Consultative
Committee for
Space Data Systems

RECOMMENDATION FOR SPACE
DATA SYSTEM STANDARDS

DATA ENTITY DICTIONARY SPECIFICATION LANGUAGE (DEDSL)— PVL SYNTAX

(CCSD0012)

647.2-B-1

BLUE BOOK

June 2001

AUTHORITY

Issue:	Blue Book, Issue 1
Date:	June 2001
Location:	Oxfordshire, England

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 authorization of CCSDS Recommendations is detailed in [C1], and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This document is published and maintained by:

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

STATEMENT OF INTENT

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of member space Agencies. The Committee meets periodically to address data systems 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 Committee actions 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 this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

o Whenever an Agency establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommendation**. Establishing such a **standard** does not preclude other provisions which an Agency may develop.

o 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.

o Specific service arrangements shall be made via memoranda 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 canceled.

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

FOREWORD

This Recommendation is a technical Recommendation providing the PVL implementation for the Abstract Syntax defined in the *DEDSL—Abstract Syntax* Recommendation (reference [1]) in order to provide a computer processable standardisation of the expression of the semantic information which is to be carried with data.

Through the process of normal evolution, it is expected that expansion, deletion or modification to this document may occur. This Recommendation is therefore subject to CCSDS document management and change control procedures defined in reference [C1]. Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

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

DOCUMENT CONTROL

Document	**Title and Issue**	**Date**	**Status**
CCSDS 647.2-B-1.	Data Entity Dictionary Specification Language (DEDSL)—PVL Syntax	June 2001	Original Issue

CONTENTS

1 INTRODUCTION

1.1 PURPOSE AND SCOPE

The purpose of this Recommendation is to provide a standard method to represent the attributes and their values, as has been defined by the Abstract Syntax of the Data Entity Dictionary Specification Language (DEDSL) (reference [1]), using the Parameter Value Language (PVL) (references [2] and [C3]) for the construction and interchange of data entity dictionaries.

This Recommendation is registered under the CCSDS Authority and Description Identifier (ADID): CCSD0012.

This Recommendation does not exclude other implementation Recommendations of the *DEDSL—Abstract Syntax* Recommendation.

1.2 APPLICABILITY

This Recommendation is intended to be used by implementers of Data Entity Dictionaries, and 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 who 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 RATIONALE

A given data entity may take on a range of values that are represented differently within different formats or in native formats. However there is information about that data entity, such as its definition and other semantic attributes, which are independent of the values and their representation in any given format. These include:

– the exchange of data entity dictionaries among disciplines and organizations which typically use differing standard formats;

- the exchange of data entity dictionary information with registration authorities such as the CCSDS/ISO Control Authority (references [5] and [C4]), and
- the exchange of data entity dictionary information using general data packaging techniques such as the CCSDS/ISO Standard Formatted Data Unit (SFDU) (see references [3] and [C2]).

1.4 DOCUMENT STRUCTURE

This document presents the PVL implementation of the DEDSL Abstract Syntax in a layered manner. The reader should be familiar with both the DEDSL Abstract Syntax (see reference [1]) and the PVL Recommendation (see reference [2]) in order to fully understand this document.

In summary, the document is structured as follows:

- Section 2 introduces the use of PVL as implementation language of the DEDSL.
- Section 3 specifies the exact PVL syntax for each DEDSL dictionary attribute and the way of defining a data entity dictionary in PVL.
- Section 4 specifies the exact PVL syntax for each DEDSL data entity attribute and the way of defining a data entity in PVL.
- Section 5 specifies the exact PVL syntax for each DEDSL descriptor and the way of defining a user-defined attribute in PVL.
- 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.
- Section 7 lists the reserved keywords associated with the PVL implementation.
- Annex A provides examples of data entity definitions.
- Annex B provides an example of a complete data entity dictionary.
- Annex C provides a list of references that may be valuable to the user of this Recommendation as background material or as implementation guidelines for using this Recommendation.

1.5 DEFINITIONS

1.5.1 ACRONYMS AND ABBREVIATIONS

This subsection defines the acronyms and abbreviations that are used throughout this Recommendation:

ADID	Authority and Description Identifier
ASCII	American Standard Code for Information Interchange
CCSDS	Consultative Committee for Space Data Systems
DED	Data Entity Dictionary
DEDSL	Data Entity Dictionary Specification Language
ID	IDentifier
ISO	International Organization for Standardization
LVO	Label Value Object
MACAO	Member Agency Control Authority Office
PVL	Parameter Value Language
SFDU	Standard Formatted Data Unit

1.5.2 GLOSSARY OF TERMS

For the purpose of this document the following definitions apply:

Attribute	A piece of information that describes a Data Entity or Dictionary Entity. This information characterizes or enhances the understanding of the data that is described. Attributes are used to define the semantics of data entities.
Attribute Descriptor	A piece of information that describes an attribute. This document specifies a set of descriptors for attribute description.
Attribute Value	A value associated with an attribute instance.
Composite Data Entity	A data entity which consists of a combination of various other elementary and composite entities.
Constant	A named constant value that is used within a dictionary but is not part of the data themselves. Use of constants enables data entity dictionaries to specify values which will be used by several projects or within a domain (astronomy constants, image size, etc.).

Data Entity	A concept that can, or does, take on one or more values. The concept, and optionally constraints on the representation of its value, are defined by attributes and their values.
Data Entity Dictionary	A collection of semantic definitions of various data entities, together with a few mandatory and optional attributes about the collection as a whole. 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 designers and users as references. Some parts of a dictionary are optional. In practical terms the dictionary could be a file or a Standard Formatted Data Unit (SFDU) Label-Value Object (LVO) value field (see references [3] and [C2]). Within this Recommendation, the expression 'data entity dictionary' can refer either to the notion of data entity dictionaries or to a data entity dictionary instance. A data entity dictionary is also an entity, called Dictionary Entity.
Data Product	A collection of one or more data items that are packaged for or by a specific application.
Defaulted	Indication of an attribute or descriptor value that is understood when the attribute or descriptor is not explicitly included in the containing definition.
Descriptor Name	An Identifier that is the name of the descriptor.
Descriptor Type	The characterization of the descriptor value; e.g., text, identifier, integer.
Elementary Data Entity	A data entity whose data type is elementary, that is Integer, Real, Text or Enumerated.
Enumerated	A set containing a restricted number of discrete values, where each discrete value is named and unique within the set.
Identifier	A PVL unquoted string that designates something.
Integer	The set of integer values. It can optionally be defined more precisely by specifying a range (minimum and maximum bounds).
Model	A data entity described independently from any instance in a data product and corresponding to a re-usable data entity definition from which other data entities may inherit the attributes and apply some specialization rules.
Real	The set of real values. It can optionally be defined more precisely by specifying a range (minimum and maximum bounds).
Semantics	Information that defines the meaning rather than the physical representation of data. Semantics potentially cover a very large domain, from the simple domain, such as the units of one data entity, to a more complex one, such as the relationship between a data entity and another.

Standard Attribute	One of the attributes defined within the DEDSL Abstract Syntax Recommendation (reference [1]).
Syntax	Information defining the physical representation of data. It includes the structural arrangement of the fields within the data on the exchanged media.
Text	A PVL quoted string. The set of allowed characters is defined in the Data Entity Dictionary.
User Defined Attribute	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 within that data entity dictionary.
White Space	Consists of the equivalent 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}) (see reference [4]).

1.5.3 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 an optional specification;

d) the words 'is', 'are' and 'will' imply statements of fact.

1.5.4 CONVENTIONS

Convention 1: Conventions for the syntax of the DEDSL PVL constructs

The syntax of the DEDSL PVL constructs is described using the following conventions:

a) the item named on the left of the **::=** symbol is the statement being defined;

b) the corresponding definition is on the right of the **::=** symbol and is made up of a series of items;

c) the statement which is being defined is delimited by the two delimiters, '<' and '>';

d) items denoting reserved keywords defined by the DEDSL PVL implementation are represented as unquoted strings in bold characters;

e) optional items are enclosed by parentheses;

f) an item on the right of the ::= symbol can be a statement which is defined elsewhere;

g) repeatable items are marked by an asterisk '*';

h) items are separated by space characters;

i) when a statement is no further detailed, its definition may appear within a note, which is delimited by the two delimiters, ‘**[**‘ and ‘**]**’.

The following example presents the definition of the item Statement. Statement is defined:

– first as 3 keywords (‘BEGIN_GROUP’, ‘=’, ‘SEMANTIC_GROUP’) optionally followed by a semicolon;

– followed by a repeatable item Inner_Statement;

– followed by an optional item Optional_Statement, which is further detailed as a note;

– then ended by 3 keywords (‘END_GROUP’, ‘=’, ‘SEMANTIC_GROUP’) optionally followed by a semicolon.

<Statement> ::=	**BEGIN_GROUP = SEMANTIC_GROUP** (;)
	< Inner_Statement >*
	(< Optional_Statement >)
	END_GROUP = SEMANTIC_GROUP (;)
< Optional_Statement > ::=	[Sequence of PVL statements whose syntax and semantics are defined in section xxx]

Figure 1-1: Example of the Conventions for DEDSL PVL Constructs Convention

The Rules appearing throughout this Recommendation are numbered consecutively.

1.6 REFERENCES

The following documents contain provisions (through references within this text) which constitute provisions of this Recommendation. At the time of the 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] *Data Entity Dictionary Specification Language (DEDSL)—Abstract Syntax (CCSD0011)*. Draft Recommendation for Space Data System Standards, CCSDS 647.1-R-2. Red Book. Issue 2. Washington, D.C.: CCSDS, June 2000.

[2] *Parameter Value Language Specification (CCSD0006 and CCSD0008)*. Recommendation for Space Data System Standards, CCSDS 641.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, June 2000.

[3] *Standard Formatted Data Units—Structure and Construction Rules*. Recommendation for Space Data System Standards, CCSDS 620.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, May 1992. (ISO 12175)

[4] *ASCII Encoded English (CCSD0002)*. Recommendation for Space Data System Standards, CCSDS 643.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, November 1992. (ISO 14962)

[5] *Standard Formatted Data Units—Control Authority Procedures*. Recommendation for Space Data System Standards, CCSDS 630.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, June 1993. (ISO 13764)

[6] *Information Processing - 8-Bit Single-Byte Coded Graphic Character Sets—Part 1: Latin Alphabet No. 1*. International Standard, ISO 8859-1:1987. Geneva: ISO, 1987.

[7] *Code for the Representation of Names of Languages*. International Standard, ISO 639-2 -1998. Geneva: ISO, 1998.

[8] *Information Processing—Representation of SI and Other Units in Systems with Limited Character Sets*. International Standard, ISO 2955-1983. Geneva: ISO, 1983.

2 DEDSL IMPLEMENTATION USING PVL

The document *Data Entity Dictionary Specification Language (DEDSL)—Abstract Syntax (CCSD0011)* (reference [1]) defines an abstract standard.

One recommended method of constructing and conveying a Data Entity Dictionary is by using the CCSDS developed Parameter Value Language (PVL, references [2] and [C3]).

PVL is designed to support the conveyance of named values, therefore is suitable for the purpose of implementing the abstract standard. This Recommendation bases its implementation on PVL but specifies additional semantic rules: new keywords and new semantic constructs (see section 7 for the complete list of keywords).

The following subsections specify the PVL implementation of the abstract standard in the following order:

– Subsection 2.1 defines the general mapping of DEDSL Abstract Syntax concepts and elements to PVL constructs, and it includes restrictions related to the PVL implementation;

– Subsection 2.2 provides the structure of a complete data entity dictionary using PVL. It is implemented as a single block of PVL statements and therefore is separate from any data which it describes.

2.1 GENERAL DEDSL ABSTRACT SYNTAX TO PVL MAPPINGS

The following mapping rules apply.

a) Descriptor names and attribute names

The descriptor names and attribute names are implemented as PVL parameter names and are case-insensitive. Therefore these names consist of a sequence of PVL unrestricted Characters. The interoperability constraints on Identifiers specified in the DEDSL Abstract Syntax (reference [1]) should be applied to be fully compliant to this Recommendation.

b) Descriptor values and attribute values

The descriptor values and attribute values are implemented as PVL values. The mappings from the DEDSL Abstract Syntax types to PVL representations are provided in table 2-1**.**

Table 2-1: DEDSL Types / PVL Types Mapping

DEDSL Types	PVL Types
INTEGER	PVL Integer
REAL	PVL Floating Point or PVL Exponential
IDENTIFIER	PVL Unquoted String (see Note 1)
TEXT	PVL Quoted String (see Rule 1)
ENUMERATED	Set of PVL Unquoted String (see Note 1)
ENTITY_TYPE	ENTITY_TYPE (see Note 2)
LIST consisting of only mandatory elements	PVL Sequence (see Note 3)
CHOICE (arg1, arg2, ...) where each arg1, arg2 ... must have a different number of items due to optional elements	PVL Sequence containing the mandatory elements; or PVL Group containing the mandatory and any desired optional elements. (see Notes 3 and 4)
(item1, item2, ...) with exact number of items	PVL Set
List(Identifier) implying an unspecified number of items	(see Note 5)

Rule 1 – Values of type Text are expressed as PVL Quoted Strings. The characters used must be consistent with both the PVL constraints on Quoted Strings and those defined by the attribute TEXT_FIELD_CHARACTER_SET. For example, the character set used to express textual values can be a super-set of the ISO646 character set, and can be for example the Latin Alphabet No1 (reference [6]).

NOTES

1 The interoperability constraints on Identifiers specified in the DEDSL Abstract Syntax (reference [1]) should be applied to be fully compliant to this Recommendation.

2 There is no PVL Type equivalent to Entity_Type, which means the data type of the entity. Therefore, the keyword Entity_Type is defined.

3 The order of the items in the lists in the DEDSL Abstract Syntax is maintained in the corresponding sequences in the PVL implementation.

4 In the DEDSL Abstract Syntax (reference[1]), attributes which have both mandatory and optional elements are shown as a Choice of Lists, where the minimum length list contains the mandatory elements and the longer list(s) contain(s) both optional and mandatory elements.

5 One attribute which appears as a list in the Abstract Syntax and is named as plural, is represented in the PVL Syntax as a separate attribute named as singular in a PVL block. This is the case of ENUMERATION_VALUE / ENUMERATION_VALUES.

c) **Grouping requirements**

Where the DEDSL Abstract Syntax requires the grouping of parameter/value pairs, the PVL aggregation statements BEGIN_GROUP and END_GROUP are used.

d) **PVL statement**

The semicolon is not mandatory at the end of PVL statements. However, use of the semicolon is recommended for a PVL implementation of a data entity dictionary.

2.2 COMPLETE DEDSL DEFINITION OF A DATA ENTITY DICTIONARY

The structure of a complete Data Entity Dictionary using PVL is bounded by an aggregation block called 'DEDSL_DICTIONARY'.

The dictionary attributes, the user-defined attributes as well as the data entity definitions appear in separate blocks as indicated below:

– The 'DICTIONARY_IDENTIFICATION' block appears first. It contains a 'USER_DEFINED_ATTRIBUTES' block when new dictionary or global attributes (both for the dictionary and the data entities) are defined. It then contains the dictionary attributes within a DICTIONARY_ENTITY_DEFINITION block (see section 3 for the complete definition).

– The 'DATA_ENTITY_DEFINITIONS' block follows the 'DICTIONARY_ IDENTIFICATION' block. It contains a 'USER_DEFINED_ATTRIBUTES' block whenever attributes specific to data entities are defined. It then contains data entity definitions. There must be at least one of these data entity definitions ('ENTITY_DEFINITION' block) within a dictionary and there is no limit on the total number (see section 4 for the complete definition).

User-defined attributes must be defined before they are used.

This structure is described using the conventions defined in 1.5.4 as follows:

< DEDSL_Dictionary > ::= < Begin_DEDSL_Statement >
< Dictionary_Identification_Block >
< Data_Entity_Definitions_Block >
< End_DEDSL_Statement >

< Begin_DEDSL_Statement > ::= **BEGIN_GROUP = DEDSL_DICTIONARY** (**;**)

< End_DEDSL_Statement > ::= **END_GROUP = DEDSL_DICTIONARY** (**;**)

Figure 2-1: Structure of a DEDSL_Dictionary Block

3 IMPLEMENTATION OF DICTIONARY ATTRIBUTES

3.1 GENERAL

3.1.1 The ‘DICTIONARY_IDENTIFICATION’ block contains:

– a ‘USER_DEFINED_ATTRIBUTES’ block when new dictionary or global attributes (both for the dictionary and the data entities) are defined (see section 5), followed by

– the DICTIONARY_ENTITY_DEFINITION block specifying the dictionary attributes described by this section.

3.1.2 The structure is described using the conventions defined in 1.5.4 as follows:

```
< Dictionary_Identification_Block > ::= < Begin_Dictionary_Statement >
                                        ( < User_Defined_Attributes_Block > )
                                        < Dictionary_Entity_Definition >
                                        < End_Dictionary_Statement >

< Begin_Dictionary_Statement > ::=
            BEGIN_GROUP = DICTIONARY_IDENTIFICATION (;)

< End_Dictionary_Statement > ::=
            END_GROUP = DICTIONARY_IDENTIFICATION (;)

< Dictionary_Entity_Definition > ::=  < Begin_Dictionary_Entity_Statement >
                                      < Dictionary_Attribute_Statement > *
                                      < End_Dictionary_Entity_Statement >

< Begin_Dictionary_Entity_Statement > ::=
            BEGIN_GROUP = DICTIONARY_ENTITY_DEFINITION (;)

< End_Dictionary_Entity_Statement > ::=
            END_GROUP = DICTIONARY_ENTITY_DEFINITION (;)

< Dictionary_Attribute_Statement > ::= [ Sequence of PVL Statements whose syntax and
                                         semantics are defined in section 3 ]
```

Figure 3-1: Structure of a Dictionary_Identification Block

NOTE – The following is an example showing a 'DICTIONARY_IDENTIFICATION' block conforming to the previously defined structure:

```
BEGIN_GROUP = DICTIONARY_IDENTIFICATION ;

        BEGIN_GROUP = DICTIONARY_ENTITY_DEFINITION ;
                    DICTIONARY_NAME = CDPP_Plasma_Dictionary ;
                    DICTIONARY_DEFINITION = 'This dictionary contains data entity
                    definitions relative to planetary science and which may be re-used for
                    defining data products';
                    TEXT_FIELD_CHARACTER_SET = 'ISO-LATIN ALPHABETNo1';
                    CASE_SENSITIVITY = NOT_CASE_SENSITIVE ;
                    LANGUAGE = ('English', en) ;
                    DICTIONARY_VERSION = '1.a' ;
                    DICTIONARY_IDENTIFIER = FCST0172 ;
                    DEDSL_VERSION = 'CCSDS 647.2-B-1.0' ;
        END_GROUP = DICTIONARY_ENTITY_DEFINITION ;

END_GROUP = DICTIONARY_IDENTIFICATION ;
```

Example 3-1: Example of a Dictionary_Identification Block

3.2 LIST OF DICTIONARY ATTRIBUTES

The table 3-1 provides for each category the standard attributes that are defined for dictionary entities by this Recommendation. The obligation column indicates whether an attribute is mandatory (M), conditional (C), optional (O) or defaulted (D) in the definition of each data entity appearing in a conforming Data Entity Dictionary. The occurrence column indicates the number of times the attribute can appear in a dictionary definition.

Table 3-1: List of Dictionary Attributes

Category	Attribute or block name	Obligation	Occurrence
Identifying	DICTIONARY_NAME	M	1
Definitional	DICTIONARY_DEFINITION	O	1
Relational	EXTERNAL_DICTIONARY_REFERENCE	C (see Rule 2)	‘n’
Representational	TEXT_FIELD_CHARACTER_SET	M	1
	CASE_SENSITIVITY	D	1
	LANGUAGE	M	1
Administrative	DICTIONARY_VERSION	O	1
	DICTIONARY_IDENTIFIER	O	1
	DEDSL_VERSION	M	1

Rule 2 – The attribute EXTERNAL_DICTIONARY_REFERENCE is mandatory when a reference to a Data Entity Dictionary is made in the current data entity dictionary in one of the INHERITS_FROM_BLOCK or RELATION_BLOCK attributes defined for data entities.

3.3 IDENTIFYING ATTRIBUTES

3.3.1 DICTIONARY_NAME

DEFINITION	Human readable name for the Data Entity Dictionary.
PVL TYPE	Unquoted String
EXAMPLE VALUE	DICTIONARY_NAME = CDPP_Plasma_Dictionary ;

3.4 DEFINITIONAL ATTRIBUTES: DICTIONARY_DEFINITION

DEFINITION	Human readable definition for the Data Entity Dictionary.
PVL TYPE	Quoted String
EXAMPLE VALUE	DICTIONARY_DEFINITION = ‘This dictionary contains data entity definitions relative to planetary science and which may be re-used for defining data products’;

3.5 RELATIONAL ATTRIBUTES: EXTERNAL_DICTIONARY_REFERENCE

DEFINITION	Reference to another Data Entity Dictionary whose models are re-used in the current one, defined as the local name of the Data Entity Dictionary, followed by its identifier and its associated Control Authority.
PVL TYPE	(Unquoted String, Unquoted String, Quoted String)
EXAMPLE VALUE	EXTERNAL_DICTIONARY_REFERENCE = (CDPP_Plasma_Dictionary, FCST0172, ‘CCSDS_Control_Authority’) ;

3.6 REPRESENTATIONAL ATTRIBUTES

3.6.1 TEXT_FIELD_CHARACTER_SET

DEFINITION	Name of the Character Set that is valid for TEXT value types within the dictionary.
PVL TYPE	Quoted String
EXAMPLE VALUE	TEXT_FIELD_CHARACTER_SET = ‘ISO-LATIN ALPHABET No1’ ;

3.6.2 CASE_SENSITIVITY

DEFINITION	Specifies the case sensitivity for the Identifiers used as values for the attributes of the data entities contained in the dictionary.
PVL TYPE	{CASE_SENSITIVE, NOT_CASE_SENSITIVE}with NOT_CASE_SENSITIVE as default value
EXAMPLE VALUE	CASE_SENSITIVITY = NOT_CASE_SENSITIVE ;

3.6.3 LANGUAGE

DEFINITION	Main natural language that is valid for any value of type TEXT given to the attributes of the current entity. When used in a data entity, the value of the attribute overrides the value specified for the dictionary entity. It is defined as the English name of the language and its associated 2 or 3 letter code as specified in ISO 639-2 (reference [7]).
PVL TYPE	(Quoted String, Unquoted String)
EXAMPLE VALUE	LANGUAGE = (‘French’, fr) ;

3.7 ADMINISTRATIVE ATTRIBUTES

3.7.1 DICTIONARY_VERSION

DEFINITION	Version of the data Entity Dictionary.
PVL TYPE	Quoted String
EXAMPLE VALUE	DICTIONARY_VERSION = ‘1.a’;

3.7.2 DICTIONARY_IDENTIFIER

DEFINITION	Identifier under which the Data Entity Dictionary has been registered at a registration Authority.
PVL TYPE	Unquoted String
EXAMPLE VALUE	DICTIONARY_IDENTIFIER = FCST0172;

3.7.3 DEDSL_VERSION

DEFINITION	CCSDS document number of the document corresponding to the PVL implementation of the Abstract Syntax. Note that this reference contains the version.
PVL TYPE	Quoted String
EXAMPLE VALUE	DEDSL_VERSION = ‘CCSDS 647.2-B-1.0’;

4 IMPLEMENTATION OF DATA ENTITY ATTRIBUTES

4.1 GENERAL

4.1.1 The ‘DATA_ENTITY_DEFINITIONS’ block contains:

– a ‘USER_DEFINED_ATTRIBUTES’ block when new data entity attributes are defined (see section 5), followed by

– at least one ENTITY_DEFINITION block specifying the attributes described by this section for a specific data entity. There are as many ENTITY_DEFINITION blocks as there are data entities to describe.

4.1.2 The structure is described using the conventions defined in 1.5.4 as follows:

```
< Data_Entity_Definitions_Block > ::=       < Begin_DED_Statement >
                                            ( < User_Defined_Attributes_Block > )
                                            < Entity_Definition_Block > *
                                            < End_DED_Statement >

< Begin_DED_Statement > ::=
                  BEGIN_GROUP = DATA_ENTITY_DEFINITIONS (;)

< End_DED_Statement > ::=
                  END_GROUP = DATA_ENTITY_DEFINITIONS (;)

< Entity_Definition_Block > ::=             < Begin_ED_Statement >
                                            < Entity_Attribute_Statement > *
                                            < End_ED_Statement >

< Begin_ED_Statement > ::= BEGIN_GROUP = ENTITY_DEFINITION (;)

< End_ED_Statement > ::= END_GROUP = ENTITY_DEFINITION (;)

< Entity_Attribute_Statement > ::= [Sequence of PVL Statements whose syntax and
                                   semantics are defined in section 4]
```

Figure 4-1: Structure of a Data_Entity_Definitions Block

NOTE – The following is an example showing a 'DATA_ENTITY_DEFINITIONS' block conforming to the previously defined structure:

```
BEGIN_GROUP = DATA_ENTITY_DEFINITIONS ;

        BEGIN_GROUP = ENTITY_DEFINITION ;
                NAME = YEAR;
                CLASS = MODEL;
                DEFINITION = 'It corresponds to the year expressed as an integer' ;
                DATA_TYPE = Integer;
                RANGE = (1900,2100);
        END_GROUP = ENTITY_DEFINITION;

        BEGIN_GROUP = ENTITY_DEFINITION;
                NAME = MONTH;
                CLASS = MODEL;
                DEFINITION = 'It corresponds to a month expressed as an enumerated';
                DATA_TYPE = Enumerated;
                ENUMERATION_VALUES = { JANUARY, FEBRUARY, MARCH,
                APRIL, MAY, JUNE, JULY, AUGUST, SEPTEMBER, OCTOBER,
                NOVEMBER, DECEMBER};
        END_GROUP = ENTITY_DEFINITION;

        BEGIN_GROUP = ENTITY_DEFINITION;
                NAME = DAY;
                CLASS = MODEL;
                DEFINITION = 'It corresponds to the day expressed as an integer' ;
                DATA_TYPE = Integer;
                RANGE = (1,31);
        END_GROUP = ENTITY_DEFINITION;

        BEGIN_GROUP = ENTITY_DEFINITION;
                NAME = DATE;
                CLASS = MODEL;
                DEFINITION = 'It corresponds to the definition of a date';
                DATA_TYPE = Composite;
                COMPONENT = YEAR;
                COMPONENT = MONTH;
                COMPONENT = DAY;
        END_GROUP = ENTITY_DEFINITION;
END_GROUP = DATA_ENTITY_DEFINITIONS;
```

Example 4-1: Example of a Data_Entity_Definitions Block

4.2 LIST OF DATA ENTITY ATTRIBUTES AND BLOCKS

Table 4-1 provides for each category the standard attributes or PVL blocks that are defined for data entities by this Recommendation. The obligation column indicates whether an attribute is mandatory (M), conditional (C), optional (O) or defaulted (D) in the definition of each data entity appearing in a conforming Data Entity Dictionary. The occurrence column indicates the number of times the attribute or the block can appear in the definition of each data entity.

Table 4-1: List of Data Entity Attributes and Blocks

Category	Attribute or Block Name	Obligation	Occurrence
Identifying	NAME	M	1
	ALIAS	O	'n'
	CLASS	D	1
Definitional	DEFINITION	M	1
	SHORT_DEFINITION	O	1
	COMMENT	O	'n'
	UNITS	C (see Rule 3)	1
	SPECIFIC_INSTANCE	O	'n'
Relational	INHERITS_FROM	O (see Rule 4)	1
	INHERITS_FROM_BLOCK	O (see Rule 4)	1
	COMPONENT	O (see Rule 5)	'n'
	COMPONENT_BLOCK	O (see Rules 5 and 13)	'n'
	KEYWORD	O	'n'
	RELATION	O	'n'
	RELATION_BLOCK	O (see Rule 13)	'n'
Representational	DATA_TYPE	C (see Rule 6)	1
	ENUMERATION_VALUES	C (see Rules 7 and 8)	1
	ENUMERATION_VALUES _BLOCK	C (see Rules 7, 8 and 13)	1
	RANGE	O (see Rule 9)	1
	TEXT_SIZE_MAX	C (see Rules 10 and 11)	1
	TEXT_SIZE_BLOCK	C (see Rules 10, 11 and 13)	1
	CASE_SENSITIVITY	O	1
	LANGUAGE	O	1
	CONSTANT_VALUE	C (see Rule 12)	1

Rule 3 – If the data entity is non-scalar then the attribute shall not be specified.

Rule 4 – At most one of the INHERITS_FROM and INHERITS_FROM_BLOCK can appear in a single ‘ENTITY_DEFINITION’ block. If the parent entity resides in the same data entity dictionary either form may be used, if not, only the block form may be used.

Rule 5 – At most one of the COMPONENT and COMPONENT_BLOCK can appear in a single ‘ENTITY_DEFINITION’ block. The COMPONENT attribute can only be used when the number of times the particular component appears is 1, and in the other cases, either form may be used.

Rule 6 – This attribute must be present for a product data field definition and for a constant definition (CLASS attribute set to DATA_FIELD or CONSTANT) and is optional for a model definition (CLASS attribute set to MODEL).

Rule 7 – This attribute is mandatory if the DATA_TYPE is ENUMERATED and if only the permitted values are given.

Rule 8 – At most one of the ENUMERATION_VALUES and ENUMERATION_VALUES_BLOCK can appear in a single ‘ENTITY_DEFINITION’ block. The ENUMERATION_VALUES attribute can be used when only the set of permitted values is given, and in other cases (when either the meanings or the conventions are given) the ENUMERATION_VALUES_BLOCK should be used.

Rule 9 – This attribute is allowed if the DATA_TYPE is Integer or Real. It is mandatory when the class attribute is set to DATA_FIELD or CONSTANT.

Rule 10 – This attribute is mandatory if the DATA_TYPE is Text.

Rule 11 – At most one of the TEXT_SIZE_MAX and TEXT_SIZE_BLOCK can appear in a single ‘ENTITY_DEFINITION’ block. When the minimum and the maximum number of characters that a text may contain are different, only TEXT_SIZE_BLOCK may be used.

Rule 12 – This attribute is mandatory if the CLASS attribute is CONSTANT.

Rule 13 – Some attributes can only appear in the construct of a PVL block. These attributes are:

- OCCURRENCE_MIN and OCCURRENCE_MAX which are part of the COMPONENT_BLOCK block,
- REFERRED_ENTITY and EXTERNAL_DICTIONARY which are part of the RELATION_BLOCK block,

- ENUMERATION_VALUE, ENUMERATION_MEANING and ENUMERATION_CONVENTION which are part of the ENUMERATION_VALUES_BLOCK block,

- TEXT_SIZE_MIN which is part of the TEXT_SIZE_BLOCK block.

4.3 IDENTIFYING ATTRIBUTES

4.3.1 NAME

DEFINITION	A name that may be used to link a collection of attributes with an equivalent identifier in, or associated with, the data entity.
PVL TYPE	Unquoted String
EXAMPLE VALUE	NAME = ACQ_STATION;

4.3.2 ALIAS

DEFINITION	Single- or multi-word designation that differs from the value of NAME, but represents the same data entity concept, followed by the context in which this name is applied.
PVL TYPE	(Quoted String, Quoted String)
EXAMPLE VALUE	ALIAS = (‘TIME_LINE’, ‘used within the ground segment’);

4.3.3 CLASS

DEFINITION	Statement of what kind of entity is defined by the current entity definition. This definition can be a MODEL definition, a DATA_FIELD definition or a CONSTANT definition.
PVL TYPE	{MODEL, DATA_FIELD, CONSTANT} with DATA_FIELD as default value
EXAMPLE VALUE	CLASS = DATA_FIELD;

4.4 DEFINITIONAL ATTRIBUTES

4.4.1 DEFINITION

DEFINITION	Statement that expresses the essential nature of a data entity and permits its differentiation from all the other data entities. This attribute is intended for readership.
PVL TYPE	Quoted String
EXAMPLE VALUE	DEFINITION = ‘This describes my entity definition’;

4.4.2 SHORT_DEFINITION

DEFINITION	Statement that expresses the essential nature of a data entity in a shorter and more concise manner than the statement of the mandatory attribute DEFINITION. This attribute is intended for display purposes.
PVL TYPE	Quoted String
EXAMPLE VALUE	SHORT_DEFINITION = ‘my data entity’;

4.4.3 COMMENT

DEFINITION	Associated information about a data entity. It enables to add information which does not correspond to definition information.
PVL TYPE	Quoted String
EXAMPLE VALUE	COMMENT = ‘This is a very key concept’;

4.4.4 UNITS

DEFINITION	Specifies the scientific units that should be associated with the value of the data entity so as to make the value meaningful to applications.
PVL TYPE	set of Quoted String {} to indicate there is no units See Rules 14 and 15

EXAMPLE VALUE	UNITS = {'km/s2', 'm/s2', 'cm/s2'} ;
	UNITS = {'km/s2'} ;
	UNITS = {} ;

Rule 14 – The contents of the quoted string must conform to the representation specified in ISO 2955 (reference [8]).

Rule 15 – The value of the UNITS attribute for a data_field or a constant must be a set with 0 or 1 member. For a model the set may contain any number of values.

4.4.5 SPECIFIC_INSTANCE

DEFINITION	Provides a real-world meaning for a specific instance (a value) of the data entity being described.
PVL TYPE	(Entity_Type, Quoted String)

EXAMPLE VALUE	SPECIFIC_INSTANCE = (+00.00, 'Equator');
	SPECIFIC_INSTANCE = (100, 'Boiling point of H_2O');

4.5 RELATIONAL ATTRIBUTES

4.5.1 INHERITS_FROM

DEFINITION	Gives the name of a model or a data field from which the current entity definition inherits attributes. This name must be the value of the NAME attribute found in the referred entity definition.
PVL TYPE	Unquoted String
EXAMPLE VALUE	INHERITS_FROM = CCSDS_calendar_time;

4.5.2 INHERITS_FROM_BLOCK

DEFINITION	Gives the name of a model or a data field from which the current entity definition inherits attributes. This name must be the value of the NAME attribute found in the referred entity definition. When the referred entity definition belongs to another dictionary, the external dictionary must be defined in the dictionary attribute EXTERNAL_DICTIONARY_REFERENCE, and its local name must be used in the EXTERNAL_DICTIONARY statement.

PVL TYPE	BEGIN_GROUP = INHERITS_FROM_BLOCK;
	    INHERITS_FROM = Unquoted String;
	    EXTERNAL_DICTIONARY = Unquoted String;
	END_GROUP = = INHERITS_FROM_BLOCK;
	See Rule 16

EXAMPLE VALUE	BEGIN_GROUP = INHERITS_FROM_BLOCK;
	    INHERITS_FROM = CCSDS_calendar_time;
	    EXTERNAL_DICTIONARY = CCSDS_time_codes;
	END_GROUP = = INHERITS_FROM_BLOCK;

Rule 16 – The INHERITS_FROM_BLOCK must contain exactly one INHERITS_FROM statement and may contain at most one EXTERNAL_DICTIONARY statement.

4.5.3 COMPONENT

DEFINITION	Name of a component.
PVL TYPE	Unquoted String
EXAMPLE VALUE	COMPONENT = DATE;

4.5.4 COMPONENT_BLOCK

DEFINITION	Name of a component. When the component can appear more than once, the number of times it occurs is specified by a range: integer literals or constant names.

PVL TYPE	BEGIN_GROUP = COMPONENT_BLOCK;
	    COMPONENT = Unquoted String ;
	    OCCURRENCE_MIN = Integer or Unquoted String;
	    OCCURRENCE_MAX = Integer or Unquoted String;
	END_GROUP = COMPONENT_BLOCK;
	See Rules 17 and 18

EXAMPLE VALUE	BEGIN_GROUP = COMPONENT_BLOCK;
	    COMPONENT = DATE;
	    OCCURRENCE_MIN = 1;
	    OCCURRENCE_MAX = 2;
	END_GROUP = COMPONENT_BLOCK;

Rule 17 – The COMPONENT_BLOCK must contain exactly one COMPONENT statement, and may contain at most one OCCURRENCE_MIN statement and at most one OCCURRENCE_MAX statement.

Rule 18 – Each COMPONENT must be defined in the containing dictionary; this may just consist of the mandatory attributes NAME and DEFINITION.

4.5.5 KEYWORD

DEFINITION	A significant word used for retrieving data entities.
PVL TYPE	Quoted String

EXAMPLE VALUE	KEYWORD = ‘Astrophysics’;
	KEYWORD = ‘Earth Science’;

4.5.6 RELATION

DEFINITION	Used to express a relationship between the current entity definition and a referred entity definition when this relation cannot be expressed using a precise standard relational attribute. The relationship is user-defined and expressed using free text.
PVL TYPE	(Quoted String, Unquoted String)
EXAMPLE VALUE	RELATION = (‘dependent on’, Times);

4.5.7 RELATION_BLOCK

DEFINITION	Used to express a relationship between the current entity definition and a referred entity definition when this relation cannot be expressed using a precise standard relational attribute. The relationship is user-defined and expressed using free text. When the referred entity definition belongs to another dictionary, the external dictionary must be defined in the dictionary attribute EXTERNAL_DICTIONARY_REFERENCE, and its local name must be used in the EXTERNAL_DICTIONARY statement.

PVL TYPE	BEGIN_GROUP = RELATION_BLOCK;
	        RELATION = Quoted String;
	        REFERRED_ENTITY = Unquoted_String;
	        EXTERNAL_DICTIONARY = Unquoted String;
	END_GROUP = RELATION_BLOCK;
	See Rule 19

EXAMPLE VALUE	BEGIN_GROUP = RELATION_BLOCK ;
	        RELATION = ‘dependent on’;
	        REFERRED_ENTITY = Times;
	        EXTERNAL_DICTIONARY = CCSDS_calendar_time;
	END_GROUP = RELATION_BLOCK;

Rule 19 – The RELATION_BLOCK must contain exactly one RELATION statement, one REFERRED_ENTITY statement and at most one EXTERNAL_DICTIONARY statement.

4.5.8 DATA_TYPE

DEFINITION	A set of distinct values for representing the type of the data entity value. This attribute defines the conceptual data type of the entity and is not intended for specifying the physical representation of the entity.
PVL TYPE	{Enumerated, Text, Real, Integer, Composite}
EXAMPLE VALUE	DATA_TYPE = Enumerated;

4.5.9 ENUMERATION_VALUES

DEFINITION	The set of permitted values of the Enumerated data entity.
PVL TYPE	PVL Set of Unquoted Strings
EXAMPLE VALUE	ENUMERATION_VALUES = {red, green, blue} ;

4.5.10 ENUMERATION_VALUES_BLOCK

4.5.10.1 The ENUMERATION_VALUES_BLOCK block is a sequence of one or more ENUMERATION blocks, with as many ENUMERATION blocks as there are values in the enumeration.

4.5.10.2 The structure is described using the conventions defined in 1.5.4 as follows:

```
< Enumeration_Values_Block > ::=        < Begin_EVB_Statement >
                                        < Enumeration_Block > *
                                        < End_EVB_Statement >

< Begin_EVB_Statement > ::=
                BEGIN_GROUP = ENUMERATION_VALUES_BLOCK (;)

< End_EVB_Statement > ::=
                END_GROUP = ENUMERATION_VALUES_BLOCK (;)
```

4.5.10.3 The Enumeration_Block statement is an ENUMERATION block defined as follows:

DEFINITION	Used to give one of the permitted values of an Enumerated data entity while giving its associated meaning and/or convention.

PVL TYPE	BEGIN_GROUP = ENUMERATION;
	ENUMERATION_VALUE = Unquoted String;
	ENUMERATION_MEANING = Quoted String;
	ENUMERATION_CONVENTION = Quoted String;
	END_GROUP = ENUMERATION;
	
	See Rule 20

EXAMPLE VALUE	BEGIN_GROUP = ENUMERATION;
	ENUMERATION_VALUE = red;
	ENUMERATION_MEANING = ‘the range of wavelengths from 700-730A’;
	ENUMERATION_CONVENTION = ‘1’;
	END_GROUP = ENUMERATION;

Rule 20 – The ENUMERATION block must contain exactly one ENUMERATION_VALUE statement, at most one ENUMERATION_MEANING statement and at most one ENUMERATION_CONVENTION statement.

4.5.11 RANGE

DEFINITION	The minimum bound and the maximum bound of an Integer or Real data entity. The bounds can be a literal value or a constant name.
PVL TYPE	(Entity_Type, Entity_Type) or (Unquoted String, Entity_Type) or (Entity_Type, Unquoted String) or (Unquoted String, Unquoted String)

EXAMPLE VALUE	RANGE = (0.0, 360.0);
	RANGE = (0, 99);
	RANGE = (MIN, MAX);

4.5.12 TEXT_SIZE_MAX

DEFINITION	The maximum number of characters the text may contain.
PVL TYPE	TEXT_SIZE_MAX = Integer or Unquoted String
EXAMPLE VALUE	TEXT_SIZE_MAX = 10;

4.5.13 TEXT_SIZE_BLOCK

DEFINITION	The minimum and the maximum number of characters the text may contain.

PVL TYPE	BEGIN_GROUP = TEXT_SIZE_BLOCK;
	    TEXT_SIZE_MIN = Integer or Unquoted String;
	    TEXT_SIZE_MAX = Integer or Unquoted String;
	ENG_GROUP = TEXT_SIZE_BLOCK;
	See Rule 21

EXAMPLE VALUE	BEGIN_GROUP = TEXT_SIZE_BLOCK;
	    TEXT_SIZE_MIN = 100;
	    TEXT_SIZE_MAX =1000 ;
	ENG_GROUP = TEXT_SIZE_BLOCK;

Rule 21 – The TEXT_SIZE_BLOCK must contain exactly one TEXT_SIZE_MAX statement and may contain at most one TEXT_SIZE_MIN statement.

4.5.14 CASE_SENSITIVITY

DEFINITION	Specifies the case sensitivity for the unquoted Strings used as values for the attributes of the current entity. When used in a data entity, the value of the attribute overrides the value specified at the dictionary level.
PVL TYPE	{CASE_SENSITIVE, NOT_CASE_SENSITIVE}
EXAMPLE VALUE	CASE_SENSITIVITY = NOT_CASE_SENSITIVE ;

4.5.15 LANGUAGE

DEFINITION	Main natural language that is valid for any value of type TEXT given to the attributes of the current entity. When used in a data entity, the value of the attribute overrides the value specified for the dictionary entity. It is defined as the English name of the language and its associated 2 or 3 letter code as specified in ISO 639-2 (reference [7]).
PVL TYPE	(Quoted String, Unquoted String)
EXAMPLE VALUE	LANGUAGE = (‘French’, fr);

4.5.16 CONSTANT VALUE

DEFINITION	Specifies the value given to a constant (data entity whose CLASS attribute is set to CONSTANT).
PVL TYPE	Entity_Type

EXAMPLE VALUE	CONSTANT_VALUE = 1276.0;
	CONSTANT_VALUE = 1276;

5 IMPLEMENTATION OF USER-DEFINED ATTRIBUTES

5.1 GENERAL

5.1.1 The ‘USER_DEFINED_ATTRIBUTES’ block contains the definition of at least one new user-defined attribute.

5.1.2 The structure is described using the conventions defined in 1.5.4 as follows:

```
< User_Defined_Attributes_Block > ::=      < Begin_UDA_Statement >
                                           < Attribute_Definition_Block > *
                                           < End_UDA_Statement >

< Begin_UDA_Statement > ::=
                   BEGIN_GROUP = USER_DEFINED_ATTRIBUTES (;)

< End_UDA_Statement > ::=
                   END_GROUP = USER_DEFINED_ATTRIBUTES (;)

< Attribute_Definition_Block > ::=         < Begin_AD_Statement >
                                           < Attribute_Definition_Statement >
                                           < End_AD_Statement >

< Begin_AD_Statement > ::=
                   BEGIN_GROUP = ATTRIBUTE_DEFINITION (;)

< End_AD_Statement > ::=
                   END_GROUP = ATTRIBUTE_DEFINITION (;)

< Attribute_Definition_Statement > ::= [Sequence of PVL Statements whose syntax and
                                        semantics are defined in 5]
```

Figure 5-1: Structure of a USER_DEFINED_ATTRIBUTES Block

NOTE – Example 5-1 shows a ‘USER_DEFINED_ATTRIBUTES’ block conforming to the previously defined structure:

- The first user-defined attribute ASSOCIATED_UTILITIES provides the name of the utility which processes the physical values of a specific data entity within a data product and some comments about it. Therefore it can be defined as a sequence of two Text values. Moreover, as it is a user-defined attribute, the value of the defaulted descriptors ATTRIBUTE_INHERITANCE and ATTRIBUTE_SCOPE should be specified.
- The second user-defined attribute AUDIO_EXAMPLE provides the definition of an external reference as a Text.

```
BEGIN_GROUP = USER_DEFINED_ATTRIBUTES;

BEGIN_GROUP = ATTRIBUTE_DEFINITION;
        ATTRIBUTE_NAME = ASSOCIATED_UTILITY;
        ATTRIBUTE_DEFINITION = ‘Provides the name of the utility used to process the
        occurrences of the data entity with which the attribute is associated. The first text gives
        the name of the utility and the second one specifies the application context of the
        utility.’ ;
        ATTRIBUTE_OBLIGATION = Optional;
        ATTRIBUTE_MAXIMUM_OCCURRENCE = ‘n’;
        ATTRIBUTE_VALUE_TYPE = (Text, Text);
        ATTRIBUTE_MAXIMUM_SIZE = (400,400);
        ATTRIBUTE_INHERITANCE = INHERITABLE;
        ATTRIBUTE_VALUE_EXAMPLE = ‘Here is an example (embedded texts being
        delimited by simple quotes): (‘xv’, ‘tool to use for image display’)’;
        ATTRIBUTE_SCOPE = DATA;
END_GROUP = ATTRIBUTE_DEFINITION;


BEGIN_GROUP = ATTRIBUTE_DEFINITION;
        ATTRIBUTE_NAME = AUDIO_EXAMPLE;
        ATTRIBUTE_DEFINITION = ‘Represents a pointer to a sound file in .wav format
        providing an example to be heard. The value is expressed as a path.’;
        ATTRIBUTE_OBLIGATION = Optional;
        ATTRIBUTE_MAXIMUM_OCCURRENCE = ‘n’;
        ATTRIBUTE_VALUE_TYPE = Text;
        ATTRIBUTE_MAXIMUM_SIZE = 400;
        ATTRIBUTE_INHERITANCE = INHERITABLE;
        ATTRIBUTE_VALUE_EXAMPLE = ‘ Here is an example (embedded text being
        delimited by simple quotes): ‘/home/MUSIC/INSTRUMENT/SOUND/Piano.wav’’;
        ATTRIBUTE_SCOPE = DATA;
END_GROUP = ATTRIBUTE_DEFINITION;

END_GROUP = USER_DEFINED_ATTRIBUTES;
```

Example 5-1: Example of User_Defined_Attributes Block

5.2 LIST OF ATTRIBUTE DESCRIPTORS

Table 5-1 provides the set of general descriptors that are necessary to define a user-defined attribute. The obligation column indicates whether a descriptor is mandatory (M), conditional (C), optional (O) or defaulted (D).

Table 5-1: List of Attribute Descriptors

Attribute descriptor	Obligation
ATTRIBUTE_NAME	M
ATTRIBUTE_DEFINITION	M
ATTRIBUTE_OBLIGATION	M
ATTRIBUTE_CONDITION	C (see Rule 22)
ATTRIBUTE_MAXIMUM_OCCURRENCE	M
ATTRIBUTE_VALUE_TYPE	M
ATTRIBUTE_MAXIMUM_SIZE	O (see Rule 23)
ATTRIBUTE_ENUMERATION_VALUES	C (see Rule 24)
ATTRIBUTE_COMMENT	O
ATTRIBUTE_INHERITANCE	D
ATTRIBUTE_DEFAULT_VALUE	C (see Rule 25)
ATTRIBUTE_VALUE_EXAMPLE	O
ATTRIBUTE_SCOPE	D

Rule 22 – This descriptor shall be present if the ATTRIBUTE_OBLIGATION descriptor of the same data entity attribute definition has the value ‘conditional’.

Rule 23 – This descriptor shall be present if the ATTRIBUTE_VALUE_TYPE descriptor of the same data entity attribute definition has the value Identifier or Text. It may be present for composite attributes made up of Identifiers and Texts, and represents the size allowed for each component of the set.

Rule 24 – This descriptor shall be present if the ATTRIBUTE_VALUE_TYPE descriptor of the same data entity attribute definition has the value Enumerated.

Rule 25 – This descriptor shall be present if the ATTRIBUTE_OBLIGATION descriptor of the same data entity attribute definition has the value ‘defaulted’. It shall not be present for composite attributes, that is for attributes whose attribute_value_type is a List or a Choice.

5.3 ATTRIBUTE_NAME

DEFINITION	Identification of an attribute (unique within a Data Dictionary Entity).
PVL TYPE	Unquoted String
EXAMPLE VALUE	ATTRIBUTE_NAME = DICTIONARY_VERSION;

5.4 ATTRIBUTE_DEFINITION

DEFINITION	Full textual description of an attribute, that is any information enhancing the understanding of the attribute.
PVL TYPE	Quoted String
EXAMPLE VALUE	ATTRIBUTE_DEFINITION = ‘This attribute corresponds to the issue and the revision of the current dictionary separated by a period’ ;

5.5 ATTRIBUTE_OBLIGATION

DEFINITION	Presence indicator of an attribute specifying whether the attribute shall always be present or only sometimes according to specified conditions.
PVL TYPE	{mandatory, M, conditional, C, optional, O, defaulted, D}
EXAMPLE VALUE	ATTRIBUTE_OBLIGATION = optional;

5.6 ATTRIBUTE_CONDITION

DEFINITION	Presence condition of an attribute.
PVL TYPE	Quoted String
EXAMPLE VALUE	ATTRIBUTE_CONDITION = ‘The attribute TEXT_SIZE must be present when the DATA_TYPE is Text’;

5.7 ATTRIBUTE_MAXIMUM_OCCURRENCE

DEFINITION	Maximum number of attribute occurrences in a data entity definition.
PVL TYPE	{Integer, ‘n’}
EXAMPLE VALUE	ATTRIBUTE_MAXIMUM_OCCURRENCE = 1;

5.8 ATTRIBUTE_VALUE_TYPE

DEFINITION	Type of the attribute value.
PVL TYPE	{Enumerated, Integer, Real, Text, Identifier, Entity_Type, PVL Sequence, PVL Set}

EXAMPLE VALUE	ATTRIBUTE_VALUE_TYPE = Identifier;
	ATTRIBUTE_VALUE_TYPE = (Text, Text);
	ATTRIBUTE_VALUE_TYPE = (Entity_Type, Text ;
	ATTRIBUTE_VALUE_TYPE = { Identifier, (Identifier, Identifier, Text) };

5.9 ATTRIBUTE_MAXIMUM_SIZE

DEFINITION	Maximum number of characters for representing the value of the attribute.
PVL TYPE	{Integer, PVL Sequence, PVL Set}

EXAMPLE VALUE	ATTRIBUTE_MAXIMUM_SIZE = 10;
	ATTRIBUTE_MAXIMUM_SIZE = {400, (400, 400, 1000)} ;

5.10 ATTRIBUTE_ENUMERATION_VALUES

DEFINITION	The distinct and discrete values of an Enumerated attribute.
PVL TYPE	PVL Set of Unquoted Strings
EXAMPLE VALUE	ATTRIBUTE_ENUMERATION_VALUES = {enumerated, text, real, integer, composite} ;

5.11 ATTRIBUTE_COMMENT

DEFINITION	Associated information about an attribute.
PVL TYPE	Quoted String
EXAMPLE VALUE	ATTRIBUTE_COMMENT = 'for all entities in the dictionary, in addition to the standard attributes, the attribute VERSION should be stated in order to allow traceability of the versions of the data entity definitions';

5.12 ATTRIBUTE_INHERITANCE

DEFINITION	Information about the inheritance rules for the attribute in a context of data entity modeling. The context is as follows: a data entity definition A inherits from another data entity definition B. The following cases describe what may happen for the values of the attributes of A for the different possible values of ATTRIBUTE_INHERITANCE for the attributes of B. – When the value of an attribute of B cannot be inherited, the attribute may be defined locally in the definition of A. – When the value of an attribute of B can be inherited, the value of this attribute is the value of the corresponding attribute of A, to which specialization rules can be applied.
PVL TYPE	{NOT_INHERITABLE, INHERITABLE}
EXAMPLE VALUE	ATTRIBUTE_INHERITANCE = INHERITABLE;

5.13 ATTRIBUTE_DEFAULT_VALUE

DEFINITION	Default value for the attribute that is conform to the type of the attribute, i.e., compliant with the value of the attribute_value_type descriptor.
PVL TYPE	Entity_Type
EXAMPLE VALUE	ATTRIBUTE_DEFAULT_VALUE = 10;

5.14 ATTRIBUTE_VALUE_EXAMPLE

DEFINITION	Examples and explanatory information.
PVL TYPE	Quoted String
EXAMPLE VALUE	ATTRIBUTE_VALUE_EXAMPLE = ‘Here is an example of a possible value: 2.4’;

5.15 ATTRIBUTE_SCOPE

DEFINITION	The category of entities to which the attribute is applicable.
PVL TYPE	{DATA, DICTIONARY, ALL}
EXAMPLE VALUE	ATTRIBUTE_SCOPE = ALL;

6 DEDSL CONFORMANCE

6.1 GENERAL

This *DEDSL—PVL Syntax* specification is version 1.0 of the Recommendation and provides a PVL implementation for the DEDSL Abstract Syntax Recommendation (reference [1]). Note that this part of the specification does not specify how the attribute names and values are to be linked to any given physical occurrence of a data entity within a data product. This allows a variety of formatting approaches to be used for this linking.

6.2 CONFORMANCE LEVEL 1: NOTATION COMPLIANCE

Implementations which implement all of sections 3, 4 and 5 will be Notation-compliant with this Recommendation.

6.3 CONFORMANCE LEVEL 2: FULL COMPLIANCE

Implementations which implement all of sections 3, 4 and 5 and the Interoperability constraints from the Abstract Specification will be Interoperable compliant with this Recommendation.

7 RESERVED KEYWORDS

The following reserved keywords are not available for use as declared identifiers or parameter names. Some of them are reserved keywords of the PVL (reference [2]) which are used by this Recommendation. Others are keywords defined by this Recommendation. Note that the case is not significant.

a) **PVL Keywords**

   – BEGIN_GROUP
   – END_GROUP

b) ***DEDSL—PVL Syntax* Keywords**

   – **Descriptor names:**

      • ATTRIBUTE_COMMENT
      • ATTRIBUTE_CONDITION
      • ATTRIBUTE_DEFAULT_VALUE
      • ATTRIBUTE_DEFINITION
      • ATTRIBUTE_ENUMERATION_VALUES
      • ATTRIBUTE_INHERITANCE
      • ATTRIBUTE_MAXIMUM_OCCURRENCE
      • ATTRIBUTE_MAXIMUM_SIZE
      • ATTRIBUTE_NAME
      • ATTRIBUTE_OBLIGATION
      • ATTRIBUTE_SCOPE
      • ATTRIBUTE_VALUE_EXAMPLE
      • ATTRIBUTE_VALUE_TYPE

   – **Data Entity attribute names**

      • ALIAS
      • CLASS
      • COMMENT
      • COMPONENT
      • CONSTANT_VALUE
      • DATA_TYPE
      • DEFINITION
      • ENUMERATION_CONVENTION
      • ENUMERATION_MEANING
      • ENUMERATION_VALUES
      • INHERITS_FROM
      • KEYWORD
      • NAME
      • RANGE
      • RELATION
      • SHORT_DEFINITION

- SPECIFIC_INSTANCE
- UNITS

– **Dictionary attribute names**

- CASE_SENSITIVITY
- DEDSL_VERSION
- DICTIONARY_DEFINITION
- DICTIONARY_IDENTIFIER
- DICTIONARY_NAME
- DICTIONARY_VERSION
- EXTERNAL_DICTIONARY_REFERENCE
- LANGUAGE
- TEXT_FIELD_CHARACTER_SET

– **Group names**

- COMPONENT_BLOCK
- DATA_ENTITY_DEFINITIONS
- DICTIONARY_ENTITY_DEFINITION
- DICTIONARY_IDENTIFICATION
- ENTITY_DEFINITION
- ENUMERATION
- ENUMERATION_VALUES_BLOCK
- INHERITS_FROM_BLOCK
- RELATION_BLOCK
- TEXT_SIZE_BLOCK
- USER_DEFINED_ATTRIBUTES

– **Parameter names**

- ENUMERATION_VALUE
- EXTERNAL_DICTIONARY
- OCCURRENCE_MAX
- OCCURRENCE_MIN
- REFERRED_ENTITY
- TEXT_SIZE_MAX
- TEXT_SIZE_MIN

– **Value names**

- Entity_Type

ANNEX A

EXAMPLES OF DATA ENTITIES

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

The following examples show how data entity definitions would appear in a single file.

A1 REAL DATA ENTITY EXAMPLE

The following example shows the ENTITY_DEFINITION associated with a data entity of type Real:

```
BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = Measurement_Date;
        ALIAS = (‘Elapsed_Time’, ‘elapsed time since mission start’ );
        CLASS = DATA_FIELD;
        DEFINITION = ‘This field contains the date of the instrument measurement delivering,
        expressed as the elapsed time since the start of the mission’;
        SHORT_DEFINITION = ‘Elapsed time since the mission start’;
        SPECIFIC_INSTANCE = (0.0, ‘Start of mission: 2nd of July 1999 at 3h45mn
        12.0067s PM’ );
        DATA_TYPE = Real;
        UNITS = {‘s’};
END_GROUP = ENTITY_DEFINITION;
```

A2 ENUMERATED DATA ENTITY EXAMPLE

The following example shows the ENTITY_DEFINITION associated with a data entity of type Enumerated (numeric values):

```
BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = EQUIPMENT_MODE;
        CLASS = DATA_FIELD;
        DEFINITION = ‘Mode of the equipment being used for taking measures’;
        SHORT_DEFINITION = ‘Equipment Mode’;
        DATA_TYPE = Enumerated;
        BEGIN_GROUP = ENUMERATION_VALUES_BLOCK ;
                BEGIN_GROUP = ENUMERATION;
                        ENUMERATION_VALUE = NOMINAL;
                        ENUMERATION_MEANING = ‘equipment on and working’;
                            ENUMERATION_CONVENTION = ‘0’;
                END_GROUP = ENUMERATION;
                BEGIN_GROUP = ENUMERATION;
                        ENUMERATION_VALUE = CALIBRATION;
                ENUMERATION_MEANING = ‘equipment under calibration tests’;
                ENUMERATION_CONVENTION = ‘1’;
            END_GROUP = ENUMERATION;
            BEGIN_GROUP = ENUMERATION;
                ENUMERATION_VALUE = OFF;
                ENUMERATION_MEANING = ‘equipment off’;
                ENUMERATION_CONVENTION = ‘2’;
            END_GROUP = ENUMERATION;
        END_GROUP = ENUMERATION_VALUES_BLOCK;
END_GROUP = ENTITY_DEFINITION;
```

A3 ENUMERATED DATA ENTITY EXAMPLE

The following example shows the ENTITY_DEFINITION associated with a data entity of type Enumerated (ascii values):

```
BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = MISSION;
        CLASS = DATA_FIELD;
        DEFINITION = ‘Different missions associated with the satellite’;
        DATA_TYPE = Enumerated;
        BEGIN_GROUP = ENUMERATION_VALUES_BLOCK;
            BEGIN_GROUP = ENUMERATION;
                ENUMERATION_VALUE = MARS;
                ENUMERATION_MEANING = ‘MARS mission’;
                ENUMERATION_CONVENTION = ‘  MARS’;
            END_GROUP = ENUMERATION;
            BEGIN_GROUP = ENUMERATION;
                ENUMERATION_VALUE = HELIOS;
                ENUMERATION_MEANING = ‘HELIOS mission’;
                ENUMERATION_CONVENTION = ‘HELIOS’;
            END_GROUP = ENUMERATION;
            BEGIN_GROUP = ENUMERATION;
                ENUMERATION_VALUE = MOON;
                ENUMERATION_MEANING = ‘MOON mission’;
                ENUMERATION_CONVENTION = ‘  MOON’;
            END_GROUP = ENUMERATION;
        END_GROUP = ENUMERATION_VALUES_BLOCK;
        COMMENT = ‘ The values of the data entity named MISSION are ASCII encoded
            enumerated values corresponding to strings of fixed length (6 characters)’ ;
END_GROUP = ENTITY_DEFINITION;
```

A4 REAL DATA ENTITY EXAMPLE

The following example shows the ENTITY_DEFINITION associated with a data entity of type Real (with a range of values):

```
BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = RMS;
        CLASS = DATA_FIELD;
        DEFINITION = ‘Components of B in GSE coordinates obtained by taking the root-
        mean-square of the values in the observing intervals’;
        SHORT_DEFINITION = ‘Components of RMS of B (GSE)’;
        COMMENT = ‘This is an example entity to demonstrate the DEDSL’;
        DATA_TYPE = Real;
        RANGE = (1.0 , 99.0);
        UNITS = { ‘deg’ };
END_GROUP = ENTITY_DEFINITION;
```

A5 COMPOSITE DATA ENTITY EXAMPLE

The following example shows the ENTITY_DEFINITION associated with a data entity of type Composite:

```
BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = DATE_TIME;
        CLASS = MODEL;
        DEFINITION = ‘Definition of the kind of date used to date telemetry frames’ ;
        SHORT_DEFINITION = ‘Telemetry frame date’ ;
        DATA_TYPE = Composite;
        COMPONENT = YEAR;
        COMPONENT = MONTH;
        COMPONENT = DAY;
        COMPONENT = HOUR;
        COMPONENT = MINUTE;
        COMPONENT = SECOND;
END_GROUP = ENTITY_DEFINITION;
```

A6 USE OF A USER-DEFINED ATTRIBUTE EXAMPLE

The following example shows a data entity description using a user-defined attribute (see Example 5-1 for its definition and note that some of the standard attributes are not applicable and therefore not included):

```
BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = Music_Instrument;
        CLASS = MODEL;
        DEFINITION = ‘It corresponds to an instrument’;
        DATA_TYPE = Enumerated;
        ENUMERATION_VALUES = {Piano, Guitar, Violin, Saxophone};
        AUDIO_EXAMPLE = ‘/home/MUSIC/INSTRUMENTS/SOUND/Piano.wav’;
END_GROUP = ENTITY_DEFINITION;
```

A7 INHERITANCE BETWEEN TWO COMPOSITE DATA ENTITIES EXAMPLE

The following example shows a case of inheritance between two data entity definitions (a MODEL entity and a DATA FIELD entity). In this example, the model entity has 3 components and the data field entity has 3 additional components as a result of the possibility of specialization. The value of the DEFINITION attribute of the data entity called DATE_TIME is the concatenation of the DEFINITION of the model entity DATE and of its own DEFINITION.

```
BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = DATE;
        CLASS = MODEL;
        DEFINITION = ‘It corresponds to the definition of the date’;
        DATA_TYPE = Composite;
        COMPONENT = YEAR;
        COMPONENT = MONTH;
        COMPONENT = DAY ;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = DATE_TIME;
        CLASS = DATA_FIELD;
        DEFINITION = ‘The time has been added to the date information’;
        INHERITS_FROM = DATE;
        COMPONENT = HOUR;
        COMPONENT = MINUTE;
        COMPONENT = SECOND;
 END_GROUP = ENTITY_DEFINITION;
```

A8 INHERITANCE BETWEEN TWO ENUMERATED DATA ENTITIES EXAMPLE

The following example shows a case of inheritance between two data entity definitions (a MODEL entity and a DATA FIELD entity). In this example, the model entity is an Enumerated giving different second formats, and the data field entity has less allowed formats as a result of the possibility of specialization. The list of allowed enumeration values has to be explicitly stated.

```
BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = CURRENT_SECOND_FORMATS;
        CLASS = MODEL;
        DEFINITION = 'Set of the second formats applicable in the project context.';
        DATA_TYPE = Enumerated;
        ENUMERATION_VALUES = {SECOND, SECOND_E_2, SECOND_E_4,
        MICRO_SECOND, SECOND_E_8, SECOND_E_10};
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = ALLOWED_SECOND_FORMATS;
        CLASS = DATA_FIELD;
        DEFINITION = 'Set of the current second formats applicable in the project context.
        Less formats are allowed.';
        SPECIFIC_INSTANCE = ( MICRO_SECOND, 'most frequently used for defining a
        date time format' );
        INHERITS_FROM = CURRENT_SECOND_FORMATS;
        ENUMERATION_VALUES = {SECOND, SECOND_E_2, SECOND_E_4,
        MICRO_SECOND};
END_GROUP = ENTITY_DEFINITION;
```

A9 INHERITANCE BETWEEN TWO INTEGER DATA ENTITIES EXAMPLE

The following example shows a case of inheritance between two data entity definitions (a MODEL entity and a DATA FIELD entity). In this example, the model entity is an Integer offering a choice of different distance units, and the data field entity has only one unit as a result of the possibility of specialization. Several meaningful specific instances can also be defined as a possible result of specialization.

```
BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = DISTANCE;
        CLASS = MODEL;
        DEFINITION = 'Distance generally speaking';
        UNITS = { 'mm', 'cm', 'dm', 'm', 'km'};
        DATA_TYPE = Integer;
        RANGE = (0 , 36000);
END_GROUP = ENTITY_DEFINITION;
```

```
BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = COVERED_DISTANCE;
        CLASS = DATA_FIELD;
        DEFINITION = ‘Distance covered during athletic competitions.’;
        SPECIFIC_INSTANCE = ( 50,’50 meters’ );
        SPECIFIC_INSTANCE = ( 5000,’5000 meters’ );
        INHERITS_FROM = DISTANCE;
        UNITS = {‘m’} ;
END_GROUP = ENTITY_DEFINITION;
```

ANNEX B

EXAMPLES

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

In this annex a community DED is presented showing the semantic information relative to the data entities chosen as being models. Then the definitions of product DEDs are presented, using this community DED to define some of their data entities.

All the following examples are the ones used in the Abstract Syntax annexes, which are now expressed in their PVL form.

B1 COMMUNITY DED

```
BEGIN_GROUP = DEDSL_DICTIONARY;

/* Dictionary block */

BEGIN_GROUP = DICTIONARY_IDENTIFICATION ;

        /* No new dictionary or global user-defined attributes */
        /* Data Entity Dictionary attributes */

        BEGIN_GROUP = DICTIONARY_ENTITY_DEFINITION;
                DICTIONARY_NAME = ‘Planetary_Science_Data_Dictionary’;
                DICTIONARY_DEFINITION =‘This dictionary contains data entity
                definitions relative to planetary science and which may be re-used for defining
                data products.’ ;
                TEXT_FIELD_CHARACTER_SET = ‘ISO-LATIN ALPHABET No1’;
                CASE_SENSITIVITY = NOT_CASE_SENSITIVE;
                LANGUAGE = (‘English’, en);
                DICTIONARY_VERSION = ‘1.a’;
                DEDSL_VERSION = ‘CCSDS 647.2-B-1.0’;
        END_GROUP = DICTIONARY_ENTITY_DEFINITION ;

END_GROUP = DICTIONARY_IDENTIFICATION;

/* Data entities block */
BEGIN_GROUP = DATA_ENTITY_DEFINITIONS;

/* No new data entity user-defined attributes */
BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = LATITUDE_MODEL;
        ALIAS = (‘LAT’, ‘Used by the historical projects EARTH_PLANET’);
        CLASS = MODEL;
        DEFINITION = ‘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_DEFINITION = ‘Latitude’;
        UNITS = { ‘deg’ };
        SPECIFIC_INSTANCE = (+00.000, ‘Equator’);
        DATA_TYPE = REAL;
        RANGE = (-90.00, +90.00);
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = LONGITUDE_MODEL;
        ALIAS = (‘LON’, ‘Used by the historical projects EARTH_PLANET’);
        CLASS = MODEL;
        DEFINITION = ‘Longitudes east of Greenwich shall be designated by the use of
         the plus sign (+), longitudes west of Greenwich 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_DEFINITION = ‘Longitude’;
        UNITS = { ‘deg’ };
        SPECIFIC_INSTANCE = (-180.000, ‘The 180th Meridian’);
        DATA_TYPE = REAL;
        RANGE = (-180.00, +180.00);
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = PRODUCT_ID_MODEL;
        ALIAS = (‘PRODUCT_NAME’, ‘Used by the historical projects
                   EARTH_PLANET to identify their data products’);
        CLASS = MODEL;
        DEFINITION = ‘The PRODUCT_ID represents a permanent, unique identifier
        assigned to a data product by its producer.’;
        SHORT_DEFINITION = ‘Product Identification’;
        DATA_TYPE = TEXT;
        TEXT_SIZE_MAX = 40;
END_GROUP = ENTITY_DEFINITION;

END_GROUP = DATA_ENTITY_DEFINITIONS;

END_GROUP = DEDSL_DICTIONARY;
```

B2 DATA ENTITY DICTIONARY ASSOCIATED WITH PRODUCT_X

The models of LATITUDE_MODEL, LONGITUDE_MODEL and PRODUCT_ID_MODEL match the data entities appearing within the data product PRODUCT_X and therefore they are referenced within the current data entity dictionary.

```
BEGIN_GROUP = DEDSL_DICTIONARY ;

/* Data Entity Dictionary attributes */

BEGIN_GROUP = DICTIONARY_IDENTIFICATION;

        BEGIN_GROUP = DICTIONARY_ENTITY_DEFINITION;
                DICTIONARY_NAME = PRODUCT_X _Dictionary;
                DICTIONARY_DEFINITION = ‘This dictionary contains the data entity
                definitions relative to the data product PRODUCT_X.’;
                EXTERNAL_REFERENCE_DICTIONARY = (Planetary_Science_Data_
                Dictionary, FCST0172, CCSDS_Control_Authority);
                TEXT_FIELD_CHARACTER_SET = ‘ISO-LATIN ALPHABET No1’;
                CASE_SENSITIVITY = NOT_CASE_SENSITIVE;
                LANGUAGE = (‘English’, en);
                DICTIONARY_VERSION = ‘1.a’;
                DEDSL_VERSION = ‘CCSDS 647.2-B-1.0’;
        END_GROUP = DICTIONARY_ENTITY_DEFINITION ;

END_GROUP = DICTIONARY_IDENTIFICATION ;

/* Dictionary entities */
BEGIN_GROUP = DATA_ENTITY_DEFINITIONS;

BEGIN_GROUP = USER_DEFINED_ATTRIBUTES ;

        BEGIN_GROUP = ATTRIBUTE_DEFINITION;
                ATTRIBUTE_NAME = FIELD_LOCATION;
                ATTRIBUTE_DEFINITION = ‘Provides the location of the field within
                 the data product. It corresponds to the series of the names of the
                encapsulating composite entities separated by a point and ending with the
                 name of the field’;
                ATTRIBUTE_OBLIGATION = Conditional;
                ATTRIBUTE_CONDITION = ‘for data fields only’;
                ATTRIBUTE_MAXIMUM_OCCURRENCE = 1;
                ATTRIBUTE_VALUE_TYPE = Text ;
                ATTRIBUTE_MAXIMUM_SIZE = 1024 ;
                ATTRIBUTE_INHERITANCE = NOT_INHERITABLE ;
                ATTRIBUTE_VALUE_EXAMPLE = ‘date.year’;
                ATTRIBUTE_SCOPE = DATA;
        END_GROUP = ATTRIBUTE_DEFINITION;

END_GROUP = USER_DEFINED_ATTRIBUTES ;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = HEADER;
        CLASS = DATA_FIELD;
        DEFINITION = ‘It represents the header of the data product PRODUCT_X. It
        identifies an aggregation of values which are associated with an image array.’;
        SHORT_DEFINITION = ‘Image Header Values’;
        COMPONENT = PRODUCT_ID_X;
        COMPONENT = ACQ_STATION;
        COMPONENT = ACQ_TIME;
        COMPONENT = CENTRE_COORD;
        DATA_TYPE = COMPOSITE;
        FIELD_LOCATION = ‘HEADER’;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = PRODUCT_ID;
        CLASS = DATA_FIELD;
        DEFINITION = ‘It represents a permanent, unique identifier assigned to the data
        product PRODUCT_X.’;
        SHORT_DEFINITION = ‘Product Identification’;
        INHERITS_FROM = PRODUCT_ID_MODEL;
        FIELD_LOCATION = ‘PRODUCT_ID’;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = ACQ_STATION;
        ALIAS = (‘ACQUSTAT’, ‘used in the header’);
        CLASS = DATA_FIELD;
        DEFINITION = ‘It includes the identifier of the station which has acquired the
        data.’;
        SHORT_DEFINITION = ‘Identifier of the acquisition station’;
        DATA_TYPE = Enumerated;
        BEGIN_GROUP = ENUMERATION_VALUES_BLOCK;
                BEGIN_GROUP = ENUMERATION;
                        ENUMERATION_VALUE = AMERICA;
                        ENUMERATION_MEANING = , ‘station located in America’;
                ENG_GROUP = ENUMERATION ;
                BEGIN_GROUP = ENUMERATION ;
                        ENUMERATION_VALUE = EUROPE;
                        ENUMERATION_MEANING =  ‘station located in Europe’;
                END_GROUP = ENUMERATION;
                BEGIN_GROUP = ENUMERATION ;
```

```
                        ENUMERATION_VALUE = ASIA;
                        ENUMERATION_MEANING =  ‘station located in Asia’;
                END_GROUP = ENUMERATION;
        END_GROUP = ENUMERATION_VALUES_BLOCK;
        FIELD_LOCATION = ‘ACQ_STATION’ ;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = ACQ_TIME;
        ALIAS = (‘ACQUTIME’, ‘Used in the header’);
        CLASS = DATA_FIELD;
        DEFINITION = ‘It represents the date and time of the acquisition of the data. Its
         format is the following one: YYYY-MM-DDThh:mm:ss.d_>Z. It conforms to the
        CCSDS ISO rules for date/time definitions.  The acquisition time should
        correspond to the first scan line of the data.’;
        SHORT_DEFINITION = ‘Date/Time of the data acquisition’;
        DATA_TYPE = Text;
        TEXT_SIZE_MAX = 40;
        FIELD_LOCATION =‘ACQ_TIME’;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = CENTRE_COORD;
        CLASS = DATA_FIELD;
        DEFINITION = ‘It represents a coordinate centre.’;
        SHORT_DEFINITION = ‘Centre coordinates’;
        COMPONENT = LATITUDE;
        COMPONENT = LONGITUDE;
        KEYWORD = ‘LATITUDE BY LONGITUDE COORDINATE CENTRE’;
        DATA_TYPE = COMPOSITE;
        FIELD_LOCATION = ‘CENTRE_COORD’;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = LATITUDE;
        CLASS = DATA_FIELD;
        DEFINITION = ‘It represents the latitude used for the centre coordinate.’;
        INHERITS_FROM = LATITUDE_MODEL;
        FIELD_LOCATION =‘CENTRE_COORD.LATITUDE’ ;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = LONGITUDE;
        CLASS = DATA_FIELD;
        DEFINITION = ‘It represents the longitude used for the centre coordinate.’;
        INHERITS_FROM = LONGITUDE_MODEL;
        FIELD_LOCATION = ‘CENTRE_COORD.LONGITUDE’;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = W_IMAGE_SIZE;
        CLASS = CONSTANT;
        DEFINITION = ‘It represents the number of pixels for an image taken from
         spacecraft W.’;
        SHORT_DEFINITION = ‘Spacecraft W Image pixel’;
        RELATION = (‘size of’, DATA_1);
        DATA_TYPE = INTEGER;
        CONSTANT_VALUE = 1 440 000;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = DATA_1;
        CLASS = DATA_FIELD;
        DEFINITION = ‘It represents an image taken from spacecraft W.’;
        SHORT_DEFINITION = ‘Spacecraft W Image’;
        COMMENT = ‘The image is an array of W_IMAGE_SIZE items called
         DATA_1_PIXEL’;
        BEGIN_GROUP = COMPONENT_BLOCK;
                COMPONENT = DATA_1_PIXEL;
                OCCURRENCE_MIN = 1;
                OCCURRENCE_MAX = W_IMAGE_SIZE;
        END_GROUP = COMPONENT_BLOCK;
        KEYWORD = ‘IMAGE’;
        DATA_TYPE = COMPOSITE;
        FIELD_LOCATION = ‘DATA_1’;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = DATA_1_PIXEL;
        CLASS = DATA_FIELD;
        DEFINITION = ‘It represents a pixel belonging to the image taken from
        spacecraft W.’;
        SHORT_DEFINITION = ‘Spacecraft W Image pixel’;
        DATA_TYPE = INTEGER;
        RANGE = (0 , 255);
        FIELD_LOCATION = ‘DATA_1.DATA_1_PIXEL’;
END_GROUP = ENTITY_DEFINITION;

END_GROUP = DATA_ENTITY_DEFINITIONS;

END_GROUP = DEDSL_DICTIONARY;
```

B3 DATA ENTITY DICTIONARY ASSOCIATED WITH PRODUCT_Y

The models of LATITUDE_MODEL, LONGITUDE_MODEL and PRODUCT_ID_MODEL match the data entities appearing within the data product PRODUCT_Y and therefore they are referenced within the current data entity dictionary.

```
BEGIN_GROUP = DEDSL_DICTIONARY ;

/* Data Entity Dictionary attributes */

BEGIN_GROUP = DICTIONARY_IDENTIFICATION;
        BEGIN_GROUP = DICTIONARY_ENTITY_DEFINITION;
                DICTIONARY_NAME = PRODUCT_Y _Dictionary;
                DICTIONARY_DEFINITION = ‘This dictionary contains the data entity
                definitions relative to the data product PRODUCT_Y.’;
                EXTERNAL_REFERENCE_DICTIONARY = (Planetary_Science_Data_
                Dictionary, FCST0172, CCSDS_Control_Authority);
                TEXT_FIELD_CHARACTER_SET = ‘ISO-LATIN ALPHABET No1’;
                CASE_SENSITIVITY = NOT_CASE_SENSITIVE;
                LANGUAGE = (‘English’, en);
                DICTIONARY_VERSION = ‘1.a’;
                DEDSL_VERSION = ‘ CCSDS 647.2-B-1.0’;
        END_GROUP = DICTIONARY_ENTITY_DEFINITION ;
END_GROUP = DICTIONARY_IDENTIFICATION ;

/* Dictionary entities */

BEGIN_GROUP = DATA_ENTITY_DEFINITIONS;

BEGIN_GROUP = USER_DEFINED_ATTRIBUTES ;

        BEGIN_GROUP = ATTRIBUTE_DEFINITION;
                ATTRIBUTE_NAME = FIELD_LOCATION;
                ATTRIBUTE_DEFINITION = ‘Provides the location of the field within
                 the data product. It corresponds to the series of the names of the
                encapsulating composite entities separated by a point and ending with the
                 name of the field’;
                ATTRIBUTE_OBLIGATION = Conditional;
                ATTRIBUTE_CONDITION = ‘for data fields only’;
                ATTRIBUTE_MAXIMUM_OCCURRENCE = 1;
                ATTRIBUTE_VALUE_TYPE = Text ;
                ATTRIBUTE_MAXIMUM_SIZE = 1024 ;
                ATTRIBUTE_INHERITANCE = NOT_INHERITABLE ;
                ATTRIBUTE_VALUE_EXAMPLE = ‘date.year’;
                ATTRIBUTE_SCOPE = DATA;
        END_GROUP = ATTRIBUTE_DEFINITION;
END_GROUP = USER_DEFINED_ATTRIBUTES ;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = PRODUCT_ID;
        CLASS = DATA_FIELD;
        DEFINITION = ‘It represents a permanent, unique identifier assigned to the data
        product PRODUCT_Y.’;
        SHORT_DEFINITION = ‘Product Identification’;
        INHERITS_FROM = PRODUCT_ID_MODEL;
        FIELD_LOCATION = ‘PRODUCT_ID’;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = LATITUDE;
        CLASS = DATA_FIELD;
        DEFINITION = ‘It represents the latitude used for the centre coordinate.’;
        INHERITS_FROM = LATITUDE_MODEL;
        FIELD_LOCATION = ‘LATITUDE’;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = LONGITUDE;
        CLASS = DATA_FIELD;
        DEFINITION = ‘It represents the longitude used for the center coordinate.’;
        INHERITS_FROM = LONGITUDE_MODEL;
        FIELD_LOCATION = ‘LONGITUDE’;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = W_IMAGE_SIZE;
        CLASS = CONSTANT;
        DEFINITION = ‘It represents the number of pixels for an image taken from
         spacecraft W2.’;
        SHORT_DEFINITION = ‘Spacecraft W2 Image pixel’;
        RELATION = (‘size of’, DATA_2);
        DATA_TYPE = INTEGER;
        CONSTANT_VALUE = 1 440 000;
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = DATA_2;
        CLASS = DATA_FIELD;
        DEFINITION = ‘It represents an image taken from spacecraft W2.’;
        SHORT_DEFINITION = ‘Spacecraft W2 Image’;
        COMMENT = ‘The image is an array of W_IMAGE_SIZE items called
```

```
         DATA_2_PIXEL' ;
        BEGIN_GROUP = COMPONENT_BLOCK;
                COMPONENT = DATA_2_PIXEL;
                OCCURRENCE_MIN = 1;
                OCCURRENCE_MAX = W_IMAGE_SIZE;
        END_GROUP = COMPONENT_BLOCK;
        KEYWORD = 'IMAGE';
        DATA_TYPE = COMPOSITE;
        FIELD_LOCATION = 'DATA_2';
END_GROUP = ENTITY_DEFINITION;

BEGIN_GROUP = ENTITY_DEFINITION;
        NAME = DATA_2_PIXEL;
        CLASS = DATA_FIELD;
        DEFINITION = 'It represents a pixel belonging to the image taken from
        spacecraft W2.';
        SHORT_DEFINITION = 'Spacecraft W2 Image pixel';
        DATA_TYPE = INTEGER;
        RANGE = (0 , 255);
        FIELD_LOCATION = 'DATA_2.DATA_2_PIXEL' ;
END_GROUP = ENTITY_DEFINITION;

END_GROUP = DATA_ENTITY_DEFINITIONS;

END_GROUP = DEDSL_DICTIONARY;
```

ANNEX C

INFORMATIVE REFERENCES

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

This annex provides a list of references that may be valuable to the user of this Recommendation as background material or to provide implementation guidelines for using this Recommendation.

[**C**1] *Procedures Manual for the Consultative Committee for Space Data Systems*. CCSDS A00.0-Y-7. Yellow Book. Issue 7. Washington, D.C.: CCSDS, November 1996.

[C2] *Standard Formatted Data Units—A Tutorial*. Report Concerning Space Data System Standards, CCSDS 621.0-G-1. Green Book. Issue 1. Washington, D.C.: CCSDS, May 1992.

[C3] *Parameter Value Language—A Tutorial*. Report Concerning Space Data System Standards, CCSDS 641.0-G-1. Green Book. Issue 1. Washington, D.C.: CCSDS, May 1992.

[C4] *Standard Formatted Data Units—Control Authority Procedures Tutorial*. Report Concerning Space Data System Standards, CCSDS 631.0-G-2. Green Book. Issue 2. Washington, D.C.: CCSDS, November 1994.

[C5] *UNIDATA Units Package*. NCAR, Version 1.11.5, 18 August 1997.

[C6] *Time Code Formats*. Recommendation for Space Data Systems Standards, CCSDS 301.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, April 1990. (ISO 11104.)

[C7] Douglas Steedman. *Abstract Syntax Notation One (ASN.1)—The Tutorial and Reference*. Twickenham: Technology Appraisals, 1990.