Consultative Committee for Space Data Systems

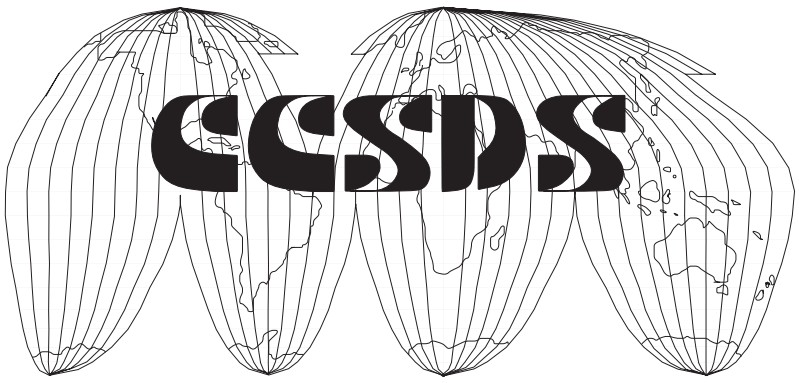
RECOMMENDATION FOR SPACE DATA SYSTEM STANDARDS

PARAMETER VALUE LANGUAGE SPECIFICATION (CCSD0006 and CCSD0008)

CCSDS 641.0-B-2

BLUE BOOK

June 2000



AUTHORITY

Issue:	Blue Book, Issue 2
Date:	June 2000
Location:	Toulouse, France

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 Reference [C3], and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This Recommendation is published and maintained by:

CCSDS Secretariat
Program Integration Division (Code 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:

- 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.

- 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 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 document is a technical Recommendation for the specification of the Parameter Value Language (PVL) and has been prepared by the Consultative Committee for Space Data Systems (CCSDS).

This Recommendation defines the Parameter Value Language that provides a human readable, machine processable language for naming and expressing data values. It allows implementing organizations within each Agency to proceed coherently with the development of compatibly derived Standards for space data systems and widely dispersed data users that are within their cognizance. Derived Agency Standards may implement only a subset of the optional features allowed by the Recommendation and may incorporate features not addressed by the Recommendation.

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 as defined in Reference [C3]. 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.

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

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- 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 e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- National Aeronautics and Space Administration (NASA)/USA.
- National Space Development Agency of Japan (NASDA)/Japan.
- Russian Space Agency (RSA)/Russian Federation.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Communications Research Laboratory (CRL)/Japan.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Federal Service of Scientific, Technical & Cultural Affairs (FSST&CA)/Belgium.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Industry Canada/Communications Research Centre (CRC)/Canada.
- Institute of Space and Astronautical Science (ISAS)/Japan.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Korea Aerospace Research Institute (KARI)/Korea.
- Ministry of Communications (MOC)/Israel.
- National Oceanic & Atmospheric Administration (NOAA)/USA.
- National Space Program Office (NSPO)/Taipei.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

DOCUMENT CONTROL

Document	Title and Issue	Date	Status
CCSDS 641.0-B-1	Parameter Value Language Specification (CCSD0006), Issue 1	May 1992	Original Issue (superseded).
CCSDS 641.0-B-2	Parameter Value Language Specification (CCSD0006 and CCSD0008), Issue 2	May 1999	Current Issue. - Format updated for consistency with current CCSDS style guidelines. - Specification of PVL character set updated to include G1 set of ISO 8859-1 to support use of accented characters used in many languages. - Purpose and Applicability updated to include usage outside of CCSDS. - ASN.1 corrected to ensure order of PVL statements is maintained. - ASN.1 DayOfMonth corrected. - Conformance Section added. - Constraints Section added. - Added clarification that alternate syntactic forms should not be used to carry meaning. - Updated Numeric and White Space definitions. - Consistently capitalized words used to denote PVL syntactical elements.

NOTE – Substantive changes from the previous issue are indicated with change bars in the outside margin.

CONTENTS

CONTENTS (continued)

Figure Page

Table

1 INTRODUCTION

1.1 PURPOSE AND SCOPE

The purpose of this document is to establish a common Recommendation for the specification of a standard keyword value type language for naming and expressing data values. It is useful for wider audiences, but it is designed be used to interchange data in a more uniform fashion within and among Agencies participating in the Consultative Committee for Space Data Systems (CCSDS). This Recommendation provides an overview and formal syntax specification of the Parameter Value Language (PVL). Two versions of PVL are defined—the basic version (CCSD0006) and an extended character set version (CCSD0008).

1.2 APPLICABILITY

The specifications in this document are applicable to applications where a keyword value language is desired. The specifications in this document shall be invoked through the normal standards program of Agencies participating in CCSDS and are applicable to all space-related science and engineering data exchanges where a keyword value language is desired.

1.3 RECOMMENDED APPROACH TO READING THE DOCUMENT

A proper understanding of this Recommendation requires familiarity with the terminology used in this document. Terms are defined as they are introduced in the text. Individuals who are accessing the document out of sequence may wish to refer to 1.4, which presents the terminology used in this document, and Annex A, which presents a summary of the acronyms used in this document. Reference [C5] is a tutorial which describes the requirements, the techniques used to fulfill the requirements, usage guidelines and parser implementation guidelines for PVL. Some readers may find it useful to read Reference [C5] prior to reading this document.

The document is structured as follows:

- Section 2 describes the PVL language, using English text and syntax diagrams.
- Section 3 describes constraints that must be followed to successfully exchange PVL objects.
- Section 4 provides the formal syntax specification written in Abstract Syntax Notation One (ASN.1, see Reference [2]). The comments in the ASN.1 are part of the specification. This is the ruling form of the specification.
- Section 5 describes the single conformance level for this specification.
- Annex A contains acronyms used in this document.
- Annex B lists the ASCII codes for the characters used in PVL.
- Annex C contains a list of informative references.

This document uses syntax diagrams to illustrate the syntax of the various language constructs. Components of the construct are called elements, are presented in boxes or circles and are connected by directional lines. The following conventions are used:

- Elements that are presented in uppercase and lowercase letters in rectangles are defined elsewhere in the document.
- Elements that are presented in a circle as a single bold character are delimiters or reserved characters.
- Elements that are presented in lowercase letters in a rectangle with rounded corners are basic items not further defined in the syntax diagrams of this document.
- Elements that are presented in bold characters in a rectangle with rounded corners are keywords.
- The item named on the left of the ::= symbol is the item being defined.
- The diagram on the right of the ::= symbol is the definition.
- A vertical branch represents a choice.
- A repetition is indicated by a loop back covering the object to be repeated.
- The termination of each structure is represented by the ○ symbol.

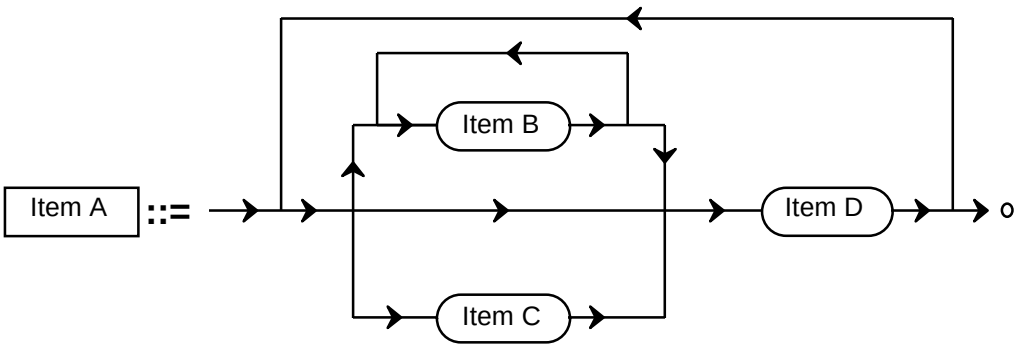


Figure 1-1: Example Structure Diagram

For example:

In this example Item A is defined as first a choice between Items B or C or nothing, where Item B itself may be repeated any number of 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 ○ symbol is taken. Of course if any items on the right (B, C or D) are an Item A or contain an Item A, the definition is recursive. Readers are warned to watch for recursive structure definitions, which are permitted in this Recommendation.

1.4 DEFINITIONS

1.4.1 TERMINOLOGY

Aggregation Block: A named collection of Assignment Statements and/or other Aggregation Blocks.

Alphanumeric character set: The set of characters comprised of the digits 0 through 9 and the letters a-z or A-Z.

Block Name: The name used to identify an Aggregation Block.

Comment: A delimited string of characters, which is treated as White Space syntactically. Comments are intended to provide explanatory information.

Comment delimiters: The character pairs (/* and */) used to delimit a Comment.

End Statement: An optional statement that terminates the PVL Module prior to the end of the provided octet space.

Numeric: A sequence of characters that conform to encoding rules that permit its interpretation as a number.

Octet: A sequence of eight bits.

Parameter Name: The name used to reference the value assigned in the Assignment Statement.

PVL Module: The externally defined octet space that may be optionally terminated by a PVL End Statement, within which PVL statements are written.

Quote String Delimiters: The symbols apostrophe or quotation mark.

Quoted String: Zero or more PVL Characters enclosed between matching Quote String Delimiters.

Reserved Characters: The set of PVL Characters that may not occur in Parameter Names, Unquoted Strings, or Block Names.

Sequence: A delimited collection of values in which the order of the enclosed values is significant.

Set: A delimited collection of values in which the order of the enclosed values is not significant.

Standard Formatted Data Unit: Data units that conform to a specific set of CCSDS Recommendations.

Unquoted String: A value consisting of a sequence of Unrestricted Characters.

Unrestricted Characters: The set of PVL Characters that may be used to form Parameter Names, Unquoted Strings, or Block Names.

White Space: One or more space or format effector characters. Used to separate syntactic elements and to promote readability between syntactic elements or within the contents of Comment or text strings.

1.5 NORMATIVE REFERENCES

[1] *Information Processing — Representation of Numerical Values in Character Strings for Information Interchange*. International Standard, ISO 6093-1985(E). Geneva: ISO, 1985.

[2] *Information Technology — Open System Interconnection — Specification of Abstract Syntax Notation One (ASN.1)*. International Standard, ISO/IEC 8824:1990(E). 2nd Ed. Geneva: ISO, 1990.

[3] *Time Code Formats*. Recommendation for Space Data Systems Standards, CCSDS 301.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, April 1990 or later issue.

2 OVERVIEW OF THE LANGUAGE

2.1 CHARACTER SET DEFINITIONS

The following sections contain character set definitions used in this specification. A clear understanding of these terms is necessary to understand this Recommendation.

2.1.1 PVL CHARACTER SET

The PVL Character Set is split into three subsets: White Space Characters, Reserved Characters, and Unrestricted Characters. The CCSD0006 version of the PVL Character Set is a subset of the ASCII character set. The specific subset is shown in Annex B. The CCSD0008 version of the PVL Character Set is the CCSD0006 version of the PVL Character Set with the Additional Character Set.

2.1.1.1 White Space Character Set

The White Space Character Set is defined as the following characters: space, carriage return, line feed, horizontal tab, vertical tab, and form feed. A sequence of one or more or these characters is known as White Space. The semantic effect of White Space between syntactic elements is not affected by its length.

NOTE – Since sequences of one or more of any of the White Space Characters between syntactic elements are syntactically equivalent, the number of White Space Characters or the use of a particular White Space Character may not be used to provide different meanings (semantics) for applications.

2.1.1.2 Reserved Character Set

The Reserved Character Set is a collection of characters reserved for specific purposes or future use. The Reserved Character Set is defined in Table 2-1.

Table 2-1: Reserved Character Set

Symbol	Name	Symbol	Name	Symbol	Name
&	Ampersand	[	Left Square Bracket	%	Percent Sign
<	Less-Than Sign (Open Angle Bracket)	]	Right Square Bracket	+	Plus Sign
>	Greater-Than Sign (Close Angle Bracket)	=	Equal Sign	"	Quotation Mark
'	Apostrophe	!	Exclamation Point	;	Semicolon
{	Left Curly Bracket (Left Brace)	#	Number Sign, (Hash)	~	Tilde
}	Right Curly Bracket, (Right Brace)	(	Left Parenthesis	|	Vertical Line
,	Comma	)	Right Parenthesis		

2.1.1.3 Unrestricted Character Set

The Unrestricted Character Set is a collection of PVL Characters that are not reserved or used as White Space. The CCSDS0006 version of the Unrestricted Character Set is defined as the alphanumeric character set (`a-z`, `A-Z`, and `0-9`) and the non-alphanumeric characters in Table 2-2. The CCSD0008 version of the Reserved Character set is the CCSD0006 version of the Unrestricted Character Set extended by the Additional Character Set.

Table 2-2: CCSD0006 Unrestricted Character Set

Symbol	Name	Symbol	Name	Symbol	Name
a-z	Lower Case Alphabetics	A-Z	Upper Case Alphabetics	0-9	Digits
*	Asterisk	$	Dollar Sign	?	Question Mark
^	Circumflex Accent, (Caret)	`	Grave Accent	/	Solidus, (Forward Slash)
:	Colon	.	Full Stop, (Period)	\	Reverse Solidus, (Backward Slash)
@	Commercial At	-	Hyphen-Minus Sign	_	Low Line, (Underscore)

2.1.1.4 Additional Character Set

The Additional Character Set is defined as the G1 character set of ISO 8859-1. The Additional Character Set is shown in Table 2-3.

Table 2-3: Additional Character Set

NBSP	°	À	Ð	à	ð
¡	±	Á	Ñ	á	ñ
¢	²	Â	Ò	â	ò
£	³	Ã	Ó	ã	ó
¤	´	Ä	Ô	ä	ô
¥	µ	Å	Õ	å	õ
|	¶	Æ	Ö	æ	ö
§	·	Ç	×	ç	÷
¨	¸	È	Ø	è	ø
©	¹	É	Ù	é	ù
ª	º	Ê	Ú	ê	ú
«	»	Ë	Û	ë	û
¬	¼	Ì	Ü	ì	ü
SHY	½	Í	Ý	í	ý
®	¾	Î	Þ	î	þ
¯	¿	Ï	ß	ï	ÿ

2.1.2 COMMENT

A Comment consists of zero or more PVL Characters enclosed between a pair of Comment Delimiters. The Begin Comment Delimiter is the forward slash-asterisk sequence (`/*`). The End Comment Delimiter is the first following asterisk-forward slash sequence (`*/`). Comments are treated the same as White Space when occurring between syntactic elements, except that Comments cannot appear within a Units Expression. Comments shall not be embedded within other Comments.

NOTE – Since Comments are normally syntactically equivalent to White Space, the presence or absence of Comments may not be used to provide different meanings (semantics) for applications.

2.2 LANGUAGE SYNTAX

PVL provides a specific syntax for the association of values with parameters. A PVL Module consists of a sequence of zero or more statements. These statements are found within an externally provided sequence of Octets. Some or all of these statements can be aggregated into named blocks. Layout (i.e., the use of White Space to promote human readability) is not significant for the interpretation of these statements.

The PVL Module is delimited by either the end of the provided Octet Sequence or by the use of an optional End Statement (see 2.5).

Figure 2-1 contains a syntax diagram for the contents of the PVL Module; it references Figure 2-2, which defines WSC to represent a possibly empty collection of White Space Characters and/or Comments. When this construct appears in syntax diagrams, it represents the capability of using optional White Space and/or Comments between syntactic elements for readability.

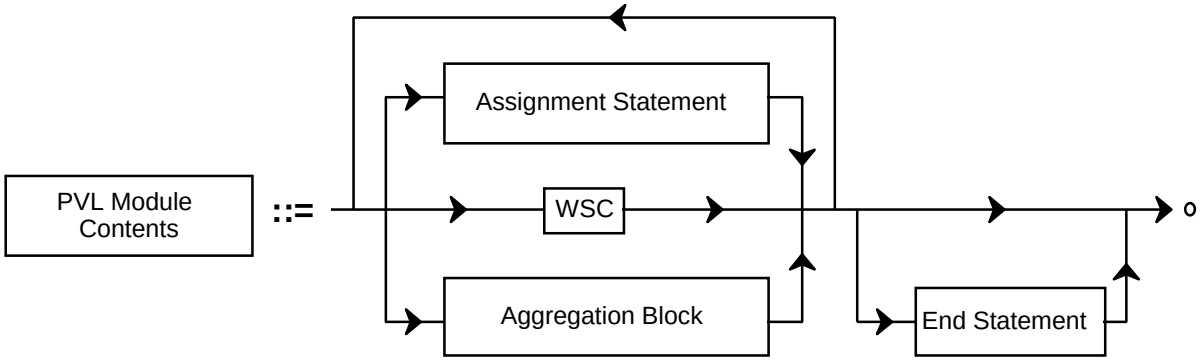


Figure 2-1: PVL Module Contents Syntax Diagram

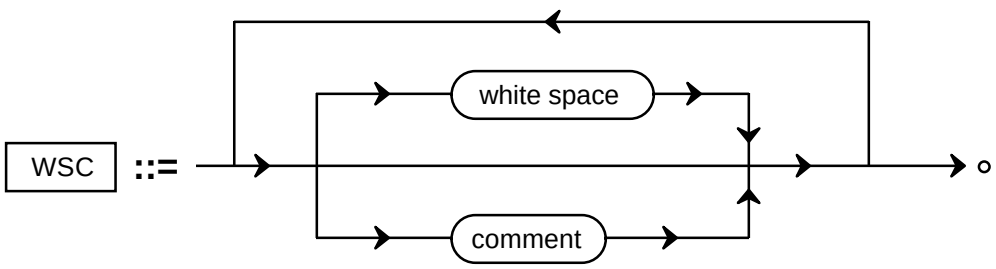


Figure 2-2: White Space/Comment Syntax Diagram

An Assignment Statement has the following general form:

Parameter = Value

An Aggregation Block has the following general form:

Begin Aggregation Statement

A collection of Assignment Statements and/or Aggregation Blocks

End Aggregation Statement

2.3 ASSIGNMENT STATEMENT

An Assignment Statement is used to assign a value to a Parameter Name. Within the Assignment Statement, White Spaces and Comments are ignored between syntactic elements, except where required for statement delimitation.

The Assignment Statement has the following format (the square brackets indicate that the semicolon is optional):

parameter name = value [;]

Figure 2-3 contains a syntax diagram for the PVL Assignment Statement.

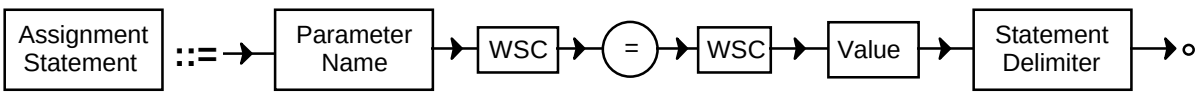


Figure 2-3: Assignment Statement Syntax Diagram

Figure 2-4 contains a syntax diagram illustrating the options for Statement Delimiter.

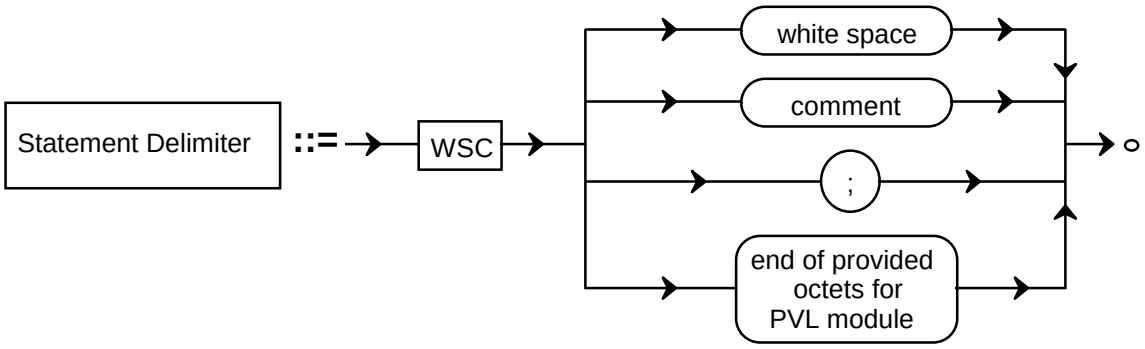


Figure 2-4: Statement Delimiter Syntax Diagram

Statements are separated by the use of a Statement Delimiter, which follows the value. Within this context, a Statement Delimiter is defined as one of the following:

– an explicit delimiter character (`;`), which can be preceded by White Space Characters and/or Comments;

– in the absence of the explicit delimiter character, a set of one or more White Space Characters or Comments;

– the end of the externally provided octet sequence.

Statement delimitation in the absence of the explicit delimiter character is the only time when White Space or Comments have semantic meaning in PVL.

NOTE – Since any of the Statement Delimiters are syntactically equivalent, the use of a particular Statement Delimiter may not be used to provide different meanings (semantics) for applications.

2.3.1 PARAMETER NAME

The Parameter Name provides a way to reference the value assigned in the Assignment Statement. Parameter Names consist of a sequence of Unrestricted Characters.

A Parameter Name must not contain a Comment Delimiter sequence (`/*` or `*/`) and it shall not conform to Numeric encoding rules (see 2.3.2.1.1) or Date/Time encoding rules (see 2.3.2.1.3). A Parameter Name terminates with the character immediately prior to a Reserved Character, a White Space Character, or the beginning of a Comment.

There are seven reserved keywords in PVL:

```
BEGIN_GROUP
BEGIN_OBJECT
END
END_GROUP
END_OBJECT
GROUP
OBJECT
```

Reserved keywords are not permitted as Parameter Names within an Assignment Statement or as Block Names in Aggregation Statements.

2.3.2 VALUE

The Value in an Assignment Statement can be a Simple Value, a Set, or a Sequence. Any Simple Value, Set, or Sequence can optionally be followed by a Units Expression.

Figure 2-5 contains a syntax diagram for Value.

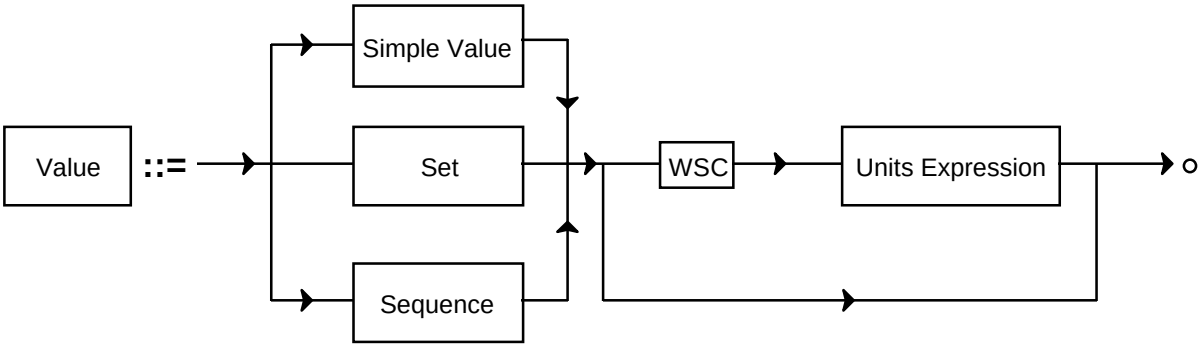


Figure 2-5: Value Syntax Diagram

2.3.2.1 Simple Value

A Simple Value can be a Numeric, String, or Date/Time Value.

Figure 2-6 contains a syntax diagram for a Simple Value.

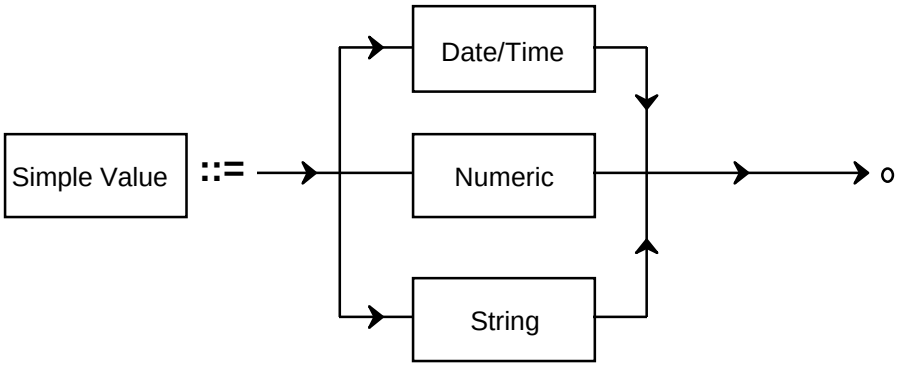


Figure 2-6: Simple Value Syntax Diagram

2.3.2.1.1 Numeric

A Numeric is a sequence of Unrestricted Characters that conform to encoding rules that permit its interpretation as a number. Numerics can be either Decimal Numbers or one of three non-decimal integer encodings: Binary, Octal, and Hexadecimal.

2.3.2.1.1.1 Decimal Numbers

Decimal Numbers follow the three numerical representations (Integer, Floating Point, and Exponential) specified in ISO 6093 (see Reference [3]) for decimal representations, with the exception that comma (,) shall not be used as a decimal point.

2.3.2.1.1.1.1 Integer

Integer numbers correspond to the First Numerical Representation (NR1) in ISO 6093. Each number is represented by at least one decimal digit. The number can be optionally prefixed by a sign symbol (+ or -). An unsigned number will be taken as positive.

Examples:
```
125
+211109
-79
```

2.3.2.1.1.1.2 Floating Point

Floating Point numbers correspond to the Second Numerical Representation (NR2) in ISO 6093. Each number is represented by at least one decimal digit and a decimal point. The decimal point is defined to be the full stop (.). The decimal point can appear anywhere within the sequence. The decimal point is used to separate the integer part of the real number from the

fractional part, at the point where the decimal point is placed. The number can be optionally prefixed by a sign symbol (+ or -). An unsigned number will be taken as positive.

Examples:
```
    69.35
+12456.345
    -0.23456
      .05
    -7.
```

2.3.2.1.1.1.3 Exponential

Exponential numbers correspond to the Third Numerical Representation (NR3) in ISO 6093. Each number is represented by two sequences of decimal digits called the significand (i.e., mantissa) and exponent, separated by the ASCII character `E` or `e`. The value of the number equals the value of the significand multiplied by the result of 10 raised to the power represented by the exponent. The significand can be optionally prefixed by the sign symbol (+ or -). The exponent is an optionally signed integer. If either the significand or exponent is unsigned, it will be taken as positive.

Examples:
```
-2.345678E12
    1.567E-10
    +4.99E+3
```

2.3.2.1.1.2 Non-Decimal Representations

Integer values can also be represented in bases other than base 10. Non-decimal Integers can have a radix of `2` (binary), `8` (octal), or `16` (hexadecimal). The Non-decimal Integer format begins with an optional sign (+ or -) and the radix (in decimal notation), followed by the non-decimal form enclosed within a pair of number signs (`#`). If the optional leading sign has been omitted, then the number will be taken as positive. The non-decimal form itself is interpreted as a positive, uncomplemented integer.

A Non-decimal Integer has the following form

[sign]radix#non_decimal_integer#

where the radix denotes whether the number is binary, octal, or hexadecimal.

NOTE – Any of the above forms of Numerics are stored internally by PVL support software in a format that may be unknown to the user. Therefore, if a particular string of bits is required or must be conserved, for instance, as a mask or flag, then this should be expressed as a Quoted String, e.g. `MASK = "01110101";` and translated to a bit pattern by the application.

2.3.2.1.1.2.1 Binary Numbers

Binary Numbers are represented with a radix value of `2`. The non-decimal portion is a sequence of the characters `0` or `1`.

Example: *`2#0101#`* is equal to the decimal value 5.

2.3.2.1.1.2.2 Octal Numbers

Octal Numbers are represented with a radix value of `8`. The non-decimal portion is a sequence of characters from the following set:

`0, 1, 2, 3, 4, 5, 6, 7.`

Example: *`8#0107#`* is equal to decimal value 71.

2.3.2.1.1.2.3 Hexadecimal Numbers

Hexadecimal Numbers are represented with a radix value of `16`. The non-decimal portion is a sequence of characters from the following set:

`0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, a, b, c, d, e, f`

Lower case letters are equivalent to their upper case counterparts.

Example: *`16#100A#`* is equal to the decimal value 4106.

2.3.2.1.2 String

A String is a sequence of PVL Characters that conforms to the requirements for either a Quoted String or an Unquoted String. Figure 2-7 contains a syntax diagram illustrating the two types of String.

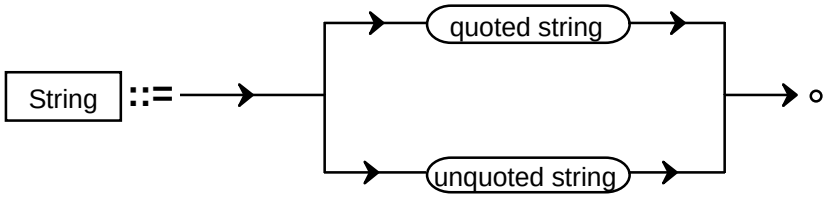


Figure 2-7: String Type Syntax Diagram

2.3.2.1.2.1 Quoted String

A Quoted String consists of zero or more PVL Characters enclosed between matching quote delimiters. The Quote String Delimiters are the quotation mark (") or the apostrophe (').

NOTES

1 A Quote String Delimiter character can be embedded within a String by the use of the Quote String Delimiter not used to enclose the String itself (e.g. "John said 'GOODBYE' and then left" or 'John said "GOODBYE" and then left').

2 If a String is to contain any of the Reserved Characters, White Space Characters, or the Comment Delimiter sequences, it must be a Quoted String rather than an Unquoted String. A String must also be quoted if it conforms to the encoding rules for either Numeric or Date/Time.

3 The above definition allows for null length (i.e., empty) Strings. A null length String may have meaning and therefore is permitted.

4 Since Quote String Delimiters are interchangeable, the use of a particular Quote String Delimiter may not be used to provide different meanings (semantics) for applications.

2.3.2.1.2.2 Unquoted String

An Unquoted String is a sequence of one or more Unrestricted Characters.

An Unquoted String shall not contain the Begin Comment Delimiter (/*) or the End Comment Delimiter (*/), nor shall it conform to Numeric or Date/Time encoding rules. An Unquoted String terminates with the character immediately prior to a White Space Character, a Reserved Character, the beginning of a Comment, or the end of the PVL Module.

2.3.2.1.3 Date/Time Value

The Date/Time Value is a strict subset of the CCSDS ASCII Time Code recommendation (Reference [3]), in which all time is represented in Universal Coordinated Time, (i.e. Greenwich Mean Time). The time construct consists of a combination of date and time constructs.

The date construct has two forms:

YYYY-DDD
where
YYYY is year (0001 to 9999)
DDD is day of year (001 to 365, 366 for leap year)

and

YYYY-MM-DD
where
YYYY is four digit year (0001 to 9999)
MM is month (01 to 12)
DD is day of month (01 to 28, 29, 30 or 31)

Note that each field has a specified width, leading zeros must be included if needed to assure field width. Figure 2-8 contains a syntax diagram of the date format.

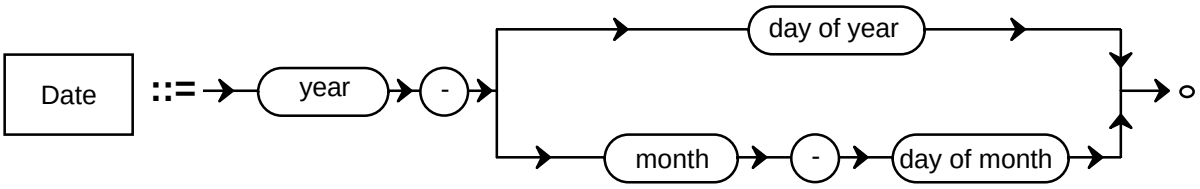


Figure 2-8: Date Syntax Diagram

Examples: *2000-012* is the twelfth day of the year 2000
1995-06-08 is June 8, 1995
1978-04-30 is April 30, 1978

The time construct has the form

hh:mm[:ss[.d...d]]
where
hh is hours (00 to 23)
mm is minutes (00 to 59)
ss is seconds (00 to 60, 60 is to accommodate leap seconds).
d...d is fractional seconds represented by 1 or more digits.

Figure 2-9 contains a syntax diagram for the time format.

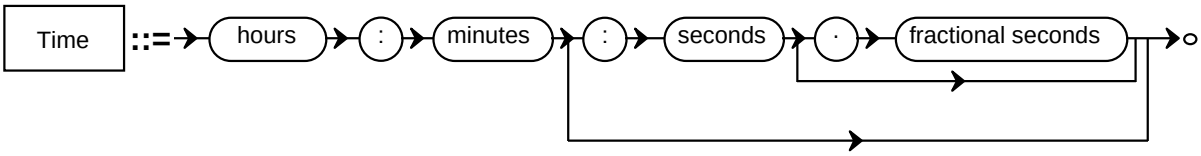


Figure 2-9: Time Syntax Diagram

Examples: *00:00:00.0*
12:01:56
23:01

The complete time construct consists of date, followed by the separator `T` followed by the time construct; all of this can be optionally followed by the character `Z` as a terminator. Separate time and date values can also be used. Figure 2-10 contains a syntax diagram of the Date/Time format.

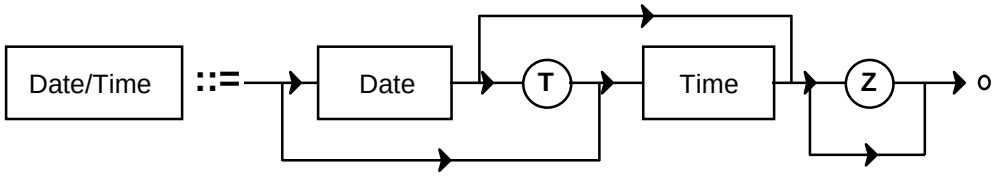


Figure 2-10: Date/Time Syntax Diagram

Examples: *1991-12-22T22:03:12.01Z*
2001-001T12:13
1998-02-12T00:00:01.00
1995-360T14:02:13.0123456Z

2.3.2.2 Set

A Set is a delimited collection of Values in which the order of the Values is not significant and need not be maintained. A Set can contain zero or more Values. If a Set contains two or more Values, they are separated by commas. The beginning of a Set is indicated by a left curly bracket ({), and the end by a right curly bracket (}).

NOTE – The above definition allows for Empty Sets. An Empty Set may have meaning and is therefore permitted.

Figure 2-11 contains a syntax diagram for the Set format.

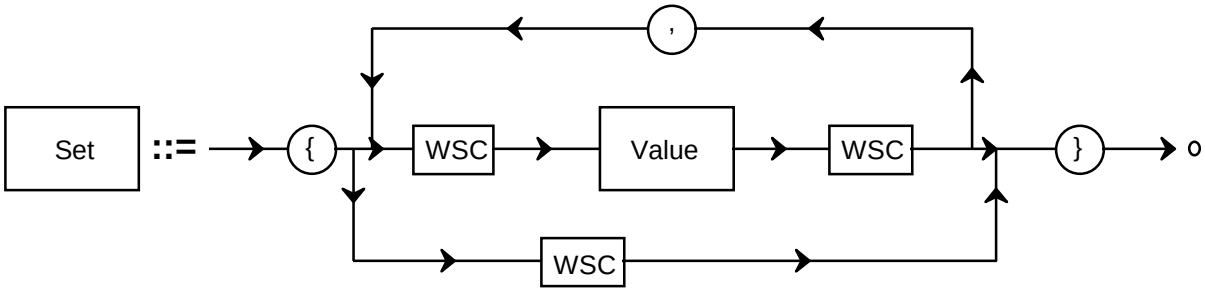


Figure 2-11: Set Syntax Diagram

2.3.2.3 Sequence

A Sequence is a delimited collection of Values in which the order of the Values is significant. A Sequence can contain zero or more Values. If two or more Values are contained in a Sequence, they are separated by commas. The beginning of a Sequence is indicated by a left parenthesis (( ) and the end by a right parenthesis ( )).

NOTE – The above definition allows for Empty Sequences. An Empty Sequence may have meaning and is therefore permitted.

Figure 2-12 contains a syntax diagram for the Sequence format.

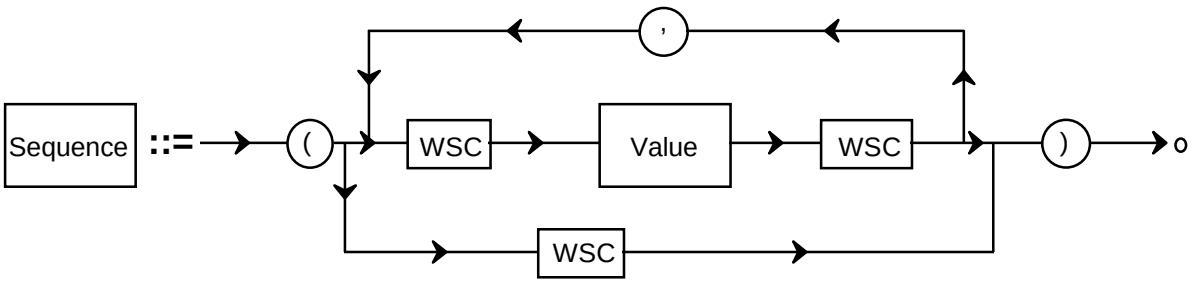


Figure 2-12: Sequence Syntax Diagram

2.3.2.4 Units Expression

Any Simple Value, Set, or Sequence can optionally be followed by a Units Expression. The Units Expression consists of a Units Value contained between an open angle bracket (<) and a close angle bracket (>). The Units Value begins with the first non–White Space Character after the open angle bracket and ends with the last non–White Space Character before the close angle bracket. A Units Value can contain any PVL Character other than the angle brackets themselves.

Figure 2-13 contains a syntax diagram for the Units Expression format. Note that White Space is a collection of one or more White Space Characters.

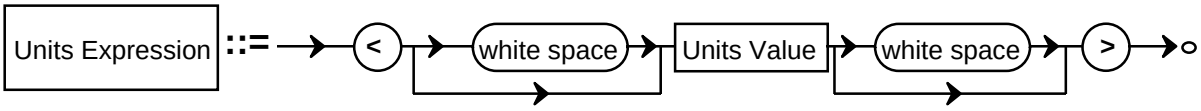


Figure 2-13: Units Expression Syntax Diagram

Figure 2-14 contains a syntax diagram for the Units Value format, in which a units character is any PVL Character other than the open angle bracket (<), close angle bracket (>), or White Space Character.

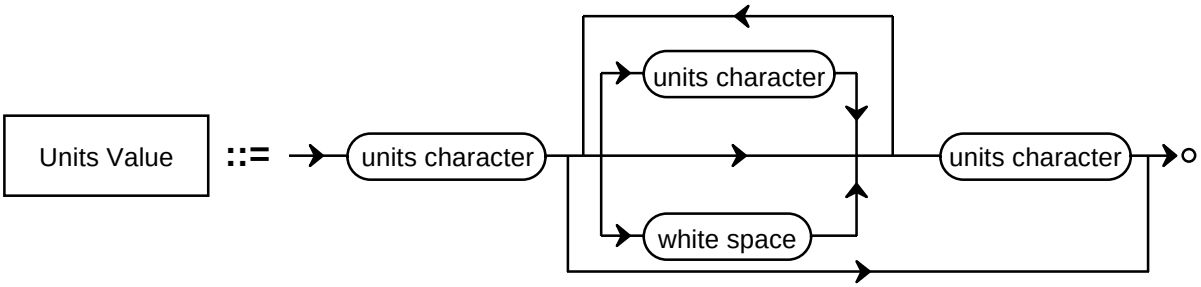


Figure 2-14: Units Value Syntax Diagram

2.4 AGGREGATION BLOCK

The Aggregation Block is a named collection of Assignment Statements and/or other Aggregation Blocks. The Aggregation Block is identified by a Block Name. The start of the block is indicated by a Begin Aggregation Statement and is terminated by an End Aggregation Statement. Figure 2-15 contains a syntax diagram for the Aggregation Block format.

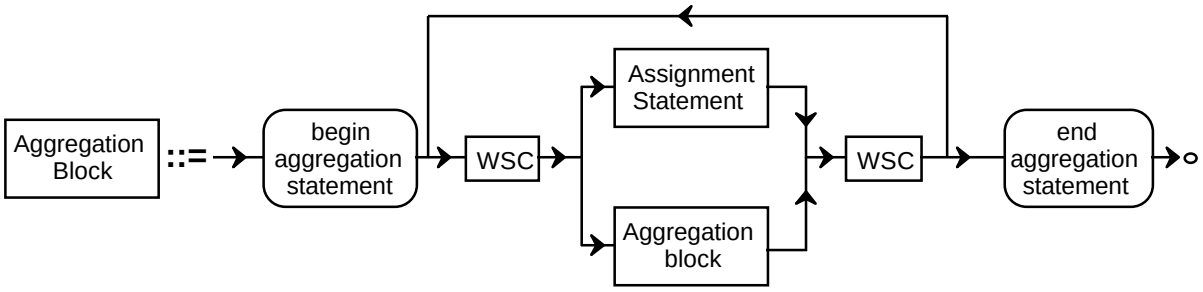


Figure 2-15: Aggregation Block Syntax Diagram

Aggregations are commonly referred to as Groups or Objects. These two keyword forms for Aggregation Statements are permitted to allow for the stylistic preferences. No semantic differentiation between the two is made by PVL. Applications are free to assign such differentiation if desired.

2.4.1 BEGIN AGGREGATION STATEMENT

The Begin Aggregation Statement is parallel in construction to the Assignment Statement. The Begin Aggregation Statement has the following format (the square brackets indicate that the semicolon is optional):

begin aggregation keyword = block name [;]

The Begin Aggregation keywords are `BEGIN_GROUP` and `BEGIN_OBJECT` and are matched with statements that use `END_GROUP` and `END_OBJECT` respectively. The keyword `OBJECT` is a synonym for `BEGIN_OBJECT` and the keyword `GROUP` is a synonym for `BEGIN_GROUP`. The form of the Block Name is identical to Parameter Name.

NOTES

1 These synonyms are allowed for historical compatibility with several existing keyword languages.

2 Since BEGIN_GROUP and GROUP are syntactically equivalent and BEGIN_OBJECT and OBJECT are syntactically equivalent, the use of synonymous forms of the keyword may not be used to provide different meanings (semantics) for applications.

3 Since BEGIN_GROUP and BEGIN_OBJECT are **not** syntactically equivalent, applications may assign different meanings (semantics) to their use.

Figure 2-16 contains a syntax diagram of the Begin Aggregation Statement formats.

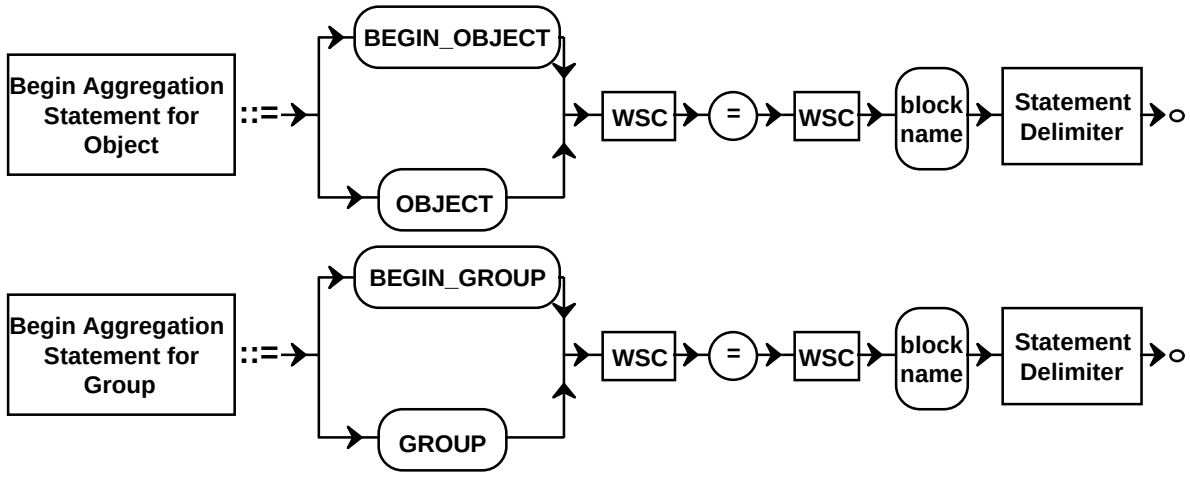


Figure 2-16: Aggregation Begin Statement Syntax Diagram

2.4.2 END AGGREGATION STATEMENT

The End Aggregation Statement is identified by the End Aggregation keyword. The full form of the End Aggregation Statement follows the same construction rules as an Assignment Statement; it has the following format (the square brackets indicate that the semicolon is optional):

end aggregation keyword = Block Name [;]

An abbreviated form of the End Aggregation Statement is allowed as a convenience to the user. The abbreviated End Aggregation Statement has the following format:

end aggregation keyword [;]

The use of the full form is encouraged.

NOTE – Since the full form and the abbreviated form are syntactically equivalent, the use of the abbreviated form rather than the full form may not be used to provide different meanings (semantics) for applications.

The defined End Aggregation keywords are `END_GROUP` and `END_OBJECT`. Figure 2-17 contains a syntax diagram for the End Aggregation Statement.

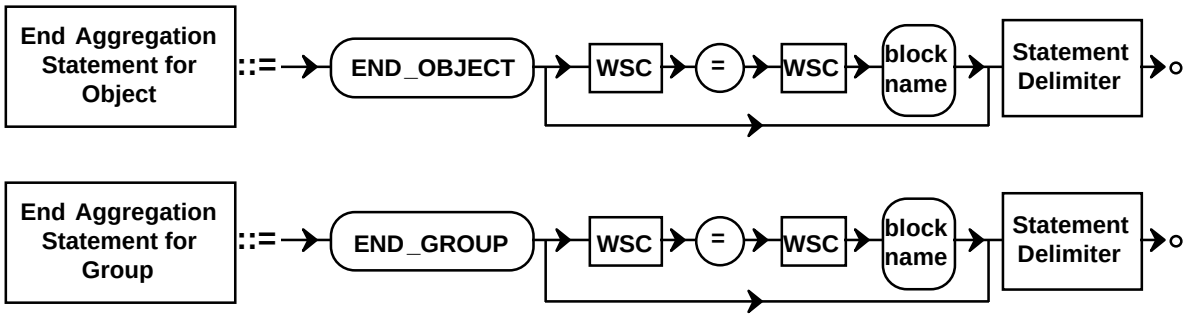


Figure 2-17: End Aggregation Statement Syntax Diagram

NOTE – The preferred form of the aggregation end statement is the full form, which includes the Block Name.

2.4.3 AGGREGATION BLOCK CONSTRUCTION RULES

The end aggregation statement must be paired with a begin aggregation statement. In other words, an Aggregation Block that starts with a `BEGIN_GROUP` statement must end with an `END_GROUP` statement. If a Block Name is used in the end aggregation statement, it must match the name used in the matching begin aggregation statement.

2.5 END STATEMENT

The End Statement is a special type of statement used to delimit a PVL Module prior to the end of the externally provided octet sequence. Figure 2-18 illustrates the syntax

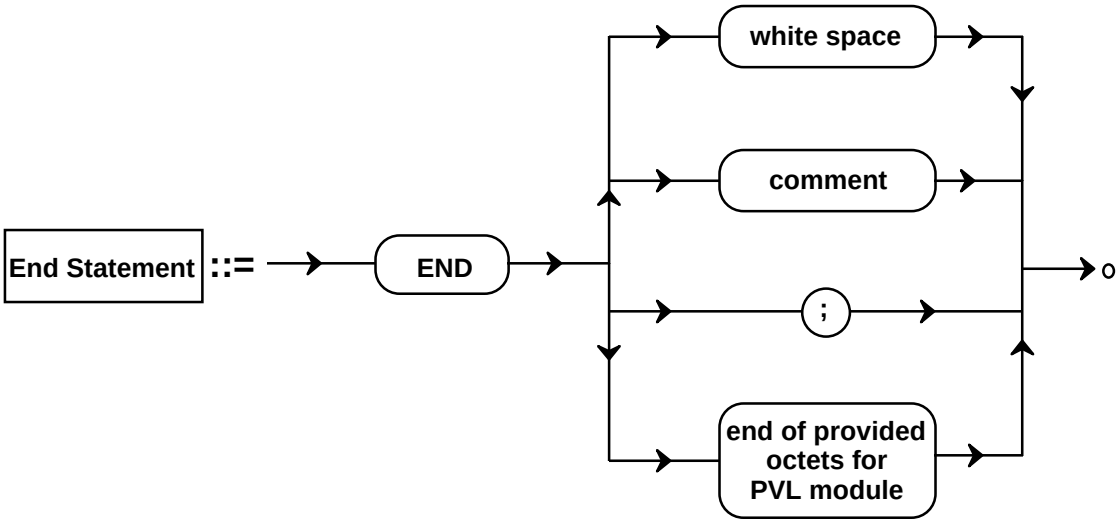


Figure 2-18: End Statement Syntax Diagram

The End Statement is delimited by one of the following: a semicolon; the first White Space Character; the end of a Comment; or the end of the provided octet space.

NOTE – The statement delimitation of the End Statement is more restrictive than for other statements since the remaining Octets in the sequence which may include White Space, Comments, or semicolons, as well as any other character, may have significance to the application.

There shall be at most one End Statement in a PVL Module, and if present it shall be the last statement of the PVL Module.

3 CONSTRAINTS FOR INFORMATION PRESERVATION

To ensure that information is preserved in the exchange of PVL objects among open systems, it is necessary to make clear which PVL formatting options may and may not be altered by these systems. Producing systems will then avoid attaching special meaning to formatting choices that may be altered by automated processes in recipient systems.

The following constraints need to be observed:

1. Statement ordering shall be preserved.

NOTE – Specific applications built on PVL are free to allow statement re-ordering as long as aggregations (`BEGIN_GROUP` and `BEGIN _OBJECT`) are correctly preserved.

2. Statement Delimiters (e.g., White Space, semicolons) may be substituted for each other.

NOTE – The use of a semicolon as the Statement Delimiter is the preferred form.

3. Comments may be added or deleted.

NOTE – Maintaining Comments is the preferred form. Additional Comments may be added as is consistent with valid PVL.

4. White Space between PVL statement elements may be altered as to amount and type.

5. String Delimiters may be added to PVL Strings or removed from PVL Strings as is consistent with valid PVL and where the meaning of the String is not changed. For example, the string "ABCD1234" (with double quotes) may be represented as ‘ABCD1234’ (with single quotes) or as ABCD1234 (without quotes), and vice-versa.

6. PVL Begin Aggregation keywords `GROUP` and `BEGIN_GROUP` may be substituted for each other.

NOTE – The use of the `BEGIN_GROUP` keyword is the preferred form.

7. PVL begin aggregation keywords `OBJECT` and `BEGIN_OBJECT` may be substituted for each other.

NOTE – The use of the `BEGIN_OBJECT` keyword is the preferred form.

8. PVL end aggregation statements `END_GROUP = 'Block Name';` and `END_GROUP;` may be substituted for each other.

NOTE – `END_GROUP = 'Block Name';` is the preferred form.

9. PVL end aggregation statements `END_OBJECT = 'Block Name';` and `END_OBJECT;` may be substituted for each other.

NOTE – `END_OBJECT = 'Block Name';` is the preferred form.

4 PARAMETER VALUE LANGUAGE FORMAL SYNTAX SPECIFICATION

Precedence: In the case of ambiguity of the preceding sections or disagreement with this formal specification, this formal specification shall take precedence. This specification is presented in Abstract Syntax Notation One (ASN.1, see Reference[2]). The comments in the ASN.1 are also part of the specification. Readers unfamiliar with ASN.1 may wish to consult an ASN.1 tutorial such as Reference [C4].

The ASN.1 specification is organized into groupings based on major constructs. Each group begins on a new page with Comment block immediately followed by the definition of the construct with its components in alphabetical order. Components used by more than one major construct are listed in the common language elements group at the end of the specification. Common language elements contain components such as Statement Delimiter, separator, the combination of White Space and Comment (WSC), and character sets.

The construct sections are found on the following pages:

NOTE – The term IA5String as used in this ASN.1 refers to the International ASCII Character Set #5.

4.1 FORMAL SPECIFICATION

```
PVLModule DEFINITIONS      ::= BEGIN
```

```
-- ********************************************************
-- *****
-- *****    PVL MODULE CONTENTS
-- *****
-- ********************************************************

PVLModuleContents       ::= SEQUENCE
                                {
                                SEQUENCE OF
                                    CHOICE
                                        {
                                        Statement,
                                        WSC
                                        },
                                EndStatement OPTIONAL
                                }

EndKeyword              ::= IA5String("END")

EndStatement            ::= SEQUENCE
                                {
                                EndKeyword,
                                CHOICE
                                    {
                                    SemiColon,
                                    WhiteSpace,
                                    Comment,
                                    EndProvidedOctetSeq
                                    }
                                }

Statement               ::=  CHOICE
                                {
                                AssignmentStmt,
                                AggregationBlock
                                }
```

```
-- ********************************************************
-- *****
-- *****   AGGREGATION BLOCK
-- *****
-- ********************************************************

AggregationBlock        ::= CHOICE
                            {
                            AggrGroup,
                            AggrObject
                            }

AggrContents            ::= SEQUENCE
                            {
                            WSC,
                            Statement,      -- Must contain at least one statement
                            SEQUENCE OF
                                CHOICE
                                {
                                WSC,
                                Statement
                                }
                            }

AggrGroup               ::= SEQUENCE
                            {
                            BeginGroupStmt,
                            AggrContents,
                            EndGroupStmt
                            }

AggrObject              ::= SEQUENCE
                            {
                            BeginObjectStmt,
                            AggrContents,
                            EndObjectStmt
                            }
```

```
BeginGroupKeywd     ::= CHOICE
                        {
                        IA5String("BEGIN_GROUP"),
                        IA5String("GROUP")
                        }

BeginGroupStmt      ::= SEQUENCE
                        {
                        BeginGroupKeywd,
                        WSC,
                        AssignmentSymbol,
                        WSC,
                        BlockName,
                        -- Block Name must match Block Name
                        -- in paired End Group Statement
                        -- if Block Name is present in End Group Statement
                        StatementDelim
                        }

BeginObjectKeywd    ::= CHOICE
                        {
                        IA5String("BEGIN_OBJECT"),
                        IA5String("OBJECT")
                        }

BeginObjectStmt     ::= SEQUENCE
                        {
                        BeginObjectKeywd,
                        WSC,
                        AssignmentSymbol,
                        WSC,
                        BlockName,
                        -- Block Name must match Block Name
                        -- in paired End Object Statement
                        -- if Block Name is present in End Object Statement
                        StatementDelim
                        }
```

```
BlockName           ::= SEQUENCE
                        {
                        -- Must not contain the sequence /* or */
                        -- Must not be reserved keyword (see 4.2)
                        -- Must not conform to Numeric encoding rules
                        -- Must not conform to Date/Time encoding rules
                        UnrestrictedChar,
                        SEQUENCE OF UnrestrictedChar
                        }

EndGroupKeywd       ::= IA5String("END_GROUP")

EndGroupLabel       ::= SEQUENCE
                        {
                        AssignmentSymbol,
                        WSC,
                        BlockName
                        -- Block Name must match Block Name
                        -- in paired Begin Group Statement
                        -- if End Group Label is present
                        -- in End Group Statement
                        }

EndGroupStmt        ::= SEQUENCE
                        {
                        EndGroupKeywd,
                        WSC,
                        EndGroupLabel OPTIONAL,
                        StatementDelim
                        }
```

```
EndObjectLabel          ::= SEQUENCE
                            {
                            AssignmentSymbol,
                            WSC,
                            BlockName
                            -- Block Name must match Block Name
                            -- in paired Begin Object Statement
                            -- if End Object Label is present
                            -- in End Object Statement
                            }

EndObjectKeywd          ::= IA5String("END_OBJECT")

EndObjectStmt           ::= SEQUENCE
                            {
                            EndObjectKeywd,
                            WSC,
                            EndObjectLabel OPTIONAL,
                            StatementDelim
                            }
```

```
-- ********************************************************
-- *****
-- *****   ASSIGNMENT STATEMENT
-- *****
-- ********************************************************

AssignmentStmt      ::= SEQUENCE
                        {
                        Name,
                        WSC,
                        AssignmentSymbol,
                        WSC,
                        Value,
                        StatementDelim
                        }

Name                ::= SEQUENCE
                        -- Must not contain the sequence /* or */
                        -- Must not be reserved keyword (see 4.2)
                        -- Must not conform to Numeric encoding rules
                        -- Must not conform to Date/Time encoding rules
                        {
                        UnrestrictedChar,
                        SEQUENCE OF UnrestrictedChar
                        }

SimpleValue         ::= CHOICE
                        {
                        Numeric,
                        String,
                        DateTimeValue
                        }

Value               ::= SEQUENCE
                        {
                        CHOICE
                            {
                            SimpleValue,
                            Set,
                            Sequence
                            },
                        WSC,
                        UnitsExpression OPTIONAL
                        }
```

```
-- ********************************************************
-- *****
-- *****   COMMENT
-- *****
-- ********************************************************

Comment             ::= SEQUENCE
                            {
                            CommentStart,
                            CommentString,
                            CommentEnd
                            }

CommentChar         ::= CHOICE
                            {
                            UnrestrictedChar,
                            WhiteSpace,
                            Apostrophe,
                            QuoteMark,
                            OpenAngleBracket,
                            CloseAngleBracket,
                            SpecialChar
                            }

CommentEnd          ::= IA5String("*/")

CommentStart        ::= IA5String("/*")

CommentString       ::= SEQUENCE OF CommentChar
                            -- Must not contain the sequence "/*" or "*/"
```

```
-- ********************************************************
-- *****
-- *****   DATE/TIME VALUE
-- *****
-- ********************************************************

DateTimeValue           ::= SEQUENCE
                                {
                                CHOICE
                                    {
                                    Date,
                                    Time,
                                    SEQUENCE
                                        {
                                        Date,
                                        DateTimeSeparator,
                                        Time
                                        }
                                    },
                                TimeCodeTerminator OPTIONAL
                                }

Colon                   ::= IA5String(":")

Date                    ::= SEQUENCE
                                {
                                Year,
                                Hyphen,
                                CHOICE
                                    {
                                    DayOfYear,
                                    MonthAndDay
                                    }
                                }
DateTimeSeparator       ::= IA5String("T")
```

```
DayOfMonth          ::= CHOICE
                        {
                        SEQUENCE
                            {
                            -- 01 to 09
                            DecimalChar0,
                            PosDecimalChar
                            },
                        SEQUENCE
                            {
                            -- 10 to 29
                            DecimalChar1to2,
                            DecimalChar
                            },
                        SEQUENCE
                            {
                            -- 30 to 31
                            DecimalChar3,
                            DecimalChar0to1
                            }
                        }
```

```
DayOfYear           ::= CHOICE
                        {
                        SEQUENCE
                            {
                            -- 001 to 009
                            DecimalChar0,
                            DecimalChar0,
                            PosDecimalChar
                            },
                        SEQUENCE
                            {
                            -- 010 to 099
                            DecimalChar0,
                            PosDecimalChar,
                            DecimalChar
                            },
                        SEQUENCE
                            {
                            -- 100 to 299
                            DecimalChar1to2,
                            DecimalChar,
                            DecimalChar
                            },
                        SEQUENCE
                            {
                            -- 300 to 366
                            DecimalChar3,
                            CHOICE
                                {
                                SEQUENCE
                                    {
                                    -- 300 to 359
                                    DecimalChar0to5,
                                    DecimalChar
                                    },
                                SEQUENCE
                                    {
                                    -- 360 to 366
                                    DecimalChar6,
                                    DecimalChar0to6
                                    }
                                }
                            }
                        }
```

```
DecFracSecond       ::= SEQUENCE
                        {
                        DecimalChar,
                        SEQUENCE OF DecimalChar
                        }

DecFracSecondSeq    ::= SEQUENCE
                        {
                        DecimalPoint,
                        DecFracSecond
                        }

DecimalChar0        ::= IA5String("0")

DecimalChar0to1     ::= IA5String(FROM ("0" | "1"))

DecimalChar0to2     ::= IA5String(FROM ("0"| "1"| "2"))

DecimalChar0to3     ::= IA5String(FROM ("0"| "1"| "2"| "3"))

DecimalChar0to5     ::= IA5String(FROM ("0" | "1"| "2"| "3"| "4"| "5"))

DecimalChar0to6     ::= IA5String(FROM ("0" | "1"| "2"| "3"| "4"| "5" |" 6"))

DecimalChar1        ::= IA5String("1")

DecimalChar1to2     ::= IA5String(FROM ("1"| "2"))

DecimalChar2        ::= IA5String("2")

DecimalChar3        ::= IA5String("3")

DecimalChar6        ::= IA5String("6")

DecimalChar60       ::= IA5String("60")
```

```
Hour            ::= CHOICE
                    {
                    SEQUENCE
                        {
                        -- 00 to 19
                        DecimalChar0to1,
                        DecimalChar
                        },
                    SEQUENCE
                        {
                        -- 20 to 23
                        DecimalChar2,
                        DecimalChar0to3
                        }
                    }

Hyphen          ::= IA5String("-")

Minute          ::= SEQUENCE
                    {
                    -- 00 to 59
                    DecimalChar0to5,
                    DecimalChar
                    }


Month           ::= CHOICE
                    {
                    SEQUENCE
                        {
                        -- 00 to 09
                        DecimalChar0,
                        PosDecimalChar
                        },
                    SEQUENCE
                        {
                        -- 10 to 12
                        DecimalChar1,
                        DecimalChar0to2
                        }
                    }
```

```
MonthAndDay             ::= SEQUENCE
                            {
                            Month,
                            Hyphen,
                            DayOfMonth
                            }

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

Second                  ::= CHOICE
                            {
                            SEQUENCE
                                {
                                -- 00 to 59
                                DecimalChar0to5,
                                DecimalChar
                                },
                            DecimalChar60
                            --  60 is allowed for leap seconds
                            }

SecondSeq               ::= SEQUENCE
                            {
                            Colon,
                            Second,
                            DecFracSecondSeq OPTIONAL
                            }

Time                    ::= SEQUENCE
                            {
                            Hour,
                            Colon,
                            Minute,
                            SecondSeq OPTIONAL
                            }

TimeCodeTerminator      ::= IA5String("Z")

Year                    ::= SEQUENCE
                            {
                            -- year 0000 is not allowed --
                            DecimalChar,
                            DecimalChar,
                            DecimalChar,
                            DecimalChar
                            }
```

```
-- ********************************************************
-- *****
-- *****   NUMERIC VALUES
-- *****
-- ********************************************************

Numeric             ::= CHOICE
                        {
                        Integer,
                        FloatingPoint,
                        Exponential,
                        BinaryNum,
                        OctalNum,
                        HexadecimalNum
                        }

BinaryChar          ::= IA5String(FROM("0" |"1"))

BinaryNum           ::= SEQUENCE
                        {
                        Sign OPTIONAL,
                        IA5String("2"),
                        RadixSymbol,
                        -- Binary characters are interpreted
                        --   as a positive and uncomplemented integer
                        BinaryChar,
                        SEQUENCE OF BinaryChar,
                        RadixSymbol
                        }
```

```
Exponential          ::= SEQUENCE
                         {
                         CHOICE
                             {
                             Integer,
                             FloatingPoint
                             },
                         ExponentMark,
                         Integer
                         }

ExponentMark         ::= IA5String(FROM("e" |"E"))

FloatingPoint        ::= SEQUENCE
                         {
                         Sign OPTIONAL,
                         -- If all digits in number are 0,
                         --   only legal value for sign is +
                         CHOICE
                             {
                             SEQUENCE
                                 {
                                 DecimalChar,
                                 -- Ensures at least one digit to
                                 --  left of decimal point
                                 SEQUENCE OF DecimalChar,
                                 DecimalPoint,
                                 SEQUENCE OF DecimalChar
                                 },
                             SEQUENCE
                                 {
                                 SEQUENCE OF DecimalChar,
                                 DecimalPoint,
                                 DecimalChar,
                                 -- Ensures at least one digit to
                                 --  right of decimal point
                                 SEQUENCE OF DecimalChar
                                 }
                             }
                         }

HexadecimalChar      ::= IA5String(FROM("0" |"1" |"2" |"3" |"4" |"5" |"6" |"7" |"8"
                             |"9" |"A" |"B" |"C" |"D" |"E" |"F" |"a" |"b" |"c" |"d"
                             |"e" |"f"))
```

```
HexadecimalNum      ::= SEQUENCE
                        {
                        Sign OPTIONAL,
                        IA5String("16"),
                        RadixSymbol,
                        -- Hexadecimal characters are interpreted
                        --   as a positive and uncomplemented integer
                        HexadecimalChar,
                        SEQUENCE OF HexadecimalChar,
                        RadixSymbol
                        }

Integer             ::= SEQUENCE
                        {
                        Sign OPTIONAL,
                        -- If all digits in number are 0,
                        --   only legal value for sign is +
                        DecimalChar,
                        SEQUENCE OF DecimalChar
                        }

OctalChar           ::= IA5String(FROM("0" |"1" |"2" |"3" |"4" |"5" |"6" |"7"))

OctalNum            ::= SEQUENCE
                        {
                        Sign OPTIONAL,
                        IA5String("8"),
                        RadixSymbol,
                        -- Octal characters are interpreted
                        --   as a positive and uncomplemented integer
                        OctalChar,
                        SEQUENCE OF OctalChar,
                        RadixSymbol
                        }

RadixSymbol         ::= IA5String("#")

Sign                ::= IA5String(FROM("+" |"-"))
```

```
                -- ********************************************************
                -- *****
                -- *****   SEQUENCE
                -- *****
                -- ********************************************************

Sequence            ::= SEQUENCE
                            {
                            SequenceStart,
                            WSC,
                            SequenceValue OPTIONAL,
                            WSC,
                            SequenceEnd
                            }

SequenceEnd         ::= IA5String(")")

SequenceStart       ::= IA5String("(")

SequenceValue       ::= SEQUENCE
                            {
                            Value,
                            SEQUENCE OF
                                SEQUENCE
                                    {
                                    WSC,
                                    SeparatorSymbol,
                                    WSC,
                                    Value
                                    }
                            }
```

```
-- ********************************************************
-- *****
-- *****   SET
-- *****
-- ********************************************************

Set                 ::= SEQUENCE
                        {
                        SetStart,
                        WSC,
                        SetValue OPTIONAL,
                        WSC,
                        SetEnd
                        }

SetEnd              ::= IA5String("}")

SetStart            ::= IA5String("{")

SetValue            ::= SEQUENCE
                        {
                        Value,
                        SEQUENCE OF
                            SEQUENCE
                                {
                                WSC,
                                SeparatorSymbol,
                                WSC,
                                Value
                                }
                        }
```

```
-- ********************************************************
-- *****
-- *****   STRING
-- *****
-- ********************************************************

String              ::= CHOICE
                        {
                        QuotedString,
                        UnquotedString
                        }

QuotedString        ::= CHOICE
                        {
                        QuotedString1,
                        QuotedString2
                        }

QuotedChar          ::= CHOICE
                        {
                        UnrestrictedChar,
                        WhiteSpace,
                        OpenAngleBracket,
                        CloseAngleBracket,
                        SpecialChar
                        }

QstringDelim1       ::= QuoteMark

QstringDelim2       ::= Apostrophe

QuotedString1       ::= SEQUENCE
                        {
                        QstringDelim1,
                        -- quotation mark
                        SEQUENCE OF
                            CHOICE
                                {
                                Apostrophe,
                                -- character used for QstringDelim2
                                QuotedChar
                                },
                        QstringDelim1
                        -- quotation mark
                        }
```

```
QuotedString2           ::= SEQUENCE
                                {
                                QstringDelim2,
                                -- apostrophe
                                SEQUENCE OF
                                    CHOICE
                                        {
                                        QuoteMark,
                                        -- character used for QstringDelim1
                                        QuotedChar
                                        },
                                QstringDelim2
                                -- apostrophe
                                }

UnquotedString          ::= SEQUENCE
                                -- Must not contain the sequence /* or */
                                -- Must not be reserved keyword (see 4.2)
                                -- Must not conform to Numeric encoding rules
                                -- Must not conform to Date/Time encoding rules
                                {
                                UnrestrictedChar,
                                SEQUENCE OF UnrestrictedChar
                                }
```

```
-- ********************************************************
-- *****
-- *****    UNIT EXPRESSION
-- *****
-- ********************************************************


UnitsExpression             ::= SEQUENCE
                                    {
                                    UnitsStart,
                                    SEQUENCE OF WhiteSpace,
                                    UnitsValue,
                                    SEQUENCE OF WhiteSpace,
                                    UnitsEnd
                                    }

RemainUnitValueChars        ::= SEQUENCE
                                    {
                                    SEQUENCE OF
                                        CHOICE
                                            {
                                            WhiteSpace,
                                            UnitsChar
                                            },
                                    UnitsChar
                                    }

UnitsChar                   ::= CHOICE
                                    {
                                    UnrestrictedChar,
                                    SpecialChar,
                                    Apostrophe,
                                    QuoteMark
                                    }

UnitsEnd                    ::= CloseAngleBracket

UnitsStart                  ::= OpenAngleBracket

UnitsValue                  ::= SEQUENCE
                                    {
                                    UnitsChar,
                                    RemainUnitValueChars OPTIONAL
                                    }
```

```
-- ********************************************************
-- *****
-- *****   COMMON LANGUAGE ELEMENTS
-- *****
-- ********************************************************

Apostrophe              ::= IA5String("'")

AssignmentSymbol        ::= IA5String("=")

CarriageRet             ::= IA5String(13)
                            -- ASCII carriage return

CloseAngleBracket       ::= IA5String(">")

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

DecimalPoint            ::= IA5String(".")
                            -- full stop

EndProvidedOctetSeq     ::= EXTERNAL
                            -- This is the token returned
                            -- by the system that indicates
                            -- that the end of the externally
                            -- provided Octet sequence has been reached.

AdditionalChar          ::= T61String(FROM(
    Nbsp |“¡” |“¢” |“£” |“¤” |“¥” |“|“ |“§” |“¨” |“©” |“ª” |“«” |“¬“ |Shy |“®” |“¯ ”
    |”°” |“±” |“²” |“³” |“´” |“µ“ |“¶” |“·” |“¸” |“¹” |“º” |“»” |“¼” |“½” |“¾” |“¿”
    |”À” |“Á” |“Â” |“Ã” |“Ä” |“Å” |“Æ” |“Ç” |“È” |“É” |“Ê” |“Ë” |“Ì” |“Í” |“Î” |“Ï”
    |”Ð” |“Ñ” |“Ò” |“Ó” |“Ô” |“Õ” |“Ö” |“×“ |“Ø” |“Ù” |“Ú” |“Û” |“Ü” |“Ý” |“Þ” |“ß”
    |”à” |”á” |”â” |”ã” |”ä” |”å” |”æ” |”ç” |”è” |”é” |”ê” |”ë” |”ì” |”í” |”î” |”ï”
    |“ð” |“ñ” |“ò” |“ó” |“ô” |“õ” |“ö” |“÷” |“ø” |“ù” |“ú” |“û” |“ü” |“ý” |“þ” |"ÿ"))
                            -- All characters from G1 character set of ISO 8859-1

FormFeed                ::= IA5String(12)
                            -- ASCII form feed

HorizontalTab           ::= IA5String(9)
                            -- ASCII horizontal tab
```

```
Letter                ::= 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" |"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"))

LineFeed              ::= IA5String(10)
                          -- ASCII line feed

Nbsp                  ::= T61String(160)
                          -- ISO 8859-1 nbsp character

OpenAngleBracket      ::= IA5String("<")

QuoteMark             ::= IA5String(34)
                          -- ASCII quote symbol

SemiColon             ::= IA5String(";")

SeparatorSymbol       ::= IA5String(",")

Shy                   ::= T61String(173)
                          -- ISO 8859-1 shy character

Space                 ::= IA5String(32)
                           -- ASCII space character

SpecialChar           ::= IA5String(FROM( "(" |")" |"{" | "}" |"#" |"," |";" |"=" |"[" |"]"
                              |"!" |"%" | "&" |"~" |"|" |"+" ))
                              -- Characters allowed in Comments, Quoted Strings
                              -- or Units but not in Unquoted Strings, Block Names
                              --  or Parameter Names
```

```
StatementDelim      ::= CHOICE
                        {
                        SEQUENCE
                            {
                            WSC,
                            CHOICE
                                {
                                SemiColon,
                                WhiteSpace,
                                Comment
                                -- ensure that a Statement Delimiter consists
                                -- of one semicolon, optionally preceded by
                                -- multiple White Spaces and/or Comments,
                                -- OR one or more Comments and/or
                                -- White Space sequences.
                                }
                            },
                        EndProvidedOctetSeq
                        }

UnrestrictedChar    ::= CHOICE
                        {
                        DecimalChar,
                        Letter,
                        AdditionalChar,  -- Included only for CCSD0008 version
                        UnrestrictedSymbol
                        }
```

```
UnrestrictedSymbol      ::= IA5String(FROM("$" |"-" |"." |"/" |":" |"?" |"@" |"\" |"^" |"_"
                                |"`" |"*" ))
VerticalTab             ::= IA5String(11)
                            -- ASCII vertical tab

WhiteSpace              ::= CHOICE
                                {
                                Space,
                                HorizontalTab,
                                VerticalTab,
                                CarriageRet,
                                LineFeed,
                                FormFeed
                                }

WSC                     ::= SEQUENCE OF
                                CHOICE
                                    {
                                    WhiteSpace,
                                    Comment
                                    }
END
```

4.2 RESERVED KEYWORDS

The following reserved keywords are not available for use as Parameter Names in Assignment Statements or as Block Names in Aggregation Statements:

```
BEGIN_GROUP
BEGIN_OBJECT
END_GROUP
END_OBJECT
END
GROUP
OBJECT
```

5 CONFORMANCE

Data conforming to a Recommendation may be said to be in conformance at some identified level. Identifying conformance levels provides a standard way to classify the required capabilities of generating and receiving systems.

This Recommendation recognizes only two conformance levels—CCSD006 Conformance and CCSD0008 Conformance.

Recipient systems that are said to be in CCSD0006 Conformance to the Recommendation shall recognize the entire specification using the CCSD0006 Character Set. Generating systems that are said to be in CCSD0006 Conformance to this Recommendation shall generate material recognizable by any CCSD0006 Conforming recipient systems.

Recipient systems that are said to be in CCSD0008 Conformance to the Recommendation shall recognize the entire specification using the CCSD0008 Character Set. Generating systems that are said to be in CCSD0008 Conformance to this Recommendation shall generate material recognizable by any CCSD0008 Conforming recipient systems.

ANNEX A

ACRONYMS

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

Purpose:

This annex defines key acronyms used throughout this Recommendation to describe the concepts and elements of the Parameter Value Language.

ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
CCSDS	Consultative Committee for Space Data Systems
ISO	International Organization for Standardization
PVL	Parameter Value Language
SFDU	Standard Formatted Data Unit

ANNEX B

CHARACTER DEFINITIONS

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

Purpose:

This annex contains the definition of the character representations used by the Parameter Value Language Specification.

PVL CHARACTER SETS

The CCSD0006 PVL Character Set is a subset of the ASCII Character Set, specifically the 7-bit portion (MSB=0) of ISO 8859-1:1987 (Reference I5) which is known as the G0 Character Set which set corresponds to ANSI 3.4-1977. This is also the same as the ISO 646 IRV set (Reference [C1]), with the exception of positions 36 and 126, which display the currency symbol and overscore, respectively, in the IRV.

The CCSD0006 PVL Character Set consists of the printable characters occupying the positions 33 to 126, inclusive (the Graphics Characters), the space (32), and the format effectors (positions 9 to 13 inclusive). These characters are listed on the following page.

The CCSD0008 PVL Character set includes all characters in the CCSD0006 Character Set and is expanded by the 8 bit portion (MSB=1) of ISO 8859-1:1987 which is identified as the G1 Character Set. These characters are listed on the page following the CCSD0006 Characters. This character set is a subset of the 16-bit Basic Multilingual Plane (BMP) of the ISO 10646 coded character set (reference). This subset is defined as the first 256 characters (row00) of the BMP, which corresponds to the ISO 8859-1, which is an 8-bit single-byte coded graphic character set, also known as “Latin Alphabet No. 1”. The corresponding codes are shown in the following tables. (The code for each character (Char) is given in decimal (Dec), and hexadecimal (Hex).)

The whole ISO 8859-1 character set is shown in the following tables. The characters shaded in the following tables are not included in the PVL character sets and they should not appear in PVL.

Some of the defined characters need some explanations:

a) A Space (SP) might be interpreted as a graphic character, or a control character or both. As a graphic character, its representation consists of no symbol, but it takes up display space.

b) A No_Break_Space (NBSP) is a graphic character for which the representation consists of no symbol, but it takes up display space. It shall be used when no break (new line) is allowed.

c) A Soft_Hyphen (SHY) is used to represent where a word may be broken at the end of a line. Its representation consists of a hyphen before the new line when a word is broken at the end of a line. Its representation consists of no symbol and it does not take up display space if it does not appear at the end of a line.

The use of an ISO 8859-1 encoding to represent the natural language also permits the incorporation of tables and figures that can be drawn with the characters listed below. For these figures or tables to be presented identically for any receiver, the interpretation of the control characters (Vertical Tab, Horizontal Tab, Form Feed, Line Feed (also known as New Line) and Carriage Return) must be standardized. The following rules apply:

a) A Carriage Return positions the next displayable character as the first position on the current line.

b) A Line Feed positions the next displayable character one line below the current displayable character position.

c) A Horizontal Tab character positions the next displayable character onto the next character position that is a multiple of 8 (i.e., character positions 8, 16, 24, 32 etc., where the leftmost displayable character position is 0).

d) A Form Feed character positions the next displayable character to the leftmost displayable position and down to the beginning of the next page. The definition of a page is as defined by the local device (e.g., a new screen for a visual display unit (VDU) or a new piece of paper for a printer).

e) If the characteristics of the display device conflict with those of the data, for example, line lengths may be greater than those permitted by the device, then some adjustment to the layout of the data, as determined by the device, will occur. (Note also that some devices may process or react to codes which this Recommendation specifies as being ignored for presentation purposes.)

NOTE – If the alignment of the displayed characters is significant to the understanding of the information, then a fixed space font should be used for presentation.

Char	Dec	Hex
NUL	0	00
SOH	1	01
STX	2	02
ETX	3	03
EOT	4	04
ENQ	5	05
ACK	6	06
BEL	7	07
BS	8	08
HT	9	09
LF	10	0A
VT	11	0B
FF	12	0C
CR	13	0D
SO	14	0E
SI	15	0F
DLE	16	10
DC1	17	11
DC2	18	12
DC3	19	13
DC4	20	14
NAK	21	15
SYN	22	16
ETB	23	17
CAN	24	18
EM	25	19
SUB	26	1A
ESC	27	1B
FS	28	1C
GS	29	1D
RS	30	1E
US	31	1F

Char	Dec	Hex
Space	32	20
!	33	21
“	34	22
#	35	23
$	36	24
%	37	25
&	38	26
‘	39	27
(	40	28
)	41	29
*****	42	2A
+	43	2B
,	44	2C
-	45	2D
.	46	2E
/	47	2F
0	48	30
1	49	31
2	50	32
3	51	33
4	52	34
5	53	35
6	54	36
7	55	37
8	56	38
9	57	39
:	58	3A
;	59	3B
<	60	3C
=	61	3D
>	62	3E
?	63	3F

Char	Dec	Hex
@	64	40
A	65	41
B	66	42
C	67	43
D	68	44
E	69	45
F	70	46
G	71	47
H	72	48
I	73	49
J	74	4A
K	75	4B
L	76	4C
M	77	4D
N	78	4E
O	79	4F
P	80	50
Q	81	51
R	82	52
S	83	53
T	84	54
U	85	55
V	86	56
W	87	57
X	88	58
Y	89	59
Z	90	5A
[	91	5B
\	92	5C
]	93	5D
^	94	5E
_	95	5F

Char	Dec	Hex
`	96	60
a	97	61
b	98	62
c	99	63
d	100	64
e	101	65
f	102	66
g	103	67
h	104	68
i	105	69
j	106	6A
k	107	6B
l	108	6C
m	109	6D
n	110	6E
o	111	6F
p	112	70
q	113	71
r	114	72
s	115	73
t	116	74
u	117	75
v	118	76
w	119	77
x	120	78
y	121	79
z	122	7A
{	123	7B
|	124	7C
}	125	7D
~	126	7E
DEL	127	7F

Char	Dec	Hex
Res	128	80
Res	129	81
Res	130	82
Res	131	83
IND	132	84
NEL	133	85
SSA	134	86
ESA	135	87
HTS	136	88
HTJ	137	89
VTS	138	8A
PLD	139	8B
PLU	140	8C
RI	141	8D
SS2	142	8E
SS3	143	8F
DCS	144	90
PU1	145	91
PU2	146	92
STS	147	93
CCH	148	94
MW	149	95
SPA	150	96
EPA	151	97
Res	152	98
Res	153	99
Res	154	9A
CSI	155	9B
ST	156	9C
OSC	157	9D
PM	158	9E
APC	159	9F

Char	Dec	Hex
NBSP	160	A0
¡	161	A1
¢	162	A2
£	163	A3
¤	164	A4
¥	165	A5
¦	166	A6
§	167	A7
¨	168	A8
©	169	A9
ª	170	AA
«	171	AB
¬	172	AC
SHY	173	AD
®	174	AE
¯	175	AF
°	176	B0
±	177	B1
²	178	B2
³	179	B3
´	180	B4
µ	181	B5
¶	182	B6
·	183	B7
¸	184	B8
¹	185	B9
º	186	BA
»	187	BB
¼	188	BC
½	189	BD
¾	190	BE
¿	191	BF

Char	Dec	Hex
À	192	C0
Á	193	C1
Â	194	C2
Ã	195	C3
Ä	196	C4
Å	197	C5
Æ	198	C6
Ç	199	C7
È	200	C8
É	201	C9
Ê	202	CA
Ë	203	CB
Ì	204	CC
Í	205	CD
Î	206	CE
Ï	207	CF
Ð	208	D0
Ñ	209	D1
Ò	210	D2
Ó	211	D3
Ô	212	D4
Õ	213	D5
Ö	214	D6
×	215	D7
Ø	216	D8
Ù	217	D9
Ú	218	DA
Û	219	DB
Ü	220	DC
Ý	221	DD
Þ	222	DE
ß	223	DF

Char	Dec	Hex
à	224	E0
á	225	E1
â	226	E2
ã	227	E3
ä	228	E4
å	229	E5
æ	230	E6
ç	231	E7
è	232	E8
é	233	E9
ê	234	EA
ë	235	EB
ì	236	EC
í	237	ED
î	238	EE
ï	239	EF
ð	240	F0
ñ	241	F1
ò	242	F2
ó	243	F3
ô	244	F4
õ	245	F5
ö	246	F6
÷	247	F7
ø	248	F8
ù	249	F9
ú	250	FA
û	251	FB
ü	252	FC
ý	253	FD
þ	254	FE
ÿ	255	FF

The following tables assign a name (according to the ISO standard) to each printable character of the set.

Hex	Name
09	HORIZONTAL TAB
0A	LINE FEED
0C	FORM FEED
0D	CARRIAGE RETURN
20	SPACE
21	EXCLAMATION MARK
22	QUOTATION MARK
23	NUMBER SIGN
24	DOLLAR SIGN
25	PERCENT SIGN
26	AMPERSAND
27	APOSTROPHE
28	LEFT PARENTHESIS
29	RIGHT PARENTHESIS
2A	ASTERISK
2B	PLUS SIGN
2C	COMMA
2D	HYPHEN, MINUS SIGN
2E	FULL STOP
2F	SOLIDUS
30	DIGIT ZERO
31	DIGIT ONE
32	DIGIT TWO
33	DIGIT THREE
34	DIGIT FOUR
35	DIGIT FIVE
36	DIGIT SIX
37	DIGIT SEVEN
38	DIGIT EIGHT
39	DIGIT NINE
3A	COLON
3B	SEMICOLON
3C	LESS THAN SIGN

Hex	Name
3D	EQUALS SIGN
3E	GREATER THAN SIGN
3F	QUESTION MARK
40	COMMERCIAL AT
41	CAPITAL LETTER A
42	CAPITAL LETTER B
43	CAPITAL LETTER C
44	CAPITAL LETTER D
45	CAPITAL LETTER E
46	CAPITAL LETTER F
47	CAPITAL LETTER G
48	CAPITAL LETTER H
49	CAPITAL LETTER I
4A	CAPITAL LETTER J
4B	CAPITAL LETTER K
4C	CAPITAL LETTER L
4D	CAPITAL LETTER M
4E	CAPITAL LETTER N
4F	CAPITAL LETTER O
50	CAPITAL LETTER P
51	CAPITAL LETTER Q
52	CAPITAL LETTER R
53	CAPITAL LETTER S
54	CAPITAL LETTER T
55	CAPITAL LETTER U
56	CAPITAL LETTER V
57	CAPITAL LETTER W
58	CAPITAL LETTER X
59	CAPITAL LETTER Y
5A	CAPITAL LETTER Z
5B	LEFT SQUARE BRACKET
5C	REVERSE SOLIDUS
5D	RIGHT SQUARE BRACKET

Hex	Name
5E	CIRCUMFLEX ACCENT
5F	LOW LINE
60	GRAVE ACCENT
61	SMALL LETTER A
62	SMALL LETTER B
63	SMALL LETTER C
64	SMALL LETTER D
65	SMALL LETTER E
66	SMALL LETTER F
67	SMALL LETTER G
68	SMALL LETTER H
69	SMALL LETTER I
6A	SMALL LETTER J
6B	SMALL LETTER K
6C	SMALL LETTER L
6D	SMALL LETTER M
6E	SMALL LETTER N
6F	SMALL LETTER O
70	SMALL LETTER P
71	SMALL LETTER Q
72	SMALL LETTER R
73	SMALL LETTER S
74	SMALL LETTER T
75	SMALL LETTER U
76	SMALL LETTER V
77	SMALL LETTER W
78	SMALL LETTER X
79	SMALL LETTER Y
7A	SMALL LETTER Z
7B	LEFT CURLY BRACKET
7C	VERTICAL LINE
7D	RIGHT CURLY BRACKET
7E	TILDE
A0	NO-BREAK SPACE

Hex	Name
A1	INVERTED EXCLAMATION MARK
A2	CENT SIGN
A3	POUND SIGN
A4	CURRENCY SIGN
A5	YEN SIGN
A6	BROKEN BAR
A7	PARAGRAPH SIGN, SECTION SIGN
A8	DIAERESIS
A9	COPYRIGHT SIGN
AA	FEMININE ORDINAL INDICATOR
AB	LEFT ANGLE QUOTATION MARK
AC	NOT SIGN
AD	SOFT HYPHEN
AE	REGISTERED TRADE MARK SIGN
AF	MACRON
B0	RING ABOVE, DEGREE SIGN
B1	PLUS-MINUS SIGN
B2	SUPERSCRIPT TWO
B3	SUPERSCRIPT THREE
B4	ACUTE ACCENT
B5	MICRO SIGN
B6	PILCROW SIGN
B7	MIDDLE DOT
B8	CEDILLA
B9	SUPERSCRIPT ONE
BA	MASCULINE ORDINAL INDICATOR
BB	RIGHT ANGLE QUOTATION MARK
BC	VULGAR FRACTION ONE QUARTER
BD	VULGAR FRACTION ONE HALF
BE	VULGAR FRACTION THREE QUARTERS
BF	INVERTED QUESTION MARK
C0	CAPITAL LATTER A WITH GRAVE
C1	CAPITAL LETTER A WITH ACUTE ACCENT
C2	CAPITAL LETTER A WITH CIRCUMFLEX

Hex	Name
C3	CAPITAL LETTER A WITH TILDE
C4	CAPITAL LETTER A WITH DIAERESIS
C5	CAPITAL LETTER A WITH RING ABOVE
C6	CAPITAL DIPHTHONG A WITH E
C7	CAPITAL LETTER C WITH CEDILLA
C8	CAPITAL LETTER E WITH GRAVE ACCENT
C9	CAPITAL LETTER E WITH ACUTE ACCENT
CA	CAPITAL LETTER E WITH CIRCUMFLEX
CB	CAPITAL LETTER E WITH DIAERESIS
CC	CAPITAL LETTER I WITH GRAVE ACCENT
CD	CAPITAL LETTER I WITH ACUTE ACCENT
CE	CAPITAL LETTER I WITH CIRCUMFLEX
CF	CAPITAL LETTER I WITH DIAERESIS
D0	CAPITAL ICELANDIC LETTER ETH
D1	CAPITAL LETTER N WITH TILDE
D2	CAPITAL LETTER O WITH GRAVE ACCENT
D3	CAPITAL LETTER O WITH ACUTE ACCENT
D4	CAPITAL LETTER O WITH CIRCUMFLEX
D5	CAPITAL LETTER O WITH TILDE
D6	CAPITAL LETTER O WITH DIAERESIS
D7	MULTIPLICATION SIGN
D8	CAPITAL LETTER O WITH OBLIQUE
D9	CAPITAL LETTER U WITH GRAVE ACCENT
DA	CAPITAL LETTER U WITH ACUTE ACCENT
DB	CAPITAL LETTER U WITH CIRCUMFLEX
DC	CAPITAL LETTER U WITH DIAERESIS
DD	CAPITAL LETTER Y WITH ACUTE ACCENT
DE	CAPITAL ICELANDIC LETTER THORN
DF	SMALL GERMAN LETTER SHARP S
E0	SMALL LETTER A WITH GRAVE ACCENT
E1	SMALL LETTER A WITH ACUTE ACCENT

Hex	Name
E2	SMALL LETTER A WITH CIRCUMFLEX
E3	SMALL LETTER A WITH TILDE
E4	SMALL LETTER A WITH DIAERESIS
E5	SMALL LETTER A WITH RING ABOVE
E6	SMALL DIPHTHONG A WITH E
E7	SMALL LETTER C WITH CEDILLA
E8	SMALL LETTER E WITH GRAVE ACCENT
E9	SMALL LETTER E WITH ACUTE ACCENT
EA	SMALL LETTER E WITH CIRCUMFLEX
EB	SMALL LETTER E WITH DIAERESIS
EC	SMALL LETTER I WITH GRAVE ACCENT
ED	SMALL LETTER I WITH ACUTE ACCENT
EE	SMALL LETTER I WITH CIRCUMFLEX
EF	SMALL LETTER I WITH DIAERESIS
F0	SMALL ICELANDIC LETTER ETH
F1	SMALL LETTER N WITH TILDE
F2	SMALL LETTER O WITH GRAVE ACCENT
F3	SMALL LETTER O WITH ACUTE ACCENT
F4	SMALL LETTER O WITH CIRCUMFLEX
F5	SMALL LETTER O WITH TILDE
F6	SMALL LETTER O WITH DIAERESIS
F7	DIVISION SIGN
F8	SMALL LETTER O WITH OBLIQUE STROKE
F9	SMALL LETTER U WITH GRAVE ACCENT
FA	SMALL LETTER U WITH ACUTE ACCENT
FB	SMALL LETTER U WITH CIRCUMFLEX
FC	SMALL LETTER U WITH DIAERESIS
FD	SMALL LETTER Y WITH ACUTE ACCENT
FE	SMALL ICELANDIC LETTER THORN
FF	SMALL LETTER Y WITH DIAERESIS

The following tables assign an identifier to each character of the set. These identifiers are the constant names of each character.

Hex	Constant Name
00	NUL
01	SOH
02	STX
03	ETX
04	EOT
05	ENQ
06	ACK
07	BEL
08	BS
09	HT
0A	LF
0B	VT
0C	FF
0D	CR
0E	SO
0F	SI
10	DLE
11	DC1
12	DC2
13	DC3
14	DC4
15	NAK
16	SYN
17	ETB
18	CAN
19	EM
1A	SUB
1B	ESC
1C	FS or IS4
1D	GS or IS3
1E	RS or IS2
1F	US or IS1

Hex	Constant Name
20	Space
21	Exclamation
22	Quotation
23	Number_Sign
24	Dollar_Sign
25	Percent_Sign
26	Ampersand
27	Apostrophe
28	Left_Parenthesis
29	Right_Parenthesis
2A	Asterisk
2B	Plus_Sign
2C	Comma
2D	Hyphen or Minus_Sign
2E	Full_Stop
2F	Solidus
30	Digit_Zero
31	Digit_One
32	Digit_Two
33	Digit_Three
34	Digit_Four
35	Digit_Five
36	Digit_Six
37	Digit_Seven
38	Digit_Eight
39	Digit_Nine
3A	Colon
3B	Semicolon
3C	Less_Than_Sign
3D	Equals_Sign
3E	Greater_Than_Sign
3F	Question

Hex	Constant Name
40	Commercial_At
41	UC_A
42	UC_B
43	UC_C
44	UC_D
45	UC_E
46	UC_F
47	UC_G
48	UC_H
49	UC_I
4A	UC_J
4B	UC_K
4C	UC_L
4D	UC_M
4E	UC_N
4F	UC_O
50	UC_P
51	UC_Q
52	UC_R
53	UC_S
54	UC_T
55	UC_U
56	UC_V
57	UC_W
58	UC_X
59	UC_Y
5A	UC_Z
5B	Left_Square_Bracket
5C	Reverse_Solidus
5D	Right_Square_Bracket
5E	Circumflex
5F	Low_Line

Hex	Constant Name
60	Grave
61	LC_A
62	LC_B
63	LC_C
64	LC_D
65	LC_E
66	LC_F
67	LC_G
68	LC_H
69	LC_I
6A	LC_J
6B	LC_K
6C	LC_L
6D	LC_M
6E	LC_N
6F	LC_O
70	LC_P
71	LC_Q
72	LC_R
73	LC_S
74	LC_T
75	LC_U
76	LC_V
77	LC_W
78	LC_X
79	LC_Y
7A	LC_Z
7B	Left Curly_Bracket
7C	Vertical Line
7D	Right_Curly_Bracket
7E	Tilde
7F	DEL

Hex	Constant Name
80	Reserved_128
81	Reserved_129
82	BPH
83	NBH
84	Reserved_132
85	NEL
86	SSA
87	ESA
88	HTS
89	HTJ
8A	VTS
8B	PLD
8C	PLU
8D	RI
8E	SS2
8F	SS3
90	DCS
91	PU1
92	PU2
93	STS
94	CCH
95	MW
96	SPA
97	EPA
98	Res
99	Res
9A	Res
9B	CSI
9C	ST
9D	OSC
9E	PM
9F	APC

Hex	Constant Name
A0	No_Break_Space or NBSP
A1	Inverted_Exclamation
A2	Cent_Sign
A3	Pound_Sign
A4	Currency_Sign
A5	Yen_Sign
A6	Broken_Bar
A7	Section_Sign
A8	Diaeresis
A9	Copyright_Sign
AA	Feminine_Ordinal_Indicator
AB	Left_Angle_Quotation
AC	Not_Sign
AD	Soft_Hyphen
AE	Registered_Trade_Mark_Sign
AF	Macron
B0	Degree_Sign or Ring_Above
B1	Plus_Minus_Sign
B2	Superscript_Two
B3	Superscript_Three
B4	Acute
B5	Micro_Sign
B6	Pilcrow_Sign or Paragraph_Sign
B7	Middle_Dot
B8	Cedilla
B9	Superscript_One
BA	Masculine_Ordinal_Indicator
BB	Right_Angle_Quotation
BC	Fraction_One_Quarter
BD	Fraction_One_Half
BE	Fraction_Three_Quarters
BF	Inverted_Question

Hex	Constant Name
C0	UC_A_Grave
C1	UC_A_Acute
C2	UC_A_Circumflex
C3	UC_A_Tilde
C4	UC_A_Diaeresis
C5	UC_A_Ring
C6	UC_AE_Diphthong
C7	UC_C_Cedilla
C8	UC_E_Grave
C9	UC_E_Acute
CA	UC_E_Circumflex
CB	UC_E_Diaeresis
CC	UC_I_Grave
CD	UC_I_Acute
CE	UC_I_Circumflex
CF	UC_I_Diaeresis
D0	UC_Icelandic_Eth
D1	UC_N_Tilde
D2	UC_O_Grave
D3	UC_O_Acute
D4	UC_O_Circumflex
D5	UC_O_Tilde
D6	UC_O_Diaeresis
D7	Multiplication_Sign
D8	UC_O_Oblique_Stroke
D9	UC_U_Grave
DA	UC_U_Acute
DB	UC_U_Circumflex
DC	UC_U_Diaeresis
DD	UC_Y_Acute
DE	UC_Icelandic_Thorn
DF	LC_German_Sharp_S

Hex	Constant Name
E0	LC_A_Grave
E1	LC_A_Acute
E2	LC_A_Circumflex
E3	LC_A_Tilde
E4	LC_A_Diaeresis
E5	LC_A_Ring
E6	LC_AE_Diphthong
E7	LC_C_Cedilla
E8	LC_E_Grave
E9	LC_E_Acute
EA	LC_E_Circumflex
EB	LC_E_Diaeresis
EC	LC_I_Grave
ED	LC_I_Acute
EE	LC_I_Circumflex
EF	LC_I_Diaeresis
F0	LC_Icelandic_Eth
F1	LC_N_Tilde
F2	LC_O_Grave
F3	LC_O_Acute
F4	LC_O_Circumflex
F5	LC_O_Tilde
F6	LC_O_Diaeresis
F7	Division_Sign
F8	LC_O_Oblique_Stroke
F9	LC_U_Grave
FA	LC_U_Acute
FB	LC_U_Circumflex
FC	LC_U_Diaeresis
FD	LC_Y_Acute
FE	LC_Icelandic_Thorn
FF	LC_Y_Diaeresis

ANNEX C

INFORMATIVE REFERENCES

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

[C1] *Information Technology — ISO 7-Bit Coded Character Set for Information Interchange*. International Standard, ISO 646-1991(E). 3rd Ed. Geneva: ISO, 1991.

[C2] *Information Processing — 8-bit Single-Byte Coded Graphic Character Set — Part 1: Latin Alphabet No.1*. International Standard, ISO 8859-1-1987. Geneva: ISO, 1987. [Note: ISO DIS 8859-1:1997 is available]

[C3] *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 or later issue.

[C4] Douglas Steedman. *Abstract Syntax Notation One (ASN.1): The Tutorial and Reference*. Isleworth, U.K.: Technology Appraisals, 1990.

[C5] *Parameter Value Language — A Tutorial*. Report Concerning Space Data Systems Standards, CCSDS 641.0-G-1. Green Book. Issue 1. Washington, D.C.: CCSDS, May 1992 or later issue.

INDEX