ISO/IEC FDIS 14651~~:2025(en)~~

ISO/IEC JTC 1/SC 2

Secretariat: JISC

Date: 2025-~~02-13~~03-27

Information technology — International string ordering and comparison — Method for comparing character strings and description of the common template tailorable ordering

Technologies de ~~l'information~~ l'information — Classement international et comparaison de chaînes de caractères — Méthode de comparaison de chaînes de caractères et description du modèle commun et adaptable d'ordre de classement

~~Seventh edition, 2025~~

FDIS stage

ISO ~~Copyright Office~~copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: + 41 22 749 01 11
~~Email:~~ E-mail: copyright@iso.org
Website: www.iso.org

Published in Switzerland~~.~~

~~CONTENTS~~

Contents

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Field Code Changed

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

Field Code Changed

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 2, *Coded character sets*.

This seventh edition cancels and replaces the sixth edition (ISO/IEC 14651:2020), which has been technically revised.

The main changes are as follows:

— ~~—~~the Common Template Tables are now referenced externally, removing the need for frequent update to this document.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

This document provides a method, applicable around the world, for ordering text data, and references Common Template Tables which, when tailored, can meet a given language's ordering requirements while retaining reasonable ordering for other scripts.

Common Template Tables require some tailoring in different local environments. Conformance to this document requires that all deviations from these templates, called "deltas", be declared to document resultant discrepancies.

This document describes a method to order text data independently of context.

ISO/IEC 30112 has specifications for ordering that informatively complement the specifications in this document and indicates where additional information can be sought on ordering keywords defined in this document.

Information technology — International string ordering and comparison — Method for comparing character strings and description of the common template tailorable ordering

1 Scope

This document defines a reference comparison method. This method is applicable to two or more character strings to determine their collating order in a sorted list. The method can be applied to strings containing characters from the full repertoire of ISO/IEC 10646. This method is also applicable to subsets of that repertoire to produce ordering results valid (after tailoring) for a given set of languages for each script. This method uses collation tables derived either from the Common Template Tables (CTT) referenced by this document or from one of their tailoring. The format of the Common Template Table is described using the Backus-Naur Form (BNF). The format is used normatively within this document.

This document also defines syntax elements to tailor these Common Template Tables used by the reference comparison method. Furthermore, it defines requirements for a declaration of the differences (delta) between a collation table and a given Common Template Table including the tailoring elements.

These Common Template Tables describe an order for all characters encoded in the current and past ISO/IEC 10646 editions, including amendments. They allow for a specification of a fully deterministic ordering. These tables enable the specification of a string ordering adapted to local ordering rules, without requiring an implementer to have knowledge of all the different scripts already encoded in the Universal Coded Character Set (UCS).

All these Common Template Tables have reference names which are related to a particular stage of development of the ISO/IEC 10646 Universal coded character set or a particular version of the Unicode Standard. These names and their relationship with ISO/IEC 10646 or the Unicode Standard repertoire are specified by an externally referenced document: Unicode Technical Standard, UTS #10, Unicode Collation Algorithm.

This document does not:

— mandate a specific comparison method; any equivalent method giving the same results is acceptable;

— mandate a specific format for describing or tailoring tables in a given implementation;

— mandate specific symbols to be used by implementations;

— mandate any specific internal format for intermediate keys used when comparing, nor for the table used. The use of numeric keys is not mandated either;

— mandate a context-dependent ordering;

— mandate any particular preparation of character strings prior to comparison.

NOTE 1 It is typical to do preparation of character strings prior to comparison even if it is not prescribed by this document (see Annex C Annex C).

NOTE 2 Annex D Annex D describes problems that gave way to this document with their anticipated solutions.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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, *Information technology — Universal coded character set (UCS)*

UNICODE TECHNICAL STANDARD. *Unicode Technical Standard #10, Unicode Collation Algorithm*. :https://www.unicode.org/reports/tr10/

COMMON TEMPLATE TABLES. *Unicode Technical Standard UTS #10, Unicode Collation Algorithm, Appendix B: Synchronization with ISO/IEC 14651*: :https://www.unicode.org/reports/tr10/#Synch_ISO14651

3 Terms and definitions

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

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1 ~~3.1~~
character string
sequence of characters considered as a single object

Note 1 to entry: A character string to be ordered does not normally include the characters that delimit it, as for example an "end of line" control character in a text file to be sorted.

3.2 ~~3.2~~
collating symbol
symbol (3.12(~~3.12~~)) used to specify weights assigned to a *collating element* (3.4(~~3.4~~))

3.3 ~~3.3~~
collation table
weighting table
mapping from *collating elements* (3.4(~~3.4~~)) to *weighting elements* (3.14(~~3.14~~))

3.4 ~~3.4~~
collating element
sequence of one or more characters that are considered a single entity for *ordering* (3.7(~~3.7~~))

3.5 ~~3.5~~
delta
list of the differences between a given *collation table* (3.3(~~3.3~~)) and another one

Note 1 to entry: The given collation table, together with a given delta, forms a new collation table.

Note 2 to entry: Unless otherwise specified in this document, the term "delta" always refers to differences from a Common Template Table as defined in this document.

3.6 ~~3.6~~
level
collation level
sequence number for a *subkey* (3.11~~(3.11)~~) in the series of subkeys forming a key

3.7 ~~3.7~~
ordering
collation
process by which, given two strings, it is determined whether the first one is less than, equal to, or greater than the second one

3.8 ~~3.8~~
ordering key
sequence of *subkeys* (3.11~~(3.11)~~) used to determine an order

3.9 ~~3.9~~
preparation
collation preparation
process in which given *character strings* (3.1~~(3.1)~~) are mapped to (other) character strings before the calculation of the *ordering key* (3.8~~(3.8)~~) for each of the strings

3.10 ~~3.10~~
reference comparison method
method for establishing an order between two *ordering keys* (3.8~~(3.8)~~)

Note 1 to entry: See Clause 6~~Clause 6~~.

3.11 ~~3.11~~
subkey
sequence of weights computed for a *character string* (3.1~~(3.1)~~)

3.12 ~~3.12~~
symbol
collating element (3.4~~(3.4)~~)

3.13 ~~3.13~~
weight
collation weight
positive integer value, used in *subkeys* (3.11~~(3.11),~~), reflecting the relative order of *collating elements* (3.4~~(3.4)~~)

3.14 ~~3.14~~
weighting element
list of a given number of weights sequentially ordered by level

4 Symbols and conventions

Following ISO/IEC 10646, characters are referenced as UX where X stands for a series of one to eight hexadecimal digits (where all the letters in the hexadecimal string are in upper case) and refers to the value of that character in ISO/IEC 10646. This convention is used throughout this document.

Any use of the term "The Common Template Table" in this document is applicable to all instances of Common Template Tables referenced in Annex A~~Annex A~~. The term can also be abbreviated as CTT.

In the Common Template Table, arbitrary symbols representing weights are used according to the BNF notation description in 6.3.16.3.1.

5 Conformance

A process is conformant to this document if it produces results identical to those that result from the application of the specifications given in 6.26.2 to 6.56.5.

A declaration of conformity to this document shall be accompanied by a statement, either directly or by reference, of the following:

— the name of the Common Template Table used by the process;

— the number of levels that the process supports; (this number shall be at least three;);

— whether the process supports the forward position processing parameter;

— whether the process supports the backward processing parameter and at which level;

— the tailoring delta described in 6.46.4 and how many levels are defined in the delta;

— if a preparation process is used, the method used shall be declared.

It is the responsibility of implementers to show how their delta declaration is related to the table syntax described in 6.36.3, and how the comparison method they use, if different from the one mentioned in Clause 6Clause 6, can be considered as giving the same results as those prescribed by the method specified in Clause 6Clause 6. The use of a preparation process is optional, and its details are not specified in this document.

It is strongly recommended that processes present available tailoring options to users.

6 String comparison

6.1 Preparation of character strings prior to comparison

Preparation is necessary only for either modification and/or duplication, or both, of original strings to render them context-independent prior to the comparison phase Although not part of the Scopescope of this document, preparation can be an important part of the ordering process. See Annex CAnnex C for some examples of preparation.

Characters of the input string shall be encoded according toin accordance with ISO/IEC 10646 (UCS) or a mapping to ISO/IEC 10646 shall be provided if another encoding scheme is used.

Therefore, it can be an important part of the preparation phase to map characters from a non-UCS encoding scheme to the UCS for input to the comparison method. This task can, amongst other things, encompass the correct handling of escape sequences in the originating encoding scheme, the mapping of characters without an allocated UCS codepoint to an application-defined codepoint in the private zone area and change the sequence of characters in strings that are not stored in logical order. For example, for visual order Arabic code sets, input strings shall be put into logical order; and for some bibliographic code sets, strings with combining accents stored before their respective base character require that the combining accents be put after their base character. The resulting string sequence may then have to be remapped into its original encoding scheme.

The Common Template Table is designed so that combining sequences and corresponding single characters (precomposed) will have precisely the same ordering. To avoid inadvertently breaking this invariant (and in the process breaking Unicode conformance), tailoring should reorder combining sequences when

corresponding precomposed characters are reordered. For example, if Ä is reordered after Z, then the sequence <A>+<combining diaeresis> should also be reordered. To avoid exposing encoding differences that can be invisible to the end-user, it is recommended that strings be normalized according to Unicode normalization form NFD to achieve this equivalence[11 ~~[11].~~].

Escape sequences and control characters constitute very sensitive data to interpret, and it is highly recommended that preparation should filter out or transform these sequences.

NOTE Since the reference method is a logical statement for the mechanism for string comparison, it does not preclude an implementation from using a non-UCS character encoding only, as long as it produces results as if it were using the reference comparison method.

6.2 Key building and comparison

6.2.1 Preliminary considerations

6.2.1.1 Assumptions

The collation table is a mapping from collating elements to weighting elements. In each weighting element, four levels are described in the Common Template Table. This number of levels can be extended or reduced, but cannot be less than ~~3~~three, in tailoring.

In the Common Template Table, levels generally have the following characteristics, although this purpose is not absolute.

— Level 1: This level generally corresponds to the set of common letters of the alphabets for that script, if the script is alphabetic, and to the set of common characters of the script if the script is ideographic or syllabic.

— Level 2: This level generally corresponds to diacritical marks affecting each basic character of the script. For some languages, letters with diacritics are always considered an integral part of the basic letters of the alphabet and are not considered at this second level, but rather at the first. For example, in Spanish, N TILDE is considered a basic letter of the Latin script. Therefore, tailoring for Spanish will change the definition of N TILDE from "the weight of an N in the first level and the weight of a TILDE in the second level" to "the weight of an N TILDE (placed after N and before O) in the first level, and indication of the absence of a diacritic in the second level". For some characters, variant letter shapes are also dealt with on level 2. An example of this is ß, the LATIN SMALL LETTER SHARP S, which is treated as equivalent to ss on level 1, but traditionally distinguished from it on level 2.

— Level 3: This level generally corresponds to case distinctions (upper and lower case) or to distinctions based on variant letter shapes (like the distinction between Hiragana and Katakana).

— Level 4: This level generally corresponds to weighting differences that are less significant than those at the other levels. Often the last level (level 4 in the Common Template Table) is intended to specify additional weighting for "special" characters, i.e. characters normally not part of the spelling of words of a language (such as dingbats, punctuation, etc.), sometimes called "ignorable" characters in the context of computerized ordering.

6.2.1.2 Processing properties

A given tailored table has specific scanning and ordering properties. These properties ~~could~~can have been changed by the tailoring.

A scanning direction (forward or backward) for each level is used to indicate how to process the string. The scanning direction is a global property of each level defined in the tailored table.

If the last level is greater than three, there is an optional property of this level of comparison called "position" option: when active, a comparison on the numeric position of each "ignorable" character in the two strings is effected, before comparing their weights. In other words, for two strings equivalent at all levels except the last one, the string having an ignorable character in the lowest position comes before the other one. In the event that corresponding ignorable characters occur in the same position, their weights are considered, until a difference is found. Support for this kind of processing is optional and is not necessary to claim conformance to this document.

NOTE The scanning direction (forward or backward) is not normally related to the natural writing direction of scripts. The scanning direction applies to the logical sequence of the coded character string.

According to ISO/IEC 10646, for scripts written right to left, such as Arabic, the first characters in the logical sequence correspond to the rightmost characters in their natural presentation sequence. Conversely, for the Latin script, written left to right, the first characters in the logical sequence correspond to the leftmost characters in their natural presentation sequence.

Scanning forward starts with the lowest position in the logical sequence, while scanning backward starts from the highest position, independently of the presentation sequence. The scanning direction for ordering purposes is a global property of each level described in the table.

In ISO/IEC 10646, the Arabic script is artificially separated into two pseudo-scripts: 1) the logical, intrinsic Arabic, coded independently of contextual shapes, and 2) the Arabic presentation forms. Both allow the complete coding of Arabic, but intrinsic Arabic is normally preferred for better processing, while presentation-form Arabic is preferred by some presentation-oriented applications. ISO/IEC 10646 does not prescribe that the presentation forms be stored in any specific order, and in some implementations, the storage order for the latter is the reverse of the storage order used for intrinsic Arabic. It is therefore advisable that the preparation phase be used to make sure that Arabic presentation forms and other Arabic characters be fed to the comparison method in logical order.

A tailored sort table can be separated into sections for ease of tailoring. Each section is then assigned a name consistent with the specification in 6.3.1~~6.3.1.~~ One of the tailoring possibilities is to assign a given order to each section and to change the relative order of an entire section relative to other sections.

6.2.2 Reference ordering key formation

6.2.2.1 General

When two strings are to be compared to determine their relative order, the two strings are first parsed into a sequence of collating elements taking into account the multi-character "collating-element" statements declared and used in a tailored table (if the syntax of 6.3.2~~6.3.2~~ is used). For the syntax used for expressing the Common Template Table, the name of a collating element consisting of a single character, is formed by the UCS value of the character, expressed as a hexadecimal string, prefixed with "U". For multi-character collating elements, the name and association to characters can be found via the collating elements declarations.

NOTE Collating elements with more characters have preference over shorter ones. As an example, if a multicharacter collating element is defined for "abc" and another one is defined for "ab" or for "bc", then if "abc" is encountered, the collating element for "abc" ~~will apply~~applies and not the one for "ab" or "bc".

Then, a sequence of *m* intermediary subkeys is formed out of a character string, where *m* is the number of levels described in a tailored collation weighting table.

Each ordering key is a sequence of subkeys. Each subkey is a list of numeric weights. A subkey is formed by successively appending the list of the weights assigned, at the level of the subkey, to each collating element of the string. The keyword "IGNORE" in the Common Template Table at the place of a sequence of collating symbols at a level indicates that the sequence of weights at that level for that collating element is an empty sequence of weights.

6.2.2.2 Weighting elements to be ignored

When forming a sort key, collating elements ignored at the first level or at the two first levels and that follow a collation element ignored at all levels but the last one, do not keep their weights as defined in the common template table (or a tailored table); each of these weights shall be zeroed (this means that "IGNORE" shall be assigned to each non-nil weight).

6.2.2.3 Implicit weights computing

If there is no entry in the tailored table for a character of the input string, then the character's weights are undefined. In this case, one shall compute an "implicit" primary weight consisting of a pair of 16bit words — call them "aaaa" and "bbbb" — and assume that lines like the following were added to the weighting table:

<UXXXX> "<R{aaaa$_{16}$}><T{bbbb$_{16}$}>";<BASE>;<MIN>;<SFFFF>

NOTE 1 <SFFFF> (at the last level) is the largest level 1 weight in the Common Template Table.

Distinctions are needed among characters with no entry in the weighting table, and implicit primary weights are thus computed. A -"base ~~weight'~~weight" is assigned to these characters, and specific functions are used to derive the "aaaa" and 'bbbb" values. These "base weight" values and functions are defined in the Unicode Technical Standard #10 (see clause 2~~clause 2). ~~). The following shows examples for some of these "base weight" and functions for specific characters and how they are used to determine the "aaaa" and "bbbb" for a given code point cp:

EXAMPLE 1

CJK Unified Ideographs

base_weight = 0xFB40 for original CJK Unified Ideographs (URO)

base_weight = 0xFB80 for CJK Extension A through Extension G

aaaa = [base_weight + (cp >> 15)]$_{16}$

bbbb = [(cp & 0x7FFF) | 0x8000]$_{16}$

EXAMPLE 2

Tangut Ideographs and Components

base_weight = 0xFB00

aaaa = [base_weight]$_{16}$

bbbb = [(cp – 0x17000) | 0x8000] $_{16}$

EXAMPLE 3

Nüshu Characters

base_weight = 0xFB01

aaaa = [base_weight]$_{16}$

bbbb = [(cp – 0x1B170) | 0x8000]$_{16}$

EXAMPLE 4

Khitan Small Script Characters

base_weight = 0xFB02

aaaa = $[base\_weight]_{16}$

bbbb = $[(cp - 0x18B00) \mid 0x8000]_{16}$

Thus, the value of the code point is used to calculate the desired weights, by operating on individual bits. The two numbers, converted to four-character hexadecimal values, will then take the following form: "<Raaaa><Tbbbb>".

Decomposable characters are excluded from these computing, as they have an entry in the Common Template Table (with dual primary weights).

If a string contains ill-formed code unit sequences, two approaches show up: the sequence can be handled as if it were UFFFD (REPLACEMENT CHARACTER), or each subsequence can be ignored. The same approaches can be adopted for any out-of-range values ($cp > 10FFFF_{16}$).

NOTE 2 When computed weights are used, characters without explicit mappings are sorted in code point order within each set they belong to and they sort properly with respect to the rest of the characters.

6.2.2.4 Subkeys formation

There are three ways of forming subkeys: subkeys formed using the "forward" processing parameter, subkeys formed using the "backward" processing parameter, and subkeys formed using the "forward, position" processing parameter. Subkeys that use the "position" option can only occur at the last level, and only if that level is greater than three. Support of the "position" option is not required for conformance. If the processing parameter "forward, position" is not supported, "forward, position" shall be interpreted as if the processing parameter had been "forward".

6.2.2.5 Formation of subkeys for the first three levels

With the "forward" or "forward, position" processing parameter, during (forward) scanning of each collating element of the input character string, one or more weights are obtained. These weights are obtained by matching the collating element in the given tailored collation weighting table, obtaining the list of weights assigned to the collating element at the particular level. The obtained weight list is appended to the end of the subkey.

Subkeys, at a particular level, formed with the "backward" level processing parameter are built by forming a subkey as with the "forward" parameter, then reversing that subkey weight by weight.

6.2.2.6 Formation of the level 4 subkey

At the last level, the subkey is built as described in the preceding paragraph. However, once the ordering key is completely formed (when the end of the string is reached), there are two possible approaches:

a) ~~a)~~ With the "forward" processing parameter: all the <SFFFF> symbols are backed out of the subkey;

b) ~~b)~~ With the "forward, position" processing parameter: any trailing sequence of <SFFFF> symbol(s) shall be removed from the subkey (leaving intact the remaining part of the subkey).