Consultative Committee for Space Data Systems

RECOMMENDATION FOR SPACE
DATA SYSTEM STANDARDS

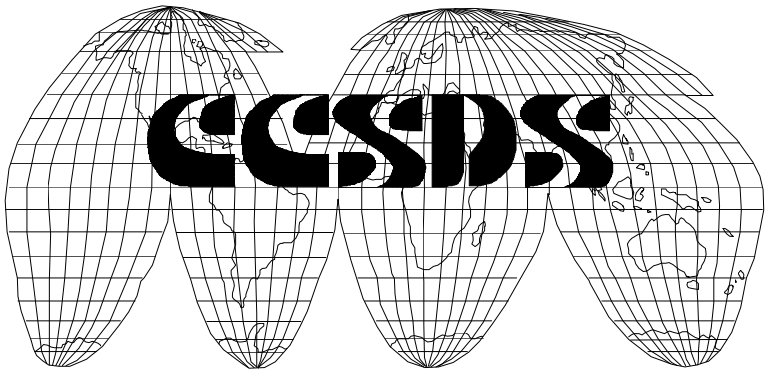
Data Entity Dictionary Specification Language (DEDSL)

(ZCSD0011/ZCSD0012)

CCSDS 6xx.0-W-0.6

WHITE BOOK

July 1996



AUTHORITY

Issue:	White Book, Issue 0.6
Date:	24 July 1996
Location:	TBD

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorisation of CCSDS Recommendations is detailed in reference [14], and the record of Agency participation in the authorisation of this document can be obtained from the CCSDS Secretariat at the address below.

This Recommendation is published and maintained by:

CCSDS Secretariat
Program Integration Division (Code-OI)
National Aeronautics and Space Administration
Washington, DC 20546, USA

STATEMENT OF INTENT

The Consultative Committee for Space Data Systems (CCSDS) is an organisation officially established by the management of the member space Agencies. The committee meets periodically to address data system problems that are common to all participants and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of the committee are termed **Recommendations** and are not considered binding on any Agency.

This **Recommendation** is issued by, and represents the consensus of, the CCSDS Plenary body. Agency endorsement of the **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever an Agency establishes a CCSDS-related **standard**, this **standard** will be in accordance with the relevant **Recommendation**. establishing such a **standard** does not preclude other provisions which an Agency may develop.
- Whenever an Agency establishes a CCSDS-related **standard**, the Agency will provide other CCSDS member Agencies with the following information:
  - The **standard** itself.
  - The anticipated date of initial operational capability.
  - The anticipated duration of operational service.
- Specific service arrangements shall be made via memorandum of agreement. Neither this **Recommendation** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommendation** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or (3) be retired or cancelled.

In those instances when a new version of a **Recommendation** is issued, existing CCSDS-related Agency **standards** and implementation are not negated or deemed to be non-CCSDS compliant. It is the responsibility of each Agency to determine when such **standards** or implementation are to be modified. Each Agency is, however, strongly encouraged to direct planning of its new **standards** and implementations towards the later versions of the **Recommendation**.

FOREWORD

This Recommendation is a technical Recommendation that provides a model and language to increase the standardisation of the expression of simple semantic concepts that are to be carried with data. These semantic concepts are given standard names and a standard way of expressing them is also provided. The semantic information may be conveyed either in a computer processable manner or via conventional (e.g. paper) documentation.

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This document is therefore subject to CCSDS document management and change control procedures which are defined in reference [14].

Questions relating to the contents or status of this document should be addressed to the CCSDS Secretariat at the address indicated on page ii.

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- British National Space Centre (BNSC)/United Kingdom.
- Canadian Space Agency (CSA)/Canada.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centre National d'Etudes Spatiales (CNES)/France.
- Deutsche Forschungsanstalt für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- National Aeronautics and Space Administration (NASA HQ)/USA.
- National Space Development Agency of Japan (NASDA)/Japan.

Observer Agencies

- Australian Space Office (ASO)/Australia.
- Austrian Space Agency (ASA)/Austria.
- Belgian Science Policy Office (SPO)/Belgium.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Space Technology (CAST)/China.
- Communications Research Laboratory (CRL)/Japan.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organisation for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Industry Canada/Communications Research Center (CRC)/Canada.
- Institute of Space and Astronautical Science (ISAS)/Japan.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Ministry of Communications (MOC)/Israel.
- National Oceanic & Atmospheric Administration (NOAA)/USA.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

DOCUMENT CONTROL

Document	Title	Date	Status/ Changes
CCSDS 6xx.0-W-0.1	Recommendation Space Data System Standards: Semantic Description Language - Specification	23 June 1995	Issue 0.1
CCSDS 6xx.0-W-0.2	Recommendation Space Data System Standards:- - Data Entity Dictionary Specification Language (DEDSL)	23 August 1995	Issue 0.2
CCSDS 6xx.0-W-0.3	Recommendation Space Data System Standards:- - Data Entity Dictionary Specification Language (DEDSL)	11 October 1995	Issue 0.3
CCSDS 6xx.0-W-0.4	Recommendation Space Data System Standards:- - Data Entity Dictionary Specification Language (DEDSL)	19 December 1995	Issue 0.4
CCSDS 6xx.0-W-0.5	Recommendation Space Data System Standards:- - Data Entity Dictionary Specification Language (DEDSL)	26 June 1996	Issue 0.5
CCSDS 6xx.0-W-0.6	Recommendation Space Data System Standards:- - Data Entity Dictionary Specification Language (DEDSL)	24 July 1996	Issue 0.6

CONTENTS

Sections **Page**

1 INTRODUCTION 1
1.1 Purpose And Scope 1
1.2 Applicability 1
1.3 Rational 2
1.4 Document Structure 2
1.5 Definitions 3
1.5.1 Glossary of Terms 3
1.5.2 Nomenclature 4
1.5.3 Conventions 5
1.6 References 6

2 OVERVIEW 8
2.1 Uses of Data Entity Dictionaries 8
2.2 Application of the DEDSL 10

3 ABSTRACT STANDARD 12
3.1 Standard Attributes 12
3.1.1 Data Entity Identification: NAME 12
3.1.2 Full Textual Description of a Data Entity: MEANING 13
3.1.3 Short Textual Description of a Data Entity: SHORT_MEANING 13
3.1.4 Scientific Units of a Data Entity: UNITS 14
3.1.5 Syntax of the Value of a Data Entity: VALUE_SYNTAX 15
3.1.6 Alternate Identification of a Data Entity: ALIAS 17
3.1.7 Specific Meaning of an Instance of a Data Entity: SPECIFIC_INSTANCE 17
3.1.8 Associated Information about a Data Entity: COMMENT 18
3.2 User Defined Attributes 19
3.2.1 Identification of an Attribute: ATTRIBUTE_NAME 19
3.2.2 Full Textual Description of an Attribute: ATTRIBUTE_MEANING 20
3.2.3 Syntax of an Attribute Value: ATTRIBUTE_VALUE_SYNTAX 20
3.2.4 Occurrences of an Attribute: ATTRIBUTE_OCCURANCE 22
3.2.5 Example of a Value of an Attribute: ATTRIBUTE_VALUE_EXAMPLE 23
3.2.6 Associated Information about an Attribute: ATTRIBUTE_COMMENT 23
3.2.7 Registration of User Defined Attributes 24
3.3 DEDSL Module Identification Attributes 24
3.3.1 DEDSL Version Information: DEDSL_VERSION 24
3.3.2 DEDSL Module Identification: MODULE_TITLE 25
3.3.3 DEDSL Module ADID: MODULE_ADID 25
3.4 Implementation Guidelines 26

4 IMPLEMENTATION USING PVL 28
4.1 Complete DEDSL Definition 28
4.2 Standard Attributes 29
4.2.1 Implementation of Standard Attributes 29
4.2.2 Aggregation of a Data Entity Definition 31
4.3 User Defined Attributes 33
4.3.1 Implementation of User Defined Attributes 33
4.3.2 Aggregation of User Defined Attribute Parameters 34

4.4 DEDSL Module Identification Attributes 36
4.4.1 Implementation of DEDSL Module Identification Attributes 36
4.4.2 Aggregation of DEDSL Module Identification Attributes 37

5 AN EXAMPLE IN THE USE OF THE DEDSL 39
5.1 Use of the DEDSL for a Discipline DED 39
5.2 Use of THE DEDSL for Product DEDs 41
5.2.1 Use of the DEDSL for Product X 41
5.2.2 Use of the DEDSL for Product Y 43

6 DEDSL CONFORMANCE 46
6.1 Conformance Level 1: Abstract DEDSL (ADID = ZCSD0011) 46
6.2 Conformance Level 2: Complete DEDSL (ADID = ZCSD0012) 46

ANNEX A: SCENARIOS OF DEDSL USAGE 47
Annex A.1: Using the DEDSL with a Formal Data Description Language 47
Annex A.2: Using the DEDSL with FITS 49
Annex A.3: Using the DEDSL with CDF 54
Annex A.4: Using the DEDSL with HDF 60

ANNEX B: PERMITTED UNIT VALUES 62

ANNEX C: UNIDATA CONFIGURATION FILE 65

ANNEX D: ASCII & RESTRICTED ASCII 67

Figures **Page**

Figure 1-1: Example Structure Diagram 5
Figure 2-1: Organisation of Two Related Data Products 9
Figure 4-1: Structure Diagram of a Complete DEDSL Description 29
Figure 4-2: Structure Diagram of a Single DEDSL Entity Definition 32
Figure 4-3: Structure Diagram of a Single DEDSL User Defined Attribute Definition 35
Figure 4-4: Structure Diagram of the DEDSL Module Identification Attributes 37
Figure A-1: DEDSL Usage with a HDF File 61

Tables **Page**

Table 2-1: Comparison of Discipline DED and Product DED Descriptions 10
Table B-1: Representation of Unit Symbols 63
Table B-2: Representation of Unit Prefixes 64

1 INTRODUCTION

1.1 Purpose And Scope

The purpose of this Recommendation is to define a language for specifying a dictionary which describes the semantics of data entities - it does not define a specific dictionary.

A dictionary is understood as a mechanism that is able to organise a set of information in a consistent and easily understood manner, and it is commonly used by humans to look up the meaning of words used in natural languages. Similarly, a data entity dictionary is used by humans and systems to look up the general meaning, and other attributes, of data entities used in the definition and generation of data products.

This Recommendation defines the concepts of name, meaning, units, and a small set of other attributes so they may be used consistently in the formation of data entity dictionaries. A method is also provided that permits the set of attributes to be extended beyond the standard ones provided within this Recommendation. Given the wide variety of data entities that may need to be described, only a few of the attributes are made mandatory by this Recommendation.

The attributes defined in this Recommendation do not provide a means for describing relationships between data entities or the physical representation of data entities.

This Recommendation also provides a standard method to represent the attributes and their values, using the Parameter Value Language (PVL, see references [5] and [6]), for the construction and interchange of data dictionaries.

1.2 Applicability

This Recommendation is intended to be used, for example,:

- By data producers to construct dictionaries that describe, in a more formal manner, data entities within their data products;
- By data users to understand data received from data producers that have used this Recommendation to construct their dictionaries.
- By an organisation that mandates the attributes used to define each entity description in dictionaries used within that organisation.
- By a particular community, such as Earth observation, space physics, archives, etc., to establish a degree of standardisation for the contents of any data dictionary associated with a data product. This would be done by using this Recommendation to define a community-wide data dictionary.
- By organisations and communities to exchange the contents of a data dictionary in a standardised manner, i.e., to facilitate interoperability.

1.3 Rational

The same data entity can be physically represented in many different formats, including within generic formats such as HDF (see reference [15]) and CDF (see reference [16]), or in native formats. However there is information about the data entities, such as their general meaning and other semantic attributes, which are independent of the physical representation.

Part of this information can be expressed in a data dictionary and this dictionary can be expressed in many different ways. For example, it may be expressed in natural language paragraphs within a document that accompanies a data product. It may be partially expressed by attributes defined within generic or native data product formats, while the rest may be in other documents. The concepts used in the description of the data entities may vary widely, or subtly. These concepts may not be documented at all.

Therefore, individuals and organisations that need to receive and understand a variety of data products may expend considerable effort in attempting to understand the data entities that comprise each data product. This also greatly hinders the use of generic tools that can assist in the recognition and presentation of this information in a way that various individuals and organisations find most understandable.

To begin to address these issues across broad organisation and community disciplines, it is first necessary to define a set of standard concepts that can be used in the formation of data dictionaries within those disciplines, and in the mapping of different dictionary concepts between disciplines. To facilitate the creation of generic tools, it is also necessary to define a standard representation for the standard concepts. This Recommendation defines a small number of concepts, in terms of attributes, that are intended to be broadly applicable. It also defines an optional representation for expressing these attributes.

Later Recommendations may build on this Recommendation in order to provide additional standardised functionality. For example, they may support the expression of the relationships between entities in a dictionary and definitions of interrelationships between dictionaries.

1.4 Document Structure

This document presents the specification in a layered manner. Firstly the abstract definition of the semantic information that is required to be conveyed is defined. This is done so that the actual technique used to convey the information is independent of the information content and therefore the same abstract standard can be used within different formatting environments. This also permits the semantic information to be translated to different representations as may be needed when data are transferred across different domains.

Secondly a standard technique for conveying the information using the CCSDS developed Parameter Value Language (PVL, see references [5] and [6]) is specified. This is the recommended method to use to convey semantic information between a data generator and end user. Finally a number of scenarios are presented that show how this DEDSL specification should be used and how it can be interfaced to both conventional fixed format data and flexible formatting techniques.

In summary, the document is structured as follows:

- Section 2 provides an overview of the data entity dictionary concept and describes, through examples, how this Recommendation may be used. It also provides a context within which to understand the standards specified in Sections 3 and 4.

- Section 3 specifies the abstract semantic description technique, including what information must be conveyed and when it is applicable.
- Section 4 specifies the PVL implementation of the abstract standard; this gives a concrete syntax to the information that is specified in Section 3 and conventions to be used.
- Section 5 continues the examples presented in Section 2, by providing full data entity dictionary listings of those examples, in accordance with the specification in Sections 3 and 4.
- Section 6 discusses the levels of conformance to the DEDSL Recommendation, in relation to the abstract specification and the PVL implementation, and the CCSDS Control Authority registration of this Recommendation.
- Annex A provides a number of scenarios that describe the usage of this recommendation, what issues can arise and compatibility with existing formatting techniques.
- Annex B provides a list of the terms to be used for units when describing data entities and their representation.
- Annex C provides the configuration file for the NCAR UNIDATA units processing package, set up to conform to the units specification provided in Annex B.
- Annex D defines the ASCII character set that is used in the representation of the DEDSL along with the subset of ASCII that conforms to the Restricted ASCII character set as used within this Recommendation.

1.5 Definitions

1.5.1 Glossary of Terms

For the purpose of this document the following definitions apply:

Attribute	An attribute is a piece of information that describes another piece of data; this information characterises or enhances understanding of the data that is being described. Attributes are the primary focus of this Recommendation in that they are used to defined the semantics of data.
Data entity	A data entity is the piece of data that is to be described by using attributes and the values of the attributes. A data entity can be of a simple type, such as integer or real; it can also be a compound type consisting of a structure of various other simple and compound entities.
Data entity dictionary	A data entity dictionary is a listing of a number of semantic definitions of various data entities. Each data entity described in the dictionary is described in terms of attributes and attribute values. Data entity dictionaries may be just for a single product, i.e. all the data entities within a single product are described in a corresponding single dictionary, or the data entity dictionary may be a discipline oriented dictionary that holds a number of previously defined data entity definitions which may be used by data definers and users for reference.
Data type	In the context of this document, data type is an indication of the conceptual

	scientific form of the data entity, such as integer, real, string, sequence of entities, etc.
DEDSL module	The DEDSL module is the combination of the DEDSL identification information, which provides information about the complete DEDSL module, the definition of any user defined attributes and the definition of the actual data entities. Various parts of a DEDSL module are optional. In practical terms the DEDSL module could be either a file, SFDU LVO value field (see references [3] and [4]), etc.
Quoted string	A quoted string is a string of characters that are enclosed within quotation marks. These may be either single or double quotation marks (although the same at each end of the string). White space characters are permitted within the quoted string (see references [5] and [6] for an exact definition).
Semantics	Semantics are the information that describes the meaning of data rather than the physical representation of that data. Semantics potentially cover a very large domain, from the simple such as the units of a data entity to the complex, such as the relationship of a data entity to another.
Standard attribute	A standard attribute is one that is defined here within the Recommendation. When the DEDSL is used there are standard attributes that are mandatory and others which are optional, but the same attribute name cannot be redefined if the semantic information is to be in conformance with this Recommendation.
Syntax	Syntax is the definition of the physical representation of data. It includes the structural arrangement of the fields within data, the physical hardware representation and results in a clear understanding of the abstract values of the data.
Unquoted string	An unquoted string is a string of characters that do not contain any white space characters and hence is delimited on either side by any white space characters, e.g. **`NAME,`** (see references [5] and [6] for an exact definition).
User defined attribute	A user defined attribute is an attribute that is defined by a particular user or project and after definition is then used in the same manner as a ‘standard attribute’. This Recommendation specifies a standard method of defining user defined attributes so as to produce unambiguous definitions and encourage reuse of user defined attributes.
White space	White space is defined to consist of the ASCII characters line feed ($0A_{hex}$), carriage return ($0D_{hex}$), horizontal tab (09_{hex}), vertical tab ($0B_{hex}$), form feed ($0C_{hex}$) and space (20_{hex}).

1.5.2 Nomenclature

The following conventions apply throughout this Recommendation:

a) The words “shall” and “must” imply binding and verifiable specification;

b) The word “should” implies an optional, but desirable, specification;

c) The word “may” implies and optional specification;

d) The words “is”, “are” and “will” imply statements of fact.

1.5.3 Conventions

This document uses structure diagrams to illustrate the structure of the DEDSL constructs. Components of a construct are called elements. The following conventions are used:

- The element named on the left of the ::= symbol is the element being defined.
- The diagram on the right of the ::= symbol is the corresponding definition.
- Elements that are presented in bold characters in a rounded box are PVL statements that are defined elsewhere in the recommendation.
- Elements that are presented in italic characters in a square box are elements that are further defined in another structure diagram.
- Elements that are represented as a single bold character in a circle are actual characters that shall appear in the syntax.
- A vertical branch represents a choice.
- A repetition is indicated by a loop-back covering the object to be repeated.

The following example presents a diagram specifying the declaration of **'Element'**. **'Element'** is defined as first **'Another element'**; this element can be repeated any number of times due to the loop back over the element (i.e. one or more '**Another element**'s must be included). **'Another element'** would be defined in another structure diagram. Following **'Another element'**, there is a choice between **'STATEMENT_1='**, **'STATEMENT_2='** or nothing. **'STATEMENT_1='** and **'STATEMENT_2='** both represent PVL statements specified elsewhere in the document. Following this there is **'STATEMENT_3='**, which represents another PVL statement specified elsewhere in the document, and finally '**STATEMENT_4='** is another PVL statement that can either be passed by altogether or included any number of times by virtue of the loop-back. (i.e. zero or more '**STATEMENT_4=**'s).

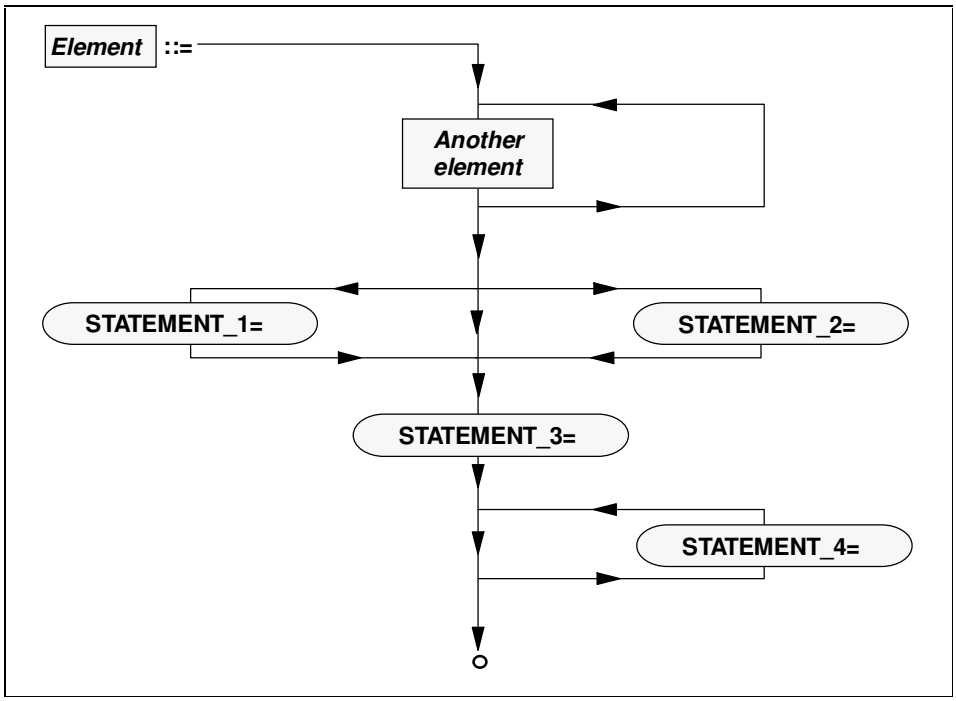


Figure 1-1: Example Structure Diagram

This document uses examples to illustrate the DEDSL. All examples of the DEDSL are shown in a bold, fixed spacing font.

1.6 References

The following documents contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Recommendation are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently available CCSDS Recommendations.

[1] ***The Data Description Language EAST -- Specification (CCSD0010),*** Draft Recommendation for Space Data System Standards, CCSDS 644.0-R-1, Red Book, Issue 1, CCSDS, November 1995.

[2] ***The Data Description Language EAST -- A Tutorial,*** Draft Report Concerning Space Data System Standards, CCSDS 645.0-G-0.1, Draft Green Book, Issue 0.1, CCSDS, November 1995.

[3] ***Standard Formatted Data Units - Structure and Construction Rules***, Recommendation for Space Data System Standards, CCSDS 620.0-B-2, Blue Book, Issue 2, CCSDS, May 1992.

[4] ***Standard Formatted Data Units - A Tutorial***, Report Concerning Space Data System Standards, CCSDS 621.0-G-1, Green Book, Issue 1, CCSDS, May 1992.

[5] ***Parameter Value Language Specification (CCSD0006),*** Recommendation for Space Data System Standards, CCSDS 641.0-B-1, Blue Book, Issue 1, CCSDS, May 1992.

[6] ***Parameter Value Language - A Tutorial,*** Report Concerning Space Data System Standards, CCSDS 641.0-G-1, Green Book, Issue 1, CCSDS, May 1992.

[7] ***ASCII Encoded English (CCSD0002)***, Recommendation for Space Data System Standards, CCSDS 643.0-B-1, Blue Book, Issue 1, CCSDS, November 1992.

[8] ***Information Processing - Representation of numerical values in character strings for information interchange***, ISO 6093-1985.

[9] ***Standard Formatted data Units -- Control Authority Procedures,*** Recommendation for Space Data System Standards, CCSDS 630.0-B-1, Blue Book, Issue 1, CCSDS, June 1993.

[10] ***Standard Formatted data Units -- Control Authority Procedures Tutorial,*** Report Concerning Space Data System Standards, CCSDS 631.0-G-2, Green Book, Issue 2, CCSDS, November 1994.

[11] ***UNIDATA Units Package,*** NCAR, Version 1.10.2, 19 October 1995, (http://www.unidata.ucar.edu/packages/udunits/index.html).

[12] ***Information Technology - Open systems Interconnection - Specification of Abstract Syntax Notation.One (ASN.1),*** ISO/IEC 8824, 2nd issue, 15 Dec 1990.

[13] ***Time Code Formats,*** Recommendation for Space Data System Standards, CCSDS 301.0-B-2, Blue Book, Issue 2, CCSDS, April 1990.

[14] ***Procedures manual for the Consultative Committee for Space Data Systems***, CCSDS A00.0-Y-6, Yellow Book, Issue 6, CCSDS, May 1994.

[15] ***Hierarchical Data Format (HDF)***, Version 4.0r1, http://hdf.ncsa.uiuc.edu, National Centre for Supercomputing Applications (NCSA)

[16] ***Common Data Format (CDF)***, Version 2.5.19a, http://nssdc.gsfc.nasa.gov/cdf/cdf_home .html, May 17 1996.

[17] ***Information Processing - Representation of SI and other units in systems with limited character sets***, ISO 2955-1983.

2 OVERVIEW

As discussed in the purpose and scope, a data entity dictionary is used by humans and systems to look up the general meaning, and other attributes, of data entities used in the definition and generation of data products. This section discusses the some of the primary uses of Data Entity Dictionaries (DEDs) and presents a basic example of DEDSL usage which will be used throughout this book. This example will also be used as the basis for Annex A of this Recommendation which presents scenarios on how DEDSL can be used in conjunction with popular science data format standards to more fully and consistently document a data product.

2.1 Uses of Data Entity Dictionaries

This Recommendation formally defines the concepts of name, meaning, units, and a small set of other attributes so they may be used consistently in the formation of data entity dictionaries. A method is also provided that permits the set of attributes to be extended beyond the standard ones provided within this Recommendation. It is anticipated that there would be two major types of DEDs, product DEDs and community (i.e., global, project or discipline) DEDs.

A product DED is used in conjunction with a formatting standard (e.g., FITS), a self describing format (e.g., CDF, HDF, etc.) or a Data Definition Language (e.g. EAST), to provide a formal definition of additional semantics associated with a specific data product. This formal definition would enable the definition of generic tools to assist producers in creating products documented to this level and to assist consumers in understanding products they receive. This version of the DEDSL focuses primarily on developing standard names and representations for the concepts required for product DEDs.

Some of the critical issues in the description of data products are associated with the assignment of names and linking the names specified in the DED to the names specified in the self describing format or data format description record. There are two conflicting requirements that affect the assignment of names to entities within data products. The first is to allow a name to translate into a unique location within the data to obtain the correct value, while the second is to use the same name in different data products when referring to the same real world concept. This leads to the concept of hierarchical naming where the complete name of a data entity includes the names of all the entities it is contained within. For example, in **`PRODUCT_X`** in Figure 2-1, the entity **`LATITUDE`** is contained within the entity **`CENTRE_COOR`** within the entity **`HEADER`**. The full name for this entity is **`HEADER.CENTRE_COORD.LATITUDE`** (note that the period is a convention adopted within this recommendation to indicate the hierarchical relationship of ‘contains’).

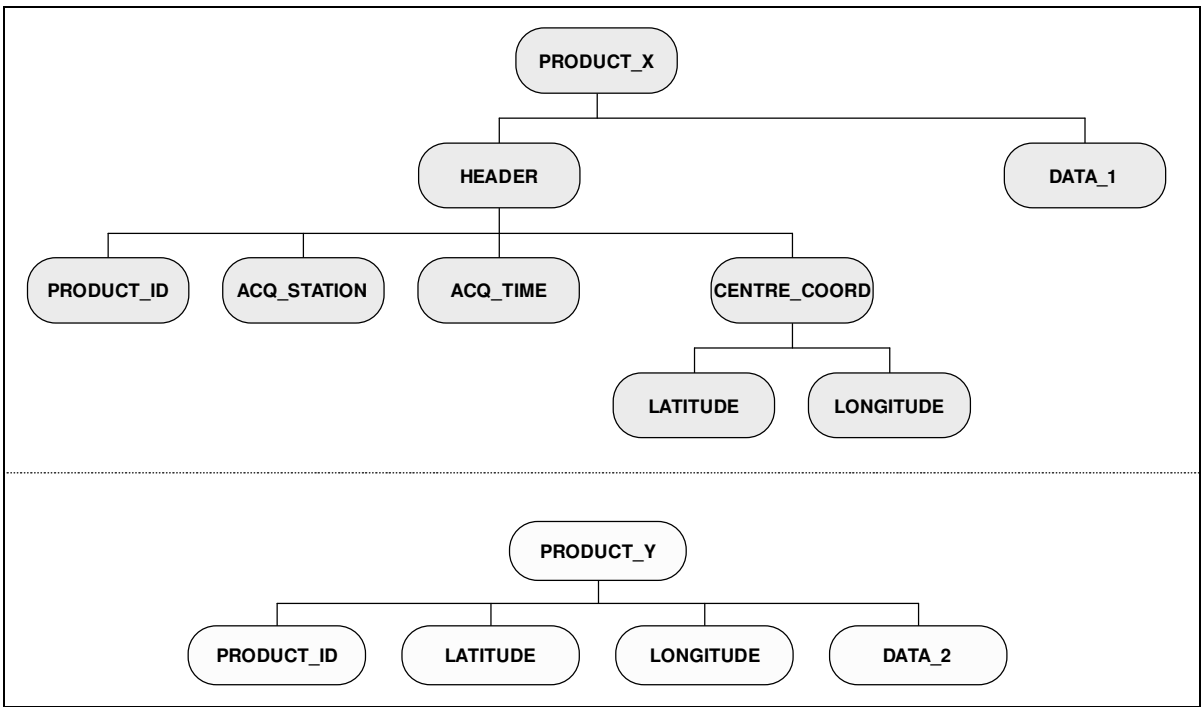


Figure 2-1: Organisation of Two Related Data Products

The techniques discussed in Sections 3 and 4 of this Recommendation can be applied to any named data entity so a project or data designer can associate attributes with that entity. Using the example above, DEDSL attributes used for **`LATITUDE`** should apply to any entity whose complete name contains the named element **`LATITUDE`** at its lowest hierarchical level (i.e. after the last period). This capability is in addition to the more common usage of this recommendation which is the description of data entities which have unique, complete, names. It should be noted that not all product description techniques (i.e., standard formats, data description languages and self describing formats) have the capability to support a hierarchical naming convention. If the product description technique does not support this, alternative names must be created to contain the combined definitions of the equivalent hierarchical levels. An example of this is show in the CDF scenario in Annex A.

A community DED can be used by an organisation to gain some degree of control/standardisation over the data descriptions created by member data producers. Unlike product dictionaries, these community DEDs are not used in conjunction with a specific product description technique. They are independent of the specific implementation of products. Examples of uses of community DEDs include:

- The creation of a standard data entity dictionary by an organisation that mandates the attributes to be used to define each entity description in dictionaries used within that organisation.
- The creation of a community DED by a particular community (e.g., planetary science, astrophysics etc.), to establish a degree of standardisation for the contents of any data dictionary associated with a data product from that community.

This recommendation provides a foundation for the creation of community DEDs by providing a base set of concepts for data entity description but does not provide the formal methods to describe relationships among data entities in a single DED or among multiple DEDs. These

concepts may be described in future recommendations. The example shown in the next section, and further illustrated in Annex A, demonstrates how product and community DED can be built using this recommendation and assumptions about the relationships among named entities or DEDs.

2.2 Application of the DEDSL

The two main application areas for the DEDSL are the following:

- To build up and define a community (i.e. discipline) DED;
- To describe a data product.

In order to highlight the difference between those application areas, Table 2-1 shows the mapping of discipline DED entries into data product DED entries for the data products shown in Figure 2-1.

Entity Name	Data Type	Discipline DED	PRODUCT_X Description	PRODUCT_Y Description
HEADER	Structure	no	yes	no
PRODUCT_ID	String	yes	yes	yes
ACQ_STATION	Enumeration	yes	yes	no
ACQ_TIME	Time	yes	yes	no
CENTRE_COORD	Structure	no	yes	no
LATITUDE	Real	yes	yes	yes
LONGITUDE	Real	yes	yes	yes
DATA 1	Array of 16-bit integers	no	yes	no
DATA 2	Array of real numbers	no	no	yes

Table 2-1: Comparison of Discipline DED and Product DED Descriptions

The discipline dictionary includes as entities: **PRODUCT_ID**, **ACQ_STATION** and **ACQ_TIME**. For each of the entities a DEDSL description is given (see Section 5.1). The purpose of a discipline dictionary is to provide, across different projects, a standard definition of parameters.

When a project defines a data product (**PRODUCT_X** in Figure 2-1), the discipline dictionary is consulted, and the **HEADER** entity is defined on the basis of **PRODUCT_ID**, **ACQ_STATION** and **ACQ_TIME**, which are obtained from the discipline dictionary (see Section 5.2.1).

Note that the DEDSL description of the entities in the data product is independent of the format (e.g. HDF, netCDF, etc.) in which the data product is represented.

In the other product in Figure 2-1, **PRODUCT_Y,** the **PRODUCT_ID**, **LATITUDE** and **LONGITUDE** entities from the discipline dictionary are used as well (see Section 5.2.2).

To define the semantics of data entities a number of parameters are required, such as the name of the entity, the meaning of the entity, the units for the entity, etc. The following sections of

this Recommendation provide the formal definition of the DEDSL, which is designed for such purposes. Following these specifications, Section 5 presents the full DEDSL definition of the discipline dictionary and both Product X and Y dictionaries discussed above. The reader is recommended to consult Section 5 when reading the specifications in the following sections for complete illustrative examples.

3 ABSTRACT STANDARD

The semantic information required to describe a data entity is seen as a collection of attributes. Each attribute describes a particular semantic characteristic of the data entity.

In the following sub-sections, the collection of attributes available to describe the basic semantics of a data entity are identified. These attributes fall into two categories; 'standard attributes' that are defined by this recommendation and 'user defined attributes' that are defined by the user in a standardised way, but which can then be used in the same manner as 'standard attributes'.

In order to facilitate the readability of attributes and support cross-comparison of data, each attribute requires a standard identifier as well as standard conventions for the specification of the attribute's value.

3.1 Standard Attributes

3.1.1 Data Entity Identification: NAME

Purpose This attribute may be used to identify the data entity that is being described. The value of this attribute is an identifier that may be used to link a collection of attributes with an equivalent identifier in, or associated with, the data entity.

The value of this attribute may also be used by the software developer to name corresponding variables in software code or as a field to be searched on for location of particular data entities.

Term to be used The standard term to be used for this attribute is: **`NAME`**

Occurrence This attribute is mandatory and can only appear once for each data entity described.

Convention for the attribute value For the value of this attribute the following conventions must be followed:

- it shall be a free format character string;
- all characters must be in a visibly displayable form, i.e. white space is not permitted;
- the value must to be unique within the DEDSL module (see Section 4.1);
- the value is case sensitive. *Note, whilst this means that two values of NAME may only differ by case of certain letters, this practice is discouraged.*
- when identifying entities within a hierarchical data structure, the commonly used 'dot' notation should be used whenever possible. *For example,* **`impQ_test.Epoch`** *or* **`impQ_test.RMS.`**
- the maximum length of the value of this attribute is 400 characters.

Example attribute value **ACQ_STATION**

3.1.2 Full Textual Description of a Data Entity: MEANING

Purpose Required to give the description and meaning of the data entity identified by the **NAME** attribute. This attribute is intended for human readership and therefore any information that will increase the understanding of the identified data entity should be included.

It is intended that the value of this attribute can be of significant length and hence provide as complete a description of the data entity as possible. The value of this attribute can be used as a field to be searched on for the location of particular data entities.

The value of this attribute may include, in natural language, the same semantic information that is carried in a more formal manner by other attributes. This is neither a requirement or illegal, but the user must make sure inconsistencies do not arise.

Term to be used The standard term to be used for this attribute is: **MEANING**

Occurrence This attribute is mandatory and can only appear once for each data entity described.

Convention for the attribute value For this attribute the following conventions must be followed for its value:

- it shall be a free format character string;
- the maximum length of the value of this attribute is 8000 characters.

Example attribute value **The ACQ_STATION includes the identifier of the station, which has acquired the data.**

3.1.3 Short Textual Description of a Data Entity: SHORT_MEANING

Purpose The value of this attribute provides a short concise description and meaning of the data entity identified by the **NAME** attribute. This attribute provides a summary of the more detailed information provided by the **MEANING** attribute

The value of this attribute can be used as a field to be searched on for the location of particular data entities. It is also intended to be used for display purposes by automated software, where the complete **MEANING** value would be too long to present in a convenient manner to the user.

Term to be used The standard term to be used for this attribute is: **SHORT_MEANING**

Occurrence This attribute is optional, but if provided it can only appear once for each data entity described.

Convention for the attribute value For this attribute the following conventions must be followed for its value:

- it shall be a free format character string;

- the maximum length of the value of this attribute is 80 characters.

Example attribute value `Identifier of acquisition station`

3.1.4 Scientific Units of a Data Entity: UNITS

Purpose The value of this attribute specifies the scientific units that should be associated with the value of the data entity so as to make the value meaningful in the 'real-world'.

Term to be used The standard term to be used for this attribute is: **`UNITS`**

Occurrence If the data entity that is being defined is of a scientific scalar type then this attribute is mandatory and may appear only once. If the data entity is non-scalar then the attribute shall not be specified.

Convention for the attribute value For this attribute the following conventions must be followed for its value:

- it shall be a free format character string;
- the maximum length of the value of this attribute is 80 characters.

The contents of the character string must conform to the representation specified in ISO 2955 (see Ref. [17]). The units and prefixes supported (as specified by ISO 2955) are duplicated for convenience in Annex B of this Recommendation. As detailed in ISO 2955, the following convention apply when combining units:

- Multiplication shall be indicated by a period (.), e.g. **`Pa.s`** to designate Pascal second, the unit of dynamic viscosity
- Division shall be indicated wither by a solidus (/), e.g. **`m/s`**, or by expressing the denominator with a negative exponent, e.g. **`m.s-1`**
- Positive exponents shall be indicated by following the unit directly with the numeric power with no sign, e.g. **`m2`** to designate m^2
- Negative exponents shall be indicated by following the unit directly with the numeric power preceded by a minus sign, e.g. **`m-3`** to designate m^{-3}
- Decimal multiples of units shall be indicated by the combination of a prefix representation (see Table B-2) immediately before the unit, e.g. **`kN`** to represent kilo Newtons

(Note: Annex C contains the configuration file for the NCAR Unidata units processing software, set up to handle the ISO 2955 units specification only. The aim of promoting software of this type is so that automated software can be used to process and convert units. This permits users to readily process data from different sources when supplied with different units.)

Example attribute values `Cel`

3.1.5 Syntax of the Value of a Data Entity: VALUE_SYNTAX

Purpose The value of this attribute specifies the syntax of the values of this data entity. In this context, syntax is defined to mean the conceptual data type and permitted values of the entity, it is not intended for specifying the physical representation of the entity. For example, an entity may be defined as having a **`VALUE_SYNTAX`** of **`INTEGER`**; physically this maybe encoded as an IEEE 16-bit 2's complement binary number or as ASCII encoded decimal, but in both bases the **`VALUE_SYNTAX`** would be **`INTEGER`**.

Term to be used The standard term to be used for this attribute is: **`VALUE_SYNTAX`**

Occurrence This attribute is optional, but if provided it can only appear once for each data entity defined.

Convention for the attribute value For this attribute the following conventions must be followed for its value:

- it shall be a free format character string;
- the maximum length of the value of this attribute is 400 characters.

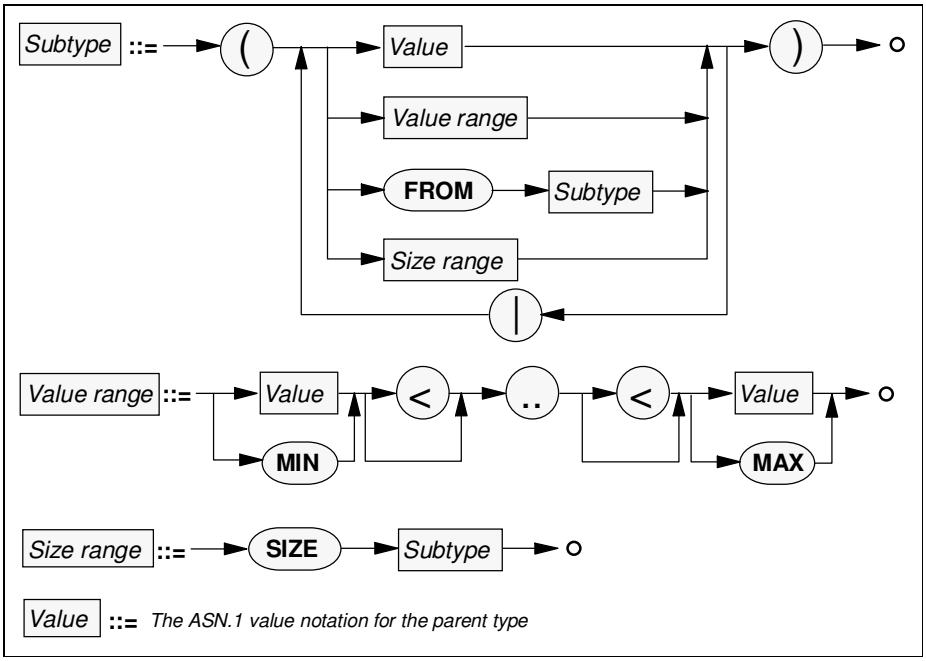
The contents of the character string must conform to the following base type and subtype definitions of ASN.1 (see Ref. [12]).

ASN.1 Base Types Supported

- **`INTEGER`** a single integer value
- **`REAL`** a single real value
- **`BOOLEAN`** a value indicating either logical true or false
- **`ENUMERATED`** a value that can be a restricted number of discrete values only
- **`BIT STRING`** a value that is a string of bits, for example, a sequence of flags
- **`OCTET STRING`** a value that is an ordered sequence of octets (8-bit bytes)
- **`VisibleString`** a value that is a character string consisting of only the visible characters plus space from the ASCII character set (see ISO 2375)

ASN.1 Subtypes Supported

An ASN.1 substype may follow the specification of the entity type to further restrict the type in some manner. For example, a more limited character string for the string types, or a limited range of values for an integer. A subtype specification is structured as follows:



Not all subtypes are applicable to all ASN.1 base types. The following summarises subtype applicability, where a ✓ indicates that the subtype is applicable and a × indicates that the substype is not applicable:

Type	***Subtype***			
	Value	***Value Range***	***FROM***	***Size Range***
`INTEGER`	✓	✓	×	×
`REAL`	✓	✓	×	×
`BOOLEAN`	✓	×	×	×
`ENUMERATED`	✓	×	×	×
`BIT STRING`	✓	×	×	✓
`OCTECT STRING`	✓	×	×	✓
`VisibleString`	✓	×	✓	✓

- *Value* is the specification of a discrete value.
- *Value Range* is the specification of a range or values, this is only applicable to scalar types.
- *FROM* is the specification of a limited alphabet subset, this is only applicable to **`VisibleString`**.
- *Size Range* is a limitation on the size of the values of the parent type. The meaning of this value is dependent upon the parent type, i.e. for **`BIT STRING`** it means number of bits, for **`OCTET STRING`** it means number of octets and for **`VisibleString`** it means number of characters.

Note that multiple subtypes can be specified for any parent type, e.g. **`INTEGER(1|3|5..9)`** or **`VisibleString(FROM("A"|"B"|"C")(SIZE (6))`**

This is only a summary of the specification of the features of ASN.1 types and subtypes that are used within the DEDSL. Ref. [12] should

be consulted for the unambiguous specification of these features.

Example attribute values

```
INTEGER
REAL(1.37..98.88)
VisibleString("ERS-1"|"ERS-2")
ENUMERATED{SAT1(5),SAT2(11),SAT3(15)}
```

3.1.6 Alternate Identification of a Data Entity: ALIAS

Purpose

The value of this attribute provides an alternative identifier of the data entity that may be required for the purpose of compatibility with historical data or data deriving from different sources. For example, different sources may produce data which includes the same entities, but which have different names. Through the use of this attribute it will be possible to define the semantic information only once. Along with the alternate identifier, this attribute value shall provide a description of the context of when the alternate identifier is used.

The value of the alternate identifier can be searched when an identifier used in a corresponding syntax description is not found within the **NAME** values.

Term to be used

The standard term to be used for this attribute is: **ALIAS**

Occurrence

This attribute is optional, although if it does appear it may appear any number of times. Each alternate identifier must be unique within the scope of the DEDSL module.

Convention for the attribute value

For this attribute the following conventions must be followed for its value:

- there shall be two values associated with this attribute: an alternative identifier and a context description;
- the alternate identifier must follow the same conventions as for that of the **NAME** attribute;
- the context information shall be a free format character string;
- that the maximum length of the context information string is 400 characters.

Example attribute value

Alternate identifier: **LAT**

Context information: **Used by the historical projects H, I, K.**

3.1.7 Specific Meaning of an Instance of a Data Entity: SPECIFIC_INSTANCE

Purpose

The value of this attribute provides a real world meaning for a specific instance of the value of the data entity being described. The reason for providing this information is so that the user can see that there is some specific meaning associated with a particular value instance that indicates something more than just the abstract value. For example, the fact that zero

degree latitude is the equator could be defined. This means that the value of this attribute must provide both an instance of the entity value and a definition of its specific meaning.

The values of the attribute can be used to enhance user interfaces and therefore user understanding. For example, instead of displaying to the user the abstract value of an entity, the 'system' could first check the DEDSL definition to see if there is a specific meaning for this abstract value, and if there is, display the specific meaning string instead. Likewise, a user could enter a meaning definition by name, e.g. **`equator`**, and the 'system' automatically (via the DEDSL definition) translate this name to a specific instance value.

Term to be used The standard term to be used for this attribute is: **`SPECIFIC_INSTANCE`**

Occurrence This attribute is optional, although if is does appear it may appear any number of times. If the attribute does appear more than once for any single entity, then each instance value that is specified must be unique as must each specific meaning definition.

Convention for the attribute value For this attribute the following conventions must be followed for its value:

- there shall be two values associated with this attribute: an instance value and a specific meaning definition.;
- the specific instance value must be expressed in visibly displayable characters (e.g. for a numeric the value must be shown in displayable characters and not expressed in binary), and be no longer than 80 characters;
- the specific meaning definition:
  - ♦ shall be a free format character string;
  - ♦ the maximum length of the specific meaning string is 80 characters

Example attribute value Specific instance: **`+00.000`**

Context information: **`Equator`**

3.1.8 Associated Information about a Data Entity: COMMENT

Purpose This attribute is for providing information which is not directly required to understand the real world meaning of the data entity, but which could still assist the user of the data in some manner. This attribute is intended for human readership.

Term to be used The standard term to be used for this attribute is: **`COMMENT`**

Occurrence This attribute is optional and can appear any number of times. This is so that additional information can be added at any time to the entity definition without having to edit existing attributes.

Convention for the attribute value For this attribute the following conventions must be followed for its value:

- it shall be a free format character string;
- the maximum length of the value of this attribute is 8000 characters.

Example attribute value **`The definition of this entity is according to ISO 6709`**

3.2 User Defined Attributes

The standard attributes specified in Section 3.1 are those predefined by this recommendation and must be recognised by any system that states conformance to this recommendation. It is recognised that there may be further attributes that are more specific to a particular domain, mission or project. This section defines a means where by users may define their own specific attributes and then use these attributes in the same manner as the standard attributes to define the semantics of their particular data entities. These attributes shall be called 'user defined attributes'.

Just as it is important to unambiguously define data entity semantics it is also important to define unambiguously the definition of user defined attributes. To do this a similar technique shall be used as that used to define the entity semantics shown in Section 3.1, where a number of parameters are defined which are required to characterise the user define attributes.

3.2.1 Identification of an Attribute: ATTRIBUTE_NAME

Purpose Specifies the user defined attribute identifier that shall be used. This identifier must be unique within any single DEDSL module, that is, it must not be the same as a standard attribute identifier and cannot be a repetition of a already existing user defined attribute identifier.

Term to be used The standard term to be used for this parameter is: **`ATTRIBUTE_NAME`**

Occurrence This parameter is mandatory and can only appear once for each user define attribute definition.

Convention for the parameter value For this parameter the following conventions must be followed for its value:

- no spaces are permitted in the value;
- the value must to be unique within the scope of the DEDSL module;
- the value is case sensitive. *Note, whilst this means that two values of* ***`ATTRIBUTE_NAME`*** *may only differ in case of certain letters, this practice is discouraged.*
- the maximum length of the value of this parameter is 40 characters.

Example parameter value **`VERSION`**

3.2.2 Full Textual Description of an Attribute: ATTRIBUTE_MEANING

Purpose Required to give the description and meaning of the user define attribute identified by the **`ATTRIBUTE_NAME`** parameter. This parameter is intended for human readership and therefore any information that will increase the understanding of the new user defined attribute should be included.

It is intended that the value of this parameter can be of a significant length and hence provide as complete a description of the user defined attribute as is possible.

Term to be used The standard term to be used for this parameter is:
`ATTRIBUTE_MEANING`

Occurrence This parameter is mandatory and can only appear once for each user defined attribute definition.

Convention for the parameter value For this parameter the following conventions must be followed for its value:

- it shall be a free format character string;
- the maximum length of the value of this parameter is 8000 characters.

Example parameter value **`This attribute states the version of the data entity. For the first official release of the dictionary the VERSION of all data entities should be 1.0.`**

3.2.3 Syntax of an Attribute Value: ATTRIBUTE_VALUE_SYNTAX

Purpose This parameter specifies the syntax of the value of the user defined attribute.

Term to be used The standard term to be used for this parameter is:
`ATTRIBUTE_VALUE_SYNTAX`

Occurrence This parameter is mandatory and can only appear once for each user define attribute definition.

Convention for the parameter value For this parameter the following conventions must be followed for its value:

- it shall be a free format character string;
- the maximum length of the value of this attribute is 400 characters.

The contents of the character string must conform to the following base type and subtype definitions of ASN.1 (see Ref. [12]).

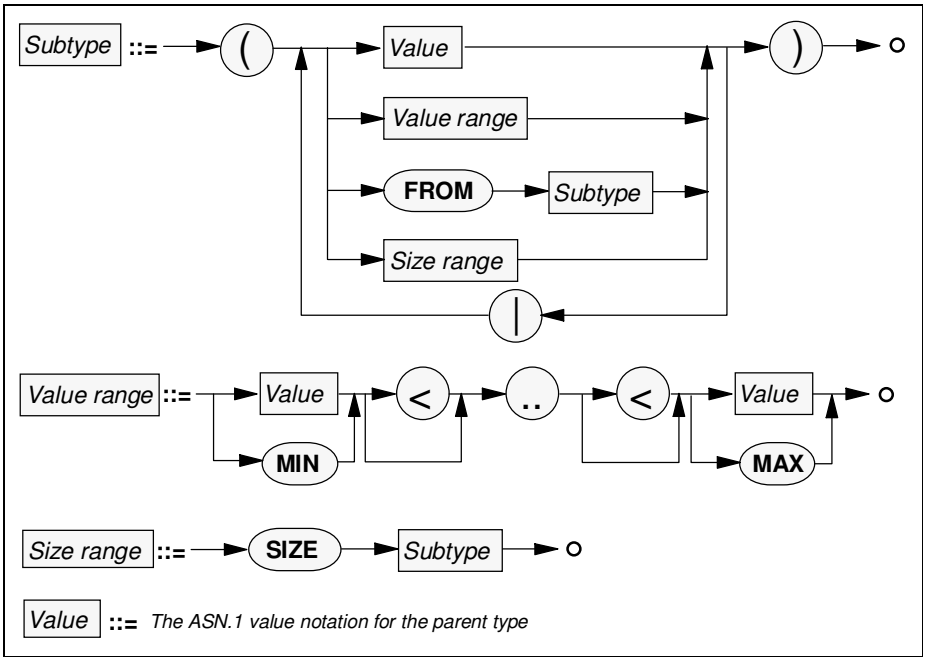
ASN.1 Base Types Supported

- **`INTEGER`** a single integer value
- **`REAL`** a single real value

- **`BOOLEAN`** a value indicating either logical true or false
- **`ENUMERATED`** a value that can be a restricted number of discrete values only
- **`BIT STRING`** a value that is a string of bits, for example, a sequence of flags
- **`OCTET STRING`** a value that is an ordered sequence of octets (8-bit bytes)
- **`VisibleString`** a value that is a character string consisting of only the visible characters plus space from the ASCII character set (see ISO 2375)

ASN.1 Subtypes Supported

An ASN.1 substype may follow the specification of the entity type to further restrict the type in some manner. For example, a more limited character string for the string types, or a limited range of values for an integer. A subtype specification is structured as follows:



Not all subtypes are applicable to all ASN.1 base types. The following summarises subtype applicability, where a ✓ indicates that the subtype is applicable and a ✗ indicates that the substype is not applicable:

Type	***Subtype***			
	Value	***Value Range***	***FROM***	***Size Range***
`INTEGER`	✓	✓	✗	✗
`REAL`	✓	✓	✗	✗
`BOOLEAN`	✓	✗	✗	✗
`ENUMERATED`	✓	✗	✗	✗
`BIT STRING`	✓	✗	✗	✓
`OCTECT STRING`	✓	✗	✗	✓

VisibleString	✓	×	✓	✓

- *Value* is the specification of a discrete value.
- *Value Range* is the specification of a range or values, this is only applicable to scalar types.
- *FROM* is the specification of a limited alphabet subset, this is only applicable to **VisibleString**.
- *Size Range* is a limitation on the size of the values of the parent type. The meaning of this value is dependent upon the parent type, i.e. for **BIT STRING** it means number of bits, for **OCTET STRING** it means number of octets and for **VisibleString** it means number of characters.

Note that multiple subtypes can be specified for any parent type, e.g. **INTEGER(1|3|5<..<9)** or **VisibleString(FROM("A"|"B"|"C")(SIZE(6))**

This is only a summary of the specification of the features of ASN.1 types and subtypes that are used within the DEDSL. Ref. [12] should be consulted for the unambiguous specification of these features.

Example parameter value

VisibleString

INTEGER

3.2.4 Occurrences of an Attribute: ATTRIBUTE_OCCURANCE

Purpose

This parameter specifies the number of times that the user defined attribute may occur when used within the definition of a single data entity. For example the standard attribute **UNITS** can only appear once for any single data entity definition, whilst **ALIAS** is optional, but if it does appear it can then appear any number of times. Similarly, if the user defined a new user defined attribute called **DEFINER**, to be used to indicate the person that defined the entity definition, this could be indicated as mandatory and appearing only once.

Term to be used

The standard term to be used for this parameter is:
ATTRIBUTE_OCCURANCE

Occurrence

This parameter is mandatory and can only appear once for each user define attribute definition.

Convention for the parameter value

The format of the parameter value is a string that is defined by the following BNF definition:

a..[b|"n"]

Where **a** is the minimum number of times that the user defined attribute may occur, **b** is the maximum number of times that the user defined attribute may occur or the character "**n**" indicates that there is no upper limit on the number of times that the user defined attribute may occur. There are no spaces permitted and **a** must be less than or equal to **b**.

For example: **1..1** exactly once

0..1 optional

0..n optional but no specified maximum

1..n at least once but no specified maximum

0..5 optional with a maximum of 5 times

1..5 at least once with a maximum of 5 times

3..8 at least 3 times with a maximum of 8 times

It is recommended that a realistic maximum number of occurrences is always specified so that practical software can be designed to handle the user defined attribute.

Example parameter value **0..1**

3.2.5 Example of a Value of an Attribute: ATTRIBUTE_VALUE_EXAMPLE

Purpose This parameter provides an example of a value of the user defined attribute as it would appear in an entity definition.

Term to be used The standard term to be used for this parameter is: **ATTRIBUTE_VALUE_EXAMPLE**

Occurrence This parameter is optional and may appear any number of times demonstrating different examples in each case.

Convention for the parameter value The format of this parameter value must conform with the value syntax as defined by the **ATTRIBUTE_VALUE_SYNTAX** parameter. The maximum length of this parameter value is also dictated by the **ATTRIBUTE_VALUE_SYNTAX** specification.

Example parameter value **2.4**

3.2.6 Associated Information about an Attribute: ATTRIBUTE_COMMENT

Purpose This parameter is for providing information which is not directly required to understand the meaning of the user define attribute, but which could still assist the user of the user defined attribute in some manner.

Term to be used The standard term to be used for this parameter is: **ATTRIBUTE_COMMENT**

Occurrence This parameter is optional and can appear any number of times. This is so that additional information can be added at any time to the user defined attribute definition without editing existing parameters..

Convention for the parameter value For this parameter the following conventions must be followed for its value:

- it shall be a free format character string;

- the maximum length of the value of this parameter is 8000 characters.

Example attribute value `For all entities in the dictionary, in addition to the standard DEDSL attributes, the attribute VERSION should be stated in order to allow traceability of entity versions.`

3.2.7 Registration of User Defined Attributes

So as to aim for maximum reuse and hence interoperability across missions, projects and agencies, it is desirable that any new user defined attributes that are created by projects are submitted for central registration. This means that they can be reused by other projects, eventually leading to a uniform data entity dictionary across many mission and projects.

The advantage of commonality of the data entity dictionary is that software can be developed to handle the entity definitions, which can then be reused by many other projects.

To register user defined attributes, the data description registration capabilities detailed in the CCSDS Recommendations on Control Authorities (se references [9] and [10]) should be followed.

When a user registers a user defined attribute, the following information must be included:

- Identification of the user - this information shall be as defined in the registration of data descriptions with the CCSDS Control Authority (see references [9] and [10]).
- A specification of the user defined attribute following the conventions defined in Sections 3.2.1 to 3.2.6.
- If software is available to support processing of the value of the user defined attribute, this should be submitted with the definition.

3.3 DEDSL Module Identification Attributes

It is frequently useful to be able to identify information about a particular DEDSL module, whether the module defines a data entity dictionary intended for a single data structure or whether it is a discipline oriented dictionary. So as to provide this type of information in a coherent and common manner this section identifies a number of attributes that can be considered global to the complete product or domain. These attributes are called Module Identification Attributes.

The primary purpose in providing this information is so that automated software can use it to detect whether it can process the particular version of the DEDSL module.

To convey the necessary information the following attributes are required. They are all ASCII strings.

3.3.1 DEDSL Version Information: DEDSL_VERSION

Purpose This parameter indicates the version of the DEDSL Recommendation that the DEDSL module conforms to. All versions are backwards compatible with previous versions. The version is indicated on the front cover of the Recommendation.

Term to be used The standard term to be used for this parameter is: **`DEDSL_VERSION`**

Occurrence If the module identification information is provided, then this parameter is mandatory and can appear only once.

Convention for the parameter value The format of this parameter value is a decimal number represented in ASCII, with a maximum length of 30 characters.

Example parameter value **`0.6`**

3.3.2 DEDSL Module Identification: MODULE_TITLE

Purpose This parameter indicates a human readable name for the DEDSL module.

Term to be used The standard term to be used for this parameter is: **`MODULE_TITLE`**

Occurrence This parameter is optional and can appear only once.

Convention for the parameter value For this parameter the following conventions must be followed for its value:

- it shall be a free format character string that can span a number of lines;
- it may include any number of white space characters;
- the maximum length of the value of this parameter is 400 characters.

Example parameter value **`impQ_test dictionary`**

3.3.3 DEDSL Module ADID: MODULE_ADID

Purpose This parameter indicates the CCSDS Authority and Description Identifier (ADID) that the DEDSL module has been registered under at a Control Authority (see references [9] and [10]).

Term to be used The standard term to be used for this parameter is: **`MODULE_ADID`**

Occurrence This parameter is optional and can appear only once.

Convention for the parameter value The format of this parameter value is an unquoted ASCII string of eight consecutive Restricted ASCII[1] characters that constitute a registered MACAO ADID (see references [9] and [10]).

Example parameter value **`NSSD1393`**

---

[1] *For a definition of the characters that constitue the Restricted ASCII (RA) character set see Annex C.*

3.4 Implementation Guidelines

This section gives some guidelines concerning the use of attributes, whether 'standard', 'user defined' or 'identification' ones.

- The identifier for all attributes (independent if standard, user defined or module identification attributes) should follow the following guidelines:
  - Whilst the identifier is case sensitive, that is, the identifiers **`UNITS`**, **`Units`** and **`units`** are all different attribute identifiers (only **`UNITS`** being a 'standard' definition), it is **STRONGLY** recommended that users do not discern different attributes purely by case;
  - It is recommended that all identifiers only use uppercase letters, numerics and the underscore character (_).
- Whilst for any single semantic entity definition, the standard attributes can be presented or accessed in any order, it is recommended that the following order is used whenever possible so as to present a common style to all users (the optional attributes are indicated in italics):

  `NAME`
  `MEANING`
  `SHORT_MEANING`
  `UNITS`
  `VALUE_SYNTAX`
  `ALIAS`
  `SPECIFIC_INSTANCE`
  `COMMENT`

  user defined attributes should follow the standard attributes.
- Whilst for any single user defined attribute definition, the parameters that define the user defined attribute can be presented or accessed in any order, it is recommended that the following order is used whenever possible so as to present a common style to all users (the optional attributes are indicated in italics):

  `ATTRIBUTE_NAME`
  `ATTRIBUTE_MEANING`
  `ATTRIBUTE_VALUE_SYNTAX`
  `ATTRIBUTE_OCCURANCE`
  `ATTRIBUTE_VALUE_EXAMPLE`
  `ATTRIBUTE_COMMENT`
- The attributes for each entity definition (or parameters for each attribute definition) must be grouped in some manner so as to keep them separate from the attributes of other entity definitions. The methodology for grouping the attributes must be defined formally in the implementation syntax.
- When specifying the definition of entities that are described syntactically by a separate syntax language description, it is recommended to use the commonly understood 'dot' notation, for naming the data entities within the DEDSL implementation, that are part of a hierarchical data structure. For example, **`packet_header.time.minutes`**. Annex A.1

demonstrates this naming convention when using the syntax description language EAST (see References [1] and [2]).

4 IMPLEMENTATION USING PVL

Section 3 defines the DEDSL as an abstract standard, this permits the actual conveyance of the information via a number of techniques which may be necessary due to the manner that the data entities being described are actually formatted. The recommended method of conveying the information is by using the CCSDS developed Parameter Value Language (PVL, see references [5] and [6]).

PVL is designed to support the conveyance of named values, therefore it is ideal for the purpose of implementing the abstract standard defined in Section 3. The additional advantage of using PVL for the implementation is that it permits the inclusion of white-space characters and comments to assist layout and human readership. This makes the PVL implementation easily expandable and flexible.

If the data entity definitions or user defined attribute definitions are carried separately to the data entities being described, then each definition must be grouped into a collection of statements using the PVL statement aggregation techniques.

The following sections specify the implementation of the abstract standard in the order:

- Section 4.1 provides a structure diagram of a complete DEDSL module. This assumes that the definition is wholly implemented in PVL and is carried in a separate data block to the actual data.
- Section 4.2.1 defines the PVL implementation of the individual standard attributes.
- Section 4.2.2 defines the PVL implementation of how to group the standard attributes for each entity definition.
- Section 4.3.1 defines the PVL implementation of how to specify the parameters which define a user defined attribute.
- Section 4.3.2 defines the PVL implementation of how to group the parameters required to specify each user defined attribute.
- Section 4.4.1 defines the PVL implementation of the module identification attributes.
- Section 4.4.2 defines the PVL implementation of how to group the module identification attributes.

4.1 Complete DEDSL Definition

Figure 4-1 shows the structure of a complete DEDSL module. If the module identification attributes, user defined attributes and data entity definitions appear in a single data block, such as a file, then they must appear in the order indicated in this structure diagram. This is so as to present a consistent view to the user and for more efficient software processing. The contents of each of the three block are shown in Figure 4-2: Structure Diagram of a Single DEDSL Entity Definition, Figure 4-3: Structure Diagram of a Single DEDSL User Defined Attribute Definition and Figure 4-4: Structure Diagram of the DEDSL Module Identification Attributes. The structure diagram clearly shows which block of PVL statements are optional (off the centre line) and which are mandatory (on the centre line), those on the left of the centre line

can appear at most only once and those on the right of the centre line can appear any number of times.

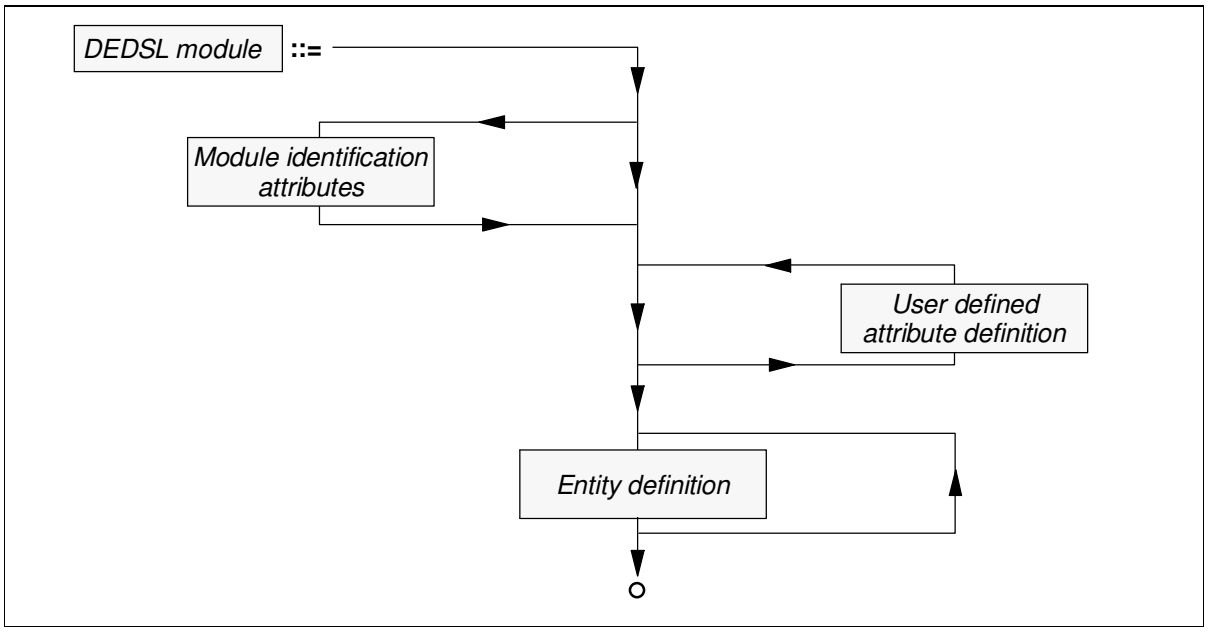


Figure 4-1: Structure Diagram of a Complete DEDSL Description

- Firstly appears the module identification attributes; whilst these are optional it is recommended that they should appear to assist in more efficient software support and configuration control.
- Secondly there are the user defined attribute definitions. There can be any number of these statement blocks.
- Thirdly, there are the entity definitions. There must be at least one of these within a DEDSL module and there is no limit on the total number.

A complete example using the recommended structure as defined in Figure 4-1 is shown in Section 2.

4.2 Standard Attributes

4.2.1 Implementation of Standard Attributes

Abstract Attribute	Data entity identification: NAME
PVL Parameter	**NAME**
PVL Value Syntax	An unquoted PVL string of maximum length 400 characters
Attribute occurrence	**1..1**
Example	**NAME = ACQ_STATION ;** **NAME = RMS ;**

Abstract Attribute	Full textual description of a data entity: MEANING

PVL Parameter	**MEANING**
PVL Value Syntax	A quoted PVL string of maximum length 8000 characters
Attribute occurrence	**1..1**
Example	**MEANING = "The ACQ_STATION includes the identifier of the station, which has acquired the data";**

Abstract Attribute	Short textual description of a data entity: SHORT_MEANING
PVL Parameter	**SHORT_MEANING**
PVL Value Syntax	A quoted PVL string of maximum length 80 characters
Attribute occurrence	**0..1**
Example	**SHORT_MEANING = "Identifier of acquisition station" ;**

Abstract Attribute	Syntax of the value of a data entity: VALUE_SYNTAX
PVL Parameter	**VALUE_SYNTAX**
PVL Value Syntax	A PVL string (quoted or unquoted as necessary) of maximum length 400 characters formatted according to the syntax specified in Section 3.1.5.
Attribute occurrence	**0..1**
Example	**VALUE_SYNTAX = "INTEGER(1..10)" ;** **VALUE_SYNTAX = "VisibleString(SIZE(20))" ;**

Abstract Attribute	Scientific units of a data entity: UNITS
PVL Parameter	**UNITS**
PVL Value Syntax	A PVL string (quoted or unquoted as necessary) of maximum length 80 characters
Attribute occurrence	**0..1**
Example	**UNITS = "0.225.km/s2" ;** **UNITS = "Cel" ;**

Abstract Attribute	Alternate identification of a data entity: ALIAS
PVL Parameter	**ALIAS**
PVL Value Syntax	A two value PVL sequence, the first value being an unquoted PVL string, of maximum length 400 characters, which is the alternate identifier for the data entity and the second value a quoted PVL

	string of maximum length 400 characters that explains the context for the use of the alternate identifier
Attribute occurrence	**0..n**
Example	**ALIAS = (LAT , "Used by the historical projects H, I, K") ;** **ALIAS = (TIME_LINE, "This entity is identified by this name within the ESA ground segement, rather than the primary name within the NASA ground segment") ;**

Abstract Attribute	Specific meaning of an instance of a data entity: SPECIFIC_INSTANCE
PVL Parameter	**SPECIFIC_INSTANCE**
PVL Value Syntax	A two value PVL sequence: • the first value being an ASCII representation (i.e. for a binary number, the ASCII representation is as defined in Ref. [12]) of the specific instance of the data entity that has a specific meaning, this shall be no longer than 80 characters; • the second value a quoted PVL string of maximum length 80 characters that explains the meaning of the specific instance.
Attribute occurrence	**0..n**
Example	**SPECIFIC_INSTANCE = (+00.000, "Equator") ;** **SPECIFIC_INSTANCE = (100, "Boiling point of h-2-O") ;** **SPECIFIC_INSTANCE = (19920812T17:54:23.12, "Experiment switch on time") ;**

Abstract Attribute	Associated information about a data entity: COMMENT
PVL Parameter	**COMMENT**
PVL Value Syntax	A quoted PVL string of maximum length 8000 characters
Attribute occurrence	**0..n**
Example	**MEANING = "The definition of this entity is according to ISO 6709" ;**

4.2.2 Aggregation of a Data Entity Definition

Section 4.2.1 has specified the exact PVL syntax for each of the standard attributes required to define a single data entity. In the situation where more than one entity definition are conveyed in a single DEDSL module, such as a file, it is necessary to group all the attributes that describe a single entity, so as to distinguish each of the entity definitions. Using PVL, this is done with the GROUP construct.

Preceding the first of the grouped PVL statements, there shall be the PVL start aggregation statement:

```
BEGIN_GROUP = ENTITY_DEFINITION ;
```

Following the last of the grouped statements, there shall be the PVL end aggregation statement:

```
END_GROUP = ENTITY_DEFINITION ;
```

Figure 4-2 shows with a structure diagram the structure of the PVL statements that constitute a DEDSL entity definition. Whilst the order of the statements are not mandated to be in the order shown, this is the recommended order to enhance standardisation and understanding. The structure diagram clearly shows which PVL statements are optional (off the centre line) and which are mandatory (on the centre line), those on the left of the centre line can appear at most only once and those on the right of the centre line can appear any number of times.

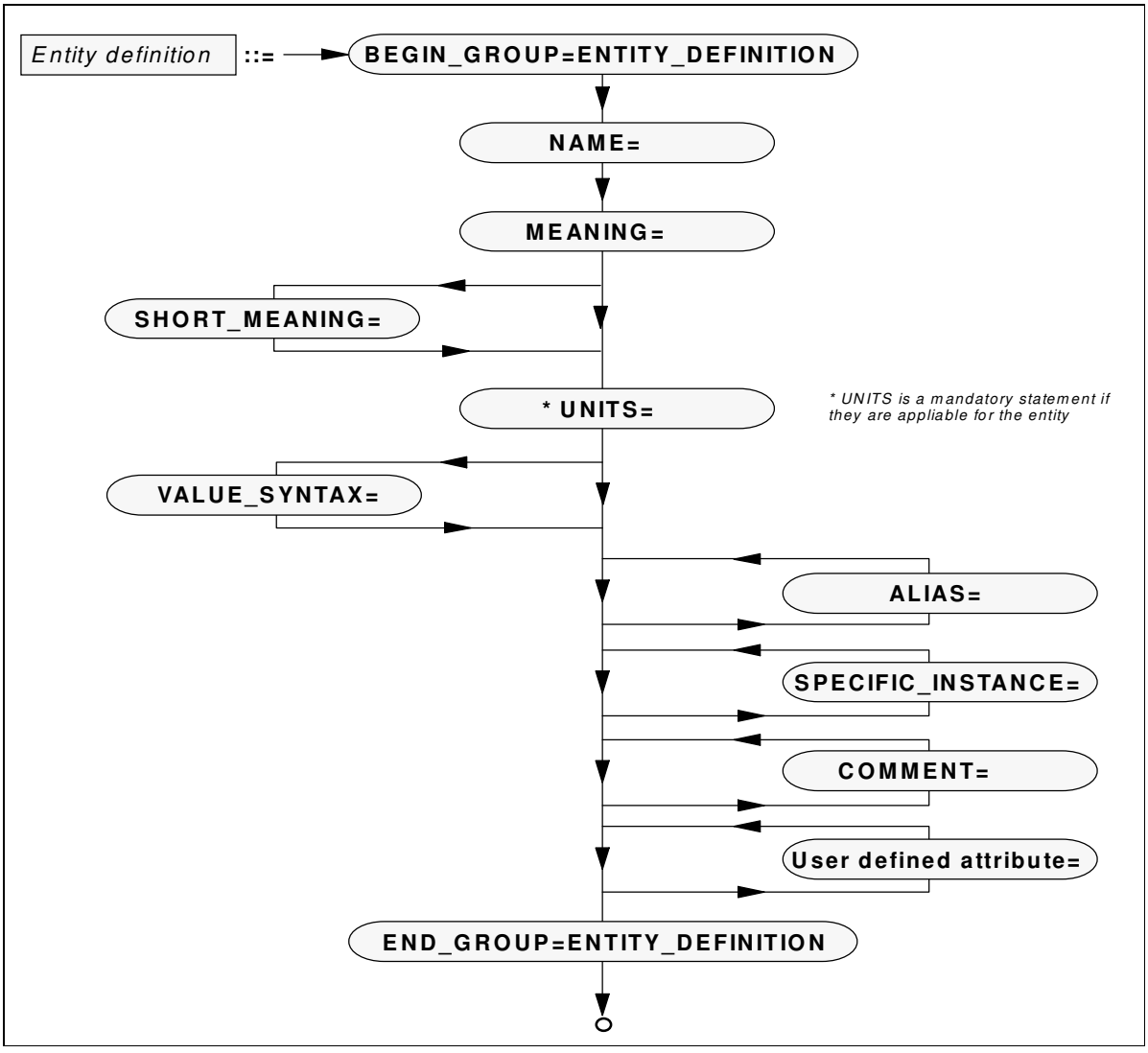


Figure 4-2: Structure Diagram of a Single DEDSL Entity Definition

Using the recommended structure above, the following shows how two entity definitions would appear in a single file (note in the second entity definition some of the standard attributes are not applicable and therefore not included):

```
BEGIN_GROUP = ENTITY_DEFINITION;
    NAME = Epoch ;
    MEANING = "Time of mid-points of the observation intervals since
              A.D." ;
    SHORT_MEANING = "Time Line" ;
    UNITS = ms ;
     VALUE_SYNTAX = VisibleString(SIZE(20)) ;
    ALIAS = (Sample_Time, "The name of the entity used by the central
             processing algorithm" );
    SPECIFIC_INSTANCE = ( 19900212T08:17:24.04, "Start of mission" );
    COMMENT = "This is an example entity to demonstrate the DEDSL" ;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION ;
    NAME = RMS ;
    MEANING = "Components of B in GSE coordinates obtained by taking
               the root-mean-square of the values in the observing
               intervals" ;
    SHORT_MEANING = "Components of RMS of B (GSE)" ;
     VALUE_SYNTAX = REAL ;
    UNITS = "nT" ;
END_GROUP = ENTITY_DEFINITION;
```

4.3 User Defined Attributes

4.3.1 Implementation of User Defined Attributes

Abstract Parameter	Identification of an attribute: ATTRIBUTE_NAME
PVL Parameter	**ATTRIBUTE_NAME**
PVL Value Syntax	An unquoted PVL string of maximum length 40 characters
Parameter occurrence	**1..1**
Example	**ATTRIBUTE_NAME = VERSION ;**

Abstract Parameter	Full textual description of an attribute: ATTRIBUTE_MEANING
PVL Parameter	**ATTRIBUTE_MEANING**
PVL Value Syntax	A quoted PVL string of maximum length 8000 characters
Parameter occurrence	**1..1**
Example	**ATTRIBUTE_MEANING = " This attribute states the version of the data entity. For the first official release of the dictionary the VERSION of all data entities should be 1.0" ;**

Abstract Parameter	Syntax of an attribute value: ATTRIBUTE_VALUE_SYNTAX

PVL Parameter	**ATTRIBUTE_VALUE_SYNTAX**
PVL Value Syntax	A PVL string (quoted or unquoted as necessary) of maximum length 400 characters formatted according to the syntax specified in Section 3.2.3.
Parameter occurrence	**1..1**
Example	**ATTRIBUTE_VALUE_SYNTAX = REAL ;** **ATTRIBUTE_VALUE_SYNTAX = VisibleString ;**

Abstract Parameter	Example of the value of an attribute: ATTRIBUTE_VALUE_EXAMPLE
PVL Parameter	**ATTRIBUTE_VALUE_EXAMPLE**
PVL Value Syntax	The value of this parameter shall be the ASCII representation of the syntax that is defined by the parameter **ATTRIBUTE_VALUE_ SYNTAX** parameter
Parameter occurrence	**0..n**
Example	**ATTRIBUTE_VALUE_EXAMPLE = 2.4 ;**

Abstract Parameter	Associated information about an attribute: ATTRIBUTE_COMMENT
PVL Parameter	**ATTRIBUTE_COMMENT**
PVL Value Syntax	A quoted PVL string of maximum length 8000 characters
Parameter occurrence	**0..n**
Example	**ATTRIBUTE_COMMENT = " For all entities in the dictionary, in addition to the standard DEDSL attributes, the attribute VERSION should be stated in order to allow traceability of entity versions";**

4.3.2 Aggregation of User Defined Attribute Parameters

Section 4.3.1 has specified the exact PVL syntax for each of the parameters required to define a single user defined attribute. In the situation where more than one user defined attribute definition is conveyed in a single DEDSL module, such as a file, it is necessary to group all the parameters that describe a single user defined attribute, so as to distinguish each of the separate definitions. Using PVL, this is done with the GROUP contract.

Preceding the first of the grouped PVL statements, there shall be the PVL start aggregation statement:

BEGIN_GROUP = ATTRIBUTE_DEFINITION ;

Following the last of the grouped statements, there shall be the PVL end aggregation statement:

```
END_GROUP = ATTRIBUTE_DEFINITION ;
```

Figure 4-3 shows with a structure diagram the structure of the PVL statements that constitute a DEDSL user defined attribute definition. Whilst the order of the statements are not mandated to be in the order shown, this is the recommended order to enhanced standardisation and understanding. The structure diagram clearly shows which PVL statements are optional (off the centre line) and which are mandatory (on the centre line), those on the right of the centre line can appear any number of times.

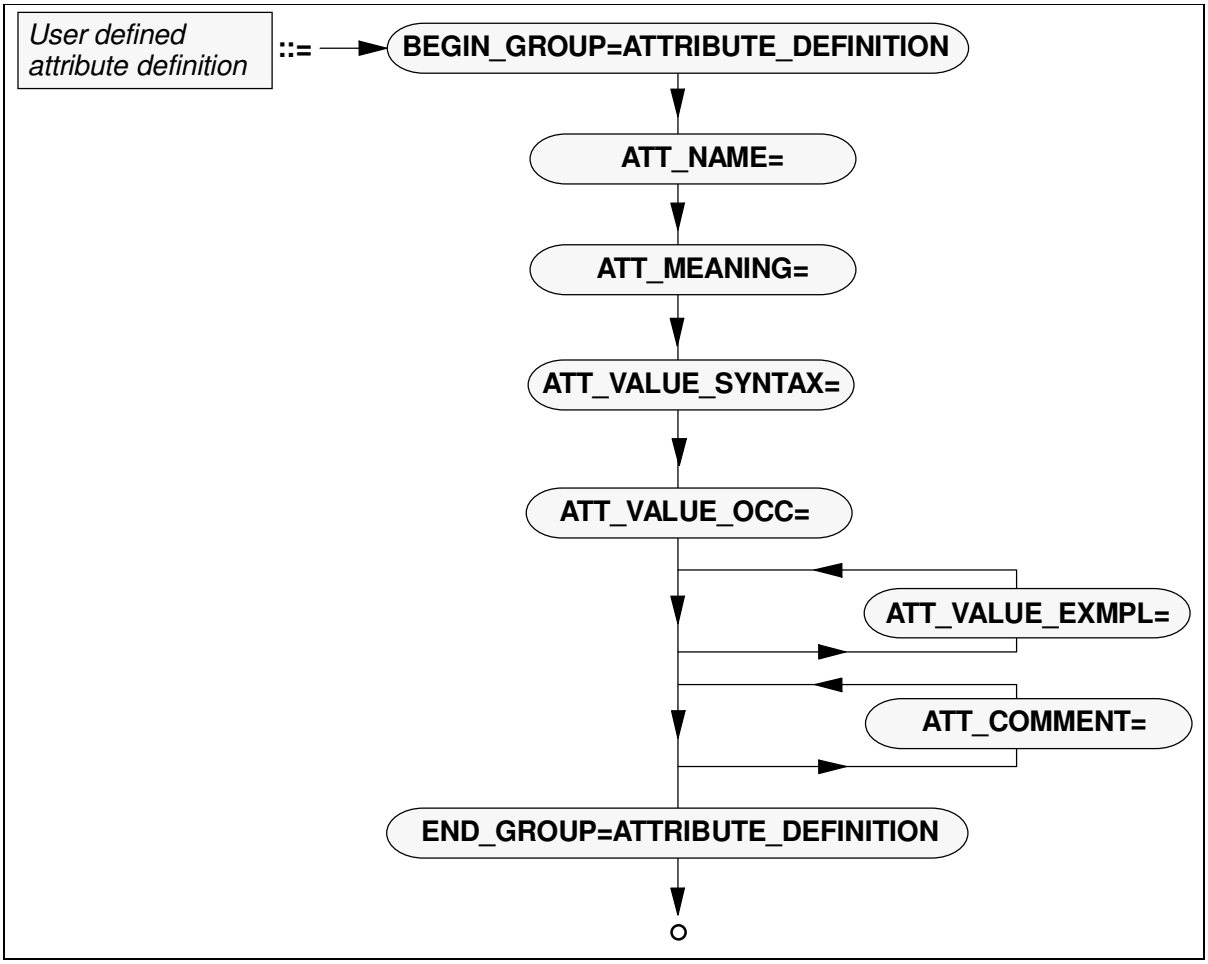


Figure 4-3: Structure Diagram of a Single DEDSL User Defined Attribute Definition

Using the recommended structure above, the following shows how two user defined attribute definitions would appear in a single file:

```
BEGIN_GROUP = ATTRIBUTE_DEFINITION ;
    ATTRIBUTE_NAME = PROJECT ;
    ATTRIBUTE_MEANING = "This attribute specifies the name of the project
                  for which the data entity attribute has been defined";
    ATTRIBUTE_VALUE_SYNTAX= VisibleString ;
    ATTRIBUTE_OCCURANCE = 0..1 ;
    ATTRIBUTE_VALUE_EXAMPLE= ISTP ;
    ATTRIBUTE_COMMENT = "A formal list of recognised project names can be
                         found out by contacting J. Smith, within
                         NASA/NSSDC." ;
END_GROUP = ATTRIBUTE_DEFINITION ;

BEGIN_GROUP = ATTRIBUTE_DEFINITION ;
    ATTRIBUTE_NAME = VALIDMAX ;
    ATTRIBUTE_MEANING = "This is a single scalar value that indicates
                         the maximum value that the entity may be
                         within any single instance of the data
                         structure." ;
    ATTRIBUTE_VALUE_SYNTAX = INTEGER ;
    ATTRIBUTE_OCCURANCE = 1..1 ;
END_GROUP = ATTRIBUTE_DEFINITION ;
```

4.4 DEDSL Module Identification Attributes

4.4.1 Implementation of DEDSL Module Identification Attributes

Abstract Attribute	Version of the DEDSL recommendation used: DEDSL_VERSION
PVL Parameter	**DEDSL_VERSION**
PVL Value Syntax	A PVL floating point number of maximum length 30 characters
Attribute occurrence	**1..1**
Example	**DEDSL_VERSION = 0.6 ;**

Abstract Attribute	Identification of the DEDSL module: MODULE_TITLE
PVL Parameter	**MODULE_TITLE**
PVL Value Syntax	A quoted PVL string of maximum 400 characters
Attribute occurrence	**0..1**
Example	**MODULE_TITLE = "impQ_test dictionary" ;**

Abstract Attribute	MACAO ADID that the DEDSL module is registered under: MODULE_ADID
PVL Parameter	**MODULE_ADID**

PVL Value Syntax	A PVL unquoted string consisting of 8 Restricted ASCII[2] characters
Attribute occurrence	`0..1`
Example	`MODULE_ADID = NSSD1393;`

4.4.2 Aggregation of DEDSL Module Identification Attributes

Section 4.4.1 has specified the exact PVL syntax for each of the DEDSL module identification attributes. In the situation where all the module identification attributes are conveyed in a single data block, such as a file, it is necessary to group them, so as to distinguish them from the actual entity definitions. Using PVL, this is done with the GROUP construct.

Preceding the first of the grouped PVL statements, there shall be the PVL start aggregation statement:

```
BEGIN_GROUP = MODULE_IDENTIFICATION ;
```

Following the last of the grouped stamens, there shall be the PVL end aggregation statement:

```
END_GROUP = MODULE_IDENTIFICATION ;
```

Figure 4-4 shows with a structure diagram the structure of the PVL statements that constitute the DEDSL module identification attributes. Whilst the order of the statements are not mandated to be in the order shown, this is the recommended order to enhanced standardisation and understanding. The structure diagram clearly shows which PVL statements are optional (off the centre line) and which are mandatory (on the centre line), those on the left of the centre line can appear only once.

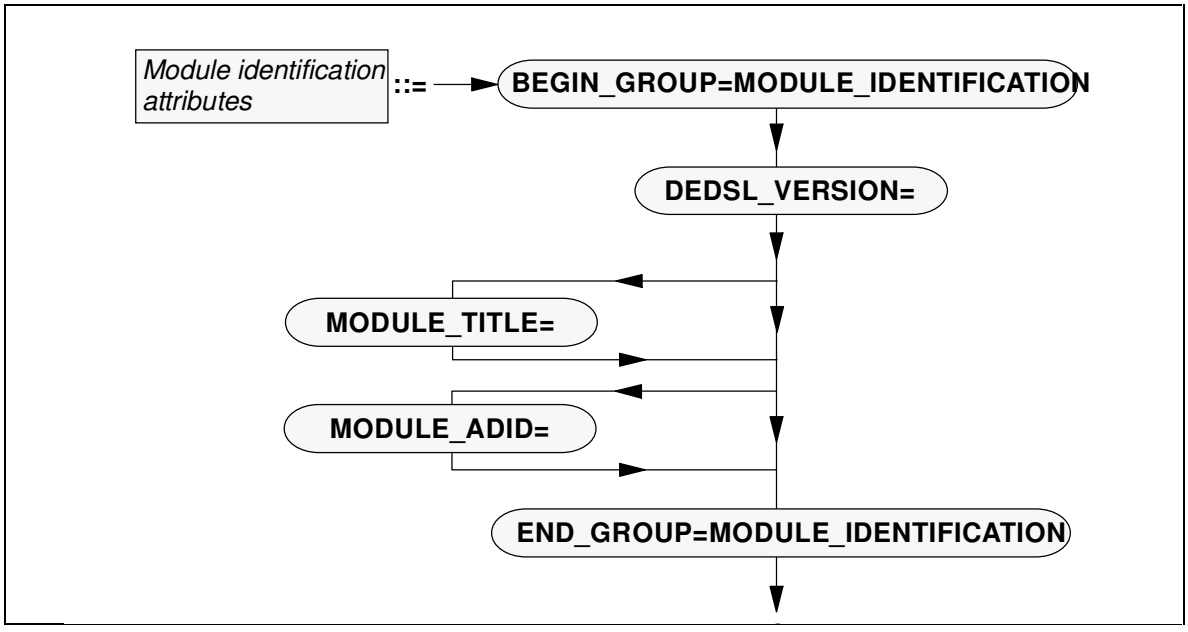


Figure 4-4: Structure Diagram of the DEDSL Module Identification Attributes

Using the recommended structure above, the following figure shows an example of DEDSL module identification attributes:

---

[2] *For a definition of the characters that constitue the Restricted ASCII (RA) character set see Annex C.*

```
BEGIN_GROUP = MODULE_IDENTIFICATION ;
    DEDSL_VERSION = 0.6 ;
    MODULE_TITLE = "impQ_test dictionary" ;
    MODULE_ADID = NSSD1393 ;
END_GROUP = MODULE_IDENTIFICATION ;
```

5 AN EXAMPLE IN THE USE OF THE DEDSL

This sections presents the full data entity dictionary specifications for the products introduced in Section 2. Firstly the discipline dictionary is presented and then both dictionaries for Products X and Y. Further examples, relating the DEDSL to particular scenarios, are presented in Annex A.

5.1 Use of the DEDSL for a Discipline DED

The following the an example of a DED, specified using the DEDSL, that has been produced for a particular discipline, in this case Earth observation. This discipline DED is used in the subsequent sections as a basis for the product specific DEDs, for Products X and Y in Table 2-1.

```
/*************************************************************************/
/***           Identification of the discipline dictionary             ***/
/*************************************************************************/

BEGIN_GROUP = MODULE_IDENTIFICATION;
     DEDSL_VERSION = 0.6;
     MODULE_TITLE = "This is the Dictionary of Project ABC. All standard
                    parameters used to define data products within the
                    domain of Project ABC are included here";
     MODULE_ADID = NSSD1393;
END_GROUP = MODULE_IDENTIFICATION;

/*************************************************************************/
/***             Definition of user defined attributes                 ***/
/*************************************************************************/

BEGIN_GROUP = ATTRIBUTE_DEFINITION;
     ATTRIBUTE_NAME = VERSION;
     ATTRIBUTE_MEANING = "This attribute states the version of the data
                         entity definition. For the first official release
                         of the definition the VERSION of all data entity
                         definitions should be 1.0.";
     ATTRIBUTE_VALUE_SYNTAX = REAL;
     ATTRIBUTE_VALUE_OCCURENCE = 1..1;
     ATTRIBUTE_VALUE_EXAMPLE = 2.4;
     ATTRIBUTE_COMMENT = "For all entities in the dictionary, in addition
                          to the standard DEDSL attributes, the attribute
                          VERSION should be stated in order to allow
                          traceability of entity versions.";
END_GROUP = ATTRIBUTE_DEFINITION;

/*************************************************************************/
/***  Entity definitions in terms of standard attributes and           ***/
/***  previously stated user defined attributes                        ***/
/*************************************************************************/

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  PRODUCT_ID;
```

```
     MEANING = "The PRODUCT_ID parameter represents a permanent, unique
               identifier assigned to a data product by its producer.";
     SHORT_MEANING = "Product Identification parameter";
     ALIAS = (PRODUCT_NAME, "Used by the historical projects H, I, K to
                             identify their data products");
     VALUE_SYNTAX = VisibleString(SIZE(1..40))
     COMMENT = "The definition has been extracted from the Planetary
               Science Data Dictionary Document, Issue V6 MOSO0099--4-00,
               20 November 1992;
     VERSION = 1.0;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  ACQ_STATION;
     MEANING = "The ACQ_STATION includes the identifier of the station,
               which has acquired the data.";
     SHORT_MEANING = "Identifier of acquisition station";
     VALUE_SYNTAX = ENUMERATION {STAT_1(1), STAT_2(2), STAT_3(3)};
     SPECIFIC_INSTANCE = (STAT_1, "Station 1 identifier");
     SPECIFIC_INSTANCE = (STAT_2, "Station 2 identifier");
     SPECIFIC_INSTANCE = (STAT_3, "Station 3 identifier");
     VERSION = 1.0;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  ACQ_TIME;
     MEANING = "Date and time of acquisition of the data. The rules to
               format this parameter are: 'YYYY-MM-DDThh:mm:ss.d->Z'. An
               example is: '1988-01-18T17:20:43.123456Z'. The acquisition
               time should correspond to the first scan line of the
               data.";
     SHORT_MEANING = "Date/Time of data acquisition";
     VALUE_SYNTAX = VisibleString(FROM("0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"
                    |"9"|"-"|":"|"."|"T"|"Z"))(SIZE(21..40));
     COMMENT = "The date/time is defined according to the CCSDS/ISO rules
               for data/time definitions";
     VERSION = 1.0;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  LATITUDE;
     MEANING = "Latitudes north of the equator shall be designated by the
               use of the plus (+) sign, latitudes south of the equator
               shall be designated by the use of the minus sign (-). The
               equator shall be designated by the use of the plus sign
               (+).";
     SHORT_MEANING = "Latitude";
     UNITS = deg ;
     ALIAS = (LAT, " Used by the historical projects H, I, K ");
     VALUE_SYNTAX = REAL (-90.000..+90.000);
     SPECIFIC_INSTANCE = (+00.000, "Equator");
     COMMENT = "The definition of this entity is according to ISO 6709";
     VERSION = 1.0;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  LONGITUDE;
```

```
     MEANING = "Longitudes east of Greewhich shall be designated by the use
                of the plus sign (+), longitudes west of Greenwhich shall
                be designated by the use of the minus sign (-).  The Prime
                Meridian shall be designated by the use of the plus sign
                (+). The 180th meridian shall be designated by the use of
                the minus sign (-)";
     SHORT_MEANING = "Longitude";
     UNITS = deg ;
     ALIAS = (LON, " Used by the historical projects H, I, K ");
     VALUE_SYNTAX = REAL (-180.000 .. +180.000);
     SPECIFIC_INSTANCE = (-180.000, "The 180th Meridian");
     COMMENT = "The definition of this entity is according to ISO 6709";
     VERSION = 1.0;
END_GROUP = ENTITY_DEFINITION;
```

5.2 Use of THE DEDSL for Product DEDs

The following sections use the example Earth observation discipline DED shown in the previous section as a basis for product specific dictionaries. The DEDs are independent of the physical format of the product, i.e. the product may be formatted in CDF, HDF, etc., but the same DED is applicable to both. The format of the DED shown here is complaint with this DEDSL Recommendation.

Whilst the DED entries for these product DEDs are based upon the discipline DED, in an effort to promote standardisation, once the DED has been produced for a product, then the discipline DED has no further relevance as the product DED is self standing.

5.2.1 Use of the DEDSL for Product X

In the following product DED, the following changes from the discipline DED have been made:

1. Removed the user defined attribute definition **`Version`**, as this attribute is not used here in the product DED.
2. Changed **`MODULE_TITLE`** value to reflect **`PRODUCT_X`**, not the original discipline source.
3. Added **`HEADER`**, **`CENTRE_COORD`**, and **`DATA_1`** entity definitions.

```
/*************************************************************************/
/***           Identification of the Product X dictionary              ***/
/*************************************************************************/

BEGIN_GROUP = MODULE_IDENTIFICATION;
     DEDSL_VERSION = 0.6;
     MODULE_TITLE = "Product X DED. It is based on the Project ABC DED
                     registered as NSSD1393 and it extends those definitions
                     as needed";
     MODULE_ADID = NSSD2245;
END_GROUP = MODULE_IDENTIFICATION;

/*************************************************************************/
/***  Entity definitions in terms of standard attributes                 ***/
```

```
/*************************************************************************/

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  PRODUCT_ID;
     MEANING = "The PRODUCT_ID parameter represents a permanent, unique
                identifier assigned to a data product by its producer.";
     SHORT_MEANING = "Product Identification parameter";
     ALIAS = (PRODUCT_NAME, "Used by the historical projects H, I, K to
                             identify their data products");
     VALUE_SYNTAX = VisibleString(SIZE(1..40))
     COMMENT = "The definition has been extracted from the Planetary
                Science Data Dictionary Document, Issue V6 MOSO0099--4-00,
                20 November 1992;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  ACQ_STATION;
     MEANING = "The ACQ_STATION includes the identifier of the station,
                which has acquired the data.";
     SHORT_MEANING = "Identifier of acquisition station";
     VALUE_SYNTAX = ENUMERATION {STAT_1(1), STAT_2(2), STAT_3(3)};
     SPECIFIC_INSTANCE = (STAT_1, "Station 1 identifier");
     SPECIFIC_INSTANCE = (STAT_2, "Station 2 identifier");
     SPECIFIC_INSTANCE = (STAT_3, "Station 3 identifier");
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  ACQ_TIME;
     MEANING = "Date and time of acquisition of the data. The rules to
                format this parameter are: 'YYYY-MM-DDThh:mm:ss.d->Z'. An
                example is: '1988-01-18T17:20:43.123456Z'. The acquisition
                time should correspond to the first scan line of the
                data.";
     SHORT_MEANING = "Date/Time of data acquisition";
     VALUE_SYNTAX = VisibleString(FROM("0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"
                    |"9"|"-"|":"|"."|"T"|"Z"))(SIZE(21..40));
     COMMENT = "The date/time is defined according to the CCSDS/ISO rules
                for data/time definitions";
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  LATITUDE;
     MEANING = "Latitudes north of the equator shall be designated by the
                use of the plus (+) sign, latitudes south of the equator
                shall be designated by the use of the minus sign (-). The
                equator shall be designated by the use of the plus sign
                (+).";
     SHORT_MEANING = "Latitude";
     UNITS = deg ;
     ALIAS = (LAT, " Used by the historical projects H, I, K ");
     VALUE_SYNTAX = REAL (-90.000..+90.000);
     SPECIFIC_INSTANCE = (+00.000, "Equator");
     COMMENT = "The definition of this parameter is according to ISO 6709";
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  LONGITUDE;
```

```
     MEANING = "Longitudes east of Greewhich shall be designated by the use
                of the plus sign (+), longitudes west of Greenwhich shall
                be designated by the use of the minus sign (-).  The Prime
                Meridian shall be designated by the use of the plus sign
                (+). The 180th meridian shall be designated by the use of
                the minus sign (-)";
     SHORT_MEANING = "Longitude";
     UNITS = deg ;
     ALIAS = (LON, " Used by the historical projects H, I, K ");
     VALUE_SYNTAX = REAL (-180.000..+180.000);
     SPECIFIC_INSTANCE = (-180.000, "The 180th Meridian");
     COMMENT = "The definition of this parameter is according to ISO 6709";
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  HEADER;
     MEANING = "Identifies an aggregation of values which are associated
                with an image array.  These values are named PRODUCT_ID,
                ACQ_STATION, ACQ_TIME and CENTRE_COORD";
     SHORT_MEANING = "Image Header Values";
     COMMENT = "The definition of this entity is not included in the
                discipline dictionary registered as NSSD1393, revision 0.";
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  CENTRE_COORD;
     MEANING = "Identifies a pair of latitude and longitude values which
                are taken as the centre coordinates for the associated data
                array.  These values are named LATITUDE and LONGITUDE.";
     SHORT_MEANING = "Center latitude and longitude values";
     COMMENT = "The definition of this entity is not included in the
                discipline dictionary, version 0.1, registered as NSSD1393,
                revision 0.;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  "DATA_1";
     MEANING = "An image taken from spacecraft W.
     SHORT_MEANING = "Spacecraft W Image";
     COMMENT = "The definition of this entity is not included in the
                discipline dictionary registered as NSSD1393, revision 0.";
END_GROUP = ENTITY_DEFINITION;
```

5.2.2 Use of the DEDSL for Product Y

In the following DED, the following changes from the Product X DED above have been made:

1. Removed definitions of entities not present in Product Y.
2. Specialised the meanings of **LATITUDE** and **LONGITUDE** by adding a sentence to give the semantics of **CENTER_COORD**; also removed the **ALIAS** field under the view that the **LAT** and **LON** fields may not have been **CENTER_COORD** when used in past projects; also changed the comment to reflect this specialisation.
3. Replaced the entity definition for **DATA_1** with that for **DATA_2**. Only the entity names have changed in this example, the entity definition has stayed the same.

```
/*************************************************************************/
/***           Identification of the Product Y dictionary              ***/
/*************************************************************************/

BEGIN_GROUP = MODULE_IDENTIFICATION;
     DEDSL_VERSION = 0.6;
     MODULE_TITLE = "Product Y DED. It is based on the Project ABC DED
                     registered as NSSD1393 and it extends those definitions
                     as needed";
     MODULE_ADID = NSSD2246;
END_GROUP = MODULE_IDENTIFICATION;

/*************************************************************************/
/***  Entity definitions in terms of standard attributes               ***/
/*************************************************************************/

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  PRODUCT_ID;
     MEANING = "The PRODUCT_ID parameter represents a permanent, unique
                identifier assigned to a data product by its producer.";
     SHORT_MEANING = "Product Identification parameter";
     ALIAS = (PRODUCT_NAME, "Used by the historical projects H, I, K to
                             identify their data products");
     VALUE_SYNTAX = VisibleString(SIZE(1..40))
     COMMENT = "The definition has been extracted from the Planetary
                Science Data Dictionary Document, Issue V6 MOSO0099--4-00,
                20 November 1992;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  LATITUDE;
     MEANING = "Latitudes north of the equator shall be designated by the
                use of the plus (+) sign, latitudes south of the equator
                shall be designated by the use of the minus sign (-). The
                equator shall be designated by the use of the plus sign
                (+). The latitude is taken as a centre coordinate for the
                associated data array.";
     SHORT_MEANING = "Centre Latitude";
     UNITS = deg ;
     VALUE_SYNTAX = REAL (-90.000..+90.000);
     SPECIFIC_INSTANCE = (+00.000, "Equator");
     COMMENT = "The definition of this parameter has been specialised from
                that given in the discipline dictionary registered as
                NSSD1393";
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  LONGITUDE;
     MEANING = "Longitudes east of Greewhich shall be designated by the use
                of the plus sign (+), longitudes west of Greenwhich shall
                be designated by the use of the minus sign (-).  The Prime
                Meridian shall be designated by the use of the plus sign
                (+). The 180th meridian shall be designated by the use of
                the minus sign (-). The longitude is taken as a centre
                coordinate for the associated data array ";
     SHORT_MEANING = "Centre Longitude";
```

```
     UNITS = deg ;
     ALIAS = (LON, " Used by the historical projects H, I, K ");
     VALUE_SYNTAX = REAL (-180.000..+180.000);
     SPECIFIC_INSTANCE = (-180.000, "The 180th Meridian");
     COMMENT = "The definition of this parameter has been specialised from
                that given in the discipline dictionary registered as
                NSSD1393";
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  "DATA_2";
     MEANING = "An image taken from spacecraft W.
     SHORT_MEANING = "Spacecraft W Image";
     COMMENT = "The definition of this entity is not included in the
                discipline dictionary registered as NSSD1393, revision 0.";
END_GROUP = ENTITY_DEFINITION;
```

6 DEDSL CONFORMANCE

The assignment of unique identifiers, in the form of CCSDS ADIDs, is needed for the DEDSL material in order to facilitate the automated interpretation of data objects conforming to the DEDSL.

This DEDSL specification, is organised to permit two primary ways of using the specification. Section 3 ('ABSTRACT STANDARD') defines a set of standard attributes by name with restrictions on their permitted values. Annex B provides further detailed restrictions on the attribute values. Note that this part of the specification does not specify how the attribute names and values are to be linked to any given data object. This allows a variety of formatting approaches to be used for this linking and this is the first primary way of using the DEDSL. For example, the CDF format and supporting software already provides attribute-value linking. CDF data objects could be written conforming to the specifications of Section 3. Such CDF data objects could be SFDU encapsulated and thus described by a registered description. This registered description could indicate that the CDF data objects were constrained to conform to DEDSL Section 3. This would be done by referencing the ADID assigned to DEDSL Section 3 within the registered description.

The second primary way of using the DEDSL is to use the material of Section 4 ('IMPLEMENTATION USING PVL') with that of the previous section. As an example of usage, consider a data object with an associated registered description. The description could contain an LVO with Class ID = E (meaning semantic description data), which in turn, contains a set of attributes and values that apply to numerous instances of the data object type. If the attributes in the LVO with Class ID = E conformed to the DEDSL specification of both Sections 3 and 4, then it would carry the ADID assigned jointly to both Sections 3 and 4 of this Recommendation.

Therefore, to support automated recognition of data conforming to the DEDSL, it is necessary to assign two unique identifiers or ADIDs. This results in two levels of DEDSL conformance.

6.1 Conformance Level 1: Abstract DEDSL (ADID = ZCSD0011)

A data object, recognised as being described by an ADID of **`ZCSD0011`**, shall be in conformance with the following sections of this Recommendation:

Section 1, Section 2, Section 3 and Annex B

6.2 Conformance Level 2: Complete DEDSL (ADID = ZCSD0012)

A data object, recognised as being described by an ADID of **`ZCSD0012`**, shall be in conformance with the following sections of this Recommendation:

Section 1, Section 2, Section 3, Section 4 and Annex B

ANNEX A: SCENARIOS OF DEDSL USAGE

(This annex **is not** part of the Recommendation)

Annex A.1: Using the DEDSL with a Formal Data Description Language

This scenario describes the process of creating the semantic description for a data structure which has been described using the formal syntax data description language EAST (see references [1] and [2]). This scenario assumes that the syntax description has been produced prior to the creation of the formal semantic description information.

This scenario uses the example in Figure 2-1 for Product X. The EAST description itself is not shown here. The following is the stages that would be followed so as to produce the semantic description of the data:

- The data producer uses an interpreter of the EAST language to provide the list of fully qualified names of the data entities in the product. This includes the names of the base type entities and also the names of the higher level aggregated data structures. The names are:

```
PRODUCT_X
PRODUCTS_X.HEADER
PRODUCTS_X.HEADER.PRODUCT_ID
PRODUCTS_X.HEADER.ACQ_STATION
PRODUCTS_X.HEADER.ACQ_TIME
PRODUCTS_X.HEADER.CENTRE_COORD
PRODUCTS_X.HEADER.CENTRE_COORD.LATITUDE
PRODUCTS_X.HEADER.CENTRE_COORD.LOGITUDE
PRODUCTS_X.DATA_1
```

- The data producer creates a DEDSL module using a word processor, or a DEDSL creation utility if available, which takes as input the ASCII file of entity names produced by the EAST interpreter. The resultant DEDSL is shown below. This DEDSL module is subtlety different to that shown in 5.2.1 as the entity **NAME** parameter values match those generated by the EAST interpreter tool. One decision the data producer must make is what data objects are relevant from a semantic viewpoint and therefore must have data entity definitions provided.

```
/*************************************************************************/
/***           Identification of the Product X dictionary              ***/
/*************************************************************************/

BEGIN_GROUP = MODULE_IDENTIFICATION;
     DEDSL_VERSION = 0.6;
     MODULE_TITLE = "Product X DED. It is based on the Project ABC DED
                     registered as NSSD1393 and it extends those definitions
                     as needed";
     MODULE_ADID = FNES1234;
END_GROUP = MODULE_IDENTIFICATION;

/*************************************************************************/
```

```
/***  Entity definitions in terms of standard attributes              ***/
/*************************************************************************/

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  PRODUCT_X.HEADER.PRODUCT_ID;
     MEANING = "This parameter represents a permanent, unique identifier
                assigned to a data product by its producer.";
     SHORT_MEANING = "Product Identification parameter";
     ALIAS = (PRODUCT_NAME, "Used by the historical projects H, I, K to
                             identify their data products");
     VALUE_SYNTAX = VisibleString(SIZE(1..40))
     COMMENT = "The definition has been extracted from the Planetary
                Science Data Dictionary Document, Issue V6 MOSO0099--4-00,
                20 November 1992;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME = PRODUCT_X.HEADER.ACQ_STATION;
     MEANING = "The ACQ_STATION includes the identifier of the station,
                which has acquired the data.";
     SHORT_MEANING = "Identifier of acquisition station";
     VALUE_SYNTAX = ENUMERATION {STAT_1(1), STAT_2(2), STAT_3(3)};
     SPECIFIC_INSTANCE = (STAT_1, "Station 1 identifier");
     SPECIFIC_INSTANCE = (STAT_2, "Station 2 identifier");
     SPECIFIC_INSTANCE = (STAT_3, "Station 3 identifier");
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME = PRODUCT_X.HEADER.ACQ_TIME;
     MEANING = "Date and time of acquisition of the data. The rules to
                format this parameter are: 'YYYY-MM-DDThh:mm:ss.d->Z'. An
                example is: '1988-01-18T17:20:43.123456Z'. The acquisition
                time should correspond to the first scan line of the
                data.";
     SHORT_MEANING = "Date/Time of data acquisition";
     VALUE_SYNTAX = VisibleString(FROM("0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"
                    |"9"|"-"|":"|"."|"T"|"Z"))(SIZE(21..40));
     COMMENT = "The date/time is defined according to the CCSDS/ISO rules
                for data/time definitions";
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME = PRODUCT_X.HEADER.CENTRE_COORD.LATITUDE;
     MEANING = "Latitudes north of the equator shall be designated by the
                use of the plus (+) sign, latitudes south of the equator
                shall be designated by the use of the minus sign (-). The
                equator shall be designated by the use of the plus sign
                (+).";
     SHORT_MEANING = "Latitude";
     UNITS = deg ;
     ALIAS = (LAT, " Used by the historical projects H, I, K ");
     VALUE_SYNTAX = REAL (-90.000..+90.000);
     SPECIFIC_INSTANCE = (+00.000, "Equator");
     COMMENT = "The definition of this parameter is according to ISO 6709";
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
```

```
     NAME = PRODUCT_X.HEADER.CENTRE_COORD.LONGITUDE;
     MEANING = "Longitudes east of Greewhich shall be designated by the use
                of the plus sign (+), longitudes west of Greenwhich shall
                be designated by the use of the minus sign (-).  The Prime
                Meridian shall be designated by the use of the plus sign
                (+). The 180th meridian shall be designated by the use of
                the minus sign (-)";
     SHORT_MEANING = "Longitude";
     UNITS = deg ;
     ALIAS = (LON, " Used by the historical projects H, I, K ");
     VALUE_SYNTAX = REAL (-180.000..+180.000);
     SPECIFIC_INSTANCE = (-180.000, "The 180th Meridian");
     COMMENT = "The definition of this parameter is according to ISO 6709";
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  PRODUCT_X.HEADER;
     MEANING = "Identifies an aggregation of values which are associated
                with an image array.  These values are named PRODUCT_ID,
                ACQ_STATION, ACQ_TIME and CENTRE_COORD";
     SHORT_MEANING = "Image Header Values";
     COMMENT = "The definition of this entity is not included in the
                discipline dictionary, version 0.1, registered as NSSD1393,
                revision 0.";
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME = PRODUCT_X.HEADER.CENTRE_COORD;
     MEANING = "Identifies a pair of latitude and longitude values which
                are taken as the centre coordinates for the associated data
                array.  These values are named LATITUDE and LONGITUDE.";
     SHORT_MEANING = "Center latitude and longitude values";
     COMMENT = "The definition of this entity is not included in the
                discipline dictionary, version 0.1, registered as NSSD1393,
                revision 0.;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  PRODUCT_X.DATA_1;
     MEANING = "An image taken from spacecraft W.
     SHORT_MEANING = "Spacecraft W Image";
     COMMENT = "The definition of this entity is not included in the
                discipline dictionary, version 0.1, registered as NSSD1393,
                revision 0.";
END_GROUP = ENTITY_DEFINITION;
```

- The data producer registers the syntax description written in EAST and the semantic description written in DEDSL (**ZCCSD0012**), at a Control Authority Office (see references [9] and [10]) and is assigned an ADID which can then be attached to all the application data.

Annex A.2: Using the DEDSL with FITS

This section describes the data Product X from Section 2 as it would appear if formatted as a FITS file. It also demonstrates the packaging of the DEDSL with the formatted data, in this case FITS, within an SFDU. For FITS products, there are FITS keywords that appear in a

header section, with the length of keywords limited to 8 characters. This limitation leads to some changes to the DEDSL but these can be kept to a minimum.

The approach here has been to use the DEDSL **ALIAS** attribute to indicate the name of the corresponding FITS header keyword to each of the entity names in Product X and Y. An alternative would have been to use the truncated keyword as the **NAME** and indicate the longer name as the **ALIAS**.

Note also that the concept of **CENTRE_COORD** has not been captured here, however in the particular case of coordinates FITS has its own semantics and it may have been better to use these standard FITS keywords. For example **CRPIX1**, **CRVAL1** and **CRDELT1** to indicate the central pixel along axis 1, its coordinate value and the rate of change of coordinate with pixel number.

The size of the data array and its data type is indicated by the standard FITS keywords as follows:

BITPIX a value of 16 indicates 16 bit signed integers, had the values been unsigned 16 bit integers then the data would to have been transformed into 32 bit integers

NAXIS '2' indicates a 2-D array

NAXIS1 the first dimension

NAXIS2 the second dimension

The assumption has been made here that the 16 bit integers are signed integers. The header of the FITS file containing an instance of the data Product X would appear as follows:

```
SIMPLE  =                    T
BITPIX  =                   16
NAXIS   =                    2
NAXIS1  =                  256
NAXIS2  =                  256
PROD_ID = 'PRODUCT_X'                    /Product_id
ACQUTIME= '1996-06-19T17:20:43.12345Z' /Acquisition_time
ACQUSTAT= 'STAT_1'                       /Acquisition_station
LATITUDE=              -37.456           /Latitude
LONGITUD=              +46.996           /Longitude
  -
  -
DEDID   = 'NSSD1234'
  -
END
¾$@¿ý%¹ü@;ür@´LÍ!ýó¥ü‹÷¿@--‹è²@é•$vis4Òÿ»éá@¸LÍs!¸@ûÿ?d@±@Óà•À¿þÿò°Ãüò®ã@
ë@ñ•ó@øÃ&•=•uíúßG}@úuæù@òßÿþù@suØù@}ôuÑù@ëÊù÷ß@üuÐ}@@u¼ù¿•@uduµùøù÷ïýuùu™
ù±•Âu'ù@u‹ÿù@èu„&Ç@.ÿûïÃE@.Aú@kúwÞ@î@.EúÇø@î@gùÃŠ>ÿq3´@°@Í@¾Ù¿Æúkƒu@ú•Çlu
@úmuún#@u@úo•@Ð••@àñ,Ì‹MMAûUO»•ÄÆ@Ís@3ÀúNJVÿñ¾¬@´Óëõ^ÿñÃ‹óƒÆ@ƒÇ@¹ëó¦tè@@à
yà=ú@»•@‹ú@‹@ü@³@;çÓ^Ó@ti'ï?¿@¶¾Æßo‹Ls‹TOŠ@@@µ{ßa°°=ïOuá¨~wï=Ã×ïðç@í@••?@
þuã•j&uúkƒu@ú•Çl.u@úmuún#@u@úo•@Ð••
```

The DED entries expressed in DEDSL would be as follows:

```
/*************************************************************************/
/***           Identification of the Product X dictionary              ***/
/*************************************************************************/

BEGIN_GROUP = MODULE_IDENTIFICATION;
     DEDSL_VERSION = 0.6;
     MODULE_TITLE = "Product X DED. It is based on the Project ABC DED
                     registered as NSSD1393 and it extends those definitions
                     as needed";
     MODULE_ADID = BRAL3266;
END_GROUP = MODULE_IDENTIFICATION;

/*************************************************************************/
/***  Entity definitions in terms of standard attributes                 ***/
/*************************************************************************/

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME =  PRODUCT_X;
     MEANING = "This parameter represents a permanent, unique identifier
                assigned to a data product by its producer.";
     SHORT_MEANING = "Product Identification parameter";
     ALIAS = (PRODUCT_NAME, "Used by the historical projects H, I, K to
                             identify their data products");
     ALIAS = (PROD_ID, "Used in the FITS header");
     VALUE_SYNTAX = VisibleString(SIZE(1..40))
     COMMENT = "The definition has been extracted from the Planetary
                Science Data Dictionary Document, Issue V6 MOSO0099--4-00,
                20 November 1992;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME = ACQ_STATION;
     MEANING = "The ACQ_STATION includes the identifier of the station,
                which has acquired the data.";
     SHORT_MEANING = "Identifier of acquisition station";
     VALUE_SYNTAX = ENUMERATION {STAT_1(1), STAT_2(2), STAT_3(3)};
     ALIAS = (ACQUSTAT, "Used in the FITS header");
     SPECIFIC_INSTANCE = (STAT_1, "Station 1 identifier");
     SPECIFIC_INSTANCE = (STAT_2, "Station 2 identifier");
     SPECIFIC_INSTANCE = (STAT_3, "Station 3 identifier");
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME = ACQ_TIME;
     MEANING = "Date and time of acquisition of the data. The rules to
                format this parameter are: 'YYYY-MM-DDThh:mm:ss.d->Z'. An
                example is: '1988-01-18T17:20:43.123456Z'. The acquisition
                time should correspond to the first scan line of the
                data.";
     SHORT_MEANING = "Date/Time of data acquisition";
     VALUE_SYNTAX = VisibleString(FROM("0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"
                    |"9"|"-"|":"|"."|"T"|"Z"))(SIZE(21..40));
     ALIAS = (ACQUTIME, "Used in the FITS header);
     COMMENT = "The date/time is defined according to the CCSDS/ISO rules
                for data/time definitions";
END_GROUP = ENTITY_DEFINITION;
```

```
BEGIN_GROUP = ENTITY_DEFINITION;
     NAME = COORD.LATITUDE;
     MEANING = "Latitudes north of the equator shall be designated by the
                use of the plus (+) sign, latitudes south of the equator
                shall be designated by the use of the minus sign (–). The
                equator shall be designated by the use of the plus sign
                (+).";
     SHORT_MEANING = "Latitude";
     UNITS = deg ;
     ALIAS = (LAT, " Used by the historical projects H, I, K ");
     VALUE_SYNTAX = REAL (–90.000..+90.000);
     SPECIFIC_INSTANCE = (+00.000, "Equator");
     COMMENT = "The definition of this parameter is according to ISO 6709";
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
     NAME = LONGITUDE;
     MEANING = "Longitudes east of Greewhich shall be designated by the use
                of the plus sign (+), longitudes west of Greenwhich shall
                be designated by the use of the minus sign (–).  The Prime
                Meridian shall be designated by the use of the plus sign
                (+). The 180th meridian shall be designated by the use of
                the minus sign (–)";
     SHORT_MEANING = "Longitude";
     UNITS = deg ;
     ALIAS = (LON, " Used by the historical projects H, I, K ");
     ALIAS = (LONGITUD, " Used in the FITS header");
     VALUE_SYNTAX = REAL (–180.000..+180.000);
     SPECIFIC_INSTANCE = (–180.000, "The 180th Meridian");
     COMMENT = "The definition of this parameter is according to ISO 6709";
END_GROUP = ENTITY_DEFINITION;
```

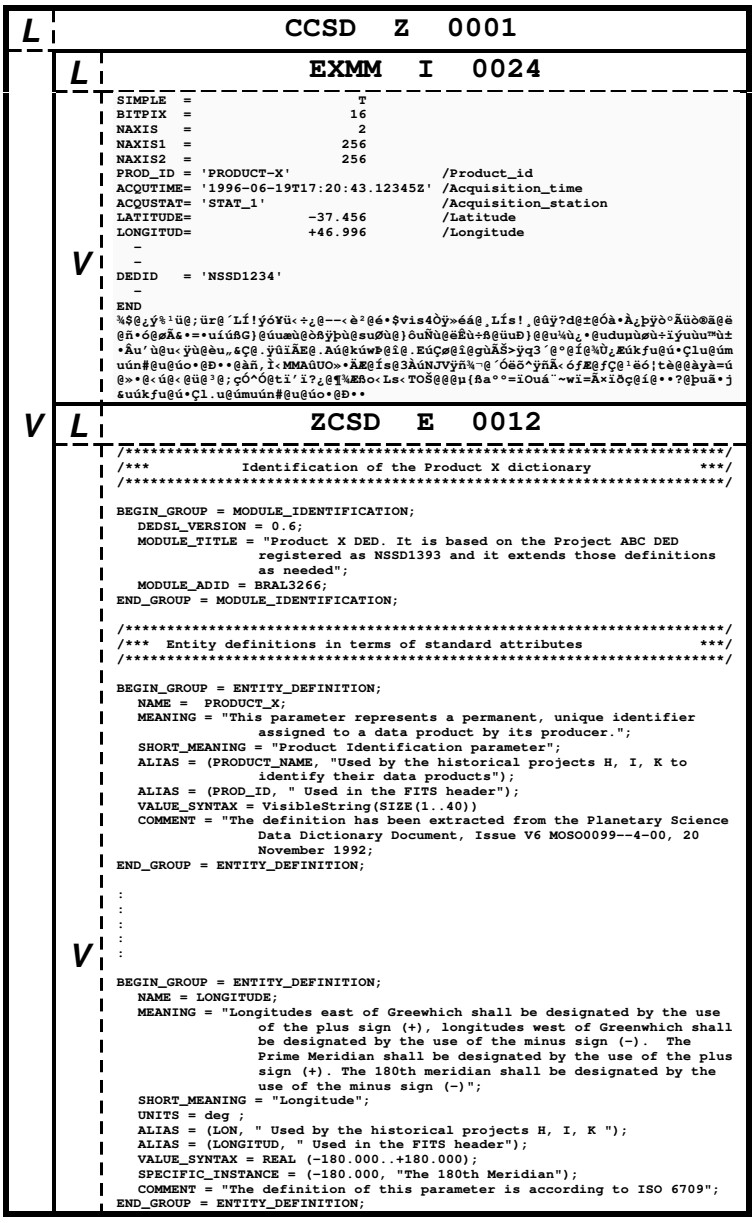
Note that the **DEDID** keyword in the FITS header is a local convention that indicates the discipline DED that is being used as the basis of the DED for this particular product. A similar technique could be used for any formatting technique which includes some form of header information, but in each case a convention would have to be agreed in the relevant formatting domain so that this non-standard keyword can be generally understood.

For Product Y the FITS header would be changed to indicate the fact that the data values are held as REALs - in this case as IEEE 4-byte REAL numbers.

```
SIMPLE  =                    T
BITPIX  =                   32
NAXIS   =                    2
NAXIS1  =                  256
NAXIS2  =                  256
PROD_ID = 'PRODUCT_Y'                  /Product_id
ACQUTIME= '1996-06-19T17:20:43.12345Z' /Acquisition_time
ACQUSTAT= 'STAT_1'                     /Acquisition_station
LATITUDE=              -37.456         /Latitude
LONGITUD=              +46.996         /Longitude
  –
  –
DEDID   = 'NSSD1234'
  –
END
üò®ã@ë@ñ•ó@øÃ&•=•uíúßG}@úuæù@òßÿþù@s¾Æßo<Ls<TOŠ@@@µ{ßa°°=ïOuá¨~wï=Ã×ïðç@í
```

```
@••?@þuã•j&uúkƒu@ú•Çl.u@úmuún#@u@úo•@Ð••uØù@}ôuÑù@ëÊù÷ß@üuÐ}@@u¼ù¿•@uduµù
øù÷ï¾$@¿ý%¹ü@;ür@´LÍ!ýó¥ü‹÷¿@−−‹è²@é•$vis4Òÿ»éá@¸LÍs!¸@ûÿ?d@±@Óà•À¿þÿò°Ã
ýuùu™ùúmuún#@u@úo•@Ð•±•Âu'ù@u‹ÿù@èu„&Ç@.ÿûïÃE@.Aú@kúwÞ@î@.EúÇø@î@gùÃŠ>ÿq3
´@°@Í@¾Ù¿Æúkƒu@ú•Çlu@•@àñ,Ì‹MMAûUO»•ÄÆ@Ís@3ÀúNJVÿñ¾¬@´ÓëõˆÿñÃ‹ófÆ@ƒÇ@¹ëó¦
tè@@àyà=ú@»•@‹ú@‹@ü@³@;çÓ^Ó@tï'ï?¿@¶
```

In both cases, the data in the FITS file and the semantic definition in the DEDSL file could then be packaged together in an SFDU as shown schematically below, so as to convey the full meaning of the FITS data with the actual data itself, therefore leading to a greater probability of the data being understood and hence of value in the future.



Rather than package the Class E LVO with the FITS data each time, it would also be possible to register the DEDSL module with a Control Authority (see References [9] and [10]) and then request and retrieve it from the Control Authority just the once.

Annex A.3: Using the DEDSL with CDF

In this scenario, we will begin by using the data representation capabilities of the Common Data Format (CDF) to carry the data entities defined as Product X in Section 2 (see Figure 2-1 and Table 2-1). Then, we will add the DEDSL defined attributes following conformance to DEDSL level 1 (see Section 6.1). Finally, we will also show a CDF, with DEDSL attributes, for Product Y of Section 2.

A CDF models its data as containing a set of global attributes (describing the overall CDF content) together with one or more variables. Each variable may have any number of additional attributes, called 'variable attributes' associated with it. A variable has values that may be a simple data type or as complex as a multi-dimensional array of simple data types. Attributes have simple data types. The values of CDF variables are conceptually organised into a sequence of records, and therefore it is necessary to specify whether or not the values of specific variables vary from record to record.

A project has chosen the Common Data Format (CDF) as the file format and access mechanism that it will use to carry most of its primary data of interest. The project will attempt to conform to the discipline dictionary of Section 5.1 while incorporating the data of Product X into a CDF. For simplicity and inter comparison of examples, only one instance of Product X will be put into a single CDF file.

There are many ways such a simple data product could be mapped into a CDF. **DATA_1**, which is an array, will certainly be a variable. However, the parameters **PRODUCT_ID**, **ACQ_STATION**, **ACQ_TIME**, **LATITUDE**, and **LONGITUDE** could be global attributes because they are constant for the single data array **DATA_1** and therefore for the whole CDF. They could also be incorporated as attributes of the variable **DATA_1**, or they could be incorporated as additional variables. We choose to map them as additional variables so that we can use the variable attribute capability of CDF to carry the DEDSL defined attributes that will enhance the understanding of the variables. Note, for a general CDF where it is desirable to enhance the understanding of not only the variables, but also some of the global attributes and variable attributes, we would generate a separate DED module containing DEDSL attributes following conformance to DEDSL level 2 (see Section 6.2) and we would associate this DED module with the CDF file in one of several ways.

A simple CDF, consisting of one global attribute and seven 'variable' entities, is created for Product X as described in Figure 2-1 and Table 2-1. A structured textual overview of this CDF file is shown below. This view is similar to the CDF standard text object called a 'CDF Skeleton Table' which serves as a template for producing multiple instances of the CDF type being described.

```
P_Xyyddd            (CDF file name with yy for year and ddd for day)

Global attributes:  (Attributes that apply to the whole product; each has a
                     'name', 'CDF data type' and 'value')

  Product Name      CDF_CHAR "Product X"

zVariables and their attributes: (Each variable [called a zVariable] is
                     given by its 'name', its 'CDF data type', and its
                     dimensionality and sizes; each attribute (if present)
```

```
                    is given by it 'name', 'CDF data type', and its
                    'value'.  No variable attributes are included in this
                    figure.)

  DATA_1            CDF_UINT2  #Elements=1, #Dim=2, Dim Sizes=256/256,
                               Rec Var= F, Dim Var=T/T

  PRODUCT_ID                CDF_CHAR   #Elements=40, #Dim=0, Rec Var= F

  ACQ_STATION               CDF_CHAR   #Elements=32, #Dim=0, Rec Var= F

  ACQ_TIME                  CDF_CHAR   #Elements=32, #Dim=0, Rec Var= F

  LATITUDE                  CDF_REAL4  #Elements=1, #Dim=0, Rec Var= F

  LONGITUDE                 CDF_REAL4  #Elements=1, #Dim=0, Rec Var= F

  ACQ_TIME_CDF              CDF_EPOCH  #Elements=1, #Dim=0, Rec Var= F
```

In this example, the template gives the form of the name to be used for the CDF file as **P_Xyyddd**. The file name is not included as part of the CDF internal file structure.

The first, and only, global attribute is called **Product Name** and is given the value of **Product X**.

The first variable of the CDF is named **DATA_1**. It is given a data type of CDF_UINT2 which is a 16-bit, unsigned, integer. The information starting with "#Elements=1" indicates that the integer values are understood to comprise a 2-dimensional array whose dimensions are 256 by 256. Further, these values are understood to not vary from record to record, but they do vary within each of the dimensions. Note that there will be only one CDF conceptual record because there is only one instance of the array **DATA_1** that we will put into each CDF file.

The second variable is named **PRODUCT_ID**. It is given a data type of CDF_CHAR which is an ASCII character. The value is a string of up to 40 characters (#Elements = 40), in conformance with the discipline dictionary definition in Section 5.1 and of course it does not vary from record to record.

The other variables are similarly defined. Note that **ACQ_TIME_CDF** is not one of the entities defined for Product X in Figure 2-1. It has been included here because CDF provides specific support for a time data type designated as "CDF_EPOCH." Therefore the ASCII time string values of **ACQ_TIME** have been converted to a variable with a data type of "CDF_EPOCH."

Note also that the Product X definition of Figure 2-1 includes the identifier of **HEADER** for the aggregation of **PRODUCT_ID, ACQ_STATION, ACQ_TIME**, and **CENTRE_COORD**. Further, note that **CENTRE_COOR** is an aggregation of **LATITUDE** and **LONGITUDE**. Neither **HEADER** nor **CENTRE_COOR** have been included in the CDF above because there is no standard way to show the hierarchical naming of collections of variables. One approach might have been to build this hierarchy into the variable names, such as **HEADER.CENTRE_COORD.LATITUDE**. This was not done to keep the example simple and more in line with a more likely mapping into CDF.

Few data producers using CDF would produce a CDF as shown above because none of the variables have been given any attribute values. Attributes are usually used to give **UNITS** and

other types of information (e.g. minimum and maximum values) associated with each variable so they are more understandable and useful to recipients. The DEDSL recommendation provides standard names to be used for some common attributes associated with entities. An approach to applying the DEDSL recommendations for level 1 conformance to the example above results in the CDF example below.

```
P_Xyyddd            (CDF file name with yy for year and ddd for day)

Global attributes:  (Attributes that apply to the whole product; each has a
                     'name', 'CDF data type' and 'value')

  Product Name      CDF_CHAR "Product X"

zVariables and their attributes: (Each variable [called a zVariable] is
                     given by its 'name', its 'CDF data type', and its
                     dimensionality and sizes; each attribute (if present)
                     is given by its 'name', 'CDF data type', and its
                     'value'.)

  DATA_1            CDF_UINT2  #Elements=1, #Dim=2, Dim Sizes=256/256,
                               Rec Var= F, Dim Var=T/T
    Attributes:
    NAME            CDF_CHAR   "DATA_1"
    MEANING         CDF_CHAR   "Image taken from spacecraft R using the Z
                                instrument"
    SHORT_MEANING   CDF_CHAR   "Image from Z Instrument"

  PRODUCT_ID        CDF_CHAR   #Elements=40, #Dim=0, Rec Var= F
    Attributes:
    NAME            CDF_CHAR   "PRODUCT_ID"
    MEANING         CDF_CHAR   "The PRODUCT_ID parameter represents a
                                permanent, unique, identifier, assigned to a
                                data product by its producer."
    SHORT_MEANING   CDF_CHAR   "Product Identification parameter"
    ALIAS           CDF_CHAR   "(PRODUCT_NAME, 'Used by the historical
                                projects H, I, K to identify their data
                                products.')"

  ACQ_STATION       CDF_CHAR   #Elements=32, #Dim=0, Rec Var= T
    Attributes:
    NAME            CDF_CHAR   "ACQ_STATION"
    MEANING         CDF_CHAR   "The ACQ_STATION includes the identifier of
                                the station, which has acquired the data."
    SHORT_MEANING   CDF_CHAR   "Identifier of acquisition station"
    VALUE_SYNTAX    CDF_CHAR   "ENUMERATION {STAT_1(1), STAT_2(2),
                                STAT_3(3)}"
    SPECIFIC_INSTANCE  CDF_CHAR "((STAT_1, 'Station 1 Identifier'),(STAT_2,
                                'Station 2 Identifier'), (STAT_3, 'Station 3
                                Identifier'))"

  ACQ_TIME          CDF_CHAR   #Elements=32, #Dim=0, Rec Var= T
    Attributes:
    NAME            CDF_CHAR   "ACQU_TIME"
    MEANING         CDF_CHAR   "Date and time of acquisition of the data.
                                The rules to format this parameter are:
```

```
                                'YYYY-MM-DDThh:mm:ss.d->Z'. An example is:
                                '1988-01-18T17:20:43.123456Z'. The
                                acquisition time should correspond to the
                                first scan line of the data."
   SHORT_MEANING   CDF_CHAR     "Date/Time of data acquisition"
   COMMENT         CDF_CHAR     "The date/time is defined according to the
                                CCSDS/ISO rules for date/time def."

 LATITUDE          CDF_REAL4    #Elements=1, #Dim=0, Rec Var= T
   Attributes:
   NAME            CDF_CHAR     "LATITUDE"
   MEANING         CDF_CHAR     "Latitudes north of the equator shall be
                                designated by the use of the plus (+) sign,
                                latitudes south of the equator shall be
                                designated by the use of the minus sign (-).
                                 The equator shall be designated by the use
                                of the plus sign (+)."
   SHORT_MEANING   CDF_CHAR     "Latitude"
   UNITS           CDF_CHAR     "deg"

 LONGITUDE         CDF_REAL4    #Elements=1, #Dim=0, Rec Var= T
   Attributes:
   NAME            CDF_CHAR     "LONGITUDE"
   MEANING         CDF_CHAR     "Longitudes east of Greenwhich shall be
                                designated by the use of the plus sign (+),
                                longitudes west of Greenwhich shall be
                                designated by the use of the minus sign (-).
                                 The Prime Meridian shall be designated by
                                the use of the plus sign (+). The 180th
                                meridian shall be designated by the use of
                                the minus sign (-)"
   SHORT_MEANING   CDF_CHAR     "Longitude"
   UNITS           CDF_CHAR     "deg"

 ACQ_TIME_CDF      CDF_EPOCH    #Elements=1, #Dim=0, Rec Var= T
   Attributes:
   NAME            CDF_CHAR     "ACQ_TIME_CDF"
   MEANING         CDF_CHAR     "Time interval, from the reference date/time
                                of 01 Jan 0000, 0H,0M,0S to acquisition of
                                the data. The acquisition time should
                                correspond to the first scan line of the
                                data."
   SHORT_MEANING   CDF_CHAR     "Time to data acquisition"
   UNITS           CDF_CHAR     "ms"
```

Starting with the variable **DATA_1**, the first DEDSL attribute added is **NAME**. **NAME** is mandatory and its value is given as **DATA_1**. Normally **NAME** is needed to make the link from the DEDSL attributes to the data entity being described. However for a CDF variable, the **NAME** information expressed as a variable attribute is redundant because there is no ambiguity about the association of variable attributes with each variable. The **MEANING** gives a text description of the variable **DATA_1**, while **SHORT_MEANING** gives a short form of this meaning that may be used for labelling purposes.

The attributes for the other variables are handled similarly. The attribute **ALIAS** under **PRODUCT_ID** requires two string fields for its value. The CDF standard does not recognise

multiple strings for an attributes value, so a local convention using parenthesis within the text string has been employed. This has also been done for the attributes **VALUE_SYNTAX** and **SPECIFIC_INSTANCE** under the variable **ACQ_STATION**. Since a given CDF variable can not have multiple occurrences of the same attribute name, the association of pairs of values with **SPECIFIC_INSTANCE** is extended by a local convention using pairs of parenthesis. Three pairs of values are associated with **SPECIFIC_INSTANCE** in this example.

The meanings for **LATITUDE** and **LONGITUDE** are, currently, as given in the discipline dictionary. However, because we are unable to associate these variables as an aggregation under **CENTRE_COORD**, we have lost the information that they are also centre coordinates. There are various non-standard ways to add this information back. We could extend the **MEANING** attributes by adding such a sentence. Alternatively, we could refer to the discipline dictionary and indicate that the additional semantics in this attribute is a specialisation of the same term in that dictionary. Additional attributes, which should be standardised, would need to be defined to accomplish this function.

Turning now to Product Y of Section 2, we produce a nearly identical CDF with DEDSL attributes mapped into the variable attributes. Below is a textual (template) view of such a CDF. It is nearly identical to the template above for PRODUCT X, with the following exceptions:

1. A new file name format to reflect new product type
2. Replace **Product X** with **Product Y**
3. Replace **DATA_1** with **DATA_2**
4. Make data type of **DATA_2** equal to CDF_REAL4 which is a 32-bit real number, in accordance with Table 2-1.
5. Delete variables **ACQ_STATION**, **ACQ_TIME**, and **ACQ_TIME_CDF**.

Note that the same issue, with regard to incorporating the semantics of centre coordinates, still arises.

```
P_Yyyddd            (CDF file name with yy for year and ddd for day)

Global attributes:  (Attributes that apply to the whole product; each has a
                     'name', 'CDF data type' and 'value')

  Product Name      CDF_CHAR "Product Y"

zVariables and their attributes: (Each variable [called a zVariable] is
                     given by its 'name', its 'CDF data type', and its
                     dimensionality and sizes; each attribute (if present)
                     is given by its 'name', 'CDF data type', and its
                     'value'.)


  DATA_2            CDF_REAL4  #Elements=1, #Dim=2, Dim Sizes=256/256,
                               Rec Var= F, Dim Var=T/T
    Attributes:
    NAME            CDF_CHAR   "DATA_2"
```

```
    MEANING         CDF_CHAR   "Image taken from spacecraft R using the Z
                                instrument"
    SHORT_MEANING   CDF_CHAR   "Image from Z Instrument"

PRODUCT_ID        CDF_CHAR   #Elements=40, #Dim=0, Rec Var= F
    Attributes:
    NAME            CDF_CHAR   "PRODUCT_ID"
    MEANING         CDF_CHAR   "The PRODUCT_ID parameter represents a
                                permanent, unique, identifier, assigned to a
                                data product by its producer."
    SHORT_MEANING   CDF_CHAR   "Product Identification parameter"
    ALIAS           CDF_CHAR   "(PRODUCT_NAME, 'Used by the historical
                                projects H, I, K to identify their data
                                products.')"

LATITUDE          CDF_REAL4  #Elements=1, #Dim=0, Rec Var= T
    Attributes:
    NAME            CDF_CHAR   "LATITUDE"
    MEANING         CDF_CHAR   "Latitudes north of the equator shall be
                                designated by the use of the plus (+) sign,
                                latitudes south of the equator shall be
                                designated by the use of the minus sign (-).
                                 The equator shall be designated by the use
                                of the plus sign (+)."
    SHORT_MEANING   CDF_CHAR   "Latitude"
    UNITS           CDF_CHAR   "deg"

  LONGITUDE         CDF_REAL4  #Elements=1, #Dim=0, Rec Var= T
    Attributes:
    NAME            CDF_CHAR   "LONGITUDE"
    MEANING         CDF_CHAR   "Longitudes east of Greenwhich shall be
                                designated by the use of the plus sign (+),
                                longitudes west of Greenwhich shall be
                                designated by the use of the minus sign (-).
                                 The Prime Meridian shall be designated by
                                the use of the plus sign (+). The 180th
                                meridian shall be designated by the use of
                                the minus sign (-)"
    SHORT_MEANING   CDF_CHAR   "Longitude"
    UNITS           CDF_CHAR   "deg"
```

In summary, it is reasonable to attempt to use most of the DEDSL attributes at level 1 conformance in the CDF variable attributes to extend understanding of CDF variables. To fully implement all the standard DEDSL attributes, however, an external DED module conforming to level 2 should be constructed and each CDF instance should be tied to this DED module.

The use of a discipline dictionary can provide a good starting point for product documentation, and ways can be found to specialise/extent the semantics form a discipline dictionary to a product dictionary. However, the current DEDSL recommendation provides no direct support for this process other than allowing for a unique, registered, identifier to be assigned to a DED module and carried within that module. Further standardisation in this area would appear desirable.

Annex A.4: Using the DEDSL with HDF

The Hierarchical Data Format (HDF) is designed for the interchange of raster image data and multi-dimensional scientific data sets across heterogeneous environments. It is a multi-object file format, with a number of predefined object types, such as arrays, but with the ability to extend the object types in a relatively simple manner. Recently HDF has been extended to handle arbitrary scientific data, rather than just uniform array oriented data, and also annotation attribute data.

HDF can store several types of data objects within one file, such as raster images, palettes, text and table style data. Each 'object' or ‘entity’ in a HDF file has a predefined tag that indicates the data type and a reference number that identifies the instance. A table of contents is maintained within the file and as the user adds data to the file the pointers in the table of contents are updated. To illustrate the use of the DEDSL with HDF, this section shall demonstrate a technique for integrating the DED (expressed in the DEDSL) with Product X from Section 2 (see Figure 2-1 and Table 2-1).

In this example, all the data within the single HDF product is stored within a single file. The physical format of the file in unknown to the user and the only means of accessing the data is through a software library. The physical storage is in a canonical form and therefore the files can be transferred to other platforms (assuming the software library is also available on the target platform).

HDF has the capability of adding attributes to either the complete file, the groups of entities or each entity within a group. There is no defined format or standard for these attributes they are purely textual strings. Therefore to use the DEDSL in collaboration with a HDF file, one would still use the PVL representation as presented in Section 4 of this Recommendation to define each of the semantic pieces of information, i.e. **`MEANING`**, **`UNITS`**, etc. but the PVL grouping statements (i.e., **`BEGIN_GROUP=ENTITY_ DEFINITION`**) would not be required as the DEDSL definition can be ‘attached’ to each object in the HDF as a single textual string. This is shown schematically in Figure A-1 .

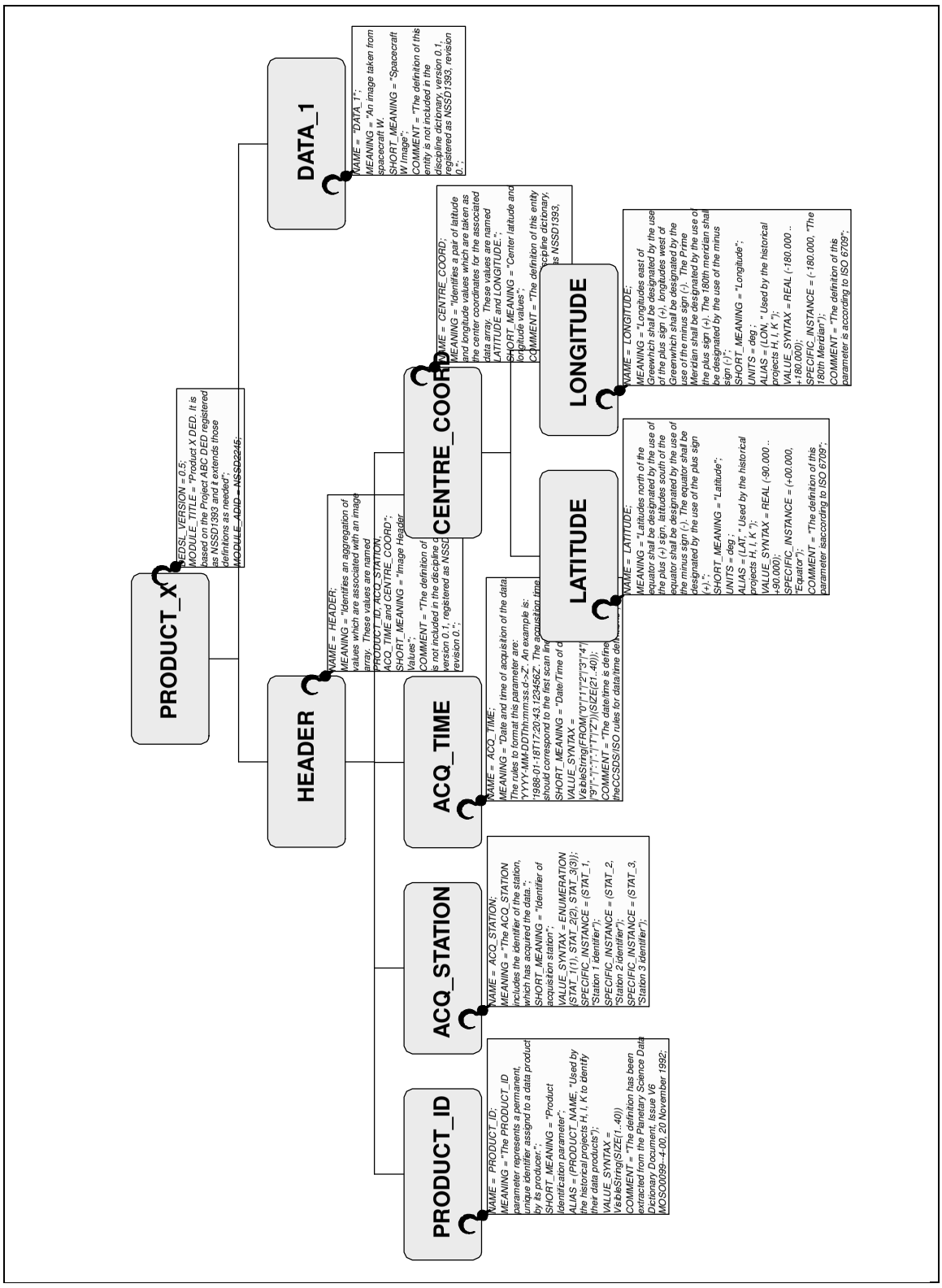


Figure A-1: DEDSL Usage with a HDF File

As can be seen the DEDSL Module Identification attributes are attached as a HDF annotation at the complete file level and then the entity definitions are attached to the separate objects within the HDF file at the object level. Each of the annotations are formatted in PVL as defined in Section 4.

ANNEX B: PERMITTED UNIT VALUES

(This annex **is** part of the Recommendation)

The value of the **UNITS** attribute shall conform to the SI system of units. The representation of these units shall conform to ISO 2955 (see Ref. [17]), this standard should be consulted for full details. This annex presents a list of the 'Form I' (mixed case) units symbols and prefixes that are defined by ISO 2955. These shall be used in the value of a **UNITS** attribute complying with this Recommendation.

Name of Unit	International Symbol	Representation in ISO 2955 'Form I'
Base SI Units		
Metre	m	**m**
kilogram	kg	**kg**
second	s	**s**
ampere	A	**A**
Kelvin	K	**K**
mole	mol	**mol**
candela	cd	**cd**
Supplementary SI Units		
radian	rad	**rad**
steradian	sr	**sr**
Derived SI Units with Special Names		
hertz	Hz	**Hz**
Newton	N	**N**
Pascal	Pa	**Pa**
joule	J	**J**
watt	W	**W**
coulomb	C	**C**
volt	V	**V**
farad	F	**F**
ohm	Ω	**Ohm**
siemens	S	**S**
weber	Wb	**Wb**
tesla	T	**T**
henry	H	**H**
degree Celsius	°C	**Cel**
lumen	lm	**lm**
lux	lx	**lx**
becquerel	Bq	**Bq**

Name of Unit	International Symbol	Representation in ISO 2955 'Form I'
gray	Gy	**Gy**
sievert	Sv	**Sv**
Other Units from ISO 1000		
grade (angle)	g	**gon**
degree (angle)	°	**deg**
minute (angle)	'	**'**
second (angle)	"	**"**
litre	l	**l**
are	a	**a**
hectare	ha	**ha**
minute (time)	min	**min**
hour	h	**h**
day	d	**d**
year	a	**a**
gram	g	**g**
tonne	t	**t**
bar	bar	**bar**
poise	P	**P**
stokes	St	**St**
electronvolt	eV	**eV**
atomic mass unit	u	**u**
astronomic unit	AU	**AU**
parser	pc	**pc**

Table B-1: Representation of Unit Symbols

Any of the units listed above may be multiplied by one of the factors in the following table, this is done by preceding the unit with the string indicated in the 'Representation in ISO 2955 'Form I'' column:

Prefix	Factor by which the unit is multiplied	Internationally Recognised Scientific Symbol	Representation in ISO 2955 'Form I'
exa	10^{18}	E	**E**
peta	10^{15}	P	**P**
tera	10^{12}	T	**T**
giga	10^{9}	G	**G**
mega	10^{6}	M	**M**
kilo	10^{3}	k	**k**
hecto	10^{2}	h	**h**
deca	10^{1}	da	**da**

Prefix	Factor by which the unit is multiplied	Internationally Recognised Scientific Symbol	Representation in ISO 2955 'Form I'
deci	10^{-1}	d	**d**
centi	10^{-2}	c	**c**
milli	10^{-3}	m	**m**
micro	10^{-6}	m	**u**
nano	10^{-9}	n	**n**
pico	10^{-12}	p	**p**
femto	10^{-15}	f	**f**
atto	10^{-18}	a	**a**

Table B-2: Representation of Unit Prefixes

ANNEX C: UNIDATA CONFIGURATION FILE

(This annex **is not** part of the Recommendation)

This Annex shows the configuration file (**udunits.dat**) to be used with the NCAR Unidata units processing software (see Ref. [11]. This file is set up to only accept the units that comply with those listed in Annex B and hence encourage standardisation of units across data sets. The NCAR Unidata units processing software is an example of the type of software that can be used to automate the processing of units.

```
# udunits.dat
#
# The first column is the unit name as should be used as the value for a
# UNITS attribute.
# The second column indicates whether or not the unit name has a plural
# form (i.e. can have an 's' appended).
# A 'P' indicates that the unit has a plural form, whereas, a 'S' indicates
# that the unit has a singular form only.  The remainder of the line is the
# definition for the unit.
#
# Anything after the '#', to-end-of-line is a comment.
#

#
# BASE UNITS.  These must be first and are identified by a nil definition.
#
m                       P                       # length - metre
kg                      P                       # mass - kilogram
s                       P                       # time - second
A                       P                       # electric current - ampere
K                       P                       # temperature - kelvin
mol                     P                       # amount of substance - mole
cd                      P                       # luminous intensity - candela

#
# SUPPLIMENTARY UNITS
#
rad                     P                       # plane angle - radian
sr                      P rad2                  # steradian

#
# DERIVED SI UNITS WITH SPECIAL NAMES
#
Hz                      S 1/s                   # hertz
N                       P kg.m/s2               # Newton
Pa                      P N/m2                  # Pascal
J                       P N.m                   # joule
W                       P J/s                   # watt
C                       P A.s                   # coulomb
V                       P W/A                   # volt
F                       P C/V                   # farad
Ohm                     P V/A                   # ohm
S                       S A/v                   # siemens
Wb                      P V.S                   # weber
```

```
T                       P W/m2              # tesla
H                       P W/A               # henry
Cel                     S K @ 273.15        # degree Celsius
lm                      P cd sr             # lux
lx                      S lm/m2
Bq                      P 1/s               # radionuclide activity
Gy                      P J/kg              # gray
Sv                      P J/kg              # sievert

#
# OTHER UNITS FROM ISO 1000
#
gon
deg
'
"
l
a
ha
min                     P 60 s                  # minute
h                       P 3.6e3 s               # hour
d                       P 8.64e4 s              # day
a                       P 3.153600e7 s          # year
g                       S kg/1000               # gram
t                       P 1e3 kg                # tonne
bar                     P 1e5 Pa                # bar (pressure)
P                       S 1e-1 Pa s             # absolute viscosity-poise
St                      S 1e-4 m2/s             # stokes
eV                      P 1.60219e-19 J         # electronvolt
u                       P 1.66044e-27 kg        # atomic mass unit
AU                      S 1.495979e11 m         # astronomic unit
pc                      P 3.085678e16 m         # parsec
```

ANNEX D: ASCII & RESTRICTED ASCII

(This annex **is** part of the Recommendation)

A code is a correspondence between a symbol and a number of digits of a number system. The American Standard Code for Information Interchange (ASCII) is a seven-bit code also known as the USA Standard Code for Information Interchange (USASCII). The latest updated American National Standards Institute ANSI-X3 standard for this is ANSI X3.4-1977. This code has been incorporated into the ISO code of the same nature (ISO 646-1983) which includes other symbols and alphabets. Since the ISO code is an eight-bit code, the ASCII code is embedded in an eight-bit field in which the higher order bit is set to zero. The Restricted ASCII set of characters (denoted here by the shaded codes with the ⇒ symbol pointing to them) is in this Recommendation. The primary reference to be used should be ISO 646-1983.

The ASCII and Restricted ASCII or RA codes are given in Table D-1. (The code for each character (Char) is given in decimal (Dec), and hexadecimal (Hex)).

Char	Dec	Hex		Char	Dec	Hex		Char	Dec	Hex		Char	Dec	Hex
NUL	0	00		space	32	20		@	64	40		`	96	60
SOH	1	01		!	33	21	⇒	**A**	**65**	**41**		a	97	61
STX	2	02		”	34	22	⇒	**B**	**66**	**42**		b	98	62
ETX	3	03		#	35	23	⇒	**C**	**67**	**43**		c	99	63
EOT	4	04		$	36	24	⇒	**D**	**68**	**44**		d	100	64
ENQ	5	05		%	37	25	⇒	**E**	**69**	**45**		e	101	65
ACK	6	06		&	38	26	⇒	**F**	**70**	**46**		f	102	66
BEL	7	07		’	39	27	⇒	**G**	**71**	**47**		g	103	67
BS	8	08		(	40	28	⇒	**H**	**72**	**48**		h	104	68
HT	9	09		)	41	29	⇒	**I**	**73**	**49**		i	105	69
LF	10	0A		*	42	2A	⇒	**J**	**74**	**4A**		j	106	6A
VT	11	0B		+	43	2B	⇒	**K**	**75**	**4B**		k	107	6B
FF	12	0C		,	44	2C	⇒	**L**	**76**	**4C**		l	108	6C
CR	13	0D		-	45	2D	⇒	**M**	**77**	**4D**		m	109	6D
SO	14	0E		.	46	2E	⇒	**N**	**78**	**4E**		n	110	6E
SI	15	0F		/	47	2F	⇒	**O**	**79**	**4F**		o	111	6F
DLE	16	10	⇒	**0**	**48**	**30**	⇒	**P**	**80**	**50**		p	112	70
DC1	17	11	⇒	**1**	**49**	**31**	⇒	**Q**	**81**	**51**		q	113	71
DC2	18	12	⇒	**2**	**50**	**32**	⇒	**R**	**82**	**52**		r	114	72
DC3	19	13	⇒	**3**	**51**	**33**	⇒	**S**	**83**	**53**		s	115	73
DC4	20	14	⇒	**4**	**52**	**34**	⇒	**T**	**84**	**54**		t	116	74
NAK	21	15	⇒	**5**	**53**	**35**	⇒	**U**	**85**	**55**		u	117	75
SYN	22	16	⇒	**6**	**54**	**36**	⇒	**V**	**86**	**56**		v	118	76
ETB	23	17	⇒	**7**	**55**	**37**	⇒	**W**	**87**	**57**		w	119	77
CAN	24	18	⇒	**8**	**56**	**38**	⇒	**X**	**88**	**58**		x	120	78
EM	25	19	⇒	**9**	**57**	**39**	⇒	**Y**	**89**	**59**		y	121	79
SUB	26	1A		:	58	3A	⇒	**Z**	**90**	**5A**		z	122	7A
ESC	27	1B		;	59	3B		[	91	5B		{	123	7B
FS	28	1C		<	60	3C		\	92	5C		|	124	7C
GS	29	1D		=	61	3D		]	93	5D		}	125	7D
RS	30	1E		>	62	3E		^	94	5E		~	126	7E
US	31	1F		?	63	3F		_	95	5F		DEL	127	7F

Table D-1: ASCII and Restricted ASCII Codes