Consultative Committee for Space Data Systems

RECOMMENDATION FOR SPACE DATA SYSTEM STANDARDS

STANDARD FORMATTED DATA UNITS — STRUCTURE AND CONSTRUCTION RULES

CCSDS 620.0-B-2

BLUE BOOK

May 1992

Note:
This current
issue includes
all updates through
Technical Corrigendum 1,
dated November 1996.

AUTHORITY

Issue:	Blue Book, Issue 2
Date:	May 1992
Location:	CCSDS Panel 2 Meeting, May 1992, Oberpfaffenhofen, Germany

This Recommendation reflects the consensus technical agreement of the following member Agencies of the Consultative Committee for Space Data Systems (CCSDS):

- British National Space Centre (BNSC) / United Kingdom
- Canadian Space Agency (CSA) / Canada
- Centre National D'Etudes Spatiales (CNES) / France
- Deutsche Forschungsanstalt für Luft und Raumfahrt (DLR) / FRG
- European Space Agency (ESA) / Europe
- Instituto de Pesquisas Espaciais (INPE) / Brazil
- National Aeronautics and Space Administration (NASA) / USA
- National Space Development Agency of Japan (NASDA) / Japan

The following observer Agencies also concur with this Recommendation:

- Department of Communication/Communications Research Centre (DOC/CRC) / Canada
- Institute for Space Astronautics and Science (ISAS) / Japan

This Recommendation is published and maintained by:

CCSDS Secretariat
Communications and Data Systems Division, (Code-OS)
National Aeronautics and Space Administration
Washington, DC 20546, USA

FOREWORD

This document is a technical Recommendation for the standardisation of the structure and construction rules of Standard Formatted Data Units (SFDU), for the interchange of digital space-related data in an open data system and has been prepared by the Consultative Committee for Space Data Systems (CCSDS). Other aspects of the SFDU concept are described in documents listed in the Reference section.

This Recommendation defines SFDU structures that will handle some of the problems of digital data interchange and several construction rules that will limit the SFDUs to a practical set that can exist in an open data system environment. It allows implementing organisations within each Agency to proceed coherently with the development of compatible derived Standards for space data systems and widely dispersed data users that are within their cognisance.

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 which are defined in Reference [1].

Questions relating to the contents or status of this document should be addressed to the CCSDS Secretariat.

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 to 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.

DOCUMENT CONTROL

Document	***Title***	***Date***	***Status/ Remarks***
CCSDS 620.0-B-1	Recommendation for Space Data System Standards: Standard Formatted Data Units -- Structure and Construction Rules, Blue Book, Issue 1	Feb 1988	Issue 1
CCSDS 620.0-B-2	Recommendation for Space Data System Standards: Standard Formatted Data Units -- Structure and Construction Rules, Blue Book, Issue 2	May 1992	Issue 2, see note below

Note: The major changes from Issue 1 to Issue 2 are the inclusion of techniques for the following:

- For linking a data description with its identifier (ADID).
- For delimiting data objects other than by length.
- For referencing external labelled and unlabelled data objects.

Issue 2 is forwards compatible with the whole of Issue 1.

CONTENTS

Sections

Figures

Tables

REFERENCES

[1] "Procedures Manual for the Consultative Committee for Space Data Systems", CCSDS A00.0-Y-4, Yellow Book, Issue 4, Consultative Committee for Space Data Systems, September 1990.

[2] "Recommendation for Space Data System Standards: Standard Formatted Data Units -- Control Authority Procedures", CCSDS 630.0-R-2, Red Book, Issue 2, Consultative Committee for Space Data Systems, April 1992 or later.

[3] "Report Concerning Space Data System Standards: Standard Formatted Data Units -- A Tutorial", CCSDS 621.0-G-1, Green Book, Issue 1, Consultative Committee for Space Data Systems, May 1992 or later.

[4] "Information Technology - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)", ISO 8824:1990E, Second Edition, December 1990.

[5] "Recommendation for Space Data System Standards: Parameter Value Language Specification (CCSD0006)", CCSDS 641.0-B-1, Blue Book, Issue 1, Consultative Committee for Space Data Systems, May 1992 or later.

[6] "Recommendation for Space Data System Standards: ASCII Encoded English (CCSD0002)", CCSDS 643.0-R-1, Red Book, Issue 1, Consultative Committee for Space Data Systems, May 1992 or later.

Cor. 1

[7] "Recommendation for Space Data System Standards: Standard Formatted Data Units -- Control Authority Data Structures", CCSDS 632.0-B-1, Blue Book, Issue 1, Consultative Committee for Space Data Systems, November 1994 or later.

[8] "Recommendation for Space Data System Standards: Standard Formatted Data Units -- Referencing Environment", CCSDS 622.0-R-1, Red Book, Issue 1, Consultative Committee for Space Data Systems, November 1994 or later.

1 INTRODUCTION

1.1 Purpose and Scope

The purpose of this document is to establish a Recommendation for the implementation of standard data structures for the interchange of data in a more uniform and automated fashion within and between the Agencies participating in the Consultative Committee for Space Data Systems (CCSDS).

This Recommendation defines the Standard Formatted Data Unit (SFDU) Concept. This concept covers the following areas:

1) A method for labelling data objects to provide a general classification and a link to a unique description of each data object;

2) A method of organising data objects into a hierarchial structure to provide a complete set of information. This includes referencing of data objects that are defined externally, such as files.

1.2 Applicability

This Recommendation serves as a guideline for the development of compatible agency standards in the field of digital data interchange. The specifications in this document are to be invoked through the normal standards program of each member agency and are applicable, at a minimum, to those missions and services for which cross support based on the need for open system data interchange is anticipated.

To be compatible with the CCSDS SFDU concept, an agency must use the structures as defined by this Recommendation.

1.3 Recommended Approach to Reading the Document

A proper understanding of this Recommendation requires familiarity with the SFDU concept, the rationale underlying the various product building and packaging techniques and the specific terminology used in this document. It is recommended that Reference [3] be read prior to this Recommendation as it describes both the requirements and the rationale behind the SFDU concept, and introduces the technical material contained in this Recommendation through examples.

The document is structured as follows:

- Section 2 gives a general overview of the SFDU concept, and introduces the basic data structuring components;

- Section 3 describes the details of the LABEL-VALUE structure used by the SFDU concept and defines all its sub-fields;

- Section 4 describes CCSDS defined Class IDs and, where applicable, the relationships between them;

- Section 5 describes the CCSDS defined Authority and Description Identifiers (ADIDs) which are relevant to this Recommendation, with regard to the rules for constructing and parsing the corresponding VALUE fields;
- Section 6 describes the CCSDS defined combinations of ADIDs and Class IDs;
- Annexes A and B present a complete summary of the acronyms and the terminology used in this document;
- Annex C gives a formal specification in Abstract Syntax Notation One (ASN.1, see Reference [4]), of the structures presented in this Recommendation;
- Annexes D and E specify the ASCII character codes and octet/bit numbering conventions used throughout the document;
- An index is supplied covering all the major terms in the document, the first page referenced by the index points to the definition of the term.

Throughout this Recommendation structure diagrams are used to explain the structures presented. The following conventions are used in these diagrams:

- The item named to the left of the **:=** symbol is the item being defined;
- The diagram on the right of the **:=** symbol is the definition;
- A vertical branch point represents a choice;
- A repetition is indicated by a loop back covering the object to be repeated. If there are the symbols **n<x** next to the loop back, then it means the loop can be repeated only 0 to **x** number of times;
- The termination of each structure is represented by an **o** symbol.

For example:

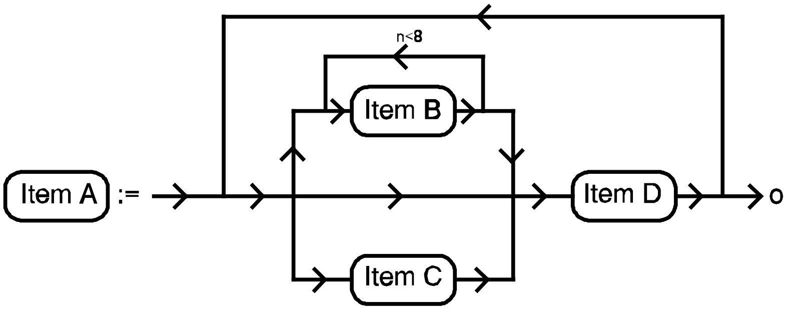


Figure 1-1: Example Structure Diagram

*In this example Item A is defined as first a choice between Items B or C or nothing, if Item B is selected then it may be repeated further from 1 to 7 times. Then this structure is followed by one Item D. Once this structure is built up, it may then all be repeated any number of times, until the choice to pass onto the **o** symbol is taken. Of course if any items on the right (B, C or D) contain an Item A, the definition is recursive. Recursive structure definitions are permitted in this Recommendation.*

Where possible Abstract Syntax Notation One (ASN.1, see Reference [4]) definitions of the structures presented in this Recommendation are given in Annex C. In the case of any unintentional inconsistency with the ASN.1 specification, the specification given in Sections 1 to 6 is the ruling specification (This is because the ASN.1 cannot wholly describe the structures presented here, without additional natural language comments).

2 SFDU OVERVIEW

2.1 Introduction

This Recommendation defines under the Standard Formatted Data Unit (SFDU) concept, methods for packaging supplementary data and metadata with space related science and engineering data to create data products that contain complete sets of information for the purpose of information interchange.

Any type of data can be integrated into the SFDU domain; what is standardised is the technique of packaging together the various data objects into an SFDU data product. There is no constraint on the format of the user data.

Data instances are of little value without descriptions of their contents and organisation. The SFDU concept integrates data and metadata and provides a technique to identify the different types of packaged information.

2.2 The SFDU Building Block - The LABEL-VALUE-OBJECT

The basic SFDU building block is comprised of a LABEL field and a VALUE field, and is referred to as a Label-Value-Object (LVO). This structure is the fundamental structural element used to build SFDUs. The LVOs themselves are made up of a sequence of octets.

In the SFDU approach, data exchanged between open (independent) data systems are tagged with a LABEL, as shown in Figure 2-1.

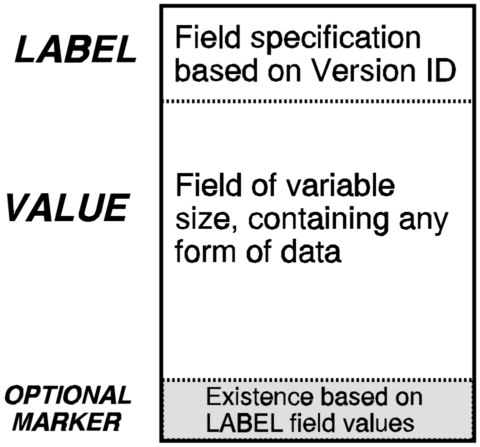

LABEL
Field specification based on Version ID
VALUE
Field of variable size, containing any form of data
OPTIONAL MARKER
Existence based on LABEL field values

Figure 2-1: The LABEL-VALUE-OBJECT Structure

The LABEL contains the following sub-fields:

- An identifier of the format and meanings of all the other LABEL sub-fields;
- An identifier of the description of the format and meaning of the data in the VALUE field;

- An identifier that gives an indication of the type of data in the VALUE field;
- The necessary information required to delimit the VALUE field.

A complete definition of the sub-fields of the LABEL is given in Section 3.

The VALUE field may contain any form of data that can be described by a user defined data description or by a CCSDS recognised data description as defined in Sections 5 and 6. The method used to delimit this field, and a description of the data in this field, are identified through the associated LABEL field.

The optional marker field is required by some delimitation techniques to delimit the VALUE field (See Section 3.3).

2.3 SFDU Structuring

SFDU data products are constructed from the basic LVO in one of two ways. If the VALUE field of the LVO contains purely user data it is termed a "Simple LVO". If, on the other hand, the VALUE field of the LVO contains purely LVOs, it is termed a "Compound LVO". There are various CCSDS defined categories of Simple and Compound LVOs depending upon the type of data or LVOs respectively that they contain, as detailed in the following sections.

2.3.1 Simple LVOs

Data in a Simple LVO may be viewed as belonging to one of the following categories:

- Application data; that is the data which is of primary interest (typically actual measurements or data derived from actual measurements);
- Supplementary data; that is data that is considered to enhance the understanding of the associated data;
- Data description information, telling how the application data are formatted, including such details as size of the data fields, numerical or other representations used and the meanings of the fields;
- Data cataloguing/production information, telling how the data may be divided, for example, by date, instrument used, instrument location or general information about the way the data was collected, relayed or processed.

Any of these types of data may be contained in the VALUE field of a single LVO. The structure of a Simple LVO can be described by Figure 2-2[(1)] (overleaf).

[(1)] *For clarity in this diagram, the optional marker component is not shown.*

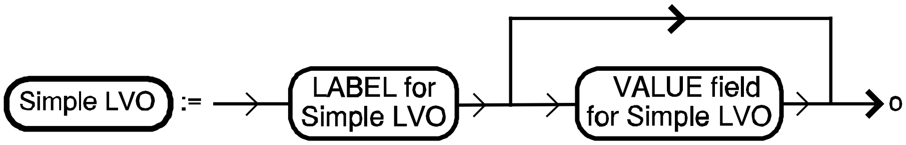


Figure 2-2: Structure Diagram of a Simple LVO

2.3.2 Compound LVOs

Compound LVOs are LVOs which contain within their VALUE field a sequence of one or more LVOs, each of which can be a Simple or Compound LVO itself; this sequence of LVOs is deemed to be one "Structure Level" lower than that of the containing Compound LVO. Any Compound LVOs in this sequence will themselves contain a sequence of LVOs; this sequence is at the next lower "Structure Level". This process may continue indefinitely leading to a succession of structure levels. This process is the way in which LVOs are nested. The structure of a Compound LVO can be described by Figure 2-3[2].

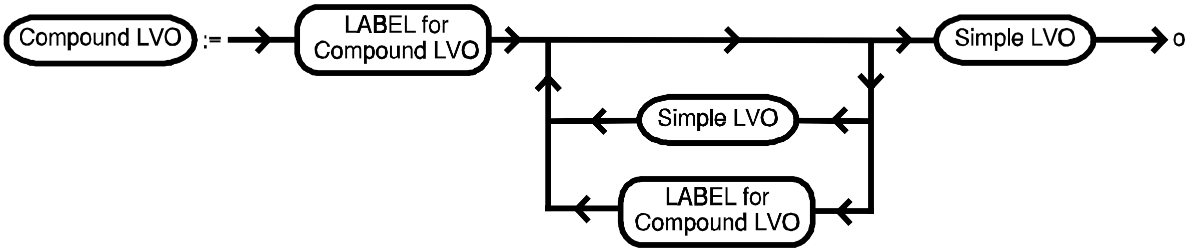


Figure 2-3: Structure Diagram of a Compound LVO

There are three Compound LVOs defined in this Recommendation; there is the Exchange Data Unit (EDU), and two particular structures which must be packaged within an EDU. These are the Application Data Unit (ADU), which explicitly does not contain any data description information, and the Description Data Unit (DDU), which must contain only data description information.

2.3.2.1 Exchange Data Unit (EDU)

Typically an EDU data product consists of not only the data *(e.g., an image, a set of measurement samples)*, but also all the supporting metadata that is needed to understand the data product. Any type of data may be contained within an EDU, whether it be packaged in Simple LVOs or Compound LVOs.

For example, the following could be packaged within an EDU:

- Information on the production of the product;
- Catalogue data pertaining to the product *(e.g., platform ID, data type, time span)*;

[2] *For clarity in this diagram, the optional marker component is not shown.*

- A number of data instances comprising of application data and supplementary data *(e.g., images and image coordinates)*;

- A number of data descriptions that describe each of the application data formats and the supplementary data formats.

2.3.2.2 Application Data Unit (ADU)

The purpose of an ADU is to package application data instances *(e.g., measurement samples)* together with any necessary ancillary data *(e.g., sampling rate)* and identification data *(e.g., catalogue information)*, and to explicitly exclude any data description information. Typically an ADU will be used to "subset" a set of application data within an EDU. Software may be designed so that data stored within an ADU is directed towards one particular processing task, which selects the ADUs out of the data structure and skips over any others. Several such units typically appear within a data product. An ADU must always be packaged within an EDU. An example of the use of ADUs is shown in Figure 2-4.

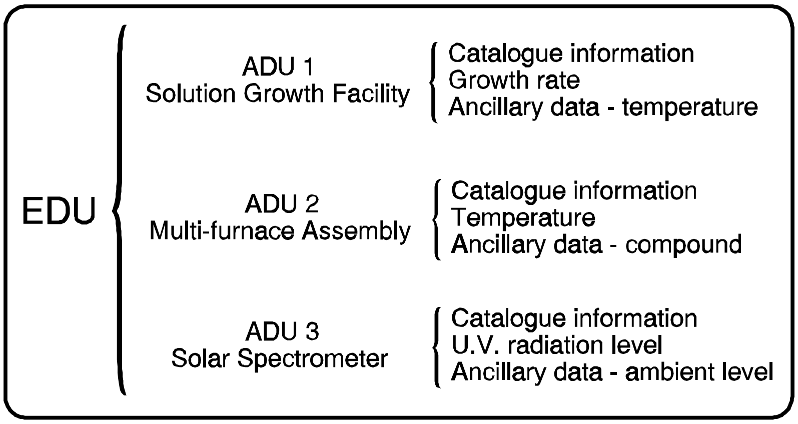


Figure 2-4: Example of the Use of ADUs

2.3.2.3 Description Data Unit (DDU)

A Description Data Unit consists of the following:

- It carries the description of a data object (typically syntactic information such as the format of a sample, and semantic information such as the name and units of the components of the sample);

- It explicitly links the data description to the data object to which it applies;

- It does not include any application data instances.

Like an ADU, a DDU must always be packaged within an EDU. Figure 2-5 (overleaf) shows a DDU that could be used to describe the data within the ADU shown in Figure 2-4. This data description data may be packaged together with the application data or held separately.

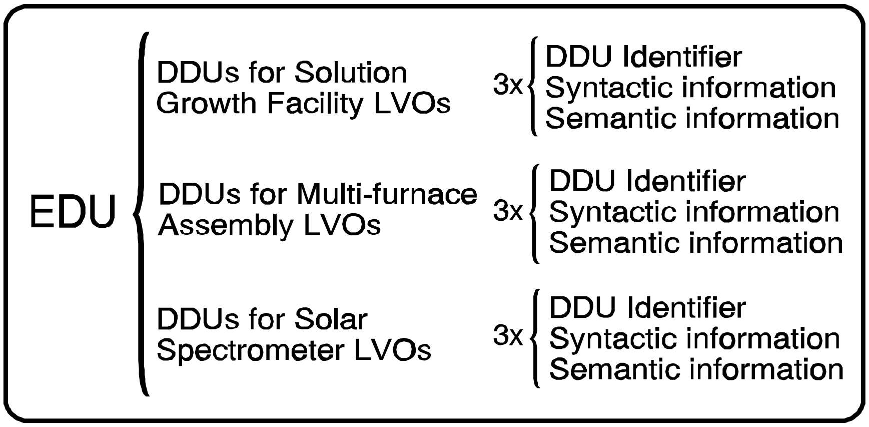
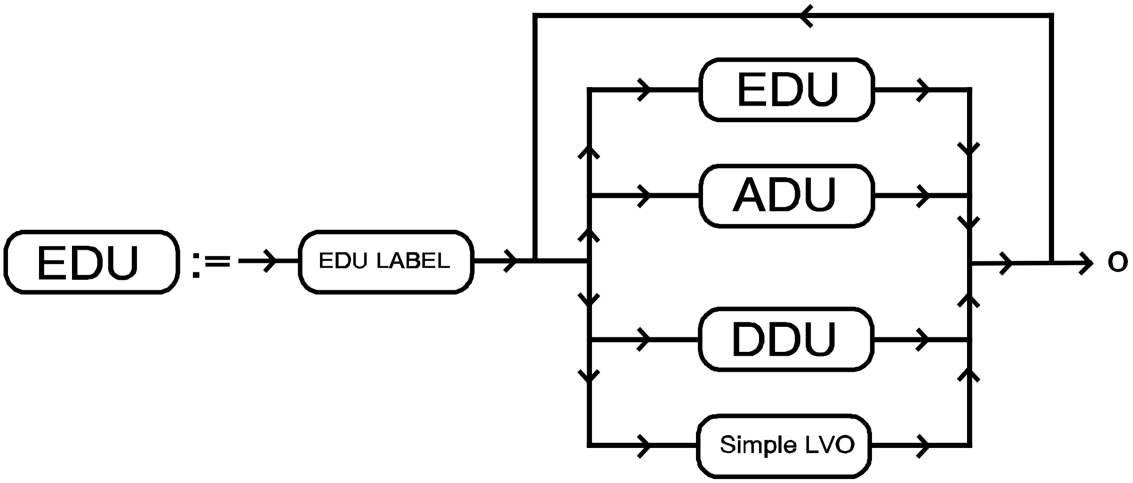


Figure 2-5: Example of the Use of DDUs

2.4 EDU Structure Diagram

The four structures that have been illustrated in the previous section are the Simple Label-Value-Object (LVO), the Exchange Data Unit (EDU), the Application Data Unit (ADU) and the Description Data Unit (DDU). These structures may be packaged together as indicated in the structure diagram of Figure 2-6[3]. Not all the components on the right of the **:=** must be included in all EDUs, but at least one Simple LVO must be present. This shows a clear hierarchical approach to packaging, with all data objects being packaged within an EDU. Recall that ADUs and DDUs contain further LVOs.



where: EDU = an Exchange Data Unit
ADU = an Application Data Unit
DDU = a Description Data Unit
and a Simple LVO is an LVO that has no further LVOs in its VALUE field

Figure 2-6: Structure Diagram of an EDU

[3] *For clarity in this diagram, the optional marker component is not shown.*

3 LVO LABEL FIELD SPECIFICATIONS USED IN SFDUS

Sections 3.1 to 3.3 specify the three versions of the LVO LABEL defined in this Recommendation. The version of the LABEL is indicated by octet 4 of the LABEL in all cases. This sub-field of the LABEL is called the **Version ID**.

(An SFDU product must be immediately identifiable as falling within the CCSDS/SFDU domain; therefore octets 0 to 3 must contain the string ***CCSD****, which is the Control Authority Identifier (CAID) for the CCSDS. Therefore, that even though the Version ID dictates the format of the LABEL, it may not appear prior to the string* ***CCSD*** *and thus appears in octet 4.)*

The octet numbering conventions, as described in Annex E, apply only to the LABEL field, not to the VALUE field. In addition, a restricted ASCII character set, denoted by RA (see Annex D), is used in many sub-fields of the LABEL.

3.1 LVO LABEL Specification - Version ID = 1

Figure 3-1 shows the LVO LABEL specification that is identified by the RA character **1** in the Version ID sub-field. Figure 3-1 indicates the sub-field names and their octet positions within the LABEL. The definitions of the terms used in the figure are given below in Sections 3.1.1 to 3.1.6.

CAID (ADID 1 of 2)	Version ID	Class ID	Spare		DDID (ADID 2 of 2)	Length	LABEL sub-field
			1	2			
0 - 3	4	5	6	7	8 - 11	12 - 19	Octet number

Figure 3-1: CCSDS LABEL Specification - Version ID = 1

3.1.1 Version ID

The Version ID contains the RA character **1**. This Version ID, specified in the second sub-field (octet 4), identifies the format and meaning of the LABEL.

3.1.2 Authority and Description Identifier (ADID)

The ADID is an identifier which is used to uniquely identify the data description information that applies to the associated VALUE field. The ADID is partitioned into two four octet sub-fields, one in octets 0 to 3 and the other in octets 8 to 11. The Control Authority Identifier occupies the first sub-field of the ADID. The second sub-field of the ADID contains the Data Description Identifier.

3.1.2.1 Control Authority Identifier (CAID)

The Control Authority Identifier specified in the first sub-field (octets 0 to 3) of the LABEL identifies the organisation that has assigned the ADID to the data description information. This Control Authority

Office has the responsibility for maintaining the data description information and disseminating it in response to user requests according to the procedures recommended in Reference [2].

Only RA characters are allowed in the CAID sub-field.

3.1.2.2 Data Description Identifier (DDID)

The Data Description Identifier, specified in the sixth sub-field (octets 8 to 11) of the LABEL contains the identifier of the data description information held at the Control Authority Office, as identified by the CAID.

Only RA characters are allowed in the DDID sub-field.

3.1.3 Class ID

The Class ID, specified in the third sub-field (octet 5) of the LABEL, indicates the kind of data contained in the VALUE field following the LABEL. The Class ID must be selected from those approved by the CCSDS. A list of approved Class IDs is found in Section 4.

Only RA characters are allowed in the Class ID sub-field.

3.1.4 Spare 1

Octet 6 of the LABEL is a spare and shall be set to RA character **0** (zero).

3.1.5 Spare 2

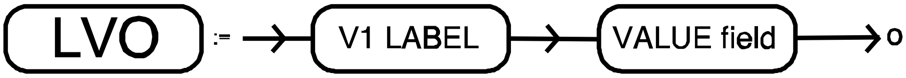
Octet 7 of the LABEL is a spare and shall be set to RA character **0** (zero).

3.1.6 Length

The length of the VALUE field, in units of octets, is specified in the seventh sub-field (octets 12 to 19) of the LABEL. The length is specified in decimal and represented in the numeric character subset of RA. The eight octets specified for this field makes possible a VALUE field length of up to 10^8-1 octets.

3.1.7 Structure Diagram

An LVO with a Version ID = 1 can be described by the structure diagram shown in Figure 3-2.



where: LVO = a LABEL-VALUE object
V1 LABEL = a LABEL field with Version ID = 1
VALUE field must conform to the corresponding LABEL specification

Figure 3-2: Structure Diagram for an LVO with Version ID = 1

3.2 LVO LABEL Specification - Version ID = 2

Figure 3-3 shows the CCSDS LABEL specification that is identified by the RA character **2** in the Version ID sub-field. Figure 3-3 indicates the sub-field names and their octet positions within the LABEL. The definitions of the terms used in the figure are given below in Sections 3.2.1 to 3.2.6.

CAID (ADID 1 of 2)	Version ID	Class ID	Spare		DDID (ADID 2 of 2)	Length	LABEL sub-field
			1	2			
0 - 3	4	5	6	7	8 - 11	12 - 19	Octet number

Figure 3-3: CCSDS LABEL Specification - Version ID = 2

<u>Note</u>: *A LABEL with Version ID = 2 differs from a LABEL with Version ID = 1 only in the value in the Version ID sub-field and that the Length sub-field specifies the length of the VALUE field in binary, not in RA characters.*

3.2.1 Version ID

The Version ID contains the RA character **2**. This Version ID, specified in the second sub-field (octet 4), identifies the format and meaning of the LABEL.

3.2.2 Authority and Description Identifier (ADID)

The ADID is an identifier which is used to uniquely identify the data description information that applies to the associated VALUE field. The ADID is partitioned into two four octet sub-fields, one in octets 0 to 3 and the other in octets 8 to 11. The Control Authority Identifier occupies the first sub-field of the ADID. The second sub-field of the ADID contains the Data Description Identifier.

3.2.2.1 Control Authority Identifier (CAID)

The Control Authority Identifier specified in the first sub-field (octets 0 to 3) of the LABEL identifies the organisation that has assigned the ADID to the data description information. This Control Authority

Office has the responsibility for maintaining the data description information and disseminating it in response to user requests according to the procedures recommended in Reference [2].

Only RA characters are allowed in the CAID sub-field.

3.2.2.2 Data Description Identifier (DDID)

The Data Description Identifier, specified in the sixth sub-field (octets 8 to 11) of the LABEL contains the identifier of the data description information held at the Control Authority Office, as identified by the CAID.

Only RA characters are allowed in the DDID sub-field.

3.2.3 Class ID

The Class ID, specified in the third sub-field (octet 5) of the LABEL, indicates the kind of data contained in the VALUE field following the LABEL. The Class ID must be selected from those approved by the CCSDS. A list of approved Class IDs is found in Section 4.

Only RA characters are allowed in the Class ID sub-field.

3.2.4 Spare 1

Octet 6 of the LABEL is a spare and shall be set to RA character **0** (zero).

3.2.5 Spare 2

Octet 7 of the LABEL is a spare and shall be set to RA character **0** (zero).

3.2.6 Length

The length, specified in the seventh sub-field (octets 12 to 19) of the LABEL, is used to specify the length of the VALUE field in octets and is represented in binary. The eight octet field specifies a 64 bit binary number that provides for a VALUE field length of up to 2^{64}-1 octets.

3.2.7 Structure Diagram

An LVO with a Version ID = 2 can be described by the structure diagram shown in Figure 3-4.

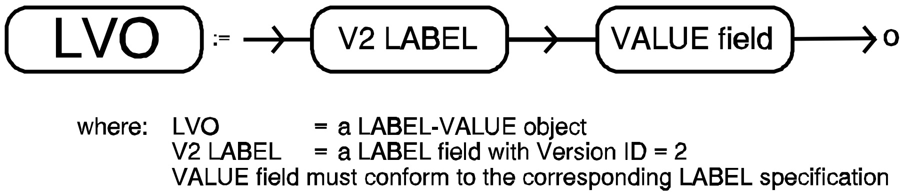

LVO
:=
V2 LABEL
VALUE field
o
where: LVO = a LABEL-VALUE object
V2 LABEL = a LABEL field with Version ID = 2
VALUE field must conform to the corresponding LABEL specification

Figure 3-4: Structure Diagram for an LVO with Version ID = 2

3.3 LVO LABEL Specification - Version ID = 3

Figure 3-5 shows the LVO LABEL Specification that is identified by the RA character **3** in the Version ID sub-field. Figure 3-5 indicates the sub-field names and their octet positions within the LABEL. The definitions of the terms used in the figure are given below in Sections 3.3.1 to 3.3.6.

CAID (ADID 1 of 2)	Version ID	Class ID	Delim. ID	Spare	DDID (ADID 2 of 2)	Delimitation Parameter	LABEL sub-field
0 - 3	4	5	6	7	8 - 11	12 - 19	Octet number

Figure 3-5: CCSDS LABEL Specification - Version ID = 3

> <u>Note</u>: *A LABEL with Version ID = 3 differs from a LABEL with Version ID = 1 or 2 only in how octets 6 and 12 to 19 are used. Octet 6 specifies how the VALUE field is delimited (as opposed to being "spare" for the previous Version IDs) , and octets 12 to 19 specify a parameter to be used for delimitation.*

3.3.1 Version ID

The Version ID contains the RA character **3**. This Version ID, specified in the second sub-field (octet 4), identifies the format and meaning of the LABEL.

3.3.2 Authority and Description Identifier (ADID)

The ADID is an identifier which is used to uniquely identify the data description information that applies to the associated VALUE field. The ADID is partitioned into two four octet sub-fields, one in octets 0 to 3 and the other in octets 8 to 11. The Control Authority Identifier occupies the first sub-field of the ADID. The second sub-field of the ADID contains the Data Description Identifier.

3.3.2.1 Control Authority Identifier (CAID)

The Control Authority Identifier specified in the first sub-field (octets 0 to 3) of the LABEL identifies the organisation that has assigned the ADID to the data description information. This Control Authority Office has the responsibility for maintaining the data description information and disseminating it in response to user requests according to the procedures recommended in Reference [2].

Only RA characters are allowed in the CAID sub-field.

3.3.2.2 Data Description Identifier (DDID)

The Data Description Identifier, specified in the sixth sub-field (octets 8 to 11) of the LABEL contains the identifier of the data description information held at the Control Authority Office, as identified by the CAID.

Only RA characters are allowed in the DDID sub-field.

3.3.3 Class ID

The Class ID, specified in the third sub-field (octet 5) of the LABEL, indicates the kind of data contained in the VALUE field following the LABEL. The Class ID must be selected from those approved by the CCSDS. A list of approved Class IDs is found in Section 4.

Only RA characters are allowed in the Class ID sub-field.

3.3.4 Delimitation ID

This single octet sub-field (octet 6) of the LABEL specifies the manner in which the LVO is delimited. Table 3-1 lists the defined Delimitation IDs and their corresponding delimitation techniques. These techniques use, as necessary, the value contained within the Delimitation Parameter sub-field to complete the delimitation technique description.

Delimitation ID	Delimitation Technique
A	Length (specified in ASCII)
B	Length (specified in binary)
S	Marker Pattern
E	Sequential End-of-File
C	Contiguous End-of-File
F	Shared End-of-File

Table 3-1: Delimitation IDs

<u>Note</u>: *End-of File (EOF) is a concept common to many storage systems and is widely used for delimitation of data sets. However its physical implementation is specific to individual media/operating systems. Any of the above delimitation techniques that make use of EOF treat it conceptually without defining its implementation. It is assumed that the creation and recognition of EOFs are operations which can be performed for the specific media/operating systems on which these LVOs are stored or transmitted.*

3.3.4.1 Length (Specified in ASCII) - Delimitation ID = A[4]

The length of the VALUE field, in units of octets, is specified in the Delimitation Parameter sub-field (octets 12 to 19) of the LABEL. The length is specified in decimal and represented in the numeric character subset of RA. The eight octets specified for the Delimitation Parameter sub-field makes possible a VALUE field length of up to 10^8-1 octets.

[4] *This delimitation technique replicates the functionality of that indicated by a Version ID = 1. Support for LVOs with Version ID = 1 are retained for compatibility with previous versions of this Recommendation.*

3.3.4.2 Length (Specified in Binary) - Delimitation ID = B[5]

The length of the VALUE field is specified in binary in the Delimitation Parameter sub-field (octets 12 to 19) of the LABEL. The eight octet sub-field specifies a 64 bit binary number that provides for a VALUE field length of up to 2^{64}-1 octets.

3.3.4.3 Marker Pattern - Delimitation ID = S

Delimitation by Marker Pattern employs a twenty octet marker pattern to indicate the end of the VALUE field. This delimitation technique is intended for use when the length of the VALUE field cannot be determined at the time of generation of the LABEL, or when transfer protocols or other processing may change the actual length of the LVO. Therefore an in-line 20 octet marker pattern is inserted when the VALUE field is complete. The first twelve octets of this marker pattern is a fixed pattern and must be '**CCSD$$MARKER**'; the remaining eight octets are user specified in the Delimitation Parameter sub-field of the LABEL. Schematically an LVO delimited by marker pattern can be described as shown in Figure 3-6. *(Note: the lower case letters in Figure 3-6 represent variable user-specified sub-fields)*

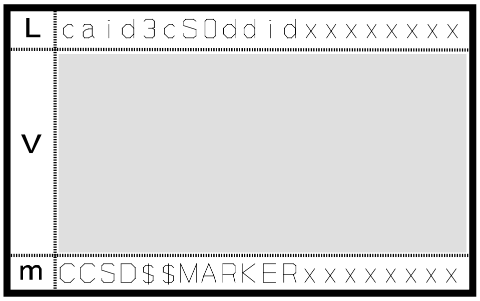


Figure 3-6: Schematic of a Marker Pattern Delimited LVO

As shown, the marker pattern has the following format:

CCSD$$MARKERxxxxxxxx

The eight octet **xxxxxxxx** part of the marker pattern is the user specified part and can be any characters from the ASCII character sub-set, decimal codes 32 to 126 (See Annex D).

An LVO employing this delimitation technique is terminated when the complete marker pattern (i.e., **CCSD$$MARKERxxxxxxxx**, where **xxxxxxxx** is the user specified portion) is encountered following the LABEL. The following should be noted:

i) If a marker pattern delimited LVO is a Compound LVO then all the LVOs that are nested within this LVO must themselves be terminated before any search for the marker pattern is performed;

ii) Even though nested LVOs may have the same marker pattern, each single instance of a marker pattern is used to terminate only one LVO. That is, a marker pattern cannot be shared by nested LVOs and thus the number of terminating marker patterns must be the same as the number of LVO LABELs which specify marker patterns.

[5] This delimitation technique replicates the functionality of that indicated by a Version ID = 2. Support for LVOs with Version ID = 2 are retained for compatibility with previous versions of this Recommendation.

3.3.4.4 Sequential End-of-File (EOF) - Delimitation ID = E

1. The VALUE field is delimited by counting the number of EOFs encountered following the LABEL until the count reaches a specified number.

2. The number of sequential EOFs is specified by the RA formatted decimal integer value contained in the Delimitation Parameter sub-field of the LABEL.

3. Any EOFs encountered in the VALUE field of LVOs at lower structure levels or used to delimit LVOs at lower structure levels shall be ignored.

Note: *Delimitation by Sequential EOF is primarily intended for use with sequential access media, such as tape. If delimitation by Sequential EOF is used on random access media (i.e., disks) in which the concept of continuing beyond an EOF makes no sense, then only the RA character string* **`00000001`** *for the Delimitation Parameter is meaningful.*

3.3.4.5 Contiguous End-of-File (EOF) - Delimitation ID = C

1. The VALUE field is delimited by a number of contiguous EOFs.

2. The number of contiguous EOFs is specified by the RA formatted decimal integer value contained in the Delimitation Parameter sub-field of the LABEL.

3. Any EOFs encountered in the VALUE field of LVOs at lower structure levels or used to delimit LVOs at lower structure levels shall be ignored.

Note: *Delimitation by Contiguous EOF is primarily intended for use with sequential access media, such as tape. If delimitation by Contiguous EOF is used on random access media (i.e., disks) in which the concept of continuing beyond an EOF makes no sense, then only the RA character string* **`00000001`** *for the Delimitation Parameter is meaningful.*

3.3.4.6 Shared End-of-File (EOF) - Delimitation ID = F

1. The VALUE field is delimited by the first EOF that is encountered. A single EOF is shared by all nested LVOs using this delimitation technique, i.e., they all terminate at the same EOF.

2. The value within the Delimitation Parameter sub-field of the LABEL must be set to the RA string **`00000001`**.

3. LVOs with Delimitation ID = E or C shall not be nested within an LVO with Delimitation ID = F.

Note: *Delimitation by Shared EOF is primarily intended for use with random access media such as disks, in which the concept of continuing beyond an EOF makes no sense.*

3.3.5 Spare

Octet 7 of the LABEL is a spare and shall be set to RA character **0** (zero).

3.3.6 Delimitation Parameter

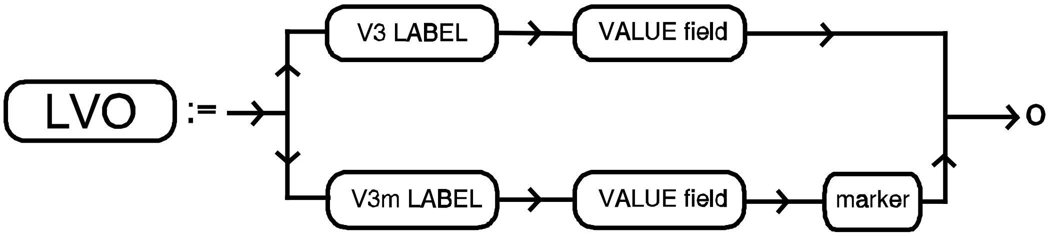
This sub-field (octets 12 to 19) of the LABEL provides the parameter required by the delimitation technique. The detailed use of the Delimitation Parameter associated with each technique is explained in Section 3.3.4. Table 3-2 summarises the definition of the Delimitation Parameter and its representation for each delimitation technique.

Delimitation ID	Delimitation Parameter Meaning	Delimitation Parameter Representation
A	Length	8 octet RA decimal integer
B	Length	8 octet 64 bit binary integer
S	Marker completion pattern	8 characters (ASCII codes 32-126, see Annex D)
E	Number of sequential EOFs	8 octet RA decimal integer
C	Number of contiguous EOFs	8 octet RA decimal integer
F	No delimitation parameter required for this delimitation technique	Fixed 8 octet RA string **`00000001`**

Table 3-2: Delimitation Parameter Definitions

3.3.7 Structure Diagram

An LVO with a Version ID = 3 can be described by the structure diagram shown in Figure 3-7. As shown here, the structure of the LVO can vary depending upon how the VALUE field is delimited *(i.e., if delimited by marker pattern, then the marker pattern is required).*



where: LVO = a LABEL-VALUE object
V3 LABEL = a LABEL field with Version ID = 3 and non-marker pattern delimitation
V3m LABEL = a LABEL field with Version ID = 3 and marker pattern delimitation
marker = the marker pattern corresponding to the specification in the V3m LABEL
VALUE fields must conform to their corresponding V3/V3m LABEL

Figure 3-7: Structure Diagram for an LVO with Version ID = 3

4 CLASS ID SPECIFICATIONS

This section defines the meaning of the Class IDs which may be used in all versions of LVO LABELs. The Class ID provides an indication of the type of the data contained in the LVO VALUE field. The interpretation of that data is made using the ADID appearing in the associated LABEL field.

The Class IDs can be split into three basic categories, as shown in Figure 4-1. The Structure Classes handle the packaging of LVOs, the Service Classes provide CCSDS service mechanisms and the Data Classes contain the actual user data.

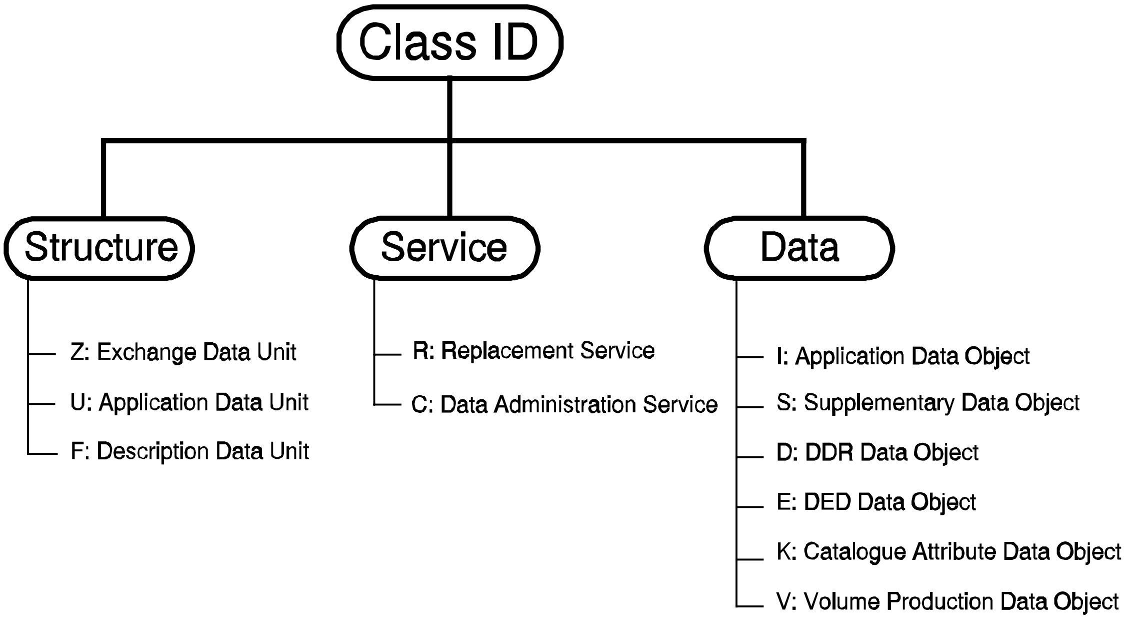


Figure 4-1: SFDU Class ID Breakdown

4.1 Structure Classes

The VALUE field of an LVO with a Structure Class ID contains further LVOs. Three Structure Class IDs, identified by the Class IDs Z, U and F are defined in this Recommendation; they differ in the types of data permitted within their VALUE fields.

4.1.1 Exchange Data Unit - Class ID = Z

The VALUE field of an LVO with Class ID = Z contains further LVOs, thus forming an Exchange Data Unit (EDU). This EDU structure is used to group together any data which are related to each other in some manner as decided by the user.

4.1.2 Application Data Unit - Class ID = U

The VALUE field of an LVO with Class ID = U contains further LVOs, thus forming an Application Data Unit (ADU). These LVOs include objects of application data and any related ancillary or identification data objects required to process them. This unit shall not include data description data objects.

4.1.3 Description Data Unit - Class ID = F

The VALUE field of an LVO with Class ID = F contains further LVOs, thus forming a Description Data Unit (DDU). Included among these LVOs must be LVOs which contain the following:

i) The unique DDU identifier *(i.e., the ADID)* with which described LVOs can point to the DDU;

ii) All data description objects *(e.g., DDRs, DEDs, supplementary description data)* necessary to interpret the described LVOs.

Only LVOs which contain data description information are allowed in an LVO with Class ID = F.

4.2 Service Classes

The VALUE field of an LVO with a Service Class ID contains information to support certain CCSDS defined service mechanisms, as opposed to carrying application data or metadata. Two Service Classes, identified by Class IDs R and C, are defined in this Recommendation. LVOs having Service Class ID cannot contain other LVOs in their VALUE fields.

4.2.1 Replacement Service Object - Class ID = R

The VALUE field of an LVO with Class ID = R contains parameters for use by replacement services. These services include the following:

i) Logically replace the LVO with Class ID = R with the sequence of LVOs formed from external data objects;

ii) Associate CCSDS LVO LABELs with external data objects, thereby forming logical LVOs from these external data objects;

iii) Logically concatenate external data objects which may or may not have SFDU LABELs attached.

4.2.2 Data Administration Service Object - Class ID = C

The VALUE field of an LVO with Class ID = C contains parameters for use in data administration services. These services include data object naming and referencing of registered information(6).

(6) ***Registered information*** *is any data description information that is either registered at a Control Authority or registered locally through the use of a DDU which is packaged with the application data requiring it for interpretation.*

4.3 Data Classes

The VALUE field of an LVO with a Data Class ID contains a specific type of data or metadata. Six Data Classes, identified by the Class IDs I, S, D, E, K and V are defined by this Recommendation.

Data classes cannot be used for packaging LVOs; that is an LVO with a Data Class ID cannot contain other LVOs within its VALUE field.

4.3.1 Application Data Object - Class ID = I

The VALUE field of an LVO with Class ID = I contains the data which the producer of the product considers to be of principal interest.

4.3.2 Supplementary Data Object - Class ID = S

The VALUE field of an LVO with Class ID = S contains data which the producer of the product considers to be supplementary data which can assist in the understanding, evaluation or processing of the data objects with which it is associated. This association is not defined by this Recommendation.

4.3.3 Data Description Record Object - Class ID = D

The VALUE field of an LVO with Class ID = D contains a description of the syntax of the VALUE field of an LVO. This syntactic description is called a Data Description Record (DDR).

4.3.4 Data Entity Dictionary Object - Class ID = E

The VALUE field of an LVO with Class ID = E contains a description of the semantics of the VALUE field of an LVO. This semantic description is called a Data Entity Dictionary (DED).

4.3.5 Catalogue Attribute Object - Class ID = K

The VALUE field of an LVO with Class ID = K contains catalogue attribute information for the data objects with which it is associated. This association is not defined by this Recommendation. Typical information includes data catalogue information such as an instrument identifier and the time and location of a measurement.

4.3.6 Volume Preparation Data Object - Class ID = V

The VALUE field of an LVO with Class ID = V contains information concerning the production of the data objects with which it is associated. This association is not defined by this Recommendation. This data typically includes the machine type on which it was created, the processing history, the time and the system version at creation.

4.4 Overview of Structure Class IDs

LVOs with Structure Class IDs vary in which LVOs they permit within their VALUE field. Table 4-1 indicates the Class IDs of LVOs which are allowed within each of the three LVOs with Structure Class IDs.

Class ID ↓ may contain Class ID →	Z	U	F	R	C	I	S	D	E	K	V
Z (EDU)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
U (ADU)	▨	✓	▨	✓	✓	✓	✓	▨	▨	✓	▨
F (DDU)	▨	▨	▨	✓	✓	▨	✓	✓	✓	✓	▨

✓ = permitted ▨ = not permitted

Table 4-1: LVOs Permitted within each Compound LVO

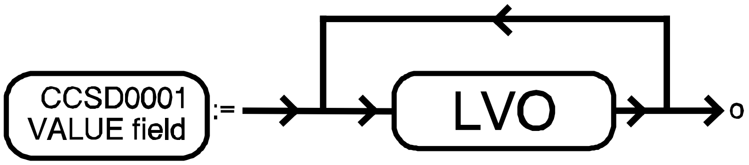
5 CCSDS ADID SPECIFICATIONS

This section defines the subset of Authority and Description Identifiers (ADIDs) that relate to structures. Each has been defined by the CCSDS and assigned a globally unique ADID which begins with the CCSDS Control Authority Identifier **CCSD**.

Where possible for each ADID, a description in English is supported by a structure diagram. There is an Abstract Syntax Notation One (ASN.1, see Reference [4]) specification in Annex C, for each ADID defined.

5.1 Specification for ADID = CCSD0001

The CCSD0001 specification provides a means for building VALUE fields composed of multiple LVOs. The structure of these LVOs is shown diagrammatically in Figure 5-1.



Where: LVO = a LABEL-VALUE-Object as defined in Section 3

Figure 5-1: Structure Diagram of the VALUE Field of an LVO with ADID=CCSD0001

As shown in Figure 5-1, the VALUE field is composed of a sequence of one or more LVOs. Note that the VALUE field cannot be empty.

5.2 Specification for ADID = CCSD0003

The CCSD0003 specification describes a Simple LVO VALUE field for specifying parameters for use by the CCSDS replacement services provided by an LVO with Class ID = R (the Referencing LVO). The Parameter Value Language, PVL (See Reference [5]), is used to specify the necessary parameters.

Three parameters are specified by this ADID. They perform the following three tasks:

1. Define the name of a referencing environment that is to be used to access external data object(s) from the Referencing LVO, for example, the filenaming convention and the type of data storage medium;

2. Provide a means to optionally add an LVO LABEL to external data object(s);

3. Provide a pointer to the start of external data object(s) within the specified referencing environment.

The three PVL statements that correspond to these three tasks are described below. The permitted ordering of these three statements in any one VALUE field is defined by Figure 5-2.

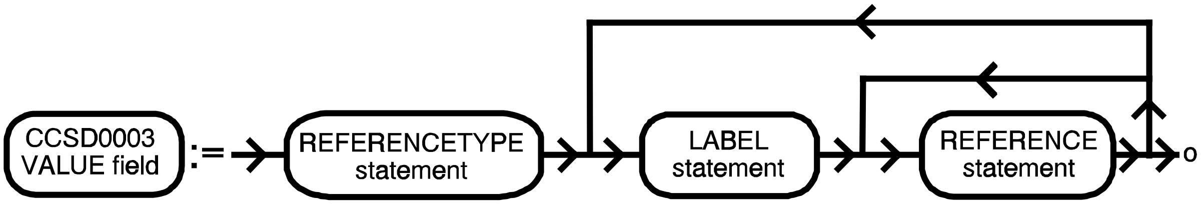


Figure 5-2: Structure Diagram of the PVL Statements within an LVO with ADID = CCSD0003

In the statement descriptions below any parameter names or values shown in upper case are keywords, whilst lower case is used to indicate user specified values. All keywords in a CCSD0003 VALUE field must be expressed in upper case.

a. **REFERENCETYPE=refenv;**
The **refenv** value names the referencing environment to be used. If **refenv** begins with the **$** character then it names a provisional CCSDS defined referencing environment~~ (Annex F specifies provisional CCSDS defined referencing environments)~~, otherwise it names an approved CCSDS defined referencing environment.

Cor. 1

b. **LABEL=string;**
The **string** value allows an LVO LABEL field to be logically added to the beginning of the external data object(s). If **string** is the keyword **ATTACHED** then it means the external data object(s) shall have an LVO LABEL at the beginning and do not require a further LABEL. Otherwise, **string** shall conform to an LVO LABEL specification that is composed of printable ASCII characters (decimal codes 32 to 126).

c. **REFERENCE=(name_1, name_2, . . . name_n);**
Each **name_x** specifies the beginning of one or more external data objects within the defined referencing environment. The parentheses are optional if there is only one **name_x**.

The following should be noted concerning Figure 5-2:

i. There shall be one, and only one, **REFERENCETYPE** statement. This shall be the first statement;

ii. There shall be at least one **REFERENCE** statement;

iii. There shall be one **LABEL** statement before the first **REFERENCE** statement;

iv. There shall be no more than one **LABEL** statement between any two consecutive **REFERENCE** statements;

v. A **LABEL** statement can apply to many **REFERENCE** statements. The most recent **LABEL** statement will be used;

vi. The last statement in a VALUE field shall be a **REFERENCE** statement.

The following rules apply:

A. External data objects are referenced in the order of the **REFERENCE** statements. If a **REFERENCE** statement specifies more than one external data object, then these external data objects are referenced in the order that they appear in the **REFERENCE** statement.

B. If a **REFERENCE** statement refers to more than one external data object, then the given LABEL applies to each of these external data objects.

C. Each external data object terminates when the VALUE field delimited by the first LABEL (whether attached or supplied by a **LABEL** statement) has been completed.

D. Each external data object shall comply with the structure rules applying at the point where the external data object was referenced *(e.g., if the Referencing LVO is within a DDU, then only LVOs with Class IDs permitted within DDUs can be referenced)*.

E. The delimitation of the Referencing LVO, and the LVOs within which it is contained, is not affected by the external data objects *(e.g., the octet count of a length delimited Referencing LVO does not take into account the octets of any external data object(s))*.

5.3 Specification for ADID = CCSD0004

The CCSD0004 specification describes a Simple LVO VALUE field for specifying parameters for use by data administration services.

A CCSD0004 VALUE field is expressed in the Parameter Value Language (PVL) as specified in Reference [5]. It shall consist of one or more "parameter=value" statements, the left hand side of each of which is one of a list of CCSDS defined parameter names. The right hand side is a user specified value, the format of which is defined below.

The permitted "parameter=value" statements are described below. In these descriptions, parameter names in upper case are CCSDS defined keywords, while lower case is used for user specified values. All keywords in a CCSD0004 VALUE field must be expressed in upper case.

1. **ADIDNAME=adid;**
   The **adid** value is an eight octet RA character string which supplies the Authority Description Identifier (ADID) by which the contents of this DDU will be identified. This statement is mandatory and may appear only once in a given CCSD0004 VALUE field. This statement must be the first PVL statement encountered in the VALUE field.

2. **DEDID=adid;**
   The **adid** value is an eight octet RA character string which supplies an Authority and Description Identifier (ADID) of registered data description information which contains one or more Data Entity Dictionary data objects (LVOs with Class ID = E) to be logically included in the DDU.

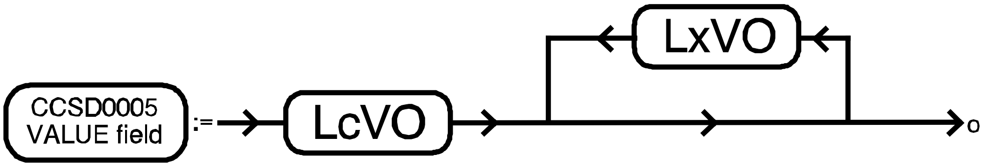
3. **`DDRID=adid;`**
   The **`adid`** value is an eight octet RA character string which supplies an Authority and Description Identifier (ADID) of registered data description information which contains one or more Data Description Record data objects (LVOs with Class ID = D) to be logically included in the DDU.

4. **`SUPID=adid;`**
   The **`adid`** value is an eight octet RA character string which supplies an Authority and Description Identifier (ADID) of registered data description information which contains one or more Supplementary Information data objects (LVOs with Class ID = S) to be logically included in the DDU.

The statements defined in items 2, 3 and 4 above are optional, and may each appear more than once in a given VALUE field and in any order.

5.4 Specification for ADID = CCSD0005

The CCSD0005 specification provides a means of specifying a VALUE field composed of multiple LVOs that meet the requirements of a Description Data Unit (DDU, an LVO with Class ID = F). That is, it provides the data description objects and includes the ADID assigned to those data description objects. The structure required for this specification is shown diagrammatically in Figure 5-3.



Where: LcVO = a LABEL-VALUE-Object with Class ID = C
LxVO = a LABEL-VALUE-Object with Class ID = D, E, K, S or R

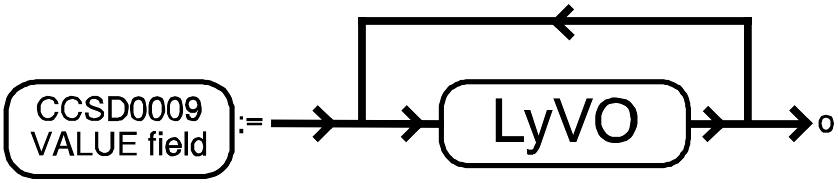
Figure 5-3: Structure Diagram of the VALUE Field of an LVO with ADID = CCSD0005

As shown in Figure 5-3, the VALUE field is composed of a sequence of LVOs. An LVO with Class ID = C provides a means of specifying the ADID assigned to the data description data in the remainder of the CCSD0005 VALUE field. It also provides a means to logically include, under this ADID, data description data previously registered under other ADIDs. For any single CCSD0005 VALUE field there shall be only one LVO with Class ID = C and this LVO shall be the first to appear.

As can also be seen from Figure 5-3 any number of LVOs with Class IDs = D, E, K, S or R may appear after the LVO with Class ID = C. The semantic definition of a given name will first be searched for in the LVOs with Class ID = E (DEDs) found inside the DDU, and the search shall be in the order that the LVOs with Class ID = E appear. Then the DEDs logically included by PVL **`DEDID`** statements within the LVO with Class ID = C will be searched, again in the order that the **`DEDID`** statements appear. The first definition encountered is the ruling one.

5.5 Specification for ADID = CCSD0009

The CCSD0009 specification provides a means for building VALUE fields composed of multiple LVOs which meet the requirements for an Application Data Unit (ADU, an LVO with Class ID = U). The structure required for this specification is shown diagrammatically in Figure 5-4.



Where: LyVO = a LABEL-VALUE-Object with Class ID = U, I, S, K, C or R

Figure 5-4: Structure Diagram of the VALUE Field of an LVO with ADID = CCSD0009

As shown in Figure 5-4, the VALUE field is composed of a sequence of one or more LVOs. Note that the VALUE field cannot be empty. Any of these LVOs may themselves also be an ADU. In line with the fact that the ADU cannot carry data description information, only LVOs with Class IDs = U, I, S, K, C or R are allowed within a CCSD0009 VALUE field.

6 COMBINATION OF ADIDS AND CLASS IDS

All LVO LABELs, by definition, include an ADID and Class ID combination. The purpose of this section is to categorise the possible combinations.

There are two types of permitted ADIDs:

1. Those defined by the CCSDS, appearing in CCSDS Recommendations, and beginning with the four RA characters **CCSD**. These are referred to as CCSDS ADIDs[7] (See Section 5);

2. Those defined by SFDU users, typically through data description registration with a Control Authority Office (See Reference [2]). These are referred to as User ADIDs.

The only permitted Class IDs are those defined by the CCSDS and appearing in CCSDS Recommendations (See Section 4).

There are two categories of permitted ADID and Class ID combinations, these are defined as follows:

1. **CCSDS defined usage**: These combinations occur only with CCSDS ADIDs and correspond to cases where a particular CCSDS ADID specification has been specifically designed by the CCSDS to support a given Class ID. Where they exist such combinations are the preferred form;

2. **User defined usage**: These combinations occur only with User ADIDs, for which the User ADID must support the CCSDS defined usage of the Class ID.

Any other combinations of ADIDs and Class IDs are not permitted.

The following table categorises the ADID and Class ID combinations, and includes CCSDS ADIDs specified in other CCSDS Recommendations. CCSDS ADIDs, which appear to be missing, correspond to specifications that are under development. New and revised documents may introduce new CCSDS ADIDs. Users of this Recommendation are encouraged to investigate the possibility of additional CCSDS ADIDs. The CCSDS Secretariat maintains a register of currently valid CCSDS Recommendations.

Cor. 1

[7] *When using a Structure or Service Class ID, a CCSDS ADID must be used whenever possible, in order to maximise universal access to the corresponding LVOs.*

ADIDs	**Class IDs**										
	Z	**F**	**U**	**R**	**C**	**I**	**S**	**D**	**E**	**K**	**V**
CCSD0001	✓✓	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
CCSD0002[†]	▨	▨	▨	▨	▨	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓
CCSD0003	▨	▨	▨	✓✓	▨	▨	▨	▨	▨	▨	▨
CCSD0004	▨	▨	▨	▨	✓✓	▨	▨	▨	▨	▨	▨
CCSD0005	▨	✓✓	▨	▨	▨	▨	▨	▨	▨	▨	▨
CCSD0006 [‡]	▨	▨	▨	▨	▨	✓✓	✓✓	✓✓	✓✓	✓✓	✓✓
CCSD0007 *	▨	▨	▨	▨	▨	▨	▨	▨	▨	✓✓	▨
CCSD0009	▨	▨	✓✓	▨	▨	▨	▨	▨	▨	▨	▨
User defined ADIDs	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Cor. 1

[†] ASCII Encoded English, see Ref. [6]

[‡] PVL, see Ref. [5]

* Control Authority Data Structures, see Ref. [7]

Cor. 1

✓✓ = Permitted

✓ = Permitted if the ADID identified data description does not contradict the Class ID identified functionality

▨ = Not permitted

Table 6-1: ADID and Class ID Combination Categorisations

To summarise, Table 6-2 specifies the CCSDS defined combinations of ADIDs and Class IDs that relate to this Recommendation.

Class ID	**ADID**	**Permitted Instance of:**
Z	CCSD0001	Exchange Data Unit (EDU)
F	CCSD0005	Description Data Unit (DDU)
U	CCSD0009	Application Data Unit (ADU)
R	CCSD0003	-
C	CCSD0004	-

Table 6-2: CCSDS Defined Combinations of Class IDs and ADIDs

In order to be a legal SFDU data product, the first label of any SFDU data products must have a Class ID = Z and an ADID = CCSD0001.

ANNEX A

STANDARD FORMATTED DATA UNIT ACRONYMS

Purpose:

This annex defines the acronyms which are used throughout this Recommendation to describe the concepts and elements of the Standard Formatted Data Unit.

Annex A: Acronyms

ADID	Authority and Description Identifier
ADU	Application Data Unit
ANSI	American National Standards Institution
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
CA	Control Authority
CAID	Control Authority Identifier
CCSDS	Consultative Committee for Space Data Systems
DDID	Data Description Identifier
DDR	Data Description Record
DDU	Description Data Unit
DED	Data Entity Dictionary
EDU	Exchange Data Unit
EOF	End-of-file
ID	Identifier
LSB	Least Significant Bit
ISO	International Organisation for Standardisation
LVO	LABEL-VALUE Object
MSB	Most Significant Bit
MSO	Most Significant Octet
PVL	Parameter Value Language
RA	Restricted ASCII
SFDU	Standard Formatted Data Unit

ANNEX B

STANDARD FORMATTED DATA UNIT GLOSSARY

Purpose:

This annex defines key terms which are used throughout this Recommendation to describe the concepts and elements of the Standard Formatted Data Unit.

Annex B: Glossary of Terms

Control Authority: An organisation under the auspices of CCSDS which supports the transfer and usage of SFDUs by providing operational services of registration, archiving and dissemination of data description data. It comprises:

- The CCSDS Secretariat supported by a CA Agent

- Member Agency Control Authority Offices (MACAOs)

Cross support: When one agency uses part of another agency's data system resources to complement its own system.

Data Description Record (DDR) (also referred to as Data Description Record Object): A syntactic description for data entities.

Data Element: The smallest named item or items of data for a given application.

Data Entity: A named collection of data elements.

Data Entity Dictionary (DED): A collection of semantic definitions for data entities.

Data Object: A collection of data elements that are packaged for or by a specific application.

Data Product: A collection of one or more data objects.

Delimitation: The method of specifying the end of a block of data.

Instance: A specific occurrence of values of a data entity.

Metadata: Data about other data.

Open system data interchange: The process of transferring data from one open system to another. An open system is one which uses publicly available formats and protocols, so that anyone can communicate with the open system by following the open system standards. It should be noted that open system does not imply an uncontrolled or unrestricted access to the data.

Reference environment: An environment in which the value of a REFERENCE statement is understood to give one or more locations at which external data objects begin.

Semantic information: Information associated with data that defines the meaning of the data.

Syntactic information: Information associated with data that defines the format of the data.

ANNEX C

STANDARD FORMATTED DATA UNIT ASN.1 DEFINITIONS

Purpose:

This annex gives full ASN.1 definitions of all the structures presented in this Recommendation. This includes the definitions for:

- The LVO structure
- The VALUE field of an LVO with ADID=CCSD0001
- The VALUE field of an LVO with ADID=CCSD0003
- The VALUE field of an LVO with ADID=CCSD0004
- The VALUE field of an LVO with ADID=CCSD0005
- The VALUE field of an LVO with ADID=CCSD0009

Annex C: ASN.1 Definitions

C.1 ASN.1 Definition of the LVO Structure

```
--*************************************************************************
--                 ASN.1 Definition of the LVO Structure
--*************************************************************************

LvoModule DEFINITIONS  ::=

BEGIN

EXPORTS Lvo;

Lvo ::= CHOICE{
               LengthLvo,
               MarkerLvo
              }

LengthLvo ::= CHOICE{
                     CompLengthLvo,
                     SimpleLengthLvo
                    }


CompLengthLvo ::= SEQUENCE{
                           CHOICE{
                                  CompVer1Label,
                                  CompVer2Label,
                                  CompVer3ALabel,
                                  CompVer3BLabel
                                 },
                           CompLvoValueField
                          }

CompVer1Label ::= SEQUENCE{
                           Caid,
                           Version1Id,
                           CompLvoClassId,
                           Spare,
                           Spare,
                           Ddid,
                           Ver13ALength
                          }


CompVer2Label ::= SEQUENCE{
                           Caid,
                           Version2Id,
                           CompLvoClassId,
                           Spare,
                           Spare,
                           Ddid,
                           Ver23BLength
                          }

CompVer3ALabel ::= SEQUENCE{
                            Caid,
                            Version3Id,
                            CompLvoClassId,
                            AscLenDelimId,
                            Spare,
                            Ddid,
                            Ver13ALength
                           }

CompVer3BLabel ::= SEQUENCE{
                            Caid,
                            Version3Id,
                            CompLvoClassId,
                            BinLenDelimId,
                            Spare,
                            Ddid,
                            Ver23BLength
                           }

CompLvoValueField ::= SET{
                          Lvo,
                          SET OF Lvo
                         }

SimpleLengthLvo ::= SEQUENCE{
                             CHOICE{
                                    SimpleVer1Label,
                                    SimpleVer2Label,
                                    SimpleVer3ALabel,
                                    SimpleVer3BLabel
                                   },
                             SimpleLvoValueField
                            }

SimpleVer1Label ::= SEQUENCE{
                             Caid,
                             Version1Id,
                             SimpleLvoClassId,
                             Spare,
                             Spare,
                             Ddid,
                             Ver13ALength
                            }


SimpleVer2Label ::= SEQUENCE{
                             Caid,
                             Version2Id,
                             SimpleLvoClassId,
                             Spare,
                             Spare,
                             Ddid,
                             Ver23BLength
                            }

SimpleVer3ALabel ::= SEQUENCE{
                              Caid,
                              Version3Id,
                              SimpleLvoClassId,
                              AscLenDelimId,
                              Spare,
                              Ddid,
                              Ver13ALength
                             }

SimpleVer3BLabel ::= SEQUENCE{
                              Caid,
                              Version3Id,
                              SimpleLvoClassId,
                              BinLenDelimId,
                              Spare,
                              Ddid,
                              Ver23BLength
                             }

SimpleLvoValueField ::= SET OF Octet

MarkerLvo ::= CHOICE{
                     SeqEofLvo,
                     ConEofLvo,
                     ShaEofLvo,
                     MarkPatLvo
                    }

SeqEofLvo ::= CHOICE{
                     SEQUENCE{
                              Caid,
                              Version3Id,
                              CompLvoClassId,
                              SeqEofDelimId,
                              Spare,
                              Ddid,
                              eofCount EofDelimParam,
                              SeqEofCompLvoValueField
                             },
                     SEQUENCE{
                              Caid,
                              Version3Id,
                              SimpleLvoClassId,
                              SeqEofDelimId,
                              Spare,
                              Ddid,
                              eofCount EofDelimParam,
                              SeqEofSimpleLvoValueField
                             }
                    }

SeqEofCompLvoValueField ::= SET SIZE (eofCount) OF SEQUENCE{
                                                            SET OF Lvo,
                                                            Eof
                                                           }

SeqEofSimpleLvoValueField ::= SET SIZE (eofCount) OF SEQUENCE{
                                                              SET OF Octet,
                                                              Eof
                                                             }

ConEofLvo ::= CHOICE{
                     SEQUENCE{
                              Caid,
                              Version3Id,
                              CompLvoClassId,
                              ConEofDelimId,
                              Spare,
                              Ddid,
                              eofCount EofDelimParam,
                              ConEofCompLvoValueField
                             },
                     SEQUENCE{
                              Caid,
                              Version3Id,
                              SimpleLvoClassId,
                              ConEofDelimId,
                              Spare,
                              Ddid,
                              eofCount EofDelimParam,
                              ConEofSimpleLvoValueField
                             }
                    }

ConEofCompLvoValueField ::= SEQUENCE{
                                     SET OF Lvo,
                                     SET SIZE (eofCount) OF Eof
                                    }

ConEofSimpleLvoValueField ::= SEQUENCE{
                                       SET OF Octet,
                                       SET SIZE (eofCount) OF Eof
                                      }

          -- any EOFs which are encountered in the above two types which
          -- are not the correct contiguous number to delimit the VALUE
          -- field are ignored, this fact is not represented here in the ASN.1

ShaEofLvo ::= CHOICE{
                     SEQUENCE{
                              Caid,
                              Version3Id,
                              CompLvoClassId,
                              ShaEofDelimId,
                              Spare,
                              Ddid,
                              eofCount EofDelimParam,
                              ShaEofCompLvoValueField
                             },
                     SEQUENCE{
                              Caid,
                              Version3Id,
                              SimpleLvoClassId,
                              ShaEofDelimId,
                              Spare,
                              Ddid,
                              eofCount EofDelimParam,
                              ShaEofSimpleLvoValueField
                             }
                    }

ShaEofCompLvoValueField ::= SEQUENCE{
                                     SET OF Lvo,
                                     Eof
                                    }

          -- any of the LVOs that are contained in the LVO delimited by a
          -- shared EOF cannot have a delimitation technique of sequential
          -- or configuous EOFs.  This fact is not represented here in the
          -- ASN.1

ShaEofSimpleLvoValueField ::= SEQUENCE{
                                       SET OF Octet,
                                       Eof
                                      }

MarkPatLvo ::= CHOICE{
                     SEQUENCE{
                              Caid,
                              Version3Id,
                              CompLvoClassId,
                              MarkPatDelimId,
                              Spare,
                              Ddid,
                              Marker,                   -- value (a)
                              MarkPatCompLvoValueField
                             },
                     SEQUENCE{
                              Caid,
                              Version3Id,
                              SimpleLvoClassId,
                              MarkPatDelimId,
                              Spare,
                              Ddid,
                              Marker,                   -- value (b)
                              MarkPatSimpleLvoValueField
                             }
                    }

MarkPatCompLvoValueField ::= SEQUENCE{
                                     SET OF Lvo,
                                     Marker             -- must be the same
                                                        -- as value (a) above
                                    }

MarkPatSimpleLvoValueField ::= SEQUENCE{
                                       SET OF Octets,
                                       Marker           -- must be the same
                                                        -- as value (b) above
                                      }

Caid ::= SET SIZE (4) OF Rachar

Ddid ::= SET SIZE (4) OF Rachar

Version1Id ::= IA5String("1")

Version2Id ::= IA5String("2")

Version3Id ::= IA5String("3")

Spare ::= IA5String("0")

AscLenDelimId ::= IA5String("A")

BinLenDelimId ::= IA5String("B")

SeqEofDelimId ::= IA5String("E")

ConEofDelimId ::= IA5String("C")

ShaEofDelimId ::= IA5String("F")

MarkPatDelimId ::= IA5String("S")

CompLvoClassId ::= Rachar(FROM("Z"|"F"|"U"))

SimpleLvoClassId ::= Rachar(FROM("I"|"S"|"K"|"V"|"D"|"E"|"R"|"C"))

Ver13ALength ::= IA5String (SIZE(8))
                           (FROM("0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"))

Ver23BLength ::= BIT STRING (SIZE(64))

EofDelimParam ::= IA5String
                  (SIZE(8))
                  (FROM("0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"))
```

```
                             -- the decimal value of this ASCII decimal
                             -- character string is used as eofCount value

Marker ::= IA5String
           (SIZE(8))
           (FROM(INCLUDES Rachar |" "|"!"|""""|"#"|"$"|"%"|"&"|"′"|"("|")"|
                                 "*"|"+"|","|"-"|"."|"/"|":"|";"|"<"|"="|">"|
                                 "?"|"@"|"["|"\"|"]"|"^"|"_"|"′"|"a"|"b"|"c"|
                                 "d"|"e"|"f"|"g"|"h"|"i"|"j"|"k"|"l"|"m"|"n"|
                                 "o"|"p"|"q"|"r"|"s"|"t"|"u"|"v"|"w"|"x"|"y"|
                                  "z"|"{"|"|"|"}"|"~"))

Rachar ::= IA5String(FROM("A"|"B"|"C"|"D"|"E"|"F"|"G"|"H"|"I"|"J"|"K"|"L"|"M"|
                          "N"|"O"|"P"|"Q"|"R"|"S"|"T"|"U"|"V"|"W"|"X"|"Y"|"Z"|
                           "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"))

Octet ::= BIT STRING (SIZE(8))

Eof ::= EXTERNAL            -- local media/operating system End-Of-File flag

END
```

C.2 ASN.1 Definition of the DDR for ADID = CCSD0001

```
--*************************************************************************
--               ASN.1 Definition of the DDR for ADID = CCSD0001
--*************************************************************************

CCSD0001Module DEFINITIONS ::=

BEGIN

IMPORTS Lvo FROM LvoModule;

CCSD0001ValueField  ::=  SEQUENCE {
                                   CCSD0001Lvo,
                                   SET OF CCSD0001Lvo
                                  }

CCSD0001Lvo ::= Lvo      -- LVOs with any combination of Class ID and ADID
                         -- are permitted within the VALUE field of an LVO
                         -- with ADID = CCSD0001

END
```

C.3 ASN.1 Definition of the DDR for ADID = CCSD0003

```
--*************************************************************************
--             ASN.1 Definition of the DDR for ADID = CCSD0003
--*************************************************************************


CCSD0003Module DEFINITIONS ::=
BEGIN

CCSD0003ValueField  ::=  SEQUENCE {
                                   ReferenceTypeExpr,
                                   LabelReferenceCombination,
                                   SET OF LabelReferenceCombination
                                  }

LabelReferenceCombination  ::=  SEQUENCE {
                                          LabelExpr,
                                          ReferenceExpr,
                                          SET OF ReferenceExpr
                                         }

ReferenceTypeExpr ::= EXTERNAL

LabelExpr ::= EXTERNAL

ReferenceExpr ::= EXTERNAL


-- The remaining ASN.1 types to be defined are listed below.  Each of these
-- corresponds to a PVL expression.  The PVL syntax is defined in the PVL
-- Recommendation, Reference [5], and the function of each expression is
-- described in Section 5.2.

--  ReferenceTypeExpr -> REFERENCETYPE=refenv;
--  LabelExpr         -> LABEL=string;
--                       LABEL=ATTACHED;
--  ReferenceExpr     -> REFERENCE=name;
--                       REFERENCE=(name_1, name_2 . . . name_n);

END
```

C.4 ASN.1 Definition of the DDR for ADID = CCSD0004

```
--*************************************************************************
--             ASN.1 Definition of the DDR for ADID = CCSD0004
--*************************************************************************


CCSD0004Module DEFINITIONS ::=
BEGIN

CCSD0004ValueField ::= SEQUENCE{
                                AdidNameExpr,
                                SET OF CHOICE{
                                              DedIdExpr,
                                              DdrIdExpr,
                                              SupIdExpr
                                             }
                               }

AdidNameExpr ::= EXTERNAL

DedIdExpr ::= EXTERNAL

DdrIdExpr ::= EXTERNAL

SupIdExpr ::= EXTERNAL


-- The remaining ASN.1 types to be defined are listed below.  Each of these
-- corresponds to a PVL expression.  The PVL syntax is defined in the PVL
-- Recommendation, Reference [5], and the function of each expression is
-- described in Section 5.3.

--                 AdidNameExpr  ->  ADIDNAME = adid;
--                 DedIdExpr     ->  DEDID = adid;
--                 DdrIdExpr     ->  DDRID = adid;
--                 SupIdExpr     ->  SUPID = adid;

END
```

C.5 ASN.1 Definition of the DDR for ADID = CCSD0005

```
--**************************************************************************
--                ASN.1 Definition of the DDR for ADID = CCSD0005
--**************************************************************************


CCSD0005Module DEFINITIONS ::=
BEGIN

IMPORTS Lvo FROM LvoModule;

CCSD0005ValueField ::= SEQUENCE{
                                ClassCLvo,
                                SET OF CCSD0005Lvo
                               }

ClassCLvo ::= Lvo    -- an LVO with Class ID = C

CCSD0005Lvo ::= Lvo  -- LVOs with Class ID = D, E, K, S or R are the only
                     -- further LVOs permitted within the VALUE field of an
                     -- LVO with ADID = CCSD0005

END
```

C.6 ASN.1 Definition of the DDR for ADID = CCSD0009

```
--*************************************************************************
--               ASN.1 Definition of the DDR for ADID = CCSD0009
--*************************************************************************

CCSD0009Module DEFINITIONS ::=

BEGIN

IMPORTS Lvo FROM LvoModule;

CCSD0009ValueField ::= SEQUENCE{
                                CCSD0009Lvo,
                                SET OF CCSD0009Lvo
                               }

CCSD0009Lvo ::= Lvo  -- LVOs with Class ID = U, I, S, K, C or R are the
                     -- only LVOs permitted within the VALUE field of an LVO
                     -- with ADID = CCSD0009

END
```

ANNEX D

STANDARD FORMATTED DATA UNIT RESTRICTED ASCII CODE

Purpose:

This Annex provides the eight-bit code for the symbols that comprise the Restricted ASCII character set used in the LABEL field of an LVO in this Recommendation.

Annex D: ASCII & Restricted ASCII Codes

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 a * next to the code) 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

ANNEX E

STANDARD FORMATTED DATA UNIT OCTET NUMBERING AND NOMENCLATURE

Purpose:

This Annex specifies the octet numbering and nomenclature used in the LABEL field of an LVO.

Annex E: Octet Numbering Convention and Nomenclature

In this document, the following convention is used to identify each octet (8-bit field) in an N-Octet field:

> The first octet in the field (to be extracted) will be drawn in the most left justified position and will be defined to be "Octet 0". The following octet will be defined as "Octet 1" and so on, up to "Octet N-1". When the field is used to express a numerical value, the Most Significant Octet (MSO), shall be the first octet of the field, *i.e., "Octet 0"*. The sequence of decreasing value will be Octet 1 to Octet N-1.
>
> According to the CCSDS convention, the Most Significant Bit (MSB) of any octet shall be the first bit transmitted and it shall be drawn in the most left justified position and designated as "Bit 0". The transmission sequence shall go from MSB to the Least Significant Bit (LSB).

ANNEX F

STANDARD FORMATTED DATA UNIT
Proposed CCSDS Referencing Environment Specifications

Purpose:

The material in this annex is deprecated. Future issues of this document will not contain the material in this annex. CCSDS 622.0-R-1, the Referencing Environment Recommendation, has been completed and in it the CCSDS Referencing Environments are defined. The material in that Recommendation replaces the guidelines suggested in this annex.

Cor. 1

No CCSDS referencing environment specifications were finalised when this Recommendation was issued. The referencing environment specifications given in this annex are being evaluated as potential CCSDS referencing environments.

This Annex is NOT part of this Recommendation.

Annex F: Proposed Referencing Environment Specifications

F.1 Basic Referencing Environment - $CCSDS1

This specification maps directly to all current major filenaming specifications. External data object names within the $CCSDS1 referencing environment are comprised of a filename, preceded by an optional directory specification and followed by an optional extension specification, as shown in Figure F-1.

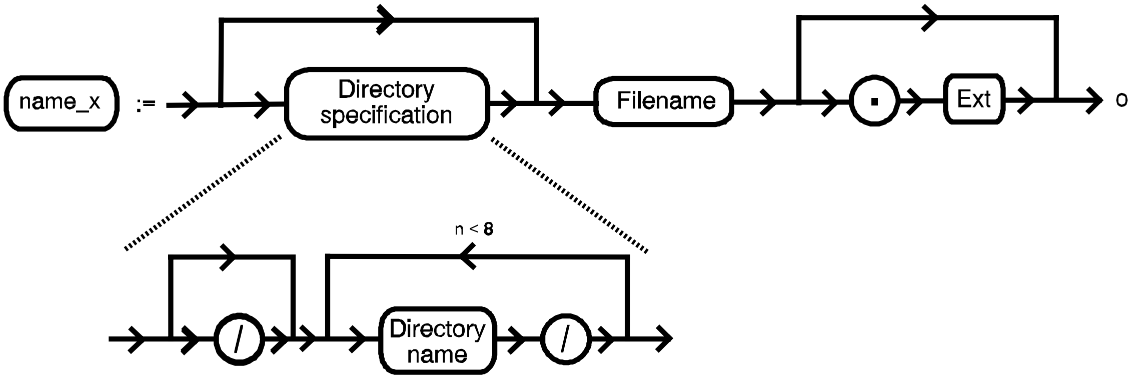


Figure F-1: Structure Diagram of $CCSDS1 Name Specification

The following rules apply:

- The characters permitted in each field of the specification are "A" to "Z", "0" to "9" and "_" (underscore).

- Each **`Directory name`** and the **`Filename`** is a string of 1 to 8 permitted $CCSDS1 characters, while **`Ext`** is a string of 1 to 3 permitted $CCSDS1 characters.

- A maximum of eight directory names are allowed.

- If there is no initial **/** (forward slash), the directory specification is relative to the directory that contains the Referencing LVO. If an initial **/** is present, the directory specification is interpreted as if it is preceded by the directory path needed to reach the highest level directory of the mounted volume or file system which contains the Referencing LVO.

- The $CCSDS1 specification provides two wildcards: ***** (asterisk) and **?** (question mark). A wildcard is permitted in the **`Filename`** and/or **`Ext`** fields. The presence of a wildcard indicates that the field is replaced by the sequence of names found in the reference environment which match the specified pattern. The order of the external data objects matched is not specified. If the order is significant then wildcards should not be used.

- The **?** (question mark) wildcard will match any $CCSDS1 character in the same position in a filename or ext.

- The **`*`** (asterisk) wildcard will match a string of zero or more $CCSDS1 characters in a filename or ext starting from the position of the **`*`**. It must be the last character in the **`Filename`** and/or **`Ext`**.

F.2 Extended Referencing Environment - $CCSDS2

This specification provides extensions to the $CCSDS1 specification for users who find the $CCSDS1 specification too restrictive. It maps directly to most major filenaming specifications, but is less portable than $CCSDS1. External data object names within the $CCSDS2 referencing environment are comprised of a filename, preceded by an optional directory specification and followed by an optional extension specification, as shown in Figure F-2.

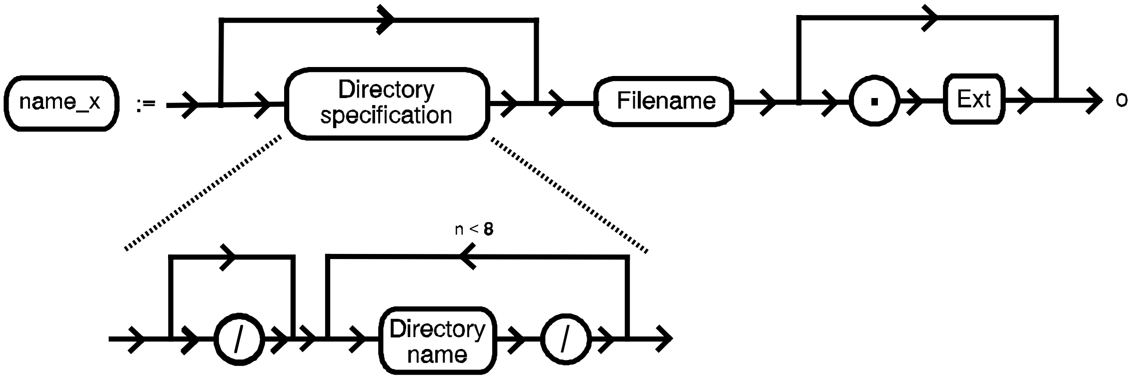


Figure F-2: Structure Diagram of $CCSDS2 Name Specification

The following rules apply:

- The characters permitted in each field of the specification are "A" to "Z", "0" to "9" and "_" (underscore).

- Each **`Directory name`** is a string of 1 to 31 permitted $CCSDS2 characters.

- A maximum of eight directory names are allowed.

- The total length of **`Filename`** and **`Ext`** cannot exceed 30 characters.

- If there is no initial `/` (forward slash), the directory specification is relative to the directory that contains the Referencing LVO. If an initial `/` is present, the directory specification is interpreted as if it is preceded by the directory path needed to reach the highest level directory of the mounted volume or file system which contains the Referencing LVO.

- The $CCSDS2 specification provides two wildcards: **`*`** (asterisk) and **`?`** (question mark). A wildcard is permitted in the **`Filename`** and/or **`Ext`** fields. The presence of a wildcard indicates that the field is replaced by the sequence of names found in the reference environment which match the specified pattern. The order of the external data objects matched is not specified. If the order is significant then wildcards should not be used.

- The **`?`** (question mark) wildcard will match any $CCSDS2 character in the same position in a filename or ext.

- The **`*`** (asterisk) wildcard will match a string of zero or more $CCSDS2 characters in a filename or ext starting from the position of the **`*`**. It must be the last character in the **`Filename`** and/or **`Ext`**.

INDEX