INTERNATIONAL STANDARD

ISO/IEC 14651

Fourth edition
2016-02-15

Corrected version
2016-05-01

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

Technologies de 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

Reference number
ISO/IEC 14651:2016(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

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.

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).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

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

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 2, *Coded character sets*.

This fourth edition cancels and replaces the third edition (ISO/IEC 14651:2011), which has been technically revised.

This corrected version of ISO/IEC 14651:2016 incorporates the following corrections: in Annex C, Thai characters have been corrected.

Introduction

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

The Common Template Table requires some tailoring in different local environments. Conformance to this International Standard requires that all deviations from the template, called "deltas", be declared to document resultant discrepancies.

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

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

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

1 Scope

This International Standard defines the following.

— A reference comparison method. This method is applicable to two 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, such as those of the different ISO/IEC 8-bit standard character sets, or any other character set, standardized or not, 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 Table defined in this International Standard or from one of its tailorings. This method provides a reference format. The format is described using the Backus-Naur Form (BNF). This format is used to describe the Common Template Table. The format is used normatively *within* this International Standard.

— A Common Template Table. A given tailoring of the Common Template Table is used by the reference comparison method. The Common Template Table describes an order for all characters encoded in the Unicode 8.0 standard, included in ISO/IEC 10646:2014 and its Amendment 1. It allows for a specification of a fully deterministic ordering. This table enables 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).

NOTE 1 This Common Template Table is to be modified to suit the needs of a local environment. The main worldwide benefit is that, for other scripts, often no modification is required and the order will remain as consistent as possible and predictable from an international point of view.

NOTE 2 The character repertoire used in this International Standard is equivalent to that of the Unicode Standard version 6.0.

— A reference name. The reference name refers to this particular version of the Common Template Table, for use as a reference when tailoring. In particular, this name implies that the table is linked to a particular stage of development of the ISO/IEC 10646 Universal coded character set.

— Requirements for a declaration of the differences (delta) between the collation table and the Common Template Table.

This International Standard does *not* mandate the following.

— A specific comparison method; any equivalent method giving the same results is acceptable.

— A specific format for describing or tailoring tables in a given implementation.

— Specific symbols to be used by implementations, except for the name of the Common Template Table.

— Any specific user interface for choosing options.

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

— A context-dependent ordering.

— Any particular preparation of character strings prior to comparison.

NOTE It is normally necessary to do preparation of character strings prior to comparison even if it is not prescribed by this International Standard (see Annex C).

Although no user interface is required to choose options or to specify tailoring of the Common Template Table, conformance requires always declaring the applicable delta, a declaration of differences with this table. It is recommended that processes present available tailoring options to users.

2 Conformance

A process is conformant to this International Standard if it produces results identical to those that result from the application of the specifications given in 6.2 to 6.5.

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

— 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 6.4 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 6.3, and how the comparison method they use, if different from the one mentioned in Clause 6, can be considered as giving the same results as those prescribed by the method specified in Clause 6. The use of a preparation process is optional and its details are not specified in this International Standard.

3 Normative references

The following referenced documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 10646:2014, *Information technology — Universal Coded Character Set (UCS)*

ISO/IEC 10646:2014/Amd.1:2015, *Information technology — Universal Coded Character Set (UCS) / Amendment 1: Cherokee supplement and other characters*

4 Terms and definitions

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

4.1
character string
sequence of characters considered as a single object

4.2
collating symbol
symbol (4.12) used to specify weights assigned to a *collating element* (4.4)

4.3
collation (weighting) table
mapping from *collating elements* (4.4) to *weighting elements* (4.14)

4.4
collating element
sequence of one or more characters that are considered a single entity for *ordering* (4.7)

4.5
delta
list of the differences between a given *collation table* (4.3) 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 International Standard, the term "delta" always refers to differences from the Common Template Table as defined in this International Standard.

4.6
(collation) level
sequence number for a *subkey* (4.11) in the series of subkeys forming a key

4.7
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

4.8
ordering key
sequence of *subkeys* (4.11) used to determine an order

4.9
(collation) preparation
process in which given *character strings* (4.1) are mapped to (other) character strings before the calculation of the *ordering key* (4.8) for each of the strings

4.10
reference comparison method
method for establishing an order between two *ordering keys* (4.8)

Note 1 to entry: See Clause 6.

4.11
subkey
sequence of weights computed for a *character string* (4.1)

4.12
symbol
collating element (4.4)

4.13
(collation) weight
positive integer value, used in *subkeys* (4.11), reflecting the relative order of *collating elements* (4.4)

4.14
weighting element
list of a given number of weights sequentially ordered by level

5 Symbols and abbreviations

Following ISO/IEC 10646, characters are referenced as U*X* 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 International Standard.

In the Common Template Table, arbitrary symbols representing weights are used according to the BNF notation description in 6.3.1.

6 String comparison

6.1 Preparation of character strings prior to comparison

It may be necessary to transform character strings before the reference comparison method is applied to them (see Annex C for an example of such preparation). Although not part of the Scope of this International Standard, preparation may be an important part of the ordering process. See Annex C for some examples of preparation.

Where applicable, 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 may be invisible to the end-user, it is recommended that strings be normalized according to Unicode normalization form NFD to achieve this equivalence (see Unicode Technical Report no. 15).

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, in tailoring.

NOTE 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 may 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 case corresponding ignorable characters are at the same position, then 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 International Standard.

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 may be separated into sections for ease of tailoring. Each section is then assigned a name consistent with the specification in 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

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 6.3.1 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 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.

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`".

If there is no entry in the tailored table for a character of the input string, then the character's weights are undefined. Characters with undefined weights should be ordered, with respect to characters that have defined weights, as if the undefined ones were given the weight named "UNDEFINED" at the first level. If there is no weight assignment to the symbol "UNDEFINED" before the symbol <SFFFF>'s weight assignment in a given tailored table, then the table shall be interpreted as if "UNDEFINED" was weighted just before <SFFFF>. The ordering of characters with undefined weights with respect to other characters with undefined weights is not specified in this International Standard.

NOTE 1 A possible way to order characters with undefined weights is *as if* there were tailoring lines like this one added to the table, in UCS code point order (call the maximal level 4 weight <PLAIN> here):

<UXXXX> "<UNDEFINED><UXXXX>";<BASE>;<MIN>;<PLAIN>

NOTE 2 <SFFFF> is the maximal level 1 weight in the Common Template Table.

6.2.2.1 Formation of a subkey with the "`forward`" level processing parameter

Subkeys, at a particular level, formed with the "`forward`" level processing parameter, are built in the following way:

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.

6.2.2.2 Formation of a subkey with the "`backward`" level processing parameter

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.

6.2.2.3 Formation of a subkey with the "`forward,position`" level processing parameter

Subkeys, at the last level, formed with the "`forward,position`" level processing parameter are formed by forming a subkey as with the "`forward`" parameter, but for collating elements that are not ignored at all levels but the last one, their last level weighting (list of weights) is replaced by a single weight (call it <PLAIN> here) that is larger than all other weights at the last level in the given tailored table. Collating elements that are ignored at all levels but the last one, retain their weighting according to the given tailored table. Finally, any trailing sequence of the maximal weight (<PLAIN>) is removed from the subkey, effectively replacing each trailing maximal weight with a zero weight.

NOTE For any level, implementations are allowed to apply an order-preserving reduction of all subkeys at that level. Such a reduction is useful for levels 2, 3, and 4. Level 2 often has long stretches of the weight named <BASE> in Annex A. Level 3 often has long stretches of the weight named <MIN> in Annex A. Level 4 often has long stretches of the weight named <PLAIN> here. One such ordering preserving subkey reduction technique effectively encodes, in the last level subkey, the (relative) position, as single number each, of each otherwise ignored character; hence the name of the "*`position`*" option.

6.2.3 Reference comparison method for ordering character strings

The reference comparison method for ordering two given character strings (*after* collation preparation, which is not part of the reference comparison method itself) is to compare ordering keys generated by the reference key formation method described in 6.2.2.

— Begin by building an ordering key, using a given tailored collation weighting table, for each of the two given character strings being compared.

— Then compare the resulting keys according to the key ordering definition below in this subclause. Keys can be compared either up to a given level, or up to the last level of the given tailored collation weighting table.

NOTE 1 The comparison may be made *while* generating the ordering keys for two strings to be compared, stopping the key generation when the order of the strings can be determined. Such a technique is sometimes termed lazy evaluation, and some systems support it by default. This avoids generating the full ordering key when an ordering difference may be found early in the keys. When a bigger set of strings are to be ordered, it may be advisable to generate the ordering keys, and store each key or an initial segment of each, before comparing the keys.

Weights for different levels should not be compared, which implies that subkeys at different levels should not be compared, nor should keys generated from different tailored tables be compared.

NOTE 2 This allows implementations to assign weightings at each level independently of the other levels, and independently of other tailorings.

m is the maximal level of a given tailored table. Recall that a key is a list, of length m, of subkeys; a subkey is a list of weights; and a weight is a positive integer. Other notations used below are the following:

— L_z is the length of the subkey z, i.e., the number of weights in the subkey;

— $z_{\mathrm{wt(a)}}$, where $1 \le a \le \mathrm{L}_z$, is the weight at index position a (an integer > 0) of the subkey z.;

— $u_{\mathrm{sk(b)}}$, where $1 \le b \le m$, is the subkey at level b (an integer > 0) of the key u.

The orderings of weights, subkeys, and ordering keys (up to a given level, or up to the last level) are total order relations, defined for a given tailored collation table as follows.

a) Weights are positive integers (in the reference method) and are compared as such for the purposes of collation.

b) A subkey v is *less than* a subkey w (written $v < w$) **if and only if** there exists an integer, i, where $1 \le i \le \mathrm{L}_v + 1$ and $i \le \mathrm{L}_w$, such that

— $i = 1$ and $v_{\mathbf{wt}(i)} < w_{\mathbf{wt}(i)}$, or

— for all integers, j, where $1 \le j < i$, the equality $v_{\mathbf{wt}(j)} = w_{\mathbf{wt}(j)}$ holds, and either

  — $i \le L_v$ and $v_{\mathbf{wt}(i)} < w_{\mathbf{wt}(i)}$, or

  — $i = L_v+1$ and $0 < w_{\mathbf{wt}(i)}$.

A subkey v is *greater than* a subkey w (written $v > w$) **if and only if** w is less than v. A subkey v is *equal to* a subkey w (written $v = w$) **if and only if** neither v is less than w, nor w is less than v.

c) An ordering key x is *less than* an ordering key y at level s (written $x <_s y$) **if and only if** there exists an integer i, where $1 \le i \le s$ and $i \le m$, such that

— $i = 1$ and $x_{\mathbf{sk}(i)} < y_{\mathbf{sk}(i)}$, or

— for all integers j, where $1 \le j < i$, the equality $x_{\mathbf{sk}(j)} = y_{\mathbf{sk}(j)}$ holds, and $x_{\mathbf{sk}(i)} < y_{\mathbf{sk}(i)}$.

An ordering key x is *greater than* an ordering key y at level s (written $x >_s y$) **if and only if** y is less than x at level s. An ordering key x is *equal to* an ordering key y at level s (written $x =_s y$) **if and only if** neither x is less than y at level s, nor y is less than x at level s.

d) For ordering keys, $<$, $>$, and $=$ are defined as $<_m$, $>_m$, and $=_m$ respectively.

NOTE 3 For ordering keys, $x <_t y$ implies $x <_{t+1} y$, $x >_t y$ implies $x >_{t+1} y$, $x =_t y$ implies $x =_{t-1} y$, $x <_0 y$ is false, $x >_0 y$ is false, and $x =_0 y$ is true. Above level m, for a given tailored table, there are no further ordering distinctions. Note that this definition implies that if two ordering keys are in the 'less than' relationship at level 1, they will also be in the 'less than' relationship at levels 2, 3, 4, etc. In general, whenever two ordering keys are less than at a given level, they will also automatically be less than at all subsequent, higher levels. Conversely, if two ordering keys are equal at a given level, they will also automatically be equal at all preceding, lower levels.

6.3 Common Template Table: Formation and interpretation

This subclause specifies the following:

— the syntax used to form the Common Template Table in Annex A or a tailored table based upon the Common Template Table as expressed in Annex A;

— conditions of well-formedness of a table using this syntax;

— interpretation of tailoring statements in deltas for tables formed using this syntax;

— evaluation from symbols to weights of tailored tables formed using this syntax;

— conditions for considering two tables as equivalent;

— conditions for considering comparison results as equivalent.

6.3.1 BNF syntax rules for the Common Template Table in Annex A

Definitions between <angle brackets> make use of terms not defined in this BNF syntax and assume general English usage.

Other conventions:

*	indicates 0 or more repetitions of a token or a group of tokens;
+	indicates 1 or more repetitions of a token or a group of tokens;
?	indicates optional occurrence of a token or a group of tokens (0 or 1 occurrences);
	parentheses are used to group tokens;
	production rules are terminated by a semicolon.

Define collation tables as sequences of lines:

```
weight_table =  common_template_table | tailored_table ;
common_template_table =
                simple_line+ ;
tailored_table = table_line+ ;
```

Define the line types:

```
simple_line =     (symbol_definition | collating_element |
                   weight_assignment | order_end)? line_completion ;
table_line =      simple_line | tailoring_line ;
tailoring_line =  (reorder_after | order_start | reorder_end |
                   section_definition | reorder_section_after)
                  line_completion ;
```

Define the basic syntax for collation weighting:

```
symbol_definition =
                  'collating-symbol' space+ symbol_element ;
symbol_element = symbol | symbol_range ;
symbol_range =   symbol '..' symbol ;
symbol =         simple_symbol | ucs_symbol ;
ucs_symbol =     ('<U'  one_to_eight_digit_hex_string  '>') |
                 ('<U-' one_to_eight_digit_hex_string '>') ;
simple_symbol =  '<' identifier '>' ;
collating_element =
                 'collating-element' space+ symbol space+
                 'from' space+ quoted_symbol_sequence ;
quoted_symbol_sequence =
                 '"' simple_weight+ '"' ;
weight_assignment =
                 simple_weight | symbol_weight ;
simple_weight =  symbol_element | 'UNDEFINED' ;
symbol_weight =  symbol_element space+ weight_list ;
weight_list =    level_token (semicolon level_token)* ;
level_token =    symbol_group | 'IGNORE' ;
symbol_group =   symbol_element | quoted_symbol_sequence ;
order_end =      'order_end' ;
```


Define the tailoring syntax:

```
reorder_after =  'reorder-after' space+ target_symbol ;
target_symbol =  symbol ;
order_start =    'order_start' space+ multiple_level_direction ;
multiple_level_direction =
                 (direction semicolon)* direction (',position')? ;
direction =      'forward' | 'backward' ;
reorder_end =    'reorder-end' ;
section_definition =
                 section_definition_simple |
                 section_definition_list ;
section_definition_simple =
                 'section' space+ section_identifier ;
section_identifier =
                 identifier ;
section_definition_list =
                 'section' space+ section_identifier space+
                 symbol_list ;
symbol_list =    symbol_element (semicolon symbol_element)* ;
reorder_section_after =
                'reorder-section-after' space+ section_identifier
                 space+ target_symbol ;
```

Define low-level tokens used by the rest of the syntax:

```
identifier =     (letter | digit) id_part* ;
id_part =         letter | digit | '-' | '_' ;
line_completion =
                 space* comment? EOL ;
comment =        comment_char character* ;
one_to_eight_digit_hex_string =
                 hex_upper |   hex_upper hex_upper |
                 hex_upper hex_upper hex_upper |
```