition>]*
        [<User-defined Attribute Definition>]*
    END_OBJECT = SDSArray
```

Additional requirements:

<SDSArrayName> can be any legal quoted name but must be unique among all SDS Array objects. This rule is different from HDF API, which allows duplicate names.

<DataType> can be DFNT_FLOAT32, DFNT_FLOAT64, DFNT_INT8, DFNT_INT16, DFNT_INT32, DFNT_UINT8, DFNT_UINT16, DFNT_UINT32 or DFNT_CHAR8.

<DimRank> is the number of dimensions in the current SDS Array object.

(<DimSize1>,<DimSize2>,…<DimeSizeN>) is a list of all dimension sizes. For example, if an SDS Array Object has three dimensions, this list should be like (5, 16, 10).

(<DimName1>,<DimName2>,…,<DimSizeN>) can be either a dimension name defined by SDS Dimension With Name Object or "-" which stands for no name. For example, for ("X_Axis", "Y_Axis", "-"), the first two dimensions refer to the ones has been defined as SDS Dimension With Name Object; the third one has no name, whose dimension scale and attributes can be defined in [<SDS Dimension Without Name Definition>].

[<SDS Dimension Without Name Definition>] is used when *<DimName>* in dimension list is "-" and user wants to add dimension attributes and scale.

An optional parameter <SDS Array Data Storage Definition> may be added later.

7.4 SDS Dimension Without Name Definition

This definition is used when <DimName > in DIMENSIONLIST is "-".

```
<SDS Dimension Without Name Definition> ::=
    OBJECT = DimensionWithoutName
        INDEX = <IndexInDimensionList>
        [<Dimension Predefined Attribute Definition>]
        [<User-defined Attribute Definition>]*
        [<Dimension Scale Definition>]
    END_OBJECT = DimensionWithoutName
```

Additional requirements:

<IndexInDimensionList> is the position of this dimension in DIMENSIONLIST in SDS Array Object definition. For the example of ("X_Axis", "Y_Axis", "-"), the index value should be 2.

[<Dimension Predefined Attribute Definition>], please refer to 7.2.2 for definition.

[<User-defined Attributes Definition>], please refer to 7.2.3 for definition.

[<Dimension Scale Definition>], please refer to 7.2.5 for definition.

7.5 SDS Array Attribute Definition

The SDS Array attributes include predefined dataset attributes and user-defined attributes.

```
<SDS Attributes Definition> ::=
    <SDS Predefined Dataset Attribute Definition> |
    <User-defined Attribute Definition>*

<SDS Predefined Dataset Attribute Definition> ::=
    OBJECT = Predefined_Dataset_Attribute
        [LABEL          = <ArrayNameString>]
        [UNIT           = <Units>]
        [FORMAT         = <FormatString>]
        [COORDINATE_SYSTEM = <CoordinateSystemString>]
        [RANGE          = <ValidRange>]
        [FILL_VALUE     = <FillValue>]
        [SCALE_FACTOR   = <ScaleFactor>]
        [SCALE_FACTOR_ERROR = <ScaleFactorError>]
        [ADD_OFFSET     = <AddOffset>]
        [ADD_OFFSET_ERROR   = <AddOffsetError>]
        [CALIBRATED_NT  = <CalibratedNt>]
    END_OBJECT = Predefined_Dataset_Attribute
```

7.6 SDS Object Example

```
OBJECT = SDS

      OBJECT = SDSDimensionWithName
            Name = "X_Axis"
            Size = 5
            OBJECT = SDSDimensionScale
                  N_Values = 5
                  DataType = DFNT_INT32
                  Data = (0, 1, 2, 3, 4)
            END_OBJECT = SDSDimensionScale
            OBJECT = User_Defined_Attribute
                  Name = "Dim_metric"
```

```
            DataType = DFNT_CHAR8
            N_Values = 7
            Data = "Seconds"
      END_OBJECT = User_Defined_Attribute
END_OBJECT = SDSDimensionWithName

OBJECT = SDSDimensionWithName
      Name = "Y_Axis"
      Size = 16
      OBJECT = SDSDimensionScale
            N_Values = 16
            DataType = DFNT_FLOAT32
            Data = (0.000, 0.100, 0.200, 0.300,
                        0.400, 0.500, 0.600, 0.700,
                        0.800, 0.900, 1.000, 1.100,
                        1.200, 1.300, 1.400, 1.500)
      END_OBJECT = SDSDimensionScale
      OBJECT = Predefined_Dimension_Attribute
            Label = "aaaa"
            Unit = "bbbb"
            Format = "cccc"
      END_OBJECT = Predefined_Dimension_Attribute
END_OBJECT = SDSDimensionWithName

OBJECT = SDSArray
      Name = "SDStemplate"
      DataType = DFNT_INT16
            DimensionRank = 3
            DimensionSize = (5, 16, 10)
      DimensionList = ("X_Axis", "Y_Axis", "-")
      OBJECT = SDSDimensionWithoutName
            Index = 2
            OBJECT = SDSDimensionScale
                  N_Values = 10
                  DataType = DFNT_FLOAT64
                  Data = (0.000, 0.100, 0.200, 0.300,
                              0.400, 0.500, 0.600, 0.700,
                              0.800, 0.900)
            END_OBJECT = SDSDimensionScale
            OBJECT = User_Defined_Attribute
                  Name = "Dim_Metrics"
                  DataType = DFNT_CHAR8
                  N_Values = 7
                  Data = "Minutes"
            END_OBJECT = User_Defined_Attribute
            OBJECT = Predefined_Dimension_Attribute
                  Label = "ffff"
                  Unit = "dddd"
                  Format = "eeee"
            END_OBJECT = Predefined_Dimension_Attribute
      END_OBJECT = SDSDimensionWithoutName

      OBJECT = User_Defined_Attribute
            Name = "Valid_range"
            DataType = DFNT_FLOAT32
            N_Values = 2
            Data = (2., 10.)
      END_OBJECT = User_Defined_Attribute
      OBJECT = Predefined_Dataset_Attribute
            Label = "aaaa"
            Unit = "bbbb"
            Format = "cccc"
            Coordinate_System = "Cardinal"
            Range = (0, 255)
            Fill_Value = (5)
```

```
                  Scale_Factor = 0.1
                  Scale_Factor_Error = 0.002
                  Add_Offset = 0.01
                  Add_Offset_Error = 0.0003
                  Calibrated_Nt = DFNT_FLOAT64
            END_OBJECT = Predefined_Dataset_Attribute
      END_OBJECT = SDSArray

      OBJECT = User_Defined_Attribute
            Name = "File_contents"
            DataType = DFNT_CHAR8
            N_Values = 16
            Data = "Storm_track_data"
      END_OBJECT = User_Defined_Attribute

END_OBJECT = SDS

OBJECT = Data_Annotation
      OwnerType = SDS
      OwnerName = "SDStemplate"
            Type = AN_DATA_LABEL
      Content = "Common A SDS"
END_OBJECT = Data_Annotation
OBJECT = Data_Annotation
      OwnerType = SDS
      OwnerName = "SDStemplate"
            Type = AN_DATA_DESC
      Content = "This is an SDS that is used to test data annotation."
END_OBJECT = Data_Annotation


END
```

8. GR Object

GR Object refers to *General Raster Image* in HDF.

8.1 GR Object Definition

```
<GR Object> :=
     <Image Array Definition>
     [<User-defined GR Attribute Definition>]

<User-defined GR Attribute Definition> ::=
     <User-defined Attribute Definition>*
```

8.2 Image Array Object Definition

```
<Image Array Object> ::=
     OBJECT = ImageArray
              NAME = <ImageName>
              N_COMPS = <NumberOfComponents>
              PIXELTYPE = <ImagePixelType>
              INTERLACEMODE = <InterlaceMode>
              DIMENDSIONSIZE = (<integer>, <integer>)
              [<Image Palette Definition>]*
              [<Image Attribute Definition>]*
     END_OBJECT = ImageArray
```

Additional requirements:

<ImageName> can be any legal quoted but must be unique among the same kind of objects.

<NumberOfComponents> is at least 1.

<ImagePixelType> can be DFNT_FLOAT32, DFNT_FLOAT64, DFNT_INT8, DFNT_INT16, DFNT_INT32, DFNT_UINT8, DFNT_UINT16, DFNT_UINT32 or DFNT_CHAR8.

<InterlaceMode> is either MFGR_INTERLACE_PIXEL(0), MFGR_INTERLACE_LINE(1) or MFGR_INTERLACE_COMPONENT(2).

(<integer>, <integer>) are the sizes of the X and Y dimensions of the image.

An optional parameter <Image Data Storage Definition> may be added later.

8.3 Image Palette Definition

Image Palette has an index and 768(256x3) data elements.

```
<Image Palette Definition> ::=
     OBJECT = Palette
          Index = <PaletteIndex>
          Data = (<integer>, …, <integer>)
     END_OBJECT = Palette
```

Additional requirements:
<PaletteIndex> starts from 0, for a certain image,.
(<integer>, …, <integer>) must be 768 entries of data elements.

8.4 Image Attribute Definition

Image Attribute has the same definition as the User-defined Attributes Definition in SDS Dimension With Name. Please refer to 7.2.3.

```
<Image Attribute Definition> ::=
     <User-defined Attribute Definition>
```

8.5 GR Object Example

```
OBJECT = GR
      OBJECT = ImageArray
                  Name = "Image Array 1"
                  N_Comps = 2
                  PixelType = DFNT_INT16
                  InterlaceMode = MFGR_INTERLACE_PIXEL
                  DimensionSize = (10, 5)
      END_OBJECT = ImageArray

      OBJECT = ImageArray
                  Name = "Image Array 2"
                  N_Comps = 1
                  PixelType = DFNT_INT16
                  InterlaceMode = MFGR_INTERLACE_PIXEL
                  DimensionList = (20, 25)
                  OBJECT = Palette
                        Index = 0
                        Data = (1, 1, 1,
                                    255, 255, 255,
                                     :       :       :
                                    0, 0, 0)
                  END_OBJECT = Palette
                  OBJECT = User_Defined_Attribute
                        Name = "Scale 1"
                        DataType = DFNT_INT32
                        N_Values = 4
                        Data = (2, 4, 6, 8)
                  END_OBJECT = User_Defined_Attribute
      END_OBJECT = ImageArray

      OBJECT = User_Defined_Attribute
                  Name = "Scale 2"
                  DataType = DFNT_CHAR8
                  N_Values = 3
```

```
                  Data = "FFF"
      END_OBJECT = User_Defined_Attribute

END_OBJECT = GR

OBJECT = Data_Annotation
      OwnerType = GR
      OwnerName = "Image Array 1"
      Type = AN_DATA_LABEL
      Content = "Common A GR"
END_OBJECT = Data_Annotation

OBJECT = Data_Annotation
      OwnerType = GR
      OwnerName = "Image Array 2"
      Type = AN_DATA_DESC
      Content = "This is 2 GR that is used to test data annotation."
END_OBJECT = Data_Annotation


END
```

9. Vdata Object

Vdata Object is the same as *Vdata* in HDF.

9.1 *Vdata Object Definition*

```
<Vdata Object> ::=
    OBJECT = Vdata
        NAME = <VdataName>
        [CLASS = <VdataClass>]
        [INTERLACEMODE = <InterlaceMode>]
        <Vdata Field Definition>*
        [<Vdata Attribute Definition>]*
    END_OBJECT = Vdata
```

Additional requirements:

<VdataName> can be any legal quoted name but must be unique among the same objects.

<VdataClass> can be any legal quoted name.

<InterlaceMode> can be either FULL_INTERLACE(default) or NO_INTERLACE.

9.2 Vdata Field Definition

```
<Vdata Field Definition> ::=
    OBJECT = Field
        NAME = <FieldName>
        DATATYPE = <DataType>
        ORDER = <EntryOrder>
        [<Field Attribute Definition>]*
    END_OBJECT = Field
```

Additional requirements:

<FieldName> is quoted name and must be unique among the same type of objects.

<DataType> can be DFNT_FLOAT32, DFNT_FLOAT64, DFNT_INT8, DFNT_INT16, DFNT_INT32, DFNT_UINT8, DFNT_UINT16, DFNT_UINT32 or DFNT_CHAR8.

<EntryOrder> is the number of entries in each field.

9.3 Vdata Attribute Definition and Field Attribute Definition

Vdata Attribute and Field Attribute have the same definition as User-defined Attribute Definition in SDS Object. Please refer to 7.2.3.

```
<Vdata Attribute Definition> |
<Field Attribute Definition> ::=
     <User-defined Attribute Definition>
```

9.4 Vdata Object Example

```
OBJECT = Vdata
          Name = "Solid Particle"
          Class = "Particle Data"
          InterlaceMode = NO_INTERLACE

          OBJECT = Field
                  Name = "Position"
                  DataType = DFNT_FLOAT32
                  Order = 3
          END_OBJECT = Field

          OBJECT = Field
                  Name = "Mass"
                  DataType = DFNT_FLOAT32
                  Order = 1
            OBJECT = User_Defined_Attribute
                        Name = "Scale1"
                        DataType = DFNT_INT16
                        N_Values = 4
                        Data = (2, 4, 6, 8)
                  END_OBJECT = User_Defined_Attribute
                  OBJECT = User_Defined_Attribute
                        Name = "Scale2"
                        DataType = DFNT_CHAR8
                        N_Values = 5
                        Data = "aaaaa"
                  END_OBJECT = User_Defined_Attribute
           END_OBJECT = Field

           OBJECT = Field
                  Name = "Temperature"
                  DataType = DFNT_FLOAT32
                  Order = 2
                  OBJECT = User_Defined_Attribute
                        Name = "Scale3"
                        DataType = DFNT_FLOAT32
                        N_Values = 4
                        Data = (10.1, 20.2, 30.3, 40.4)
                  END_OBJECT = User_Defined_Attribute
           END_OBJECT = Field

           OBJECT = User_Defined_Attribute
                  Name = "Site Ident1"
                  DataType = DFNT_FLOAT64
                  N_Values = 3
                  Data = (1.2, 3.2, 6.5)
           END_OBJECT = User_Defined_Attribute

                 OBJECT = User_Defined_Attribute
                  Name = "Site Ident2"
                  DataType = DFNT_CHAR8
                  N_Values = 3
                  Data = "ABC"
```

```
           END_OBJECT = User_Defined_Attribute
END_OBJECT = Vdata

OBJECT = Vdata
      Name = "Solid Particle2"
      Class = "Particle Data"
      InterlaceMode = FULL_INTERLACE

      OBJECT = Field
                  Name = "Position2"
                  DataType = DFNT_FLOAT32
                  Order = 3
      END_OBJECT = Field
      OBJECT = User_Defined_Attribute
                  Name = "Site Ident3"
                  DataType = DFNT_CHAR8
                  N_Values = 3
                  Data = "DEF"
      END_OBJECT = User_Defined_Attribute
END_OBJECT = Vdata

OBJECT = Data_Annotation
      OwnerType = Vdata
      OwnerName = "Solid Particle"
            Type = AN_DATA_LABEL
      Content = "Common A Vdata"
END_OBJECT = Data_Annotation

OBJECT = Data_Annotation
      OwnerType = Vdata
      OwnerName = "Solid Particle"
            Type = AN_DATA_DESC
      Content = "This is an Vdata that is used to test data annotation."
END_OBJECT = Data_Annotation

END
```

10. Vgroup Object

Vgroup Object is the same as *Vgroup* in HDF.

10.1 Vgroup Object Definition

```
<Vgroup Object> ::=
    OBJECT = Vgroup
        NAME = <VgroupName>
        [CLASS = <VgroupClass>]
        [<Vgroup Member Definition>]*
        [<Vgroup Attribute Definition>]*
    END_OBJECT = Vgroup
```

Additional requirements:
<VgroupName> is a legal quoted name and must be unique among the same objects.
<VgroupClass> can be any legal quoted name.

10.2 Vgroup Member Definition

```
<Vgroup Member Definition> ::=
    OBJECT = Member
        MEMBERTYPE = <ObjectType>
        [MEMBERNAME = <VgroupMemberName>]
        [PALETTEINDEX = <MemberIndex>]
    END_OBJECT = Member
```

<ObjectType> ::= SDS | GR | Vgroup | Vdata | Palette_Object

Additional requirements:
<VgroupMemberName> is for SDS, GR, Vgroup or Vdata object. It is used to indentify these objects. The object having this name has to be existent.
<MemberIndex> is only for Palette Object. The <VgroupMemberType> has to be 'Palette_Object' for this field to be valid. Starting from 0, this index is the same as the sequence in which palettes are inserted into file.

10.3 Vgroup Attribute Definition

Vgroup Attribute has the same definition as the one in SDS Dimension. Please refer to 7.2.3.

```
[<Vgroup Attribtute Definition> ::=
   <User-defined Attribute Definition>
```

10.4 Vgroup Object Example

```
OBJECT = Palette_Object
      Index = 0
      Data = (1, 1, 1,
                          255, 255, 255,
                          0, 0, 0,
                          1, 1, 1,
                          1, 1, 1,
                           :   :  :
                          1, 1, 1,
                          1, 1, 1)
END_OBJECT = Palette_Object

OBJECT = SDS
      OBJECT = SDSArray
            Name = "SDStemplate"
            DataType = DFNT_INT32
            DimensionRank = 1
            DimensionSize = (10)
            DimensionList = ("-")
            OBJECT = SDSDimensionWithoutName
                  Index = 0
                  OBJECT = SDSDimensionScale
                        N_Values = 10
                        DataType = DFNT_FLOAT64
                        Data = (0.000, 0.100, 0.200, 0.300,
                                    0.400, 0.500, 0.600, 0.700,
                                    0.800, 0.900)
                  END_OBJECT = SDSDimensionScale
            END_OBJECT = SDSDimensionWithoutName
      END_OBJECT = SDSArray
END_OBJECT = SDS


OBJECT = GR
      OBJECT = ImageArray
            Name = "Image Array 1"
                  N_Comps = 2
                  PixelType = DFNT_INT16
                  InterlaceMode = MFGR_INTERLACE_PIXEL
                  DimensionSize = (10, 5)
      END_OBJECT = ImageArray
END_OBJECT = GR

OBJECT = Vgroup
      Name = "Vertices2"
      Class = "Vertex Set"
END_OBJECT = Vgroup

OBJECT = Vdata
            Name = "Solid Particle"
      Class = "Particle Data"
      OBJECT = Field
```

```
                  Name = "Position2"
                  DataType = DFNT_FLOAT32
                  Order = 3
      END_OBJECT = Field
END_OBJECT = Vdata

OBJECT = Vgroup
      Name = "Vertices"
      Class = "Vertex Set"

      OBJECT = Member
            MemberType = SDS
            MemberName = "SDStemplate"
      END_OBJECT = Member
      OBJECT = Member
            MemberType = GR
            MemberName = "Image Array 1"
      END_OBJECT = Member
      OBJECT = Member
            MemberType = Vgroup
            MemberName = "Vertices2"
      END_OBJECT = Member
      OBJECT = Member
            MemberType = Vdata
            MemberName = "Solid Particle"
      END_OBJECT = Member
      OBJECT = Member
            MemberType = Palette_Object
            PaletteIndex = 0
      END_OBJECT = Member

      OBJECT = User_Defined_Attribute
            Name = "Dim_metric"
            DataType = DFNT_CHAR8
            N_Values = 7
            Data = "Minutes"
      END_OBJECT = User_Defined_Attribute
END_OBJECT = Vgroup

OBJECT = File_Annotation
      Type = AN_FILE_LABEL
      Content = "General HDF Objects"
END_OBJECT = File_Annotation

OBJECT = File_Annotation
      Type = AN_FILE_DESC
      Content = "This is an HDF file that contains general HDF objects"
END_OBJECT = File_Annotation


END
```

11. Palette Object

Palette Object is the same as *Palette* in HDF.

11.1 Palette Object Definition

Each object contains the data of 768(256x3) integers, representing a RGB mode lookup table, totaling 256 colors.

```
<Palette Object> ::=
    OBJECT = Palette_Object
        INDEX = (<paletteIndex>)
        DATA = (<data>, <data>, ...)
    END_OBJECT = Palette_Object
```

Requirements:

<paletteIndex> is a sequencial number starting from 0.

(<data>, <data>, …) must be 768 entries of 8-bit unsigned integer.

11.2 Palette Object Example

```
OBJECT = Palette_Object
      Index = 0
      Data = (0, 0, 0,
              255, 255, 255,
              0, 0, 0,
              1, 1, 1,
              :   :  :
              1, 1, 1,
              1, 1, 1)
END_OBJECT = Palette_Object

OBJECT = Palette_Object
      Index = 1
      Data = (1, 1, 1,
              255, 255, 255,
              0, 0, 0,
              1, 1, 1,
              1, 1, 1,
              :   :  :
              1, 1, 1,
              10, 10, 10,
              11, 11, 11,
              12, 12, 12)
```

```
END_OBJECT = Palette_Object

OBJECT = Data_Annotation
      OwnerType = Palette
      OwnerIndex = 0
            Type = AN_DATA_LABEL
      Content = "Common A Palette"
END_OBJECT = Data_Annotation
OBJECT = Data_Annotation
      OwnerType = Palette
      OwnerIndex = 1
      Type = AN_DATA_DESC
      Content = "This is an Palette that is used to test data
annotation."
END_OBJECT = Data_Annotation

END
```

12. Annotation Object

There are two kinds of Annotation Objects, File Annotation and Data Annotation. File Annotation is for a whole file, while Data Annotation is for HDF objects like SDS, GR, Vdata and Vgroup.

12.1 File Annotation Object Definition

```
<File Annotation Object> ::=
    OBJECT = FileAnnotation
        TYPE = <AnnotationType>
        CONTENT = <AnnotationContent>
    END_OBJECT = FileAnnotation

<AnnotationContent> ::=
     <Label> | <Description>
```

Additional requirements:

<AnnotationType> is either AN_FILE_LABEL for label, or AN_FILE_DESC for description.

<Label> is a null-terminated string of characters. It is assumed to be a short message.

<Description> can be any sequence of ASCII characters. It is assumed to be a longer message compared to *<Label>*.

12.2 Data Annotation Object Definition

```
<Data Annotation Object> ::=
    OBJECT = DataAnnotation
        OWNERTYPE = <OwnerObjectType>
        [OWNERNAME = <OwnerObjectName>]
        [OWNERINDEX = <OwnerObjectIndex>]
        Type = <AnnotationType>
        CONTENT = <AnnotationContent>
    END_OBJECT = DataAnnotation
```

<OwnerObjectType> ::=
<ObjectType>

The definition of *<ObjectType>* is in 10.2.

Additional requirements:
<OwnerObjectType> refers to the object to which this annotation will be attached. It is same as Vgroup member type.
<OwnerObjectName> is only used for SDS, GR, Vdata, Vgroup.
<OwnerObjectIndex> is only used for Palette Object.
<AnnotationType> is either AN_DATA_LABEL for label, or AN_DATA_DESC for description.
<AnnotationContent> is same as the description in 12.1.

12.3 Annotation Object Example

```
OBJECT = File_Annotation
      Type = AN_FILE_LABEL
      Content = "General HDF Objects"
END_OBJECT = File_Annotation

OBJECT = File_Annotation
      Type = AN_FILE_DESC
      Content = "This is an HDF file that contains general HDF objects"
END_OBJECT = File_Annotation


OBJECT = Vgroup
      Name = "Vertices2"
      Class = "Vertex Set"
END_OBJECT = Vgroup

OBJECT = Palette_Object
      Index = 0
      Data = (1, 1, 1,
              255, 255, 255,
              0, 0, 0,
              1, 1, 1,
              1, 1, 1,
              :   :   :
              1, 1, 1,
              1, 1, 1)
END_OBJECT = Palette_Object

OBJECT = Data_Annotation
      OwnerType = Vgroup
      OwnerName = "Vertices2"
      Type = AN_DATA_LABEL
      Content = "Common A Vgroup"
END_OBJECT = Data_Annotation
OBJECT = Data_Annotation
      OwnerType = VGroup
      OwnerName = "Vertices2"
      Type = AN_DATA_DESC
      Content = "This is a Vgroup that is used to test data annotation."
END_OBJECT = Data_Annotation
```

```
OBJECT = Data_Annotation
      OwnerType = Palette_Object
      OwnerIndex = 0
      Type = AN_DATA_LABEL
      Content = "Common A Palette"
END_OBJECT = Data_Annotation
OBJECT = Data_Annotation
      OwnerType = Palette_Object
      OwnerIndex = 0
      Type = AN_DATA_DESC
      Content = "This is a Palette that is used to test data
annotation."
END_OBJECT = Data_Annotation


END
```

PART IV. Related Documents

13. Related Documents

13.1 HDF-EOS Documents

For a detailed description of the HDF-EOS library, refer to the following documents.

[EOS96-1]
"Draft Design Document for Proposed HDF-EOS Library",
http://edhs1.gsfc.nasa.gov/ftp/hdf_eos/doc/HDFEOSLib/

[EOS96-2]
"The HDF-EOS Swath Concept",
http://edhs1.gsfc.nasa.gov/ftp/hdf_eos/doc/SwathPaper/

[EOS96-3]
"The HDF-EOS Grid Concept",
http://edhs1.gsfc.nasa.gov/ftp/hdf_eos/doc/GridPaper/

[EOS96-4]
"The HDF-EOS Point Concept",
http://edhs1.gsfc.nasa.gov/ftp/hdf_eos/doc/PointPaper/

[EOS96-5]
"Thoughts on HDF-EOS Metadata",
http://edhs1.gsfc.nasa.gov/ftp/hdf_eos/doc/MetaThought/

[EOS97-1]
"HDF-EOS Library User's Guide Volume 1: Overview and Examples",
http://edhs1.gsfc.nasa.gov/waisdata/sdp/html/tp1700503.html, April 1997.

[EOS97-2]
"HDF-EOS Library Users Guide Volume 2: Function Reference Guide",

http://edhs1.gsfc.nasa.gov/waisdata/sdp/html/tp1700602.html, April 1997.

[HDF96-2]
"HDF Configuration Record Requirements ",
ftp://hdf.ncsa.uiuc.edu/pub/HCR/Doc/HCR-Requirements.ps, April 1996.

13.2 HDF Documents

For a detailed description of the Hierarchical Data Format (HDF) software, refer to the following documents.

[HDF]
"HDF Information Server",
http://hdf.ncsa.uiuc.edu/

[HDF96]
"HDF Users Guide", version 4.0,
ftp://hdf.ncsa.uiuc.edu/pub/dist/HDF/Documentation/HDF4.0/Users_Guide

13.3 GCTP Document

For a detailed description of the GCTP software, refer to the following document.

[GCTP96]
"General Cartographic Transformation Package",
ftp://edcftp.cr.usgs.gov/pub/software/gctpc/getpc.tar.Z

13.4 Object Description Language Documents

The Metadata are stored in the form of Object Description Language (ODL) as defined in the Planetary Data System (PDS) of the Jet Propulsion Laboratory. The following are related ODL and PDS documents.

[PDS95]
"Planetary Data System Standards Reference", Version 3.2,
http://pds.jpl.nasa.gov/stdref/stdref.htm

[PDS95-12]
“Object Description Language (ODL) Specification and Usage”,
http://pds.jpl.nasa.gov/stdref/chap12.htm

13.5 Parameter Value Language Documents

ODL is related to the Parameter Value Language (PVL) as defined in the Standard Formatted Data Units (SFDU). The following are related PVL and SFDU documents.

[SFDU92]
"Recommendation for Space Data System Standards, Standard Formatted Data Units -- Structure and Construction Rules",

ftp://nssdc.gsfc.nasa.gov/pub/sfdu/p2docs/postscript/ccsds-641-0-b-1.ps

[PVL92]
"Recommendation for Space Data System Standards, Parameter Value Language Specification",
ftp://nssdc.gsfc.nasa.gov/pub/sfdu/p2docs/postscript/ccsds-641-0-b-1.ps