INTERNATIONAL STANDARD

ISO/IEC 9637-2

First edition
1992-12-01

Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Data stream binding —

Part 2:
Binary encoding

Technologies de l'information — Infographie — Interfaces pour l'infographie — Interface du flux de données CGI —

Partie 2: Codage binaire

Reference number
ISO/IEC 9637-2:1992(E)

AVA Documents
Document Sharing Hub

Full document details: ISO IEC 09637-2-1992

CONTENTS

Page

Foreword iv

Introduction v

1 Scope 1

2 Normative References 2

3 Definitions 4

4 Overall structure 5
4.1 General form of the data stream 5
4.2 General structure of the Binary Encoding 5
4.3 Encoding functions 7
4.3.1 General structure of the representation header 7
4.3.2 Basic short-form representation header 8
4.3.3 Basic long-form representation header 8
4.3.4 Extended-form representation header 10
4.4 Encoding parameter data 12
4.5 Encoding function response data 12
4.5.1 Structure of response representations 13

5 Binary Encoding primitive data forms 14
5.1 Signed Integer 14
5.1.1 Signed Integer at 8-bit precision 15
5.1.2 Signed Integer at 16-bit precision 15
5.1.3 Signed Integer at 24-bit precision 15
5.1.4 Signed Integer at 32-bit precision 15
5.2 Unsigned Integer 16
5.2.1 Unsigned Integer at 8-bit precision 16
5.2.2 Unsigned Integer at 16-bit precision 16
5.2.3 Unsigned Integer at 24-bit precision 16
5.2.4 Unsigned Integer at 32-bit precision 16
5.3 Octet 17
5.4 Fixed Point Real 17
5.4.1 Fixed Point Real at 32-bit precision 17
5.4.2 Fixed Point Real at 64-bit precision 17
5.4.3 Values of Fixed Point Real 18
5.5 Floating Point Real 18

© ISO/IEC 1992
All rights reserved. No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case Postale 56 • CH-1211 Genève 20 • Switzerland
Printed in Switzerland

Documents
Document Sharing Hub

Full document details: ISO IEC 09637-2-1992

6 Representation of abstract parameter data types 19
6.1 Special Binary Encoding rules 24
6.1.1 Encoding the precision functions 24
6.1.1.1 Binary Encoding precision functions 25
6.1.2 Encoding strings, fixed strings, and data records. . . 28
6.1.2.1 Encoding data record contents 29
6.1.2.2 String parameters and character sets 30
6.1.3 Encoding <input class> input functions 31
6.1.4 INHERITANCE FILTER enumerated values 31
6.1.5 Encoding point lists 33
6.1.6 Encoding transformation matrix components 34
6.1.7 Encoding local colour precision 34
6.1.7.1 Encoding CELL ARRAY and PATTERN TABLE local colour precision 35
6.1.7.2 Encoding PIXEL ARRAY local colour precision . . 35
6.1.8 Colour specifier lists 36
6.1.8.1 Encoding colour specifier lists 37
6.1.8.2 Encoding lists of input colour values and local colour precision 37 38
6.1.9 Encoding PATTERN TABLE and INQUIRE PATTERN colour specifiers. 38

7 Representation of each function and response 39
7.1 Opcode assignments 39
7.1.1 Class code assignments 39
7.1.2 Function id code assignments 41

8 Defaults 62

9 Classification and designation 63
9.1 Conformance 63

A Algorithms and rules for class code assignment 64

B Encoding examples 66

AVA Documents Document Sharing Hub
Full document details: ISO IEC 09637-2-1992

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 9637-2 was prepared by Joint Technical Committee ISO/IEC JTC1, *Information technology.*

ISO/IEC 9637 consists of the following parts, under the general title *Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Data stream binding*

— *Part 1: Character encoding*

— *Part 2: Binary encoding*

Annex A forms an integral part of this part of ISO/IEC 9637. Annex B is for information only.

Documents
Document Sharing Hub

Full document details: ISO IEC 09637-2-1992

Introduction

Purpose

The Binary Encoding of the Computer Graphics Interface (CGI), ISO/IEC 9636, provides a data stream representation of the CGI function syntax that can be optimized for speed of generation and interpretation, while still providing a standard means of interchange among computer systems. The encoding uses binary data formats that are more similar to the data representations used within computer systems than the data formats of the other encodings.

Some of the data formats may exactly match those of some computer systems. On most computer systems processing requirements for the Binary Encoding will be substantially lower than for the other encodings.

In cases where a computer system's architecture does not match the standard formats used in the Binary Encoding, and where absolute minimization of processing requirements is critical, and where interchange among dissimilar systems does not matter, it may be more appropriate to use a private encoding, conforming to the rules specified in ISO/IEC 9636-1.

Objectives

This encoding has the following features:

a) Partitioning of parameter lists: function/response representations are coded in the Binary Encoding by one or more partitions (see clause 4); the first (or only) partition of a representation contains the opcode (class code and id code);

b) Alignment of function representations and response representations: every function/response representation begins on a 16-bit boundary. Alignment of representations which follow partitions that require an odd number of 8-bit entities may require a partition to be padded with an 8-bit entity with all bits zero;

c) Uniformity of format: all function representations and response data records have an associated parameter length value. As a result, it is possible to ignore function representations which are not supported by the interpreter;

d) Efficiency of encoding parameter data: parameter data such as coordinates, indexes and colours are encoded as one or more 8-bit entities. The precision of every parameter is determined by the appropriate default precision or as set by a precision setting CGI function;

AVA Documents Document Sharing Hub
Full document details: ISO IEC 09637-2-1992

e) Extensibility: the arrangement of opcode class and id values has been designed to allow future growth;

f) Format of real data: real numbers are encoded using either IEEE floating point representation or a fixed-point representation;

g) Run length encoding option: if many adjacent colours have the same value, efficient encoding is possible. For each run a cell count is specified followed by the colour (or colour index);

h) Packed list encoding option: if adjacent colours do not have the same value, bit-stream lists are provided in which the values are packed as closely as possible;

i) Encoding of soliciting functions: the assignment of opcodes to functions which require a response has been designed so that all such functions can be recognized by a CGI interpreter;

j) Response Data: responses to soliciting functions have been assigned different opcodes from their associated soliciting functions. However, the response opcode can be derived in a straightforward manner from the soliciting function opcode;

k) Lists of data: there is a standard technique for representing lists of any type of data (with a few specific exceptions);

Relationship to other standards

This encoding is guided by the same objectives as the Computer Graphics Metafile Binary Encoding, ISO/IEC 8632-3:1992. For each CGI function which is identical in both semantics and parameterization to a CGM element, the encoding will be identical. That is, the opcodes will be identical and the parameters will use the same data type and appear in the same order. The extension mechanism defined in this encoding is also compatible with the CGM Binary Encoding.

The floating point representation of real data in this part of the Standard is that in ANSI/IEEE 754-1986.

The representation of character data in this part of the Standard follows the rules of ISO 646 and ISO 2022.

For certain functions and response data, the CGI defines parameter value ranges as being reserved for registration. The values and their meanings will be defined using the procedures established in ISO TR 9973.

AVA Documents
Document Sharing Hub

Full document details: ISO IEC 09637-2-1992

Information technology — Computer graphics — Interfacing techniques for dialogues with graphical devices (CGI) — Data stream binding —

Part 2:
Binary encoding

1 Scope

This part of ISO/IEC 9637 specifies a Binary Encoding of the Computer Graphics Interface (CGI) data stream. For each of the function syntaxes in clause 5 and clause 6 of ISO/IEC 9636-2, ISO/IEC 9636-3, ISO/IEC 9636-4, ISO/IEC 9636-5, and ISO/IEC 9636-6, an encoding is specified in terms of an opcode and a sequence of parameters of specified data types. For each of these data types, an explicit representation in terms of bits, 8-bit and 16-bit entities is specified. For some data types, the exact representation depends on a type and/or precision for the data as used in the data stream.

The Binary Encoding of the Computer Graphics Interface data stream will, in many circumstances, reduce the effort required to generate and interpret the data stream as compared to other encodings.

Documents
Document Sharing Hub
Full document details: ISO IEC 09637-2-1992

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9637. At the time of publication, the editions indicated were valid. All standards are subject to revisions, and parties to agreements based on this part of ISO/IEC 9637 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 646:1991, *Information technology – ISO 7-bit coded character set for information interchange.*

ISO 2022:1986, *Information processing – ISO 7-bit and 8-bit coded character sets – Code extension techniques.*

ISO/IEC 7942:1985/Amd.1:1991, *Information processing systems – Computer graphics – Graphical Kernel System (GKS) functional description – Amendment 1*

ISO 8632-1:1992, *Information technology – Computer graphics – Metafile for the storage and transfer of picture description information – Part 1: Functional specification.*

ISO 8632-3:1992, *Information technology – Computer graphics – Metafile for the storage and transfer of picture description information – Part 3: Binary encoding.*

ISO/IEC 9636-1:1991, *Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices(CGI) – Functional specification – Part 1: Overview, profiles and conformance.*

ISO/IEC 9636-2:1991, *Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices(CGI) – Functional specification – Part 2: Control.*

ISO/IEC 9636-3:1991, *Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices(CGI) – Functional specification – Part 3: Output.*

ISO/IEC 9636-4:1991, *Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices(CGI) – Functional specification – Part 4: Segments.*

ISO/IEC 9636-5:1991, *Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices(CGI) – Functional specification – Part 5: Input and echoing.*

AVA Documents
Document Sharing Hub
Full document details: ISO IEC 09637-2-1992

Normative References

ISO/IEC 9636-6:1991, *Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices(CGI) – Functional specification – Part 6: Raster.*

ISO/IEC 9637-1:1992, *Information technology – Computer graphics – Interfacing techniques for dialogues with graphical devices(CGI) – Data stream binding – Part 1: Character encoding.*

ISO/IEC TR 9973:1988, *Information technology – Computer graphics – Procedures for registration of graphical items.*

ANSI/IEEE 754, *Standard for Binary Floating Point Arithmetic.*

Documents
Document Sharing Hub
Full document details: ISO IEC 09637-2-1992

3 Definitions

3.1 representation: Portion of a binary-encoded function or response representation that contains the opcode (function/response class plus function/response id) and parameter length information. (See 4.3.4.)

3.2 octet: 8-bit entity in which all bits are significant. The bits are numbered from 7 (most significant) to 0 (least significant).

3.3 word: 16-bit entity in which all bits are significant. The bits are numbered from 15 (most significant) to 0 (least significant).

3.4 word-aligned: An entity is word-aligned when it begins on a word (16-bit) boundary within the data stream.

NOTE - Within this part of ISO/IEC 9637, the terms "octet", "word", and "word-aligned" have specific meanings. These meanings may not match those of a particular computer system on which this encoding of the data stream is used.

Documents
Document Sharing Hub
Full document details: ISO IEC 09637-2-1992

4 Overall structure

This encoding specifies representations for each of the CGI functions of ISO/IEC 9636 as well as any associated responses. A function representation is the encoded representation of a function with its *In* parameters. A response representation is the encoded representation of the *Out* parameters of a soliciting function.

4.1 General form of the data stream

All function representations in the data stream are encoded using a uniform scheme. These are represented as variable length data structures, each consisting of opcode information (function class plus function id) designating the particular function representation, the length of its parameter data and finally the parameter data itself (if any).

All response representations in the return data stream are encoded using the same uniform scheme. These are represented as variable length data structures, each consisting of opcode information (response class plus response id) designating the particular response representation, the length of its return parameter data and finally the data itself.

4.2 General structure of the Binary Encoding

The octet is the fundamental unit of organization of the binary data stream. The Binary Encoding of the CGI data stream is a logical data structure consisting of a sequential collection of octets. Fields of two different sizes are defined within the Binary Encoding structure. These two sizes correspond to the octet, an 8-bit field, and the word, a 16-bit field. These fields are used in the remainder of this part of ISO/IEC 9637 for illustrating the contents and structure of function representations and parameters. The parameter list length of a function or response representation is expressed as a number of octets.

To optimize processing of the binary data stream on a wide collection of computers, data stream function and response representations are required to consist of an even number of octets. This forces the alignment of representations in the Binary Encoding data stream to word boundaries. It is necessary to pad a representation with null octets or bits to the word boundary if the parameter data does not fill to such a boundary. This padding of an extra octet or bits does not affect any parameter list length counts within a representation.

Parameter data can be organized into subgroups called partitions. Partitions are used to accommodate parameter data larger than that supported by the basic long-form representation, as described below. Partitions need not begin on a word boundary.

AVA Documents Document Sharing Hub
Full document details: ISO IEC 09637-2-1992

The bits of an octet are numbered 0 to 7, with 7 being the most significant. The bits of a word are numbered 0 to 15, with 15 being the most significant.

```
                b7                b0
octet:          |_|_|_|_|_|_|_|_|
             msb                lsb

                b15               b8 b7               b0
word:           |_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
             msb                                   lsb
```

Figure 1 - Bit numbering for octets and words

If the consecutive bits of the binary data structure are numbered 1..*N*, the consecutive octets are numbered 1..*N*/8 (rounded to next integer), and the consecutive words are numbered 1..*N*/16 (rounded to next integer), then the logical correspondence of bits, octets, and words in the binary data structure is as illustrated in table 1.

Table 1 - Binary data structure

Data stream bit number	Octet bit number	Word bit number
1	b7/octet1	b15/word1
.	.	.
.	.	.
8	b0/octet1	b8/word1
9	b7/octet2	b7/word1
.	.	.
.	.	.
16	b0/octet2	b0/word1
17	b7/octet3	b15/word2
.	.	.
.	.	.
24	b0/octet3	b8/word2
25	b7/octet4	b7/word2
.	.	.
.	.	.

Documents
Document Sharing Hub

Full document details: ISO IEC 09637-2-1992

4.3 Encoding functions

The function descriptions in clause 5 and clause 6, as well as the formal grammars, of ISO/IEC 9636-2, ISO/IEC 9636-3, ISO/IEC 9636-4, ISO/IEC 9636-5, and ISO/IEC 9636-6, provide the basic syntax needed to encode each CGI function. They contain the function name and the input and output parameters, along with their abstract data types and value ranges. The order in which the parameters are specified in clause 5 and clause 6 is significant. This is the exact order in which they will be encoded in a representation with very few exceptions. These special parameter encoding cases are specifically detailed. (See 6.1.)

The Binary Encoding scheme has two basic components: the representation header and the parameter data. The representation header provides information concerning the function opcode (function class and function id) and the amount (in octets) of parameter data that is being supplied in the function representation. There are one or more function representation headers for each function representation in the data stream. There may be no parameter data, a fixed amount of parameter data or an indefinite amount of parameter data depending on the function being represented.

CGI functions and their *In* parameters, if any, are encoded as function representations consisting of one or more representation headers followed by any *In* parameter data. Functions with *Out* parameters are called soliciting functions. Their *Out* parameters are considered response data and are encoded in a separate response representation, consisting of one or more representation headers followed by the *Out* parameter data. (See 4.5.)

The formats of the various function and response representations and the rules for their encoding into the binary data stream are described below.

4.3.1 General structure of the representation header

Representations in the Binary Encoding have four forms - basic short-form representations, basic long-form representations, extended short-form representations and extended long-form representations. The forms differ in format with regard to the amount of parameter data accommodated and the number of function class and function id values accommodated. Specifically, the differences are:

-- the short-form representation always contains the complete function representation, including its parameter list data. The long-form representation is used to supply an indefinite amount of parameter data through the use of data partitions;

-- the short-form representation only accommodates parameter list data up to 30 octets in length. The long-form representation accommodates lengths up to 32767 octets per data partition with an indefinite number of partitions.

-- the basic form representations accommodate only 14 function classes and 127 function ids. The extended form representations accommodate an unlimited number of function classes and function ids.

AVA Documents Document Sharing Hub

Full document details: ISO IEC 09637-2-1992

The representation forms also differ in the format of the representation headers. There are specific fields, unique to each type of representation header, that identify the type of representation header. There are also differences in how the function opcode (function class and function id) and the parameter list length are encoded.

4.3.2 Basic short-form representation header

For the basic short-form, the representation header consists of a single word divided into three fields: function class, function id and parameter list length.

	15 14 13 12	11 10 9 8 7 6 5	4 3 2 1 0
Word 1	func. class	function id	parm list len

Figure 2 - Format of a basic short-form representation header.

The fields in the basic short-form representation header are as follows:

bits 15-12 function class (value range 0-14)

bits 11-5 function id (value range 0-127)

bits 4-0 parameter list length: the number of octets of parameter data that follow for this command (value range 0-30)

4.3.3 Basic long-form representation header

The representation header of the basic long-form representation consists of two words. The first word of the basic long-form representation has the same structure as the first word of the basic short-form representation with the difference that the parameter list length field contains the binary value 11111 (decimal 31). This value indicates that the header is a basic long-form representation header rather than a basic short-form representation header. The second word contains the length of the following parameter data partition and a flag that indicates if the parameter data partition is followed by another data partition or not.

Documents
Document Sharing Hub

Full document details: ISO IEC 09637-2-1992

An indefinite number of parameter partitions can be accommodated by the long-form representation. When the partition flag, bit 15 of the second word of the header has the value 1, it indicates that there will be another parameter data partition to follow. Each subsequent data partition of the function representation is preceded by a word composed of a partition flag and the partition's parameter list length in the same format as the second word of the representation header. The final data partition of a function representation is indicated by the partition flag being set to zero.

The parameter list length supplied in each of the parameter data partitions specifies the length of that partition in octets and not the length of the complete function representation data. Partitions need not start on a word boundary, therefore no padding octet needs to be appended to the data in a data partition which contains an odd number of octets for its parameter list length.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 1	func class				function id							1	1	1	1	1
Word 2	P	parameter list length														

Figure 3 - Format of a basic long-form representation header.

The fields in the basic long-form representation header are as follows:

Word 1

bits 15-12 function class (value range 0-14)

bits 11-5 function representation id (value range 0-127)

bits 4-0 binary value 11111 (decimal 31) indicating long-form

Word 2

bit 15 P, partition flag
- 0 if 'final' partition
- 1 if 'not final' partition

bits 14-0 parameter list length: the number of octets of parameter data that follow for this partition (value range 0-32767).

AVA Documents Document Sharing Hub

Full document details: ISO IEC 09637-2-1992

4.3.4 Extended-form representation header

The extended-form representation header is composed of one or more extender headers followed by either a basic short-form or a basic long-form representation header. If the last header is a basic short-form header, the representation header is referred to as an extended short-form representation header. If the last header is a basic long-form header, the representation header is referred to as an extended long-form representation header.

The first extender header is recognized by having a binary value of 1111 (decimal 15) in the field normally containing the function class code in a basic header, bits 12 through 15. Each succeeding extender header also has this value in this field. The extender header is also composed of an extension field selector (FS), a final extension header flag (FE), extension data, and a six bit constant with the value of zero. A value of 1 for FE, bit 10 of the extender header, indicates that there are more extender headers to follow. A value of 0 for FE indicates that this is the final extender header and the next word in the data stream will be one of the basic representation headers. A value of 0 for FS, bit 11 of the extender header, indicates that the extension data is to be used in constructing the function class code. A value of 1 for FS indicates that the extension data is to be used in constructing the function id code.

An extended function class is constructed from the extender headers by appending the bits in the extension data to the bits in the class code already accumulated. As each extender header that composes a particular extended function class is encountered in the data stream, the extension data bits are accumulated. That is, the higher order bits are encoded in the extender headers encountered earlier in the data stream. The bits in the class code field of the short-form or long-form basic function respresentation header trailing the last extender header are appended to the accumulated function class value to form the final function class code. An extended function id code is constructed in the same manner.

Extender headers for constructing function class and function id codes can be transmitted in any order relative to one another as long as the higher-order-bits-first rule is followed. That is, it does not matter if the extender headers forming a function class code are transmitted intermixed with the extender headers forming a function id code.

Documents
Document Sharing Hub

Full document details: ISO IEC 09637-2-1992

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 1	1	1	1	1	FS	FE	extension data					0	0	0	0	0

Figure 4 - Format of an extender header.

The fields in the extender header are as follows.

bits 15-12 binary value 1111 (decimal 15) indicating an extender header

bit 11 FS, extension field selector
- 0 if extension data applies to class code
- 1 if extension data applies to id code

bit 10 FE, final extension header flag
- 0 if 'final' extender header
- 1 if 'not final' extender header

bits 9-5 extension data

bits 4-0 binary value 00000

Note that while the function class code field of a representation header normally uses the binary value 1111 to indicate an extender header, an exception is made when the value of this field is in the basic header representation that follows a set of extender headers. This is the only means of specifying a function class with a binary value of 1111 (decimal 15) in bits 15-12. (See clause B.8.)

Also note that the use of extender headers to specify the class and id codes will not necessarily result in a unique Binary Encoding of a particular representation opcode. Refer to annex B for examples.

Theoretically, this extension technique is infinitely extensible, however, for this part of ISO/IEC 9637, a CGI interpreter will be expected to parse only a limited number of extender headers. Specifically, no CGI interpreter will be expected to accumulate more than two extension data fields for extending the class code. This allows for 16384 possible class code values. Likewise, no CGI interpreter will be expected to accumulate more than two extension data fields for extending the id code and no assigned id code will exceed 65535 (16-bits in size).

AVA Documents Document Sharing Hub

Full document details: ISO IEC 09637-2-1992

4.4 Encoding parameter data

When present in a function or response representation, the parameter values follow the parameter list length portion of a representation header for each representation form. The parameter list length is determined by the number of parameters and the type and precision of the parameters. Parameter values have the format illustrated in clause 5 of this part of ISO/IEC 9637. The possible parameter types are as specified in 5.2.10 of ISO/IEC 9636-1. Parameters are encoded in a function representation in exactly the same order as they are listed in a CGI function description in clause 5 of ISO/IEC 9636-2, ISO/IEC 9636-3, ISO/IEC 9636-4, ISO/IEC 9636-5, and ISO/IEC 9636-6. There are a few exceptions to this rule. These special cases are described in 6.1.

Every representation is constrained to begin on a word boundary. This necessitates padding a representation with a single null octet at the end if it contains an odd number of octets. In addition, in function representations with parameters whose precisions are shorter than one octet (i.e. those containing a 'local colour precision' parameter) it is necessary to pad the low order bits of the last data-containing octet with null bits if the data does not fill the octet. In all cases, the parameter list length is the count of octets actually containing parameter data - it does not include the padding octet if one is present. It is only at the end of a representation that padding is performed, with the exception of the function representations and response representations which encode their colour specifier list using the run length list mode or the packed list mode. (See 6.1.7.)

4.5 Encoding function response data

CGI functions which have *Out* parameters are called soliciting functions. These functions return the *Out* parameter values in associated response representations. The function id code for a soliciting function and its associated response representation are identical. The class code for each of these differs only in bit 4 of the accumulated class code. In the response representation, this bit has a value of 1; in the function representation, this bit has a value of 0. The response class code is derived by adding decimal 16 to the soliciting function class code.

All soliciting CGI functions have as their first return parameter a response validity value. When the response validity is INVALID, no additional return parameters need to be encoded in the response representation. If any other return parameters are present in the response representation, they should be regarded as undefined.

AVA Documents Document Sharing Hub

Full document details: ISO IEC 09637-2-1992

4.5.1 Structure of response representations

Response representations have exactly the same structure and encoding rules for their headers and parameter data as function representations. Both soliciting function representations and response representations can always be recognized as such by their assigned class codes. All soliciting functions will have bit 5=1 and bit 4=0 in their class code. All responses will have bit 5=1 and bit 4=1 in their class code. (See annex A.)

Documents Document Sharing Hub

Full document details: ISO IEC 09637-2-1992

5 Binary Encoding primitive data forms

The Binary Encoding uses five primitive data forms to represent the various abstract data types used to describe parameters in ISO/IEC 9636-2 through 6. The primitive data forms and the symbols used to represent them are as follows.

SI	Signed Integer
UI	Unsigned Integer
O	Octet
FX	Fixed Point Real
FP	Floating Point Real

Each of these primitive data forms (except Octet) may be used in a variable number of precisions. CGI functions have been defined to effect changes in these precisions. (See 6.1.1.)

The following terms are used in the following diagrams when displaying the form of numeric values.

msb	Most significant bit
lsb	Least significant bit
S	Sign bit (0 = non-negative; 1 = negative)

where the sign bit (S) is the high order bit for signed integer values.

The primitive data forms in the following diagrams are illustrated for the case that the parameter begins on a data stream word boundary. In general, parameters may align on odd or even octet boundaries because they can be preceded by an odd or even number of octets of other parameter data. Function representations containing a local colour precision parameter may have parameters shorter than one octet. It is possible in such cases that the individual parameters will not align on octet boundaries. (See 6.1.7.)

5.1 Signed Integer

Signed Integer data forms are represented in two's complement format. Four precisions are available for Signed Integers: 8-bit, 16-bit, 24-bit and 32-bit.

Documents
Document Sharing Hub
Full document details: ISO IEC 09637-2-1992

5.1.1 Signed Integer at 8-bit precision

Each value occupies one octet. The range is -128 ≤ *value* ≤ 127. The sign bit is 1 for negative values.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	*value i*						lsb	S	*value i+1*						lsb

5.1.2 Signed Integer at 16-bit precision

Each value occupies two successive octets.
The range is -32,768 ≤ *value* ≤ 32,767. The sign bit is 1 for negative values.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 1	S	*value i*														lsb
Word 2	S	*value i+1*														lsb

5.1.3 Signed Integer at 24-bit precision

Each value occupies three successive octets.
The range is -8,388,608 ≤ *value* ≤ 8,388,607. The sign bit is 1 for negative values.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 1	S	*value i*														
Word 2	*value i*							lsb	S	*value i+1*						
Word 3	*value i+1*															lsb

5.1.4 Signed Integer at 32-bit precision

Each value occupies four successive octets.
The range is -2,147,483,648 ≤ *value* ≤ 2,147,483,647. The sign bit is 1 for negative values.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 1	S	*value i*														
Word 2	*value i*															lsb
Word 3	S	*value i+1*														
Word 4	*value i+1*															lsb

Documents
Document Sharing Hub
Full document details: ISO IEC 09637-2-1992

5.2 Unsigned Integer

Four precisions may be specified for Unsigned Integer data forms: 8-bit, 16-bit, 24-bit and 32-bit.

5.2.1 Unsigned Integer at 8-bit precision

Each value occupies one octet. The range is 0 ≤ *value* ≤ 255.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
msb			*value i*				lsb	msb		*value i+1*					lsb

5.2.2 Unsigned Integer at 16-bit precision

Each value occupies two successive octets.
The range is 0 ≤ *value* ≤ 65,535.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
msb						*value*									lsb

5.2.3 Unsigned Integer at 24-bit precision

Each value occupies three successive octets.
The range is 0 ≤ *value* ≤ 16,777,215.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 1	msb						*value i*									
Word 2			*value i*					lsb	msb			*value i+1*				
Word 3							*value i+1*									lsb

5.2.4 Unsigned Integer at 32-bit precision

Each value occupies four successive octets.
The range is 0 ≤ *value* ≤ 4,294,967,295.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 1	msb						*value i*									
Word 2							*value i*									lsb

Documents
Document Sharing Hub

Full document details: ISO IEC 09637-2-1992

5.3 Octet

Each Octet primitive data form occupies a single octet.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Octet i								*Octet i+1*							

5.4 Fixed Point Real

Fixed Point Real data forms are composed of a single Signed Integer data form followed by a single Unsigned Integer data form. The first integer represents the whole part and the second integer represents the fractional part. Two precisions may be specified for Fixed Point Reals: 32-bit or 64-bit.

5.4.1 Fixed Point Real at 32-bit precision

Each value occupies 4 successive octets. The first 2 octets form a 16-bit Signed Integer data form. The remaining 2 octets form a 16-bit Unsigned Integer data form.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 1	S							Whole part								lsb
Word 2	msb							Fraction part								lsb

5.4.2 Fixed Point Real at 64-bit precision

Each value occupies 8 successive octets. The first 4 octets form a 32-bit Signed Integer data form. The remaining 4 octets form a 32-bit Unsigned Integer data form.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 1	S							Whole part								
Word 2								Whole part								lsb
Word 3	msb							Fraction part								
Word 4								Fraction part								lsb

AVA Documents Document Sharing Hub
Full document details: ISO IEC 09637-2-1992

5.4.3 Values of Fixed Point Real

The values of the represented real numbers are given by:

$$\text{for 32 bits:} \quad \text{real value} = SI + \left\{\frac{UI}{2^{16}}\right\}$$

$$\text{for 64 bits:} \quad \text{real value} = SI + \left\{\frac{UI}{2^{32}}\right\}$$

SI stands for the whole part and UI stands for the fractional part in these equations. SI, the whole part, is the largest integer less than or equal to the real number being represented.

5.5 Floating Point Real

Floating Point Real data forms are represented in the floating point format of ANSI/IEEE standard 754. This format contains three parts:

- a sign bit;
- a biased exponent part;
- a fraction part.

Two precisions may be specified for Floating Point Reals: 32-bit or 64-bit. The 32-bit precision occupies 4 successive octets. The 64-bit precision occupies 8 successive octets.

Documents
Document Sharing Hub

Full document details: ISO IEC 09637-2-1992