INTERNATIONAL STANDARD

ISO/IEC 14957

Second edition
2010-12-15

Information technology — Representation of data element values — Notation of the format

Technologies de l'information — Représentation des valeurs des éléments de données — Notation du format

Reference number
ISO/IEC 14957:2010(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. 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.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 14957 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

This second edition cancels and replaces the first edition (ISO/IEC 14957:1996), which has been technically revised.

Introduction

Data interchange is experiencing rapid expansion in the commercial, technical and public sectors. It gives rise to interworking between different communities which have often independently developed information processing applications and telecommunication networks to meet specific needs. Hence, the overall situation suffers from a lack of homogeneity.

In order to remedy this situation, an urgent standardization effort focused in particular on the representation of data elements is necessary.

The representation of a data element supposes in the first place that the format, i.e. the types of character used in the representation and the length of the representation, is specified. In order that these specifications have the same significance for everyone involved, it is necessary to express them in accordance with standardized conventions.

Such rules are likely to eliminate any and all risk of ambiguity, lack of understanding and error; they also facilitate the comparison of data element dictionaries, the design and creation of information systems, and electronic data interchange (EDI).

These notations have been partially and variously expressed in different International Standards according to the specific contexts in which they have been defined, e.g. EDIFACT (ISO 9735), banking standards (such as ISO 7982-1), character sets (ISO/IEC 8859), information processing (ISO 6093), and programming languages (ISO/IEC 9899).

The objective of this International Standard is to provide a unique source of reference on this subject for all standards utilizing these type of notations independently of their environments.

Information technology — Representation of data element values — Notation of the format

1 Scope

This International Standard specifies the notation to be used for stating the format, i.e. the character classes, used in the representation of data elements and the length of these representations. It also specifies additional notations relative to the representation of numerical figures. For example, this formatting technique might be used as part of the metadata for data elements.

The scope of this International Standard is limited to graphic characters, such as digits, letters and special characters. The scope is limited to the basic datatypes of characters, character strings, integers, reals, and pointers.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 10646:2003, *Information technology — Universal Multiple-Octet Coded Character Set (UCS)*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
data element
unit of data that is considered, in context, to be indivisible

[ISO/IEC 2382-4:1999, 04.07.01]

3.2
character set
finite set of characters that is complete for a given purpose

[ISO/IEC 2382-4:1999, 04.01.02]

3.3
character type
set of characters of the same kind or having the same use

EXAMPLE Letters, figures, special characters, etc.

3.4
length
length of representation
number of characters used to represent a data element

4 Notation relative to character types and length of representation of a data element

The format shall be a characterstring sequence. The format is composed of zero or more directives: one or more space characters as defined in ISO/IEC 10646:2003, Clause 20, an ordinary character (neither `%` nor a space character), or a conversion specification. Each conversion specification is introduced by the character `%`.

NOTE This specification of formats is based upon the "fscanf()" function in the C programming language (ISO/IEC 9899:1999).

Conceptually, the format string implies a syntax processor that consumes syntactic units, as described by the format directives. Because there is no service interface specified by this International Standard, there is no prescribed error handling for strings that do not match the formatting directives.

After the `%`, the following appear in sequence:

— an optional assignment-suppressing character `*`;

— an optional nonzero decimal integer that specifies the maximum field width (in characters);

— an optional length modifier that specifies the size of the receiving object;

— a conversion specifier character that specifies the type of conversion to be applied.

Each directive of the format is processed in turn.

A directive composed of space character(s) is executed by reading input up to the first non-space character (which remains unread), or until no more characters can be read.

A directive that is an ordinary character is executed by reading the next characters of the stream. If any of those characters differ from the ones composing the directive, the directive fails and the differing and subsequent characters remain unread.

Similarly, if end-of-file, an encoding error, or a read error prevents a character from being read, the directive fails.

A directive that is a conversion specification defines a set of matching input sequences, as described below for each specifier. A conversion specification is executed in the following steps:

Input space characters are skipped, unless the specification includes a `[`, `c`, or `n` specifier.[1]

An input item is read from the stream, unless the specification includes an `n` specifier. An input item is defined as the longest sequence of input characters which does not exceed any specified field width and which is, or is a prefix of, a matching input sequence.

The first character, if any, after the input item remains unread. If the length of the input item is zero, the execution of the directive fails. This condition is a matching failure unless end-of-file, an encoding error, or a read error prevented input from the stream, in which case it is an input failure.

Except in the case of a `%` specifier, the input item (or, in the case of a `%n` directive, the count of input characters) is converted to a type appropriate to the conversion specifier. If the input item is not a matching sequence, the execution of the directive fails: this condition is a matching failure. Unless assignment suppression was indicated by a `*`, the result of the conversion is placed in the object pointed to by the first argument following the format argument that has not already received a conversion result. If this object does not have an appropriate type, or if the result of the conversion cannot be represented in the object, the behavior is undefined.

1 These space characters are not counted against a specified field width.

The conversion specifiers and their meanings are as follows.

— `d` matches an optionally signed decimal integer, whose format is the same as expected for the subject sequence of the strtol() function with the value 10 for the base argument. Example: "%d" matches "17", but not "1.0", "17H", "ABC"; "%03d" matches "017", "000", "017", but not "1000". Note: "%d0" will not match any string.

— `i` matches an optionally signed integer, whose format is the same as expected for the subject sequence of the strtol() function with the value 0 for the base argument. Example: "%i" matches "+17", "17", "-17", but not "1.0", "+17H".

— `o` matches an optionally signed octal integer, whose format is the same as expected for the subject sequence of the strtoul() function with the value 8 for the base argument. Example: "%o" matches "0123", "-456", but not "0789".

— `u` matches an optionally signed decimal integer, whose format is the same as expected for the subject sequence of the strtoul() function with the value 10 for the base argument. Example: "%u" matches "17", but not "+17", "-17".

— `x` matches an optionally signed hexadecimal integer, whose format is the same as expected for the subject sequence of the strtoul() function with the value 16 for the base argument. Example: "%x" matches "013EF", "-013ef", but not "013EFG".

— `a,e,f,g` match[2] an optionally signed floating-point number, infinity, or NaN, whose format is the same as expected for the subject sequence of the strtod() function. The corresponding argument shall be a pointer to floating. Example: "%f" matches "1.2E10", "+1.2E+10", "-1.2e-10", "infinity, "nan" but not "1.2F10", "nan".

— `c` matches a sequence of characters of exactly the number specified by the field width (1 if no field width is present in the directive). If length modifier is present, the input shall be a sequence of characters. Examples: "%c" matches "X" but not "XY"; "%2c" matches "XY".

— `s` matches a sequence of non-space characters. If length modifier is present, the input shall be a sequence of characters. Examples: "%s" matches "123abc", "123abcd", "" but not "123 abc".

— `[` matches a nonempty sequence of characters from a set of expected characters (the scanset). The conversion specifier includes all subsequent characters in the format string, up to and including the matching right bracket (`]`). The characters between the brackets (the scanlist) compose the scanset, unless the character after the left bracket is a circumflex (^), in which case the scanset contains all characters that do not appear in the scanlist between the circumflex and the right bracket. If the conversion specifier begins with [] or [^], the right bracket character is in the scanlist and the next following right bracket character is the matching right bracket that ends the specification; otherwise the first following right bracket character is the one that ends the specification. If a - character is in the scanlist and is not the first, nor the second where the first character is a ^, nor the last character, the behavior is implementation-defined. Examples: "abc[ad-f]" matches "abca", "abcd", "abce", "abcf", but not "abc", "abcg"; "abc[^d-f]" matches "abcg" but not "abc", "abcd".

— `%` matches a single `%` character; no conversion occurs. The complete conversion specification shall be `%%`.

2 There is no distinction among `a`, `e`, `f`, and `g` formats.

Annex A
(informative)

EBNF grammar for data element description

The following syntax is based upon ISO/IEC 14977, *Information technology — Syntactic metalanguage — Extended BNF*.

```
format_description = { [non_format_component] , format_component } ,
                     [non_format_component];

non_format_component = { non_format_characters };

non_format_characters = ( ascii - "%" ) | "%%" ;

format_component = signed_decimal_format |
                   signed_integer_format |
                   signed_octal_format |
                   unsigned_integer_format |
                   signed_hexadecimal_format |
                   character_array_format |
                   character_string_format |
                   defined_character_set_format |
                   real_number_format ;

decimal_with_optinal_leading_zero = decimal_numeral , { decimal_numeral } ;

decimal_numeral = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ;

signed_decimal_format = "%" , [ "+" | "-" ] ,
                   [  "*" | decimal_with_optional_leading_zero ] , "d" ;

signed_integer_format = "%" , [ "+" | "-" ] ,
                   [  "*" | decimal_with_optional_leading_zero ] , "i" ;

signed_octal_format = "%" , [ "+" | "-" ] ,
                   [  "*" | decimal_with_optional_leading_zero ] , "o" ;

unsigned_integer_format = "%" , [ "+" | "-" ] ,
                   [  "*" | decimal_with_optional_leading_zero ] , "u" ;

signed_hexadcimal_format = "%" , [ "+" | "-" ] ,
                   [  "*" | decimal_with_optional_leading_zero ] , "x" ;

character_array_format = "%" ,
                   [  "*" | decimal_with_optional_leading_zero ] , "c" ;

character_string_format = "%" ,
                   [  "*" | decimal_with_optional_leading_zero ] , "s" ;

defined_character_set_format = "%" ,
                   [  "*" | decimal_with_optional_leading_zero ] ,
                   "[", { ascii - "]" } , "]" ;

real_number_format = "%" , [ "+" | "-" ] ,
                   [  "*" | decimal_with_optional_leading_zero ] ,
                   ( "a" | "e" | "f" | "g" );
```