HDF Configuration Record Definition

Version 1.1

Technical Document

Revised August 1997

Prepared Under Subcontract #301684

Hughes Information Technology Systems

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

Albert Cheng

Michael Folk

Paul Harten

William Whitehouse

National Center for Supercomputing Applications

University of Illinois at Urbana-Champaign

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 use of the HCR. Readers would be able to compose an HCR to create HDF-EOS 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].

1.2 The Need for a Configuration Record

End-users may wish to create an HDF-EOS file using the HDF-EOS library interface which supports grid, swath, and point objects. 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 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

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. Chapter 7 lists related documents.

1.4 Version History

1.4.1 Version 1.2

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

1.4.2 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
```

2.2 HDF-EOS Objects

HCR supports three kinds of HDF-EOS objects, namely *Swath, Grid,* and *Point.* An HCR file contains zero or more HDF-EOS object definitions ending with an End Statement. An HDF-EOS object definition can be one of *Swath Object, Grid Object,* or *Point Object.*

```
<HCR File> ::=
      [<Swath Object> | <Grid Object> | <Point Object>]*
      End
```

Requirements:
All object definitions can be in any order and are optional. A file that contains no definition but the **End** statement is a valid, though trivial HCR file.

The next three chapters define the syntax of the Swath Object, the Grid Object, and the Point Object.

3. Swath Object

3.1 Swath Object Definition

A Swath object is enclosed in the **OBJECT = Swath** statement and its corresponding **END_OBJECT** statement. Each object contains the name of the Swath and four main components as follows,

Swath Dimension Definition
Swath Geofield Definition
Swath Datafield Definition
Swath Dimension Mapping Definition (optional)

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

Requirements:

`<SwathName>` can be any legal quoted name but must be unique among all HCR object names.
`<Swath Dimension Definition>` is required.
`<Swath Geofield Definition>` is required.
`<Swath Datafield Definition>` is required.
`<Swath Dimension Mapping Definition>` is optional.

3.2 Swath Dimension Definition

The Swath Dimension Definition contains one or more Dimension Definitions. Each Dimension Definition is enclosed by the **OBJECT = Dimension** statement and its corresponding **END_OBJECT** statement and it contains the name and the size of the dimension.

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

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

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

The Swath Datafield Definition contains one or more Datafield Definitions. Each Datafield Definition is enclosed by the **OBJECT = Datafield** statement and its corresponding **END_OBJECT** statement and it contains the name and data type of the Datafield, and a list of dimensions on which the Datafield is defined. It may also contain the optional Swath Field Storage Definition.

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

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

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>.
`<Swath Field Storage Definition>` is optional.

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

The Swath Geofield Definition contains one or more Geofield Definitions. Each Geofield Definition is enclosed by the **OBJECT = Geofield** statement and its corresponding **END_OBJECT** statement and it contains the name, data type of the

Geofield, and a list of dimensions on which the Geofield is defined. It may also contain the optional Swath Field Storage Definition.

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

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

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 optional. Its 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

The Swath Dimension Mapping Definition contains one or more Dimension Mapping Definitions. Dimension mappings are categorized as either *regular mapping* or *index mapping*.

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

Requirements:

The Mapping Definitions can be in any order.

3.6.1 Regular Dimension Mapping Definition

A regular dimension mapping is enclosed by the **OBJECT = DimensionMap** statement and its corresponding **END_OBJECT** statement. It contains a name, a Datafield Dimension, a Geofield Dimension, an Offset and an Increment.

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

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

A Grid object is enclosed in the **OBJECT = Grid** statement and its corresponding **END_OBJECT** statement. Each object contains the name of the Grid and three main components as follows,

Grid Parameters Definition
Grid Dimension Definition
Grid Datafield Definition

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

Requirements:

`<GridName>` can be any legal quoted name but must be unique among all HCR Object names.
`<Grid Parameters Definition>` is required.
`<Grid Dimension Definition>` is required.
`<Grid Datafield Definition>` is required.

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>
```

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

The Grid Dimension Definition contains one or more Dimension Definitions. Each Dimension Definition is enclosed by the **OBJECT = Dimension** statement and its corresponding **END_OBJECT** statement and it contains the name and the size of the dimension.

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

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

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

The Grid Datafield Definition contains one or more Datafield Definitions. Each Datafield Definition is enclosed by the **OBJECT = Datafield** statement and its corresponding **END_OBJECT** statement and it contains the name and data type of the Datafield, and a list of dimensions on which the Datafield is defined. It may also contain the optional Swath Field Storage Definition.

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

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

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**.

<Grid Field Storage Definition> is optional.

4.4.1 Grid Field Storage Definition

The Grid Field Storage Definition contains one of Field Merge Definition, Field Compression Definition or Field Tile Definition.

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

4.4.1.1 Field Merge Definition

The Field Merge Definition is defined as

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

Requirements:

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

4.4.1.2 Field Compression Definition

The Field Compression Definition is defined as

```
<Field Compression Definition> ::=
     CompressionType = <CompressionCode>
     CompressionParameters = (<CompParamList>)
```

Requirements:

`<CompressionCode>` is required and 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

The Field Tile Definition is defined as

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

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.,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

A Point object is enclosed in the **OBJECT = Point** statement and its corresponding **END_OBJECT** statement. Each object contains the name of the Point and two main components as follows,

Point Table Definition
Point Linking Definition

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

Requirements:

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

`<Point Table Definition>` is required.

`<Point LevelLink Definition>` is optional.

5.2 Point Table Definition

The Point Table Definition contains one or more Table Definitions. Each Table Definition is enclosed by the **OBJECT = Level** statement and its corresponding **END_OBJECT** statement and it contains the name of the table and one or more Point Field Definitions. (A Point Table is also known as a Point Level.)

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

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

Requirements:

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

5.3 Point Field Definition

The Point Field Definition is enclosed by the **OBJECT = PointField** statement and its corresponding **END_OBJECT** statement and contains the name, data type and order[1] of the Field.

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

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.

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.[2] A point object is a Simple Point object unless defined otherwise by the Point LevelLink Definition.

The Point LevelLink Definition contains one or more Linked Field Definitions. Each Linked Field Definition is enclosed by the **OBJECT = LevelLink** statement and its corresponding **END_OBJECT** statement and it contains the definitions of Parent table, Child table and the Linked field.

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

<Linked Field Definition> ::=
     OBJECT = LevelLink
```

[1] *Order* as defined in the HDF Vdata.

[2] Indexed field point objects are not supported.

```
        Parent = <ParentTableName>
        Child  = <ChildTableName>
        LinkField = <FieldName>
    END_OBJECT = LevelLink
```

Requirement:
<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"
            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

```
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
```

7. Related Documents

7.1 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.

7.2 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

7.3 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

7.4 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

7.5 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