Technical Reports

L2/08-058

Proposed Update Unicode Standard Annex #14

UNICODE LINE BREAKING ALGORITHM

Version	Unicode 5.1 Draft 2
Authors	Asmus Freytag (asmus@unicode.org), Andy Heninger (andy.heninger@gmail.com)
Date	2008-01-15
This Version	http://www.unicode.org/reports/tr14/tr14-21.html
Previous Version	http://www.unicode.org/reports/tr14/tr14-20.html
Latest Version	http://www.unicode.org/reports/tr14/
Revision	21

Summary

This annex presents the Unicode line breaking algorithm along with detailed descriptions of each of the character classes established by the Unicode line breaking property. The line breaking algorithm produces a set of "break opportunities", or positions that would be suitable for wrapping lines when preparing text for display. A model implementation using pair tables is also provided.

Status

*This is a **draft** document which may be updated, replaced, or superseded by other documents at any time. Publication does not imply endorsement by the Unicode Consortium. This is not a stable document; it is inappropriate to cite this document as other than a work in progress.*

> ***A Unicode Standard Annex (UAX)** forms an integral part of the Unicode Standard, but is published online as a separate document. The Unicode Standard may require conformance to normative content in a Unicode Standard Annex, if so specified in the Conformance chapter of that version of the Unicode Standard. The version number of a UAX document corresponds to the version of the Unicode Standard of which it forms a part.*

Please submit corrigenda and other comments with the online reporting form [Feedback]. Related information that is useful in understanding this annex is found in Unicode Standard Annex #41, "Common References for Unicode Standard Annexes." For the latest version of the Unicode Standard, see [Unicode]. For a list of current Unicode Technical Reports, see [Reports]. For more information about versions of the Unicode Standard, see [Versions].

Contents

1 Overview and Scope

The text of The Unicode Standard [Unicode] presents a limited description of some of the characters with specific functions in line breaking, but does not give a complete specification of line breaking behavior. This annex provides more detailed information about default line breaking behavior reflecting best practices for the support of multilingual texts.

For most Unicode characters, considerable variation in line breaking behavior can be expected, including variation based on local or stylistic preferences. For that reason, the line breaking properties provided for these characters are informative. Some characters are intended to explicitly influence line breaking. Their line breaking behavior is therefore expected to be identical across all implementations. As described in this annex, the Unicode Standard assigns normative line breaking properties to those characters. The Unicode Line Breaking Algorithm is a tailorable set of rules that uses these line breaking properties in context to determine line break opportunities.

This annex opens with formal definitions, a summary of the line breaking task and the context in which it occurs in overall text layout followed by a brief section on conformance requirements. Three main sections follow:

- *Section 5, Line Breaking Properties*, contains a narrative description of the line breaking

behavior of the characters of the Unicode Standard, grouping them in alphabetical order by line breaking class.

- *Section 6,* Line Breaking Algorithm, provides a set of rules listed in order of precedence that constitute a line breaking algorithm.
- *Section 7,* Pair Table-Based Implementation, describes an efficient pair table-based implementation of the algorithm.

The final two sections discuss issues of customization and implementation.

- *Section 8,* Customization, provides a discussion of how to tailor the algorithm.
- *Section 9,* Implementation Notes, provides additional information to implementers using regular expression-based techniques or requiring legacy support for combining marks.

2 Definitions

All terms not defined here shall be as defined in the Unicode Standard [Unicode5.0]. The notation defined in this annex differs somewhat from the notation defined elsewhere in the Unicode Standard. All other notation used here without an explicit definition shall be as defined elsewhere in the Unicode Standard.

LD1 Line Fitting:
The process of determining how much text will fit on a line of text, given the available space between the margins and the actual display width of the text.

LD2 Line Break: The position in the text where one line ends and the next one starts.

LD3 Line Break Opportunity: A place where a line is allowed to end.

- Whether a given position in the text is a valid line break opportunity depends on the context as well as the line breaking rules in force.

LD4 Line Breaking:
The process of selecting one among several line break opportunities such that the resulting line is optimal or ends at a user-requested explicit line break.

LD5 Line Breaking Property: A character property with enumerated values, as listed in *Table 1*, and separated into normative and informative values.

- Line breaking property values are used to classify characters and, taken in context, determine the type of line break opportunity.

LD6 Line Breaking Class: A class of characters with the same line breaking property value.

The Line Breaking Classes are described in *Section 5.1,* Description of Line Breaking Properties.

LD7 Mandatory Break: A
line must break following a character that has the mandatory break property.

Such a break is also known as a *forced* break and is indicated in the rules as **B** !, where **B** is the character with the mandatory break property.

LD8 Direct Break:
A line break opportunity exists between two adjacent characters of the given line breaking classes.

A direct break is indicated in the rules below as **B** ÷ **A**, where **B** is the character class of the character *before* and **A** is the character class of the character *after* the break. If they are separated by one or more space characters, a break opportunity exists instead after the last space. In the pair table, the optional space characters are not shown.

LD9 Indirect Break:
A line break opportunity exists between two characters of the given line breaking classes *only* if they are separated by one or more spaces.

- For an indirect break, a break opportunity exists after the last space. No break opportunity exists if the characters are immediately adjacent.

An indirect break is indicated in the pair table in *Table 2* as **B** % **A**, where **B** is the character class of the character *before* and **A** is the character class of the character *after* the break. Even though space characters are not shown in the pair table, an indirect break can occur only if one or more spaces follow B. In the notation of the rules in *Section 6, Line Breaking Algorithm*, this would be represented as two rules: **B** × **A** *and* **B SP**+ ÷ **A** where the "+" sign means one or more occurrences.

LD10 Prohibited Break:
No line break opportunity exists between two characters of the given line breaking classes, even if they are separated by one or more space characters.

A prohibited break is indicated in the pair table in *Table 2* as **B** ^ **A**, where **B** is the character class of the character *before* and **A** is the character class of the character *after* the break, and the optional space characters are not shown. In the notation of the rules in *Section 6, Line Breaking Algorithm*, this would be expressed as a rule of the form: **B SP*** × **A**.

LD11 Hyphenation: Hyphenation uses language-specific rules to provide additional line break opportunities *within* a word.

- Hyphenation improves the layout of narrow columns, especially for languages with many longer words, such as German or Finnish. For the purpose of this annex, it is assumed that hyphenation is equivalent to inserting *soft hyphen* characters. All other aspects of hyphenation are outside the scope of this annex.

Table 1 lists all of line breaking classes by name, and also giving their class abbreviation and their status as tailorable or not. The examples and brief indication of line breaking behavior in this table are merely typical, not exhaustive. *Section 5.1, Description of Line Breaking Properties*, provides a detailed description of each line breaking class, including detailed overview of the line breaking behavior for characters of that class.

Table 1. Line Breaking Classes (* = non-tailorable)

Class	Descriptive Name	Examples	Characters with This Property...
		Non-tailorable Line Breaking Classes	
BK *	*Mandatory Break*	NL, PS	Cause a line break (after)
CR *	*Carriage Return*	CR	Cause a line break (after), except between CR and LF
LF *	*Line Feed*	LF	Cause a line break (after)
CM *	*Attached Characters and Combining Marks*	Combining marks, control codes	Prohibit a line break between the character and the preceding character

NL *	*Next Line*	NEL	Cause a line break (after)
SG *	*Surrogates*	Surrogates	Do not occur in well-formed text
WJ *	*Word Joiner*	WJ	Prohibit line breaks before and after
ZW *	*Zero Width Space*	ZWSP	Provide a break opportunity
GL *	*Non-breaking ("Glue")*	CGJ, NBSP, ZWNBSP	Prohibit line breaks before and after
SP *	*Space*	SPACE	Enable indirect line breaks
		Break Opportunities	
B2	*Break Opportunity Before and After*	Em dash	Provide a line break opportunity before and after the character
BA	*Break Opportunity After*	Spaces, hyphens	Generally provide a line break opportunity after the character
BB	*Break Opportunity Before*	Punctuation used in dictionaries	Generally provide a line break opportunity before the character
HY	*Hyphen*	HYPHEN-MINUS	Provide a line break opportunity after the character, except in numeric context
CB	*Contingent Break Opportunity*	Inline objects	Provide a line break opportunity contingent on additional information
		Characters Prohibiting Certain Breaks	
CL	*Closing Punctuation*	")", "]", "}", etc.	Prohibit line breaks before
EX	*Exclamation/ Interrogation*	"!", "?", etc.	Prohibit line breaks before
IN	*Inseparable*	Leaders	Allow only indirect line breaks between pairs
NS	*Nonstarter*	small kana	Allow only indirect line breaks before
OP	*Opening Punctuation*	"(", "[", "{", etc.	Prohibit line breaks after
QU	*Ambiguous Quotation*	Quotation marks	Act like they are both opening and closing
		Numeric Context	
IS	*Infix Separator (Numeric)*	. ,	Prevent breaks after any and before numeric
NU	*Numeric*	Digits	Form numeric expressions for line breaking purposes
PO	*Postfix (Numeric)*	%, ¢	Do not break following a numeric expression
PR	*Prefix (Numeric)*	$, £, ¥, etc.	Do not break in front of a numeric expression
SY	*Symbols Allowing Break After*	/	Prevent a break before, and allow a break after
		Other Characters	

AI	*Ambiguous (Alphabetic or Ideographic)*	Characters with Ambiguous East Asian Width	Act like **AL** when the resolved EAW is N; otherwise, act as **ID**
AL	*Ordinary Alphabetic and Symbol Characters*	Alphabets and regular symbols	Are alphabetic characters or symbols that are used with alphabetic characters
H2	*Hangul LV Syllable*	Hangul	Form Korean syllable blocks
H3	*Hangul LVT Syllable*	Hangul	Form Korean syllable blocks
ID	*Ideographic*	Ideographs	Break before or after, except in some numeric context
JL	*Hangul L Jamo*	Conjoining jamo	Form Korean syllable blocks
JV	*Hangul V Jamo*	Conjoining jamo	Form Korean syllable blocks
JT	*Hangul T Jamo*	Conjoining jamo	Form Korean syllable blocks
SA	*Complex Context Dependent (South East Asian)*	South East Asian: Thai, Lao, Khmer	Provide a line break opportunity contingent on additional, language-specific context analysis
XX	*Unknown*	Unassigned, private-use	Have as yet unknown line breaking behavior or unassigned code positions

3 Introduction

Lines are broken as result of one of two conditions. The first condition is the presence of a mandatory
line breaking character. The second condition results from a formatting algorithm having selected among available line break opportunities; ideally the chosen line break results in the optimal layout of the text.

Different formatting algorithms may use different methods to determine an optimal line break. For example, simple implementations consider a single line at a time, trying to find a *locally optimal*
line break. A basic, yet widely used approach is to allow no compression or expansion of the intercharacter and interword spaces and consider the longest line that fits. More complex formatting algorithms often take into account the interaction of line breaking decisions for the whole paragraph. The well-known text layout system [TEX] implements an example of such a *globally optimal*
strategy that may make complex tradeoffs across an entire paragraph to avoid unnecessary hyphenation and other legal, but inferior breaks. For a description of this strategy, see [Knuth78].

When compression or expansion is allowed, a locally optimal line break seeks to balance the relative merits of the resulting amounts of compression and expansion for different line break candidates. When expanding or compressing interword space according to common typographical practice, only the spaces marked by U+0020 SPACE, U+00A0 NO-BREAK SPACE, and U+3000 IDEOGRAPHIC SPACE are subject to compression, and only spaces marked by U+0020 SPACE,

U+00A0 NO-BREAK SPACE, and occasionally spaces marked by U+2009 THIN SPACE are subject to expansion. All other space characters normally have fixed width. When expanding or compressing intercharacter space, the presence of U+200B ZERO WIDTH SPACE or U+2060 WORD JOINER is always ignored.

Local custom or document style determines whether and to what degree expansion of intercharacter space is allowed in justifying a line. In languages, such as German, where intercharacter space is commonly used to mark e m p h a s i s (like this), allowing variable intercharacter spacing would have the unintended effect of adding random emphasis, and is therefore best avoided. In table headings that use Han ideographs, even extreme amounts of intercharacter space commonly occur as short texts are spread out across the entire available space to distribute the characters evenly from end to end.

The definition of optimal line breaks is outside the scope of this annex, as are methods for their selection. For the purpose of this annex, what is important is not so much what defines the optimal amount of text on the line, but how to determine all legal line break *opportunities*. Whether and how any given line break opportunity is actually used is up to the full layout system. Some layout systems may further evaluate the raw line break opportunities returned from the line breaking algorithm and apply additional rules. TEX, for example, uses line break opportunities based on hyphens only as a last resort.

Finally, text layout systems may support an emergency mode that handles the case of an unusual line that contains no otherwise permitted line break opportunities. In such line layout emergencies, line breaks may be placed with no regard to the ordinary line breaking behavior of the characters involved. The details of such an emergency mode are outside the scope of this annex, however, it is recommended that grapheme clusters be kept together.

3.1 Determining Line Break Opportunities

Three principal styles of context analysis determine line break opportunities.

1. *Western*: spaces and hyphens are used to determine breaks
2. *East Asian:* lines can break anywhere, unless prohibited
3. *South East Asian*: line breaks require morphological analysis

The Western style is commonly used for scripts employing the space character. Hyphenation is often used with space-based line breaking to provide additional line break opportunities—however, it requires knowledge of the language and it may need user interaction or overrides.

The second style of context analysis is used with East Asian ideographic and syllabic scripts. In these scripts, lines can break anywhere, except before or after certain characters. The precise set of prohibited line breaks may depend on user preference or local custom and is commonly tailorable.

Korean makes use of both styles of line break. When Korean text is justified, the second style is commonly used, even for interspersed Latin letters. But when ragged margins are used, the Western style (relying on spaces) is commonly used instead, even for ideographs.

The third style is used for scripts such as Thai, which do not use spaces, but which restrict word breaks to syllable boundaries, the determination of which requires knowledge of the language comparable to that required by a hyphenation algorithm. Such an algorithm is beyond the scope of the Unicode Standard.

For multilingual text, the Western and East Asian styles can be unified into a single set of specifications, based on the information in this annex. Unicode characters have explicit line breaking properties assigned to them. These properties can be utilized to implement the effect of both of these two styles of context analysis for line break opportunities. Customization for user preferences or document style can then be achieved by tailoring that specification.

In bidirectional text, line breaks are determined before applying rule L1 of the Unicode Bidirectional Algorithm [Bidi]. However, line breaking is strictly independent of directional properties of the characters or of any auxiliary information determined by the application of rules of that algorithm.

4 Conformance

Note to reviewers: *This section contains new text from the Editorial Committee that has not yet been discussed by the UTC. It has been substantially revised and should be reviewed carefully.*

There is no single method for determining line breaks; the rules may differ based on user preference and document layout. Therefore the information in this annex, including the specification of the line breaking algorithm, allows for the necessary flexibility in determining line breaks according to different conventions. However, some characters have been encoded explicitly for their effect on line breaking. Users adding such characters to a text expect that they will have the desired effect. For that reason, these characters have been given ~~non-tailorable~~ required line breaking behavior.

To handle certain situations, some line breaking implementations use techniques that cannot be expressed within the framework of the Unicode Line Breaking Algorithm. Examples include the use of dictionaries of words for languages that do not use spaces, including Chinese, Japanese, Korean and Thai; recognition of the language of the text in order to choose among different punctuation conventions; the use of dictionaries of common abbreviations or contractions to resolve ambiguities with periods or apostrophes; or a deeper analysis of common syntaxes for numbers or dates, etc. The conformance requirements permit variations of this kind.

Processes which support multiple modes for determining line breaks are also accommodated. This situation can arise with marked-up text, rich text, style sheets, or other environments in which a higher-level protocol can carry formatting instructions that prevent or force line breaks in positions that differ from those specified by the Unicode Line Break Algorithm. The approach taken here is to require that such processes have a conforming default line break behavior, and to disclose that they also include overrides or optional behaviors that are invoked via a higher-level protocol.

The methods by which a line layout process chooses optimal line breaks from among the available break opportunities is outside the scope of this specification. The behavior of a line layout process in situations where there are no suitable break opportunities is also outside of the scope of this specification.

~~The conformance requirements are spelled out in the following subsections.~~

~~At times, this specification recommends best practice. These recommendations are not normative and conformance with this specification does not depend on their realization. These recommendations contain the expression "This specification recommends ...", or some similar wording.~~

4.1 Conformance Requirements

UAX14-C1.

A process that determines line breaks in Unicode text, and that purports to implement the Unicode Line Breaking Algorithm, shall do so in accordance with the specifications in this annex. In particular, the following three subconditions shall be met:

1. The sets of mandatory break positions and of break opportunities which the implementation produces include all of those specified by the rules in Section 6.1.
2. There exist no break opportunities or mandatory breaks produced by the implementation that fall on a "non-break" position specified by the rules in Section 6.1.
3. If the implementation tailors the behavior of Section 6.2, that fact must be disclosed.

UAX14-C2.
If an implementation has a default line breaking operation which conforms to UAX14-C1, but also has overrides based on a higher-level protocol, that fact must be disclosed and any behavior that differs from that specified by the rules of section 6.1 must be documented.

> Example: An xml format provides markup which disables all line breaking over some span of text. When the markup is not in place, the default behavior is in conformance according to UAX14-C1. As long as the existence of the option is disclosed, that format can be said to conform to the Unicode Line Breaking Algorithm according to UAX14-C2.

As is the case for all other Unicode algorithms, this specification is a logical description—particular implementations can have more efficient mechanisms as long as they produce the same results. See C18 in Chapter 3, Conformance, of [Unicode] While only disclosure of tailorings is required in the conformance clauses, documentation of the differences in behaviors is strongly encouraged.

~~**4.1 Line Breaking Algorithm**~~

~~***UAX14-C1.*** *In the absence of a permissible higher-level protocol, a process that determines line breaks in Unicode text, and that purports to implement the Unicode Line Breaking Algorithm, shall do so in accordance with the specifications in this annex.*~~

- ~~As is the case for all other Unicode algorithms, this specification is a *logical* description—particular implementations can have more efficient mechanisms as long as they produce the same results. See C18 in *Chapter 3, Conformance*, of [Unicode], and the notes following.~~
- ~~AThe line breaking algorithm specifies part of the intrinsic semantics of characters specifically encoded for their line breaking behavior and, therefore, is required for conformance to the Unicode Standard where text containing such characters is broken into lines.~~

~~***UAX14-C2.*** *The permissible higher-level protocols are described in Section 4.3 Higher-Level Protocols.*~~

~~**4.2 Line Breaking Properties**~~

~~All line breaking classes are normative, but overridable, except for those line breaking classes which are not overridable. The latter are marked with * in *Table 1*, and as "non-tailorable" in Section 5.1, *Description of Line Breaking Properties*.~~

~~**4.3 Higher-Level Protocols**~~

There are many different ways to break lines of text, and the Unicode Standard does not intend to unnecessarily restrict the ways in which implementations can do this. However, for characters that are encoded solely or primarily for their line breaking behavior, interpretation of these characters must be consistent with their semantics as defined by their normative line breaking behavior. This leads to the following permissible higher-level protocols:

***UAX14-HL1**. Override rule 2 and report a break at the start of text.*

- A higher-level protocol may report a break at the start of text (**sot**). As written, the rule is intended to ensure that the line breaking algorithm always produces lines that have at least one character in them. However, an analysis in terms of text boundaries would more naturally report a boundary at the **sot**, leaving it to any client to skip past that boundary in breaking lines.

***UAX14-HL2.** Tailor any tailorable line break class.*

- A higher-level protocol may change the algorithm to produce results as if the membership of any tailorable line break class had been changed.

***UAX14-HL3.** Override any rule in Section 6.2, Tailorable Line Breaking Rules, or add new rules to that section.*

- A higher-level protocol my change the algorithm to produce results as if any of the rules in Section 6.2, *Tailorable Line Breaking Rules*, had been deleted or amended, or as if new rules had been added.

Because of the way the specification is set up, HL2 and HL3 have no effect on the results for text containing only characters of the non tailorable line breaking classes. However, they allow for unrestricted tailoring of the results for texts containing only characters from the tailorable line breaking classes as well as wide latitude in defining the behavior of mixed texts.

Note to reviewers:
late-breaking feedback from the CSS working group at the W3C has pointed to the need to have a small number of focused additional exceptions for higher level protocols. They are marked as (proposed) in the draft below. They have not been reviewed by UTC but are included here to enable advance review comments.

***UAX14-HL4.** (proposed) Allow line breaks before a SP character*

- A higher level protocol may support a mode where spaces can wrap to the beginning of a new line under certain circumstances. For example, this can occur with the use of CSS when white space is preserved.

5 Line Breaking Properties

This section provides detailed narrative descriptions of the line breaking behavior of many Unicode characters. In many instances, the descriptions in this section provide additional informative detail about handling a given character at the end of a line, or during line layout, which goes beyond the simple determination of line breaks. In some cases, the text also gives guidance as to preferred characters for achieving a particular effect in line breaking.

This section also summarizes the membership of character classes for each value of the line breaking property. Note that the mnemonic names for the line break classes are intended neither as exhaustive descriptions of their membership nor as indicators of their entire range

of behaviors in the line breaking process. Instead, their main purpose is to serve as unique, yet broadly mnemonic labels. In other words, as long as their line break behavior is identical, otherwise unrelated characters will be found grouped together in the same line break class.

The classification by property values defined in this section and in the data file is used as input into two algorithms defined in *Section 6, Line Breaking Algorithm*, and *Section 7, Pair Table-Based Implementation*. These sections describe workable default line breaking methods. *Section 8, Customization*, discusses how the default line breaking behavior can be tailored to the needs of particular languages for particular document styles and user preferences.

Data File

The full classification of all Unicode characters by their line breaking properties is available in the file LineBreak.txt [Data14] in the Unicode Character Database [UCD]. This is a tab-delimited, two-column, plain text file, with code position and line breaking class. A comment at the end of each line indicates the character name. Ideographic, Hangul, Surrogate, and Private Use ranges are collapsed by giving a range in the first column.

Future Updates

As more scripts are added to the Unicode Standard and become more widely implemented and used on computers, more line breaking classes may be added or the assignment of line breaking class may be changed for some characters. Implementers must not make any assumptions to the contrary. Any future updates will be reflected in the latest version of the data file. (See the Unicode Character Database [UCD] for any specific version of the data file.)

5.1 Description of Line Breaking Properties

Line breaking classes are listed alphabetically. Each line breaking class is marked with an annotation in parentheses with the following meanings:

(A)—the class allows a break opportunity *after* in specified contexts

(XA)—the class prevents a break opportunity *after* in specified contexts

(B)—the class allows a break opportunity *before* in specified contexts

(XB)—the class prevents a break opportunity *before* in specified contexts

(P)—the class allows a break opportunity for a *pair* of same characters

(XP)—the class prevents a break opportunity for a *pair* of same characters

Note: The use of the letters **B** and **A** in these annotations marks the position of the break opportunity relative to the character. It is not to be confused with the use of the same letters in the other parts of this annex, where they indicate the positions of the characters relative to the break opportunity.

AI: Ambiguous (Alphabetic or Ideograph)

Some characters that ordinarily act like alphabetic or symbol characters (which have the **AL** line breaking class) are treated like ideographs (line breaking class **ID**) in certain East Asian legacy contexts. Their line breaking behavior therefore depends on the context. In the absence of appropriate context information, they are treated as class **AL**, see the note at the

end of this description.

As originally defined, the line break class **AI** contained *all* characters with East_Asian_Width value A (ambiguous width) that would otherwise be **AL** in this classification. For more information on East_Asian_Width and how to resolve it, see Unicode Standard Annex #11, *East Asian Width* [EAW].

The original definition included many Latin, Greek, and Cyrillic characters. These characters are now classified by default as **AL** because use of the **AL** line breaking class better corresponds to modern practice. Where strict compatibility with older legacy implementations is desired, some of these characters need to be treated as **ID** in certain contexts. This can be done by always tailoring them to **ID** or by continuing to classify them as **AI** and resolving them to **ID** where required.

As part of the same revision, the set of ambiguous characters has been extended to completely encompass the enclosed alphanumeric characters used for numbering of bullets.

As updated, the **AI**
line breaking class includes all characters with East Asian Width A that are outside the range U+0000..U+1FFF, plus the following characters:

24EA	CIRCLED DIGIT ZERO
2780..2793	DINGBAT CIRCLED SANS-SERIF DIGIT ONE..DINGBAT NEGATIVE CIRCLED SANS-SERIF NUMBER TEN

Characters with the line break class **AI** with East_Asian_Width value A typically take the **AL** line breaking class when their *resolved* East_Asian_Width is N (narrow) and take the line breaking class **ID** when their resolved width is W (wide). The remaining characters are then resolved to **AL** or **ID** in a consistent fashion. The details of this resolution are not specified in this annex. The line breaking rules in *Section 6, Line Breaking Algorithm*, and the pair table in *Section 7, Pair Table-Based Implementation*, merely require that all ambiguous characters have been resolved appropriately as part of assigning line breaking classes to the input characters.

> *Note:* The canonical decompositions of characters of class **AI** are not necessarily of class **AI**
> themselves, or conversely. The East_Asian_Width property A on which the definition of **AI**
> is largely based, does not preserve canonical equivalence. In the context of line breaking, the fact that a character has been assigned class **AI** means that the line break implementation must resolve it to either **AL** or **ID**, in the absence of further tailoring. If preserving canonical equivalence is desired, an implementation is free to make sure that the *resolved*
> line break classes preserve canonical equivalence. Unless compatibility with particular legacy behavior is important, it may be sufficient to map all such characters to **AL**. This achieves a canonically equivalent resolution of line breaking classes, and is compatible with emerging modern practice that treats these characters increasingly like regular alphabetic characters.

AL: Ordinary Alphabetic and Symbol Characters (XP)

Ordinary characters require other characters to provide break opportunities; otherwise, no line breaks are allowed between pairs of them. However, this behavior is tailorable. In some Far Eastern documents, it may be desirable to allow breaking between pairs of ordinary characters—particularly Latin characters and symbols.

Note:
Use ZWSP as a manual override to provide break opportunities around alphabetic or symbol characters.

Except as listed explicitly below as part of another line breaking class, and except as assigned class **AI** or **ID**
based on East Asian Width, this class contains the following characters:

ALPHABETIC—all remaining characters of General Categories Lu, Ll, Lt, Lm, and Lo
SYMBOLS—all remaining characters of General Categories Sm, Sk, and So
NON-DECIMAL NUMBERS—all remaining characters of General Categories Nl, and No
PUNCTUATION—all remaining characters of General Categories Pc, Pd, and Po

Plus these characters:

0600..0603	ARABIC NUMBER SIGN..ARABIC SIGN SAFHA
06DD	ARABIC END OF AYAH
070F	SYRIAC ABBREVIATION MARK
2061..2063	FUNCTION APPLICATION..INVISIBLE SEPARATOR

These characters occur in the middle or at the beginning of words or alphanumeric or symbol sequences. However, when alphabetic characters are tailored to allow breaks, these characters should not allow breaks after.

BA: Break Opportunity After (A)

Like SPACE, the characters in this class provide a break opportunity; unlike SPACE, they do not take part in determining indirect breaks. They can be subdivided into several categories.

Breaking Spaces

Breaking spaces are the following subset of characters with General_Category Zs:

1680	OGHAM SPACE MARK
2000	EN QUAD
2001	EM QUAD
2002	EN SPACE
2003	EM SPACE
2004	THREE-PER-EM SPACE
2005	FOUR-PER-EM SPACE
2006	SIX-PER-EM SPACE
2008	PUNCTUATION SPACE
2009	THIN SPACE
200A	HAIR SPACE
205F	MEDIUM MATHEMATICAL SPACE

All of these space characters have a specific width, but otherwise behave as breaking spaces. In setting a justified line, none of these spaces normally changes in width, except for THIN SPACE when used in mathematical notation. See also the **SP** property.

The Ogham space mark may be
rendered visibly between words but it is recommended that it be elided at the end of a line.
For more information, see section 5.7 Word Separator Characters.

See the **ID** property for U+3000 IDEOGRAPHIC SPACE. For a list of all space characters in the

Unicode Standard, see *Section 6.2, General Punctuation*, in [Unicode5.0].

Tabs

0009	TAB

Except for the effect of the location of the tab stops, the tab character acts similarly to a space for the purpose of line breaking.

Conditional Hyphens

00AD	SOFT HYPHEN (SHY)

SHY marks the place where an optional line break may occur inside a word. It can be used with all scripts. SHY is rendered invisibly and has no width: it merely indicates an optional line break. The rendering of the optional line break depends on the script. For the Latin script, rendering the line break typically means displaying a hyphen at the end of the line; however, some languages require a change in spelling surrounding an optional line break. For examples, see *Section 5.4, Use of Soft Hyphen*.

Breaking Hyphens

Breaking hyphens establish explicit break opportunities immediately after each occurrence.

058A	ARMENIAN HYPHEN
2010	HYPHEN
2012	FIGURE DASH
2013	EN DASH

Hyphens are graphic characters with width. Because, unlike spaces, they are visible, they are included in the measured part of the preceding line, except where the layout style allows hyphens to hang into the margins.
For additional information about how to format line breaks resulting from the presence of hyphens, see *Section 5.3, Use of Hyphen*.

Visible Word Dividers

The following are other forms of visible word dividers that provide break opportunities:

05BE	HEBREW PUNCTUATION MAQAF
0F0B	TIBETAN MARK INTERSYLLABIC TSHEG
1361	ETHIOPIC WORDSPACE
17D8	KHMER SIGN BEYYAL
17DA	KHMER SIGN KOOMUUT

The Tibetan *tsheg*
is a visible mark, but it functions effectively like a space to separate words (or other units) in Tibetan. It provides a break opportunity after itself. For additional information, see *Section 5.6, Tibetan Line Breaking*.

The *Ethiopic word space*
is a visible word delimiter and is kept on the previous line. In contrast, U+1360 ETHIOPIC SECTION MARK
is typically used in a sequence of several such marks on a separate line, and separated by spaces. As such lines are typically marked with separate hard line breaks (**BK**), the section

mark is treated like an ordinary symbol and given line break class **AL**.

2027 HYPHENATION POINT

A hyphenation point is a raised dot, which is mainly used in dictionaries and similar works to visibly indicate syllabification of words. Syllable breaks frequently also are potential line break opportunities in the middle of words. When an actual line break falls inside a word containing hyphenation point characters, the hyphenation point is usually rendered as a regular hyphen at the end of the line.

007C VERTICAL LINE

In some dictionaries, a vertical bar is used instead of a hyphenation point. In this usage, U+0323 COMBINING DOT BELOW is used to mark stressed syllables, so all breaks are marked by the vertical bar. For an actual break opportunity, the vertical bar is rendered as a hyphen in such usage.

Historic Word Separators

Historic texts, especially ancient ones, often do not use spaces, even for scripts where modern use of spaces is standard. Special punctuation was used to mark word boundaries in such texts. For modern text processing it is recommended to treat these as line break opportunities by default. **WJ** can be used to override this default, where necessary.

16EB	RUNIC SINGLE DOT PUNCTUATION
16EC	RUNIC MULTIPLE DOT PUNCTUATION
16ED	RUNIC CROSS PUNCTUATION
2056	THREE DOT PUNCTUATION
2058	FOUR DOT PUNCTUATION
2059	FIVE DOT PUNCTUATION
205A	TWO DOT PUNCTUATION
205B	FOUR DOT MARK
205D	TRICOLON
205E	VERTICAL FOUR DOTS
2E19	PALM BRANCH
2E2A	TWO DOTS OVER ONE DOT PUNCTUATION
2E2B	ONE DOT OVER TWO DOTS PUNCTUATION
2E2C	SQUARED FOUR DOT PUNCTUATION
2E2D	FIVE DOT PUNCTUATION
2E30	RING POINT
10100	AEGEAN WORD SEPARATOR LINE
10101	AEGEAN WORD SEPARATOR DOT
10102	AEGEAN CHECK MARK
1039F	UGARITIC WORD DIVIDER
103D0	OLD PERSIAN WORD DIVIDER
1091F	PHOENICIAN WORD DIVIDER
12470	CUNEIFORM PUNCTUATION SIGN OLD ASSYRIAN WORD DIVIDER

Dandas

DEVANAGARI DANDA is similar to a full stop. The *danda* or historically related symbols are used with several other Indic scripts. Unlike a full stop, the *danda* is not used in number formatting. DEVANAGARI DOUBLE DANDA marks the end of a verse. It also has analogues in other scripts.

0964	DEVANAGARI DANDA
0965	DEVANAGARI DOUBLE DANDA
0E5A	THAI CHARACTER ANGKHANKHU
0E5B	THAI CHARACTER KHOMUT
104A	MYANMAR SIGN LITTLE SECTION
104B	MYANMAR SIGN SECTION
1735	PHILIPPINE SINGLE PUNCTUATION
1736	PHILIPPINE DOUBLE PUNCTUATION
17D4	KHMER SIGN KHAN
17D5	KHMER SIGN BARIYOOSAN
1B5E	BALINESE CARIK SIKI
1B5F	BALINESE CARIK PAREREN
A8CE	SAURASHTRA DANDA
A8CF	SAURASHTRA DOUBLE DANDA
AA5D	CHAM PUNCTUATION DANDA
AA5E	CHAM PUNCTUATION DOUBLE DANDA
AA5F	CHAM PUNCTUATION TRIPLE DANDA
10A56	KHAROSHTHI PUNCTUATION DANDA
10A57	KHAROSHTHI PUNCTUATION DOUBLE DANDA

Tibetan

0F34	TIBETAN MARK BSDUS RTAGS
0F7F	TIBETAN SIGN RNAM BCAD
0F85	TIBETAN MARK PALUTA
0FBE	TIBETAN KU RU KHA
0FBF	TIBETAN KU RU KHA BZHI MIG CAN
0FD2	TIBETAN MARK NYIS TSHEG

For additional information, see *Section 5.6, Tibetan Line Breaking*.

Other Terminating Punctuation

Termination punctuation stays with the line, but otherwise allows a break after it. This is similar to **EX**, except that the latter may be separated by a space from the preceding word without allowing a break, whereas these marks are used without spaces.

1804	MONGOLIAN COLON
1805	MONGOLIAN FOUR DOTS
~~1808~~	~~MONGOLIAN MANCHU COMMA~~
~~1809~~	~~MONGOLIAN MANCHU FULL STOP~~
1B5A	BALINESE PANTI
1B5B	BALINESE PAMADA
1B5C	BALINESE WINDU
1B5D	BALINESE CARIK PAMUNGKAH
1B60	BALINESE PAMENENG
1C3B	LEPCHA PUNCTUATION TA-ROL
1C3C	LEPCHA PUNCTUATION NYET THYOOM TA-ROL
1C3D	LEPCHA PUNCTUATION CER-WA
1C3E	LEPCHA PUNCTUATION TSHOOK CER-WA
1C3F	LEPCHA PUNCTUATION TSHOOK
1C7E	OL CHIKI PUNCTUATION MUCAAD

1C7F	OL CHIKI PUNCTUATION DOUBLE MUCAAD
2CFA	COPTIC OLD NUBIAN DIRECT QUESTION MARK
2CFB	COPTIC OLD NUBIAN INDIRECT QUESTION MARK
2CFC	COPTIC OLD NUBIAN VERSE DIVIDER
2CFF	COPTIC MORPHOLOGICAL DIVIDER
2E0E..2E15	EDITORIAL CORONIS..UPWARDS ANCORA
2E17	OBLIQUE DOUBLE HYPHEN
A60D	VAI COMMA
A60F	VAI QUESTION MARK
A92E	KAYAH LI SIGN CWI
A92F	KAYAH LI SIGN SHYA
10A50	KHAROSHTHI PUNCTUATION DOT
10A51	KHAROSHTHI PUNCTUATION SMALL CIRCLE
10A52	KHAROSHTHI PUNCTUATION CIRCLE
10A53	KHAROSHTHI PUNCTUATION CRESCENT BAR
10A54	KHAROSHTHI PUNCTUATION MANGALAM
10A55	KHAROSHTHI PUNCTUATION LOTUS

BB: Break Opportunities Before (B)

Characters of this line break class move to the next line at a line break and thus provide a line break opportunity before.

Dictionary Use

00B4	ACUTE ACCENT
1FFD	GREEK OXIA

In some dictionaries, stressed syllables are indicated with a spacing acute accent instead of the hyphenation point. In this case the accent moves to the next line, and the preceding line ends with a hyphen. The oxia is canonically equivalent to the acute accent.

02DF	MODIFIER LETTER CROSS ACCENT

A cross accent also appears in some dictionaries to mark the stress of the following syllable, and should be handled in the same way as the other stress marking characters in this section. The accent should not be separated from the syllable it marks by a break.

02C8	MODIFIER LETTER VERTICAL LINE
02CC	MODIFIER LETTER LOW VERTICAL LINE

These characters are used in dictionaries to indicate stress and secondary stress when IPA is used. Both are prefixes to the stressed syllable in IPA. Breaking before them keeps them with the syllable.

> *Note:*
> It is hard to find actual examples in most dictionaries because the pronunciation fields usually occur right after the headword, and the columns are wide enough to prevent line breaks in most pronunciations.

Tibetan and Phags-Pa Head Letters

0F01	TIBETAN MARK GTER YIG MGO TRUNCATED A
0F02	TIBETAN MARK GTER YIG MGO -UM RNAM BCAD MA

0F03	TIBETAN MARK GTER YIG MGO -UM GTER TSHEG MA
0F04	TIBETAN MARK INITIAL YIG MGO MDUN MA
0F06	TIBETAN MARK CARET YIG MGO PHUR SHAD MA
0F07	TIBETAN MARK YIG MGO TSHEG SHAD MA
0F09	TIBETAN MARK BSKUR YIG MGO
0F0A	TIBETAN MARK BKA- SHOG YIG MGO
0FD0	TIBETAN MARK BSKA- SHOG GI MGO RGYAN
0FD1	TIBETAN MARK MNYAM YIG GI MGO RGYAN
0FD3	TIBETAN MARK INITIAL BRDA RNYING YIG MGO MDUN MA
A874	PHAGS-PA SINGLE HEAD MARK
A875	PHAGS-PA DOUBLE HEAD MARK

Tibetan head letters allow a break before. For more information, see *Section 5.6, Tibetan Line Breaking*.

Mongolian

1806	MONGOLIAN TODO SOFT HYPHEN

Despite its name, this Mongolian character is not an invisible control like SOFT HYPHEN, but rather a visible character like a regular hyphen. Unlike the hyphen, MONGOLIAN TODO SOFT HYPHEN stays with the following line. Whenever optional line breaks are to be marked invisibly, SOFT HYPHEN should be used instead.

B2: Break Opportunity Before and After (B/A/XP)

2014	EM DASH

The EM DASH
is used to set off parenthetical text. Normally, it is used without spaces. However, this is language dependent. For example, in Swedish, spaces are used around the EM DASH. Line breaks can occur before and after an EM DASH. Because EM DASHes are sometimes used in pairs instead of a single quotation dash, the default behavior is not to break the line between even though not all fonts use connecting glyphs for the EM DASH.

BK: Mandatory Break (A) (Non-tailorable)

Explicit breaks act independently of the surrounding characters. No characters can be added to the **BK**
class as part of tailoring, but implementations are not required to support the VT character.

000C	FORM FEED (FF)
000B	LINE TABULATION (VT)

FORM FEED separates pages. The text on the new page starts at the beginning of the line. In some layout modes there may be no visible advance to a new “page”.

2028	LINE SEPARATOR (LS)

The text after the LINE SEPARATOR
starts at the beginning of the line. This is similar to HTML
.

2029	PARAGRAPH SEPARATOR (PS)

The text of the new paragraph starts at the beginning of the line. This character defines a paragraph break, causing suitable formatting to be applied, for example, interparagraph

spacing or first line indentation. LS, FF, VT as well as **CR**, **LF** and **NL** do not define a paragraph break.

Newline Function (NLF)

Newline Functions are defined in the Unicode Standard as providing additional mandatory breaks. They are not individual characters, but are encoded as sequences of the control characters NEL, LF, and CR. If a character sequence for a Newline Function contains more than one character, it is kept together. The particular sequences that form an NLF depend on the implementation and other circumstances as described in *Section 5.8, Newline Guidelines*, of [Unicode5.0].

This specification defines the NLF implicitly. It defines the three character classes **CR**, **LF**, and **NL**. Their line break behavior, defined in rule **LB5** in *Section 6.1, Non-tailorable Line Breaking Rules*, is to break after **NL**, **LF**, or **CR**, but not between **CR** and **LF**.

CB: Contingent Break Opportunity (B/A)

By default, there is a break opportunity both *before* and *after* any inline object. Object-specific line breaking behavior is implemented in the associated object itself, and where available can override the default to prevent either or both of the default break opportunities. Using U+FFFC OBJECT REPLACEMENT CHARACTER
allows the object anchor to take a character position in the string.

FFFC	OBJECT REPLACEMENT CHARACTER

Object-specific line break behavior is best implemented by querying the object itself, not by replacing the **CB** line breaking class by another class.

CL: Closing Punctuation (XB)

The closing character of any set of paired punctuation should be kept with the preceding character, and the same applies to all forms of wide comma and full stop. This is desirable, even when there are intervening space characters, so as to prevent the appearance of a bare closing punctuation mark at the head of a line. The **CL** line break class contains the following characters plus any characters of General_Category Pe in the Unicode Character Database.

3001..3002	IDEOGRAPHIC COMMA..IDEOGRAPHIC FULL STOP
FE11	PRESENTATION FORM FOR VERTICAL IDEOGRAPHIC COMMA
FE12	PRESENTATION FORM FOR VERTICAL IDEOGRAPHIC FULL STOP
FE50	SMALL COMMA
FE52	SMALL FULL STOP
FF0C	FULLWIDTH COMMA
FF0E	FULLWIDTH FULL STOP
FF61	HALFWIDTH IDEOGRAPHIC FULL STOP
FF64	HALFWIDTH IDEOGRAPHIC COMMA

CM: Attached Characters and Combining Marks (XB) (Non-tailorable)

Combining Characters

Combining character sequences are treated as units for the purpose of line breaking. The line breaking behavior of the sequence is that of the base character.

The preferred base character for showing combining marks in isolation is U+00A0 NO-BREAK

SPACE. If a line break before or after the combining sequence is desired, U+200B ZERO WIDTH SPACE can be used. The use of U+0020 SPACE as a base character is deprecated.

For most purposes, combining characters take on the properties of their base characters, and that is how the **CM** class is treated in rule **LB9** of this specification. As a result, if the sequence <0021, 20E4> is used to represent a triangle enclosing an exclamation point, it is effectively treated as **EX**, the line break class of the exclamation mark. If U+2061 CAUTION SIGN had been used, which also looks like an exclamation point inside a triangle, it would have the line break class of **AL**. Only the latter corresponds to the line breaking behavior expected by users for this symbol. To avoid surprising behavior, always use a base character that is a symbol or letter (Line Break **AL**) when using enclosing combining marks (General_Category Me).

The **CM**
line break class includes all combining characters with General_Category Mc, Me, and Mn, unless listed explicitly elsewhere. This includes *viramas*.

Control and Formatting Characters

Most control and formatting characters are ignored in line breaking and do not contribute to the line width. By giving them class **CM**, the line breaking behavior of the last preceding character that is not of class **CM** affects the line breaking behavior.

> *Note:*
> When control codes and format characters are rendered visibly during editing, more graceful layout might be achieved by treating them as if they had the line break class of the visible symbols instead, that is **AL** or **ID**. Such visible modes do not violate the constraint on tailorability, because they are logically equivalent to having temporarily substituted symbol *characters*, such as the characters from the Control Pictures block, or in some cases, character sequences, for the actual control characters.

The **CM**
line break class includes all characters of General_Category Cc and Cf, unless listed explicitly elsewhere.

CR: Carriage Return (A) (Non-tailorable)

000D CARRIAGE RETURN (CR)

A **CR** indicates a mandatory break after, unless followed by a **LF**. See also the discussion under **BK**.

> *Note:*
> On some platforms the character sequence <CR, CR, LF> is used to indicate the location of actual line breaks, whereas <CR, LF> is treated like a hard line break. As soon as a user edits the text, the location of all the <CR, CR, LF> sequences may change as the new text breaks differently, while the relative position of any <CR, LF> to the surrounding text stays the same. This convention allows an editor to return a buffer and the client to tell which text is displayed on which line by counting the number of <CR, CR, LF> and <CR, LF> sequences. This convention is essentially equivalent to markup that captures the result of applying the line break algorithm, not a tailoring of the CR character. The <CR, CR, LF> sequences are thus not considered part of the plain text content.

EX: Exclamation/Interrogation (XB)

Characters in this line break class behave like closing characters, except in relation to postfix (**PO**) and non-starter characters (**NS**).

0021	EXCLAMATION MARK
003F	QUESTION MARK
05C6	HEBREW PUNCTUATION NUN HAFUKHA
061B	ARABIC SEMICOLON
061E	ARABIC TRIPLE DOT PUNCTUATION MARK
061F	ARABIC QUESTION MARK
06D4	ARABIC FULL STOP
07F9	NKO EXCLAMATION MARK
0F0D	TIBETAN MARK SHAD
0F0E	TIBETAN MARK NYIS SHAD
0F0F	TIBETAN MARK TSHEG SHAD
0F10	TIBETAN MARK NYIS TSHEG SHAD
0F11	TIBETAN MARK RIN CHEN SPUNGS SHAD
0F14	TIBETAN MARK GTER TSHEG
1802	MONGOLIAN COMMA [was BA]
1803	MONGOLIAN FULL STOP [was BA]
1808	MONGOLIAN MANCHU COMMA [was BA]
1809	MONGOLIAN MANCHU FULL STOP [was BA]
1944	LIMBU EXCLAMATION MARK
1945	LIMBU QUESTION MARK
2762	HEAVY EXCLAMATION MARK ORNAMENT
2763	HEAVY HEART EXCLAMATION MARK ORNAMENT
2CF9	COPTIC OLD NUBIAN FULL STOP [was BA]
2CFE	COPTIC FULL STOP [was BA]
2E2E	REVERSED QUESTION MARK
A60C	VAI SYLLABLE LENGTHENER
A60E	VAI FULL STOP
A876	PHAGS-PA MARK SHAD
A877	PHAGS-PA MARK DOUBLE SHAD
FE15	PRESENTATION FORM FOR VERTICAL EXCLAMATION MARK
FE16	PRESENTATION FORM FOR VERTICAL QUESTION MARK
FE56..FE57	SMALL QUESTION MARK..SMALL EXCLAMATION MARK
FF01	FULLWIDTH EXCLAMATION MARK
FF1F	FULLWIDTH QUESTION MARK

GL: Non-breaking (“Glue”) (XB/XA) (Non-tailorable)

Non-breaking characters prohibit breaks on either side, but that prohibition can be overridden by **SP** or **ZW**. In particular, when NBSP follows SPACE, there is a break opportunity after the SPACE and NBSP will go as visible space onto the next line. See also **WJ**. The following lists the characters of line break class **GL** with additional description.

00A0	NO-BREAK SPACE (NBSP)
202F	NARROW NO-BREAK SPACE (NNBSP)
180E	MONGOLIAN VOWEL SEPARATOR (MVS)

NO-BREAK SPACE
is the preferred character to use where two words are to be visually separated but kept on the same line, as in the case of a title and a name “Dr.<NBSP>Joseph Becker”. When SPACE follows NBSP, there is no break, because there never is a break in front of SPACE. NARROW

NO-BREAK SPACE is used in Mongolian. The MONGOLIAN VOWEL SEPARATOR acts like a NNBSP in its line breaking behavior. It additionally affects the shaping of certain vowel characters as described in *Section 13.2*, *Mongolian*, of [Unicode5.0].

NARROW NO-BREAK SPACE (NNBSP) is a narrow version of NO-BREAK SPACE, which except for its display width behaves exactly the same in its line breaking behavior. It is regularly used in Mongolian in certain grammatical contexts (before a particle), where it also influences the shaping of the glyphs for the particle. In Mongolian text, the NNBSP is typically displayed with 1/3 the width of a normal space character.

When NARROW NO-BREAK SPACE occurs in French text, it should be interpreted as an “espace fine insécable”.

The MONGOLIAN VOWEL SEPARATOR is equivalent to a NNBSP in its line breaking behavior, but has different effects in controlling the shaping of its preceding and following characters. It constitutes a word-internal space and is typically displayed with half the width of a NNBSP.

034F	COMBINING GRAPHEME JOINER

This character has no visible glyph and its presence indicates that adjoining characters are to be treated as a graphemic unit, therefore preventing line breaks between them. The use of *grapheme joiner* affects other processes, such as sorting, therefore, U+2060 WORD JOINER should be used if the intent is to merely prevent a line break.

2007	FIGURE SPACE

This is the preferred space to use in numbers. It has the same width as a digit and keeps the number together for the purpose of line breaking.

2011	NON-BREAKING HYPHEN (NBHY)

This is the preferred character to use where words need to be hyphenated but may not be broken at the hyphen.
Because of this use as a substitute for ordinary hyphen, the appearance of this character should match that of U+2010 HYPHEN.

0F08	TIBETAN MARK SBRUL SHAD
0F0C	TIBETAN MARK DELIMITER TSHEG BSTAR
0F12	TIBETAN MARK RGYA GRAM SHAD

The TSHEG BSTAR looks exactly like a Tibetan *tsheg*, but can be used to prevent a break like *no-break space*. It inhibits breaking on either side. For more information, see *Section 5.6, Tibetan Line Breaking*.

035C..0362	COMBINING DOUBLE BREVE BELOW..COMBINING DOUBLE RIGHTWARDS ARROW BELOW

These diacritics span two characters, so no word or line breaks are possible on either side.

H2: Hangul LV Syllable (B/A)

This class includes all characters of Hangul Syllable Type LV.

Together with conjoining jamos, Hangul syllables form Korean Syllable Blocks, which are kept together; see [Boundaries]. Korean uses space-based line breaking in many styles of

documents. To support these, Hangul syllables and conjoining jamo need to be tailored to use class **AL**. The default in this specification is class **ID**, which supports the case of Korean documents not using space-based line breaking. See *Section 8.1, Types of Tailoring*. See also **JL**, **JT**, **JV**, and **H3**.

H3: Hangul LVT Syllable (B/A)

This class includes all characters of Hangul Syllable Type LVT. See also **JL**, **JT**, **JV**, and **H2**.

HY: Hyphen (XA)

002D HYPHEN-MINUS

Some additional context analysis is required to distinguish usage of this character as a hyphen from its usage as a minus sign (or indicator of numerical range). If used as hyphen, it acts like HYPHEN, which has line break class **BA**.

> *Note:* Some typescript conventions use runs of HYPHEN-MINUS to stand in for longer dashes or horizontal rules. If actual character code conversion is not performed and it is desired to treat them like the characters or layout elements they stand for, line breaking needs to support these runs explicitly.

ID: Ideographic (B/A)

> *Note:* This class includes characters other than Han ideographs.

Characters with this property do not require other characters to provide break opportunities; lines can ordinarily break before and after and between pairs of ideographic characters. The **ID** line break class consists of the following characters:

2E80..2FFF	CJK, KANGXI RADICALS, DESCRIPTION SYMBOLS
3000	IDEOGRAPHIC SPACE
3040..309F	*Hiragana* (except small characters)
30A0..30FF	*Katakana* (except small characters)
3400..4DB5	CJK UNIFIED IDEOGRAPHS EXTENSION A
4E00..9FBB	CJK UNIFIED IDEOGRAPHS
F900..FAD9	CJK COMPATIBILITY IDEOGRAPHS
A000..A48F	YI SYLLABLES
A490..A4CF	YI RADICALS
FE62..FE66	SMALL PLUS SIGN to SMALL EQUALS SIGN
FF10..FF19	WIDE DIGITS
20000..2A6D6	CJK UNIFIED IDEOGRAPHS EXTENSION B
2F800..2FA1D	CJK COMPATIBILITY IDEOGRAPHS SUPPLEMENT

It also includes all of the FULLWIDTH LATIN letters and all of the blocks in the range 3000..33FF not covered elsewhere.

> *Note:* Use U+2060 WORD JOINER
> as a manual override to prevent break opportunities around characters of class **ID**.

U+3000 IDEOGRAPHIC SPACE may be subject to expansion or compression during line justification.

Korean

Korean is encoded with conjoining jamo, Hangul syllables, or both. See also **JL**, **JT**, **JV**, **H2**, and **H3**. The following set of compatibility jamo is treated as **ID** by default.

3130..318F	HANGUL COMPATIBILITY JAMO

IN: Inseparable Characters (XP)

Leaders

These characters are intended to be used in consecutive sequence. There is never a line break between two character of this class.

2024	ONE DOT LEADER
2025	TWO DOT LEADER
2026	HORIZONTAL ELLIPSIS
FE19	PRESENTATION FORM FOR VERTICAL HORIZONTAL ELLIPSIS

HORIZONTAL ELLIPSIS can be used as a three-dot leader.

IS: Numeric Separator (Infix) (XB)

Characters that usually occur inside a numerical expression may not be separated from the numeric characters that follow, unless a space character intervenes. For example, there is no break in “100.00” or “10,000”, nor in “12:59”.

002C	COMMA
002E	FULL STOP
003A	COLON
003B	SEMICOLON
037E	GREEK QUESTION MARK (canonically equivalent to 003B)
0589	ARMENIAN FULL STOP
060C	ARABIC COMMA [moved from EX]
060D	ARABIC DATE SEPARATOR
07F8	NKO COMMA
2044	FRACTION SLASH
FE10	PRESENTATION FORM FOR VERTICAL COMMA
FE13	PRESENTATION FORM FOR VERTICAL COLON
FE14	PRESENTATION FORM FOR VERTICAL SEMICOLON

When not used in a numeric context, infix separators are sentence-ending punctuation. Therefore they always prevent breaks before.

> *Note: Figure Space*, not being a punctuation mark, has been given the line break class **GL**.

JL: Hangul L Jamo (B)

The **JL** line break class consists of all characters of Hangul Syllable Type L.

Conjoining jamos form Korean Syllable Blocks, which are kept together; see [Boundaries]. Korean uses space-based line breaking in many styles of documents. To support these, Hangul syllables and conjoining jamo need to be tailored to use class **AL**. The default in this specification is class **ID**, which supports the case of Korean documents not using space-based line breaking. See *Section 8.1, Types of Tailoring*. See also **JT**, **JV**, **H2**, and **H3**.

JT: Hangul T Jamo (A)

The **JT** line break class consists of all characters of Hangul Syllable Type T. See also **JL**, **JV**, **H2**, and **H3**.

JV: Hangul V Jamo (XA/XB)

The **JV** line break class consists of all characters of Hangul Syllable Type V. See also **JL**, **JT**, **H2**, and **H3**.

LF: Line Feed (A) (Non-tailorable)

000A	LINE FEED (LF)

There is a mandatory break after any LF character, but see the discussion under **BK**.

NL: Next Line (A) (Non-tailorable)

0085	NEXT LINE (NEL)

The **NL** class acts like **BK**
in all respects (there is a mandatory break after any NEL character). It cannot be tailored, but implementations are not required to support the NEL character; see the discussion under **BK**.

NS: Nonstarters (XB)

Nonstarter characters cannot start a line, but unlike **CL** they may allow a break in some contexts when they follow one or more space characters. Nonstarters include

17D6	KHMER SIGN CAMNUC PII KUUH
203C	DOUBLE EXCLAMATION MARK
203D	INTERROBANG
2047	DOUBLE QUESTION MARK
2048	QUESTION EXCLAMATION MARK
2049	EXCLAMATION QUESTION MARK
3005	IDEOGRAPHIC ITERATION MARK
301C	WAVE DASH
303C	MASU MARK
303B	VERTICAL IDEOGRAPHIC ITERATION MARK
309B.. 309E	KATAKANA-HIRAGANA VOICED SOUND MARK..HIRAGANA VOICED ITERATION MARK
30A0	KATAKANA-HIRAGANA DOUBLE HYPHEN
30FB..30FE	KATAKANA MIDDLE DOT..KATAKANA VOICED ITERATION MARK
A015	YI SYLLABLE WU (misnomer for YI SYLLABLE ITERATION MARK)
FE54..FE55	SMALL SEMICOLON..SMALL COLON
FF1A..FF1B	FULLWIDTH COLON.. FULLWIDTH SEMICOLON
FF65	HALFWIDTH KATAKANA MIDDLE DOT
FF70	HALFWIDTH KATAKANA-HIRAGANA PROLONGED SOUND MARK
FF9E..FF9F	HALFWIDTH KATAKANA VOICED SOUND MARK..HALFWIDTH KATAKANA SEMI-VOICED SOUND MARK

plus all Hiragana, Katakana, and Halfwidth Katakana “small” characters.

> *Note:* Optionally, the **NS** restriction may be relaxed and some or all characters treated like **ID**

to achieve a more permissive style of line breaking, especially in some East Asian document styles.

For additional information about U+30A0 KATAKANA-HIRAGANA DOUBLE HYPHEN, see *Section 5.5, Use of Double Hyphen*.

NU: Numeric (XP)

These characters behave like ordinary characters (**AL**) in the context of most characters but activate the prefix and postfix behavior of prefix and postfix characters.

Numeric characters consist of decimal digits (all characters of General_Category Nd), except those with East_Asian_Width F (Fullwidth), plus these characters:

066B	ARABIC DECIMAL SEPARATOR
066C	ARABIC THOUSANDS SEPARATOR

Unlike **IS**
characters, the Arabic numeric punctuation does not occur as sentence terminal punctuation outside numbers.

OP: Opening Punctuation (XA)

The opening character of any set of paired punctuation should be kept with the following character.
This is desirable, even when there are intervening space characters, so as to prevent the appearance of a bare opening punctuation mark at the end of a line. The **OP** line break class consists of all characters of General_Category Ps in the Unicode Character Database, plus

00A1	INVERTED EXCLAMATION MARK
00BF	INVERTED QUESTION MARK

Note: These two characters used to be classed **AI** based on their East_Asian_Width assignment of A. Such characters are normally resolved to either **ID** or **AL**. However, the two characters above are used as punctuation marks in Spanish, where they would behave more like a character of class **OP**.

PO: Postfix (Numeric) (XB)

Characters that usually follow a numerical expression may not be separated from preceding numeric characters or preceding closing characters, even if one or more space characters intervene. For example, there is no break opportunity in “(12.00) %”.

Some of these characters—in particular, *degree sign* and *percent sign*—can appear on both sides of a numeric expression. Therefore the line breaking algorithm by default does not break between **PO** and numbers or letters on either side.

The list of postfix characters is

0025	PERCENT SIGN
00A2	CENT SIGN
00B0	DEGREE SIGN
060B	AFGHANI SIGN
066A	ARABIC PERCENT SIGN [moved from EX]
2030	PER MILLE SIGN

2031	PER TEN THOUSAND SIGN
2032..2037	PRIME..REVERSED TRIPLE PRIME
20A7	PESETA SIGN
2103	DEGREE CELSIUS
2109	DEGREE FAHRENHEIT
FDFC	RIAL SIGN
FE6A	SMALL PERCENT SIGN
FF05	FULLWIDTH PERCENT SIGN
FFE0	FULLWIDTH CENT SIGN

Alphabetic characters are also widely used as unit designators in a postfix position. For purposes of line breaking, their classification as alphabetic is sufficient to keep them together with the preceding number.

PR: Prefix (Numeric) (XA)

Characters that usually precede a numerical expression may not be separated from following numeric characters or following opening characters, *even* if a space character intervenes. For example, there is no break opportunity in “$ (100.00)”.

Many currency signs can appear on both sides, or even the middle, of a numeric expression. Therefore the line breaking algorithm, by default, does not break between **PR** and numbers or letters on either side.

The **PR**
line break class consists of all currency symbols (General_Category Sc) except as listed explicitly in **PO**, as well as the following:

002B	PLUS SIGN
005C	REVERSE SOLIDUS
00B1	PLUS-MINUS
2116	NUMERO SIGN
2212	MINUS SIGN
2213	MINUS-OR-PLUS-SIGN

Note:
Many currency symbols may be used either as prefix or as postfix, depending on local convention. For details on the conventions used, see [CLDR].

QU: Ambiguous Quotation (XB/XA)

Some quotation characters can be opening or closing, or even both, depending on usage. The default is to treat them as both opening and closing. This will prevent some breaks that might have been legal for a particular language or usage, such as between a closing quote and a following opening punctuation.

Note:
If language information is available, it can be used to determine which character is used as the opening quote and which as the closing quote. See the information in *Section 6.2, General Punctuation*, in [Unicode5.0]. In such a case, the quotation marks could be tailored to either **OP** or **CL** depending on their actual usage.

The **QU**
line break class consists of characters of General_Category Pf or Pi in the Unicode Character

Database as well as

0022	QUOTATION MARK
0027	APOSTROPHE
275B	HEAVY SINGLE TURNED COMMA QUOTATION MARK ORNAMENT
275C	HEAVY SINGLE COMMA QUOTATION MARK ORNAMENT
275D	HEAVY DOUBLE TURNED COMMA QUOTATION MARK ORNAMENT
275E	HEAVY DOUBLE COMMA QUOTATION MARK ORNAMENT

U+23B6 BOTTOM SQUARE BRACKET OVER TOP SQUARE BRACKET is subtly different from the others in this class, in that it is *both*
an opening and a closing punctuation character at the same time. However, its use is limited to certain vertical text modes in terminal emulation. Instead of creating a one-of-a-kind class for this rarely used character, assigning it to the **QU** class approximates the intended behavior.

SA: Complex-Context Dependent (South East Asian) (P)

Runs of these characters require morphological analysis to determine break opportunities. This is similar to, for example, a hyphenation algorithm. For the characters that have this property, **no**
break opportunities will be found otherwise. Therefore complex context analysis, often involving dictionary lookup of some form, is required to determine non-emergency line breaks.

If such analysis is not available, it is recommended to treat them as **AL**.

> *Note:*
> These characters can be mapped into their equivalent line breaking classes as the result of dictionary lookup, thus permitting a logical separation of this algorithm from the morphological analysis.

The class **SA**
consists of all characters of General_Category Cf, Lo, Lm, Mn, or Mc in the following ranges, except as noted elsewhere:

0E00..0E7F	Thai
0E80..0EFF	Lao
1000..109F	Myanmar
1780..17FF	Khmer
1950..197F	Tai Le
1980..19DF	New Tai Lue

SG: Surrogates (XP) (Non-tailorable)

Line break class **SG**
comprises all code points with General_Category Cs. The line breaking behavior of isolated surrogates is undefined. In UTF-16, paired surrogates represent non-BMP code points. Such code points must be resolved before assigning line break properties. In UTF-8 and UTF-32 surrogate code points represent corrupted data and their line break behavior is undefined.

> *Note:*
> The use of this line breaking class is deprecated. It was of limited usefulness for UTF-16 implementations that did not support characters beyond the BMP. The correct implementation is to resolve a *pair* of surrogates into a supplementary character before

line breaking.

SP: Space (A) (Non-tailorable)

The space characters
are used as explicit break opportunities; they allow line breaks before most other characters. However, spaces at the end of a line are ordinarily not measured for fit. If there is a sequence of space characters, and breaking after any of the space characters would result in the same visible line, then the line breaking position after the last space character in the sequence is the locally most optimal one. In other words, when the last character measured for fit is *before* the space character, any number of space characters are kept together invisibly on the previous line and the first non-space character starts the next line.

0020 SPACE (SP)

Note: By default, SPACE, but none of the other breaking spaces, is used in determining an indirect break. For other breaking space characters, see **BA**.

SY: Symbols Allowing Break After (A)

The **SY**
line breaking property is intended to provide a break opportunity after, except in front of digits, so as to not break “1/2” or “06/07/99”.

002F SOLIDUS

URLs are now so common in regular plain text that they need to be taken into account when assigning general-purpose line breaking properties. Slash (*solidus*) is allowed as an additional, limited break opportunity to improve layout of Web addresses. As a side effect, some common abbreviations such as “w/o” or “A/S”, which normally would not be broken, acquire a line break opportunity. The recommendation in this case is for the layout system not to utilize a line break opportunity allowed by **SY** unless the distance between it and the next line break opportunity exceeds an implementation-defined minimal distance.

Note: Normally, symbols are treated as **AL**. However, symbols can be added to this line breaking class or classes **BA**, **BB**, and **B2** by tailoring. This can be used to allow additional line breaks—for example, after “=”. Mathematics requires additional specifications for line breaking, which are outside the scope of this annex.

WJ: Word Joiner (XB/XA) (Non-tailorable)

These characters glue together left and right neighbor characters such that they are kept on the same line.

2060 WORD JOINER (WJ)
FEFF ZERO WIDTH NO-BREAK SPACE (ZWNBSP)

The word joiner character is the preferred choice for an invisible character to keep other characters together that would otherwise be split across the line at a direct break. The character FEFF has the same effect, but because it is also used in an unrelated way as a *byte order mark,*
the use of the WJ as the preferred interword glue simplifies the handling of FEFF.

By definition, WJ and ZWNBSP take precedence over the action of **SP**, but not **ZW**.

XX: Unknown (XP)

The **XX**
line break class consists of all characters with General_Category Co and all code points with General_Category Cn.

Unassigned code positions, private-use characters, and characters for which reliable line breaking information is not available are assigned this default line breaking property by default. The default behavior for this class is identical to class **AL**. Users can manually insert ZWSP or WORD JOINER around characters of class **XX** to allow or prevent breaks as needed.

In addition, implementations can override or tailor this default behavior—for example, by assigning characters the property **ID** or another class. Doing so may give better default behavior for their users. There are other possible means of determining the desired behavior of private-use characters. For example, one implementation might treat any private-use character in ideographic context as **ID**, while another implementation might support a method for assigning specific properties to specific definitions of private-use characters. The details of such use of private-use characters are outside the scope of this standard.

For supplementary characters, a useful default is to treat characters in the range 10000..1FFFD as **AL** and characters in the ranges 20000..2FFFD and 30000..3FFFD as **ID**, until the implementation can be revised to take into account the actual line breaking properties for these characters.

For more information on handling default property values for unassigned characters, see the discussion on default property values in *Section 5.3, Unknown and Missing Characters*, of [Unicode5.0].

The line breaking rules in *Section 6, Line Breaking Algorithm*, and the pair table in *Section 7, Pair Table-Based Implementation*, assume that all unknown characters have been assigned one of the other line breaking classes, such as **AL**, as part of assigning line breaking classes to the input characters.

Implementations that do not support a given character should also treat it as unknown (**XX**).

ZW: Zero Width Space (A) (Non-tailorable)

200B	ZERO WIDTH SPACE (ZWSP)

This character is used to enable additional (invisible) break opportunities wherever SPACE cannot be used. As its name implies, it normally has no width. However, its presence between two characters does not prevent increased letter spacing in justification.

5.2 Dictionary Usage

Dictionaries follow specific conventions that guide their use of special characters to indicate features of the terms they list. Marks used for some of these conventions may occur near line break opportunities and therefore interact with line breaking. For example, in one dictionary a natural hyphen in a word becomes a tilde dash when the word is split.

Examples of conventions used in several dictionaries are briefly described in this subsection. Where possible, the line breaking properties for characters commonly used in dictionaries have been assigned to accommodate these and similar conventions by default. However, implementing the full conventions in dictionaries requires tailoring of line break classes and rules or other types of special support.

Looking up the noun “syllable” in eight dictionaries yields eight different conventions:

Dictionary of the English Language (Samuel Johnson, 1843) **SY´LLABLE** where ´ is an oversized U+02B9 and follows the vowel of the main syllable (not the syllable itself).

Oxford English Dictionary (1st Edition) **si·lă'bl** where · is a slightly raised middle dot indicating the vowel of the stressed syllable (similar to Johnson’s acute). The letter ă is U+0103. The ' is an apostrophe.

Oxford English Dictionary (2nd Edition) has gone to IPA **'slləb(ə)l** where **'** is U+02C8, l is U+026A, and ə is U+0259 (both times). The **'** comes before the stressed syllable. The () indicate the *schwa* may be omitted.

Chambers English Dictionary (7th Edition) **sil´ə-bl** where the stressed syllable is followed by ´ U+02B9, ə is U+0259, and **-** is a hyphen. When splitting a word like **abate´- ment**, the stress mark ´
goes after stressed syllable followed by the hyphen. No special convention is used when splitting at hyphen.

BBC English Dictionary **sllə<u>bl</u>** where <u>l</u> is <U+026A, U+0332> and ə is U+0259. The vowel of the stressed syllable is underlined.

Collins Cobuild English Language Dictionary **sllə<u>b</u>°<u>l</u>** where <u>l</u> is <U+026A, U+0332> and has the same meaning as in the *BBC English Dictionary*. The ə is U+0259 (both times). The **°** is a U+2070 and indicates the *schwa* may be omitted.

Readers Digest Great Illustrated Dictionary **syl·la·ble** (**sílləb'l**) The spelling of the word has hyphenation points (· is a U+2027) followed by phonetic spelling. The vowel of the stressed syllable is given an accent, rather than being followed by an accent. The ' is an apostrophe.

Webster’s 3rd New International Dictionary **syl·la·ble** /**'siləbəl**/ The spelling of the word has hyphenation points (·
is a U+2027) and is followed by phonetic spelling. The stressed syllable is preceded by **'** U+02C8. The ə’s are *schwas* as usual. *Webster’s* splits words at the end of a line with a normal hyphen. A U+2E17 DOUBLE OBLIQUE HYPHEN indicates that a hyphenated word is split at the hyphen.

Some dictionaries use a character that looks like a vertical series of four dots to indicate places where there is a syllable, but no allowable break. This can be represented by a sequence of U+205E VERTICAL FOUR DOTS followed by U+2060 WORD JOINER.

5.3 Use of Hyphen

The rules for treating hyphens in line breaking vary by language. In many instances, these rules are not supported as such in the algorithm, but the correct appearance can be realized by using a *non-breaking hyphen*.

Some languages and some transliteration systems use a hyphen at the first position in a word. For example, the Finnish orthography uses a hyphen at the start of a word in certain types of compounds of the form xxx yyy -zzz (where xxx yyy is a two-word expression that acts as the first part of a compound noun, with zzz as the second part). Line break after the hyphen is not allowed here; therefore, instead of a regular hyphen, U+2011 NON-BREAKING HYPHEN should be used.

There are line breaking conventions that modify the appearance of a line break when the line break opportunity is based on an explicit hyphen. In standard Polish orthography, explicit

hyphens are always promoted to the next line if a line break occurs at that location in the text. For example, if, given the sentence "Tam wisi czerwono-niebieska flaga" ("There hangs a red-blue flag"), the optimal line break occurs at the location of the explicit hyphen, an additional hyphen will be displayed at the beginning of the next line like this:

Tam wisi czerwono-
-niebieska flaga.

The same convention is used in Portuguese, where the use of hyphens is common, because they are mandatory for verbs forms that include a pronoun. Homographs or ambiguity may arise if hyphens are treated incorrectly: for example, "disparate" means "folly" while "dispara-te" means "fire yourself" (or "fires onto you"). Therefore the former needs to be line broken as

dispara-
te

and the latter as

dispara-
-te.

A recommended practice is to type <SHY, NBHY> instead of <HYPHEN> to achieve promotion of the hyphen to the next line. This practice is reportedly already common and supported by major text layout applications. See also *Section 5.4, Use of Soft Hyphen*.

5.4 Use of Soft Hyphen

Unlike U+2010 HYPHEN, which always has a visible rendition, the character U+00AD SOFT HYPHEN (SHY)
is an invisible format character that merely indicates a preferred intraword line break position. If the line is broken at that point, then whatever mechanism is appropriate for intraword line breaks should be invoked, just as if the line break had been triggered by another hyphenation mechanism, such as a dictionary lookup. Depending on the language and the word, that may produce different visible results—for example

- Simply inserting a hyphen glyph
- Inserting a hyphen glyph and changing spelling in the divided word parts
- Not showing any visible change and simply breaking at that point
- Inserting a hyphen glyph at the beginning of the new line

The following are a few examples of spelling changes. Each example shows the line break as " / " and any inserted hyphens. There are many other cases.

- In pre-reform German orthography, a "c" before the hyphenation point can change into a "k": "Drucker" hyphenates into "Druk- / ker".
- In modern Dutch, an *e-diaeresis*
  after the hyphenation point can change into a simple "e": "geërfde" hyphenates into "ge- / erfde", and "geëerd" into "ge-/ eerd".
- In German and Swedish, a consonant is sometimes doubled: Swedish "tuggummi"; hyphenates into "tugg- / gummi".
- In Dutch, a letter can disappear: "opaatje" hyphenates into "opa- / tje".

The inserted hyphen glyph can take a wide variety of shapes, as appropriate for the situation.

Examples include shapes like U+2010 HYPHEN, U+058A ARMENIAN HYPHEN, U+180A MONGOLIAN NIRUGU, or U+1806 MONGOLIAN TODO SOFT HYPHEN.

When a SHY is used to represent a possible hyphenation location, the spelling is that of the word without hyphenation: “tug<SHY>gummi”. It is up to the line breaking implementation to make any necessary spelling changes when such a possible hyphenation is actually used.

Sometimes it is desirable to encode text that includes line breaking decisions and will not be further broken into lines. If such text includes hyphenations, the spelling needs to reflect the changes due to hyphenation: “tugg<U+2010>/ gummi”, including the appropriate character for any inserted hyphen. For a list of dash-like characters in Unicode, see *Section 6.2, General Punctuation*, in [Unicode5.0].

Hyphenation, and therefore the SHY, can be used with the Arabic script. If the rendering system breaks at that point, the display—including shaping—should be what is appropriate for the given language. For example, sometimes a hyphen-like mark is placed on the end of the line. This mark looks like a *kashida*, but is not connected to the letter preceding it. Instead, the appearance of the mark is as if it had been placed — and the line divided — after the contextual shapes for the line have been determined. For more information on shaping, see [Bidi] and *Section 8.2, Arabic*, of [Unicode5.0].

There are three types of hyphens: explicit hyphens, conditional hyphens, and dictionary-inserted hyphens resulting from a hyphenation process. There is no character code for the third kind of hyphen. If a distinction is desired, the fact that a hyphen is dictionary-inserted and not user-supplied can only be represented out of band or by using another control code instead of SHY.

The action of a hyphenation algorithm is equivalent to the insertion of a SHY. However, when a word contains an explicit SHY, it is customarily treated as overriding the action of the hyphenator for that word.

The sequence <SHY, NBHY> is given a particular interpretation, see *Section 5.3, Use of Hyphen*.

5.5 Use of Double Hyphen

In some fonts, noticeably Fraktur fonts, it is customary to use a double-stroke form of the hyphen, usually oblique. Such use is merely a font-based glyph variation and does not affect line breaking in any way. In texts using such a font, automatic hyphenation or SHY would also result in the display of a double-stroke, oblique hyphen.

In some dictionaries, such as *Webster’s 3rd New International Dictionary*, double-stroke, oblique hyphens are used to indicate an explicit hyphen at the end of the line, in other words, a hyphen that would be retained when the term shown is not line wrapped. To support this, it is not necessary to store a special character in the data; one merely needs to substitute the glyph of any ordinary hyphen that winds up at the end of a line. For example, if the shape of the special hyphen, as in this case, matches an existing character, such as U+2E17 DOUBLE OBLIQUE HYPHEN, that character can be substituted temporarily for display purposes by the line formatter. In such convention, automatic hyphenation or SHY would result in the display of an ordinary hyphen without further substitution. (See also *Section 5.3, Use of Hyphen*).

Certain linguistic notations make use of a double-stroke, oblique hyphen to indicate specific features. The U+2E17 DOUBLE OBLIQUE HYPHEN character used in this case is not a hyphen and does not represent a line break opportunity. Automatic hyphenation or SHY would result in the display of an ordinary hyphen.

U+30A0 KATAKANA-HIRAGANA DOUBLE HYPHEN is used in scientific notation, for example, to mark the presence of a space that would otherwise have been lost in transcribing text, such as the name of a chemical compound, into Katakana. In such notation, ordinary hyphens are retained.

5.6 Tibetan Line Breaking

The Tibetan script uses spaces sparingly, relying instead on the *tsheg*. There is no punctuation equivalent to a period in Tibetan; Tibetan *shad* characters indicate the end of a "phrase," not a sentence. "Phrases" are often metrical—that is, written after every *N* syllables—and a new sentence can often start within the middle of a phrase. Sentence boundaries need to be determined grammatically rather than by punctuation.

Traditionally there is nothing akin to a paragraph in Tibetan text. It is typical to have many pages of text without a paragraph break—that is, without an explicit line break. The closest thing to a paragraph in Tibetan is a new section or topic starting with U+0F12 or U+0F08. However, these occur inline: one section ends and a new one starts on the same line, and the new section is marked only by the presence of one of these characters.

Some modern books, newspapers, and magazines format text more like English with a break before each section or topic—and (often) the title of the section on a separate line. Where this is done, authors insert an explicit line break. Western punctuation (full stop, question mark, exclamation mark, comma, colon, semicolon, quotes) is starting to appear in Tibetan documents, particularly those published in India, Bhutan, and Nepal. Because there are no formal rules for their use in Tibetan, they get treated generically by default. In Tibetan documents published in China, CJK bracket and punctuation characters occur frequently; it is recommended to treat these as in Chinese written horizontally.

> *Note:*
> The detailed rules for formatting Tibetan texts are complex, and the original assignment of line break classes was found to be wholly insufficient for the purpose. In [Unicode4.1], the assignment of line break classes for Tibetan was revised significantly in an attempt to better model Tibetan line breaking behavior. No new rules or line break classes were added.

The set of line break classes for Tibetan are expected to provide a good starting point, even though there is limited practical experience in their implementation. As more experience is gained, some modifications, possibly including new rules or additional line break classes, can be expected.

It is the stated intention of the Unicode Consortium to review these assignments in a future version and to furnish a more detailed and complete description of Tibetan line breaking and line formatting behavior.

5.7 Word Separator Characters

Visible word separator characters may behave in one of three ways at line breaks. As an example, consider the text "The:quick:brown:fox:jumped.", where the colon (:) represents a visible word separator, with a break between "brown" and "fox". The desired visual appearance could be one of the following:

1. suppress the visible word separator

   The:quick:brown
   fox:jumped.

2. break before the visible word separator

The:quick:brown
:fox:jumped.

3. break after the visible word separator

The:quick:brown:
fox:jumped.

Both (2) and (3) can be expressed with the Unicode Line Breaking Algorithm by tailoring the Line Break property value for the word separator character to be Break Before or Break After, respectively.

For case (1), the line break opportunity is positioned after the word separator character, as in case (3), but the visual display of the character is suppressed. The means by which a line layout and display process inhibits the visible display of the separator character are outside of the scope of the Line Break algorithm. U+1680 OGHAM SPACE MARK is an example of a character which may exhibit this behavior.

6 Line Breaking Algorithm

Unicode Standard Annex #29, "Text Boundaries" [Boundaries], describes a particular method for boundary detection. It is based on a set of hierarchical rules and character classifications. That method is well suited for implementation of some of the advanced heuristics for line breaking.

A slightly simplified implementation of such an algorithm can be devised that uses a two-dimensional table to resolve break opportunities between pairs or characters. It is described in *Section 7, Pair Table-Based Implementation*.

The line breaking algorithm presented in this section can be expressed in a series of rules that take line breaking classes defined in *Section 5.2, Description of Line Breaking Properties*, as input. The title of each rule contains a mnemonic summary of the main effect of the rule. The formal statement of each line breaking rules consists either of a remap rule or of one or more regular expressions containing one or more line breaking classes and one of three special symbols indicating the type of line break opportunity:

! Mandatory break at the indicated position

× No break allowed at the indicated position

÷ Break allowed at the indicated position

The rules are applied in order. That is, there is an implicit "otherwise" at the front of each rule following the first. It is possible to construct alternate sets of such rules that are fully equivalent. To be equivalent, an alternate set of rules must have the same effect.

The distinction between a direct break and an indirect break as defined in *Section 2, Definitions*, is handled in rule **LB18**, which explicitly considers the effect of **SP**. Because rules are applied in order, allowing breaks following **SP** in rule **LB18** implies that any prohibited break in rules **LB19**–**LB30** is equivalent to an indirect break.

The examples for each rule use representative characters, where 'H' stands for an ideographs, 'h' for small kana, and '9' for digits. Except where a rule contains no expressions,

the italicized text of the rule is intended merely as a handy summary.

The algorithm consists of a part for which tailoring is prohibited and a freely tailorable part.

6.1 Non-tailorable Line Breaking Rules

The rules in this subsection and the membership in the classes **BK**, **CM**, **CR**, **GL**, **LF**, **NL**, **SP**, **WJ**, and **ZW** ~~are not tailorable~~
define behavior that is required of all line break implementations; see *Section 4, Conformance*.

Resolve line breaking classes:

LB1 *Assign a line breaking class to each code point of the input. Resolve **AI**, **CB**, **SA**, **SG**, and **XX** into other line breaking classes depending on criteria outside the scope of this algorithm.*

In the absence of such criteria, it is recommended that classes **AI**, **SA**, **SG**, and **XX** be resolved to **AL**, except that characters of class **SA** that have General_Category Mn or Mc be resolved to **CM** (see **SA**). Unresolved class **CB** is handled in rule **LB20**.

Start and end of text:

There are two special logical positions: **sot**, which occurs before the first character in the text, and **eot,**
which occurs after the last character in the text. Thus an empty string would consist of **sot** followed immediately by **eot**. With these two definitions, the line break rules for start and end of text can be specified as follows:

LB2 *Never break at the start of text.*

sot ×

LB3 *Always break at the end of text.*

! eot

These two rules are designed to deal with degenerate cases, so that there is at least one character on each line, and at least one line break for the whole text. Emergency line breaking behavior usually also allows line breaks anywhere on the line if a legal line break cannot be found. This has the effect of preventing text from running into the margins.

Mandatory breaks:

A hard line break can consist of **BK** or a Newline Function (NLF) as described in *Section 5.8, Newline Guidelines*, of [Unicode5.0]. These three rules are designed to handle the line ending and line separating characters as described there.

LB4 *Always break after hard line breaks.*

BK !

LB5 *Treat **CR** followed by **LF**, as well as **CR**, **LF**, and **NL** as hard line breaks.*

CR × LF

CR !

LF !

NL !

LB6 *Do not break before hard line breaks.*

× (BK | CR | LF | NL)

Explicit breaks and non-breaks:

LB7 *Do not break before spaces or zero width space.*

× SP

× ZW

LB8 *Break after zero width space.*

ZW ÷

Combining marks:

See also *Section 9.2, Legacy Support for Space Character as Base for Combining Marks.*

LB9 *Do not break a combining character sequence; treat it as if it has the line breaking class of the base character in all of the following rules.*

Treat X CM* as if it were X.

where X is any line break class except **BK**, **CR**, **LF**, **NL**, **SP**, or **ZW**.

At any possible break opportunity between **CM** and a following character, **CM** behaves as if it had the type of its base character. Note that despite the summary title of this rule it is not limited to standard combining character sequences. For the purposes of line breaking, sequences containing most of the control codes or layout control characters are treated like combining sequences.

LB10 *Treat any remaining combining mark as **AL**.*

Treat any remaining CM as it if were AL.

This catches the case where a **CM** is the first character on the line or follows **SP**, **BK**, **CR**, **LF**, **NL**, or **ZW**.

Word joiner:

LB11 *Do not break before or after Word joiner and related characters.*

× WJ

WJ ×

Non-breaking characters:

LB12 *Do not break ~~before or~~ after NBSP and related characters.*

~~[^SP] × GL~~

GL ×

6.2 Tailorable Line Breaking Rules

The following rules and the classes referenced in them provide a reasonable default set of line break opportunities. Implementations SHOULD implement them unless alternate approaches produce better results for some classes of text or applications. When using alternative rules or algorithms, implementations must ensure that the mandatory breaks, break opportunities and non-break positions determined by the algorithm and rules of section 6.1 are preserved.

~~*The following rules and classes referenced in them can be tailored by a conformant implementation;*~~ *see Section 4, Conformance.*

Non-breaking characters:

LB12a *Do not break before NBSP and related characters, except after spaces and hyphens*

[^SP, BA, HY] × GL

The expression [^SP, BA, HY] designates any line break class other than **SP**, **BA** or **HY**. The symbol ^ is used, instead of !, to avoid confusion with the use of ! to indicate an explicit break. Unlike the case for **WJ**, inserting a **SP** overrides the non-breaking nature of a **GL**. Allowing a break after **BA** or **HY**
matches wide-spread implementation practice and supports a common way of handling special line breaking of explicit hyphens, such as in Polish and Portuguese . See *Section 5.3, Use of Hyphen*.

Opening and closing:

These have special behavior with respect to spaces, and therefore come before rule LB18.

LB13 *Do not break before ‘]’ or ‘!’ or ‘;’ or ‘/’, even after spaces.*

× CL

× EX

× IS

× SY

LB14 *Do not break after ‘[’, even after spaces.*

OP SP* ×

LB15 *Do not break within ‘”[’, even with intervening spaces.*

QU SP* × OP

For more information on this rule, see the note in the description for the **QU** class.

LB16 *Do not break between closing punctuation and a nonstarter (lb=**NS**), even with intervening spaces.*

CL SP* × NS

LB17 *Do not break within '——', even with intervening spaces.*

B2 SP* × B2

Spaces:

LB18 *Break after spaces.*

SP ÷

Special case rules:

LB19 *Do not break before or after quotation marks, such as ' " '.*

× QU

QU ×

LB20 *Break before and after unresolved* ***CB****.*

÷ CB

CB ÷

Conditional breaks should be resolved external to the line breaking rules. However, the default action is to treat unresolved **CB** as breaking before and after.

LB21 *Do not break before hyphen-minus, other hyphens, fixed-width spaces, small kana, and other non-starters, or after acute accents.*

× BA

× HY

× NS

BB ×

LB22 *Do not break between two ellipses, or between letters or numbers and ellipsis.*

AL × IN

ID × IN

IN × IN

NU × IN

Examples: '9...', 'a...', 'H...'

Numbers:

Do not break alphanumerics.

LB23 *Do not break within 'a9', '3a', or 'H%'.*

ID × PO

AL × NU

NU × AL

LB24 *Do not break between prefix and letters or ideographs.*

PR × ID

PR × AL

PO × AL

In general, it is recommended to not break lines inside numbers of the form described by the following regular expression:

(**PR** | **PO**) ? (**OP** | **HY**) ? **NU** (**NU** | **SY** | **IS**) * **CL** ? (**PR** | **PO**) ?

Examples: $(12.35) 2,1234 (12)¢ 12.54¢

The default line breaking algorithm approximates this with the following rule. Note that some cases have already been handled, such as '9,', '[9'. For a tailoring that supports the regular expression directly, as well as a key to the notation see *Section 8.2, Examples of Customization*.

LB25 *Do not break between the following pairs of classes relevant to numbers:*

CL × PO

CL × PR

NU × PO

NU × PR

PO × OP

PO × NU

PR × OP

PR × NU

HY × NU

IS × NU

NU × NU

SY × NU

Example pairs: '$9', '$[', '$-', '-9', '/9', '99', ',9', '9%' ']%'

Korean syllable blocks

Conjoining jamo, Hangul syllables, or combinations of both form Korean Syllable Blocks. Such blocks are effectively treated as if they were Hangul syllables; no breaks can occur in

the middle of a syllable block. See Unicode Standard Annex #29, “Text Boundaries” [Boundaries], for more information on Korean Syllable Blocks.

LB26 *Do not break a Korean syllable.*

JL × (JL | JV | H2 | H3)

(JV | H2) × (JV | JT)

(JT | H3) × JT

where the notation (JT | H3) means JT or H3. The effective line breaking class for the syllable block matches the line breaking class for Hangul syllables, which is **ID** by default. This is achieved by the following rule:

LB27 *Treat a Korean Syllable Block the same as **ID**.*

(JL | JV | JT | H2 | H3) × IN

(JL | JV | JT | H2 | H3) × PO

PR × (JL | JV | JT | H2 | H3)

When Korean uses SPACE for line breaking, the classes in rule **LB26**, as well as characters of class **ID**, are often tailored to **AL**; see *Section 8, Customization.*

Finally, join alphabetic letters into words and break everything else.

LB28 *Do not break between alphabetics (“at”).*

AL × AL

LB29 *Do not break between numeric punctuation and alphabetics (“e.g.”).*

IS × AL

LB30 *Do not break between letters, numbers, or ordinary symbols and opening or closing punctuation.*

(AL | NU) × OP

CL × (AL | NU)

The purpose of this rule is to prevent breaks in common cases where a part of a word appears between delimiters—for example, in “person(s)”.

LB31 *Break everywhere else.*

ALL ÷

÷ ALL

7 Pair Table-Based Implementation

A two-dimensional table can be used to resolve break opportunities between pairs of characters. This section defines such a table. The rows of the table are labeled with the possible values of the line breaking property of the leading character in the pair. The columns are labeled with the line breaking class for the following character of the pair. Each

intersection is labeled with the resulting line break opportunity.

The Japanese standard JIS X 4051-1995 [JIS] provides an example of a similar table-based definition. However, it uses line breaking classes whose membership is not solely determined by the line breaking property (as in this annex), but in some cases by heuristic analysis or markup of the text.

The implementation provided here directly uses the line breaking classes defined previously.

7.1 Minimal Table

If two rows of the table have identical values and the corresponding columns also have identical values, then the two line breaking classes can be coalesced. For example, the JIS standard uses 20 classes, of which only 14 appear to be unique. Any minimal table representation is unique, except for trivial reordering of rows and columns. Minimal tables for which the rows and columns are sorted alphabetically can be mechanically compared for differences. This is in contrast to the rules, where identical results can be achieved by sets of rules that cannot be easily compared by looking at their textual representation. However, any set of rules that is equivalent to a minimal pair table can be used to automatically generate such a table, which can then be used for comparison. The rules in *Section 6, Line Breaking Algorithm* can be expressed as minimal pair tables if the extended context used as described below.

7.2 Extended Context

Most of the rules in *Section 6, Line Breaking Algorithm*, involve only pairs of characters, or they apply to a single line break class preceded or followed by any character. These rules can be represented directly in a pair table. However, rules **LB14**–**LB17** require extended context to handle spaces.

By broadening the definition of a pair from **B A**, where **B** is the line breaking class before a break and **A** the one after, to **B SP* A**, where **SP*** is an optional run of space characters, the same table can be used to distinguish between cases where **SP** can or cannot provide a line break opportunity (that is, direct and indirect breaks). Rules equivalent to the ones given in *Section 6, Line Breaking Algorithm*, can be formulated without explicit use of **SP** by using **%** to express indirect breaks instead. These rules can then be simplified to involve only pairs of classes—that is, only constructions of the form:

B ÷ A

B % A

B × A

where either **A** or **B**
may be empty. These simplified rules can be automatically translated into a pair table, as in *Table 2*. Line breaking analysis then proceeds by pair table lookup as explained below. (For readability in table layout, the symbol ^ is used in the table instead of × and _ is used instead of ÷.)

Rule **LB9**
requires extended context for handling combining marks. This extended context must also be built into the code that interprets the pair table. For convenience in detecting the condition where **A** = **CM**, the symbols # and @ are used in the pair table, instead of % and ^, respectively. See *Section 7.5, Combining Marks*.

7.3 Example Pair Table

Table 2
implements the line breaking behavior described in this annex, with the limitation that only context of the form **B SP* A** is considered. **BK**, **CR**, **LF**, **NL**, and **SP** classes are handled explicitly in the outer loop, as given in the code sample below. Pair context of the form **B CM*** can be handled by handling the special entries @ and # in the driving loop, as explained in *Section 7.5, Combining Marks*. Conjoining jamos are considered separately in *Section 7.6, Conjoining Jamos*. In *Table 2,* the rows are labeled with the **B** class and the columns are labeled with the **A** class.

Table 2. Example Pair Table

	OP	CL	QU	GL	NS	EX	SY	IS	PR	PO	NU	AL	ID	IN	HY	BA	BB	B2	ZW	CM	WJ	H2	H3	JL	JV	JT
OP	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	@	^	^	^	^	^	^
CL	_	^	%	%	^	^	^	^	%	%	%	%	_	_	%	%	_	_	^	#	^	_	_	_	_	_
QU	^	^	%	%	%	^	^	^	%	%	%	%	%	%	%	%	%	%	^	#	^	%	%	%	%	%
GL	%	^	%	%	%	^	^	^	%	%	%	%	%	%	%	%	%	%	^	#	^	%	%	%	%	%
NS	_	^	%	%	%	^	^	^	_	_	_	_	_	_	%	%	_	_	^	#	^	_	_	_	_	_
EX	_	^	%	%	%	^	^	^	_	_	_	_	_	_	%	%	_	_	^	#	^	_	_	_	_	_
SY	_	^	%	%	%	^	^	^	_	_	%	_	_	_	%	%	_	_	^	#	^	_	_	_	_	_
IS	_	^	%	%	%	^	^	^	_	_	%	%	_	_	%	%	_	_	^	#	^	_	_	_	_	_
PR	%	^	%	%	%	^	^	^	_	_	%	%	%	_	%	%	_	_	^	#	^	%	%	%	%	%
PO	%	^	%	%	%	^	^	^	_	_	%	%	_	_	%	%	_	_	^	#	^	_	_	_	_	_
NU	%	^	%	%	%	^	^	^	%	%	%	%	_	%	%	%	_	_	^	#	^	_	_	_	_	_
AL	%	^	%	%	%	^	^	^	_	_	%	%	_	%	%	%	_	_	^	#	^	_	_	_	_	_
ID	_	^	%	%	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	#	^	_	_	_	_	_
IN	_	^	%	%	%	^	^	^	_	_	_	_	_	%	%	%	_	_	^	#	^	_	_	_	_	_
HY	_	^	%	_	%	^	^	^	_	_	%	_	_	_	%	%	_	_	^	#	^	_	_	_	_	_
BA	_	^	%	_	%	^	^	^	_	_	_	_	_	_	%	%	_	_	^	#	^	_	_	_	_	_
BB	%	^	%	%	%	^	^	^	%	%	%	%	%	%	%	%	%	%	^	#	^	%	%	%	%	%
B2	_	^	%	%	%	^	^	^	_	_	_	_	_	_	%	%	_	^	^	#	^	_	_	_	_	_
ZW	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	^	_	_	_	_	_	_	_
CM	_	^	%	%	%	^	^	^	_	_	%	%	_	%	%	%	_	_	^	#	^	_	_	_	_	_
WJ	%	^	%	%	%	^	^	^	%	%	%	%	%	%	%	%	%	%	^	#	^	%	%	%	%	%
H2	_	^	%	%	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	#	^	_	_	_	%	%
H3	_	^	%	%	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	#	^	_	_	_	_	%
JL	_	^	%	%	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	#	^	%	%	%	%	_
JV	_	^	%	%	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	#	^	_	_	_	%	%
JT	_	^	%	%	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	#	^	_	_	_	_	%

Resolved outside the pair table: **AI**, **BK**, **CB**, **CR**, **LF**, **NL**, **SA**, **SG**, **SP**, **XX**

Table 2 uses the following notation:

^ denotes a *prohibited break*: B ^ A is equivalent to **B SP* × A**; in other words, never break before A and after B, even if one or more spaces intervene.

% denotes an *indirect break opportunity:* B % A is equivalent to **B × A** *and* **B SP+ ÷ A**; in other words, do not break before A, unless one or more spaces follow B.

@ denotes a *prohibited break for combining marks:* B @ A is equivalent to **B SP* × A**, where A is of class **CM**. For more details, see *Section 7.5, Combining Marks*.

denotes an *indirect break opportunity for combining marks following a space*: B # A is equivalent to (**B × A** *and* **B SP+ ÷ A**), where A is of class **CM**.

_ denotes a *direct break opportunity* (equivalent to ÷ as defined above).

Note:
In the online edition, hovering over the cells in a browser with tool-tips enabled reveals the rule number that determines the breaking status for the pair in question. When a pair must be tested both with and without intervening spaces, multiple rules are given. Hovering over a line breaking class name gives a representative member of the class and additional information. Clicking on any line break class name anywhere in the document jumps to the definition.

7.4 Sample Code

The following two sections provide sample code [Code14] that demonstrates how the pair table is used. For a complete implementation of the line breaking algorithm, `if` statements to handle the line breaking classes **CR**, **LF**, and **NL** need to be added. They have been omitted here for brevity, but see *Section 7.7, Explicit Breaks*.

The sample code assumes that the line breaking classes **AI**, **CB**, **SG**, and **XX** have been resolved according to rule **LB1** as part of initializing the `pcls` array. The code further assumes that the complex line break analysis for characters with line break class **SA** is handled in function `findComplexBreak`, for which the following placeholder is given:

```
// placeholder function for complex break analysis
// cls - resolved line break class, may differ from pcls[0]
// pcls - pointer to array of line breaking classes (input)
// pbrk - pointer to array of line breaking opportunities (output)
// cch - remaining length of input
int
findComplexBreak(enum break_class cls, enum break_class *pcls,
                         enum break_action *pbrk, int cch)
{
        if (!cch)
            return 0;
        for (int ich = 1; ich < cch; ich++) {

            // .. do complex break analysis here
            // and report any break opportunities in pbrk ..

            pbrk[ich-1] = PROHIBITED_BRK; // by default, no break

            if (pcls[ich] != SA)
                break;
        }
        return ich;
}
```

The entries in the example pair table correspond to the following enumeration. For diagnostic purposes, the sample code returns these value to indicate not only the location but also the type of rule that triggered a given break opportunity.

```
enum break_action {
       DIRECT_BRK = 0,             // _ in table
       INDIRECT_BRK,               // % in table
       COMBINING_INDIRECT_BRK,     // # in table
       COMBINING_PROHIBITED_BRK,   // @ in table
       PROHIBITED_BRK,             // ^ in table
       EXPLICIT_BRK };             // ! in rules
```

Because the contexts involved in indirect breaks of the form **B SP* A** are of indefinite length, they need to be handled explicitly in the driver code. The sample implementation of a `findLineBrk`
function below remembers the line break class for the last characters seen, but skips any occurrence of **SP** without resetting this value. Once character **A** is encountered, a simple lookback is used to see if it is preceded by a **SP**. This lookback is necessary only if **B** % **A**.

To handle the case of a **SP** following **sot**, it is necessary to set `cls` to a dummy value. Using **WJ** gives the correct result and, as required, is unaffected by any tailoring.

```
// handle spaces separately, all others by table
// pcls - pointer to array of line breaking classes (input)
// pbrk - pointer to array of line break opportunities (output)
// cch - number of elements in the arrays (“count of characters”) (input)
// ich - current index into the arrays (variable) (returned value)
// cls - current resolved line break class for 'before' character (variable)

int
findLineBrk(enum break_class *pcls, enum break_action *pbrk, int cch)
{
    if (!cch) return 0;

    enum break_class cls = pcls[0];   // class of 'before' character

    // treat SP at start of input as if it followed a WJ
    if (cls == SP)
        cls = WJ;

    // loop over all pairs in the string up to a hard break
    for (int ich = 1; (ich < cch) && (cls != BK); ich++) {

        // to handle explicit breaks, replace code from "for" loop condition
        // above to comment below by code given in Section 7.7

        // handle spaces explicitly
        if (pcls[ich] == SP) {
            pbrk[ich-1] = PROHIBITED_BRK;   // apply rule LB7: × SP
            continue;                       // do not update cls
        }

        // handle complex scripts in a separate function
        if (pcls[ich] == SA) {
            ich += findComplexBreak(cls, &pcls[ich-1], &pbrk[ich-1],
                        cch - (ich-1));
            if (ich < cch)
                cls = pcls[ich];
            continue;
        }

        // lookup pair table information in brkPairs[before, after];
        enum break_action brk = brkPairs[cls][pcls[ich]];

        pbrk[ich-1] = brk;                      // save break action in output array

        if (brk == INDIRECT_BRK) {              // resolve indirect break
            if (pcls[ich - 1] == SP)            // if context is A SP + B
                pbrk[ich-1] = INDIRECT_BRK;     //       break opportunity
            else                                // else
                pbrk[ich-1] = PROHIBITED_BRK;   //       no break opportunity
        }

        // handle breaks involving a combining mark (see Section 7.5)

        // save cls of 'before' character (unless bypassed by 'continue')
        cls = pcls[ich];
    }
    pbrk[ich-1] = EXPLICIT_BRK;                 // always break at the end

    return ich;
}
```

The function returns all of the break opportunities in the array pointed to by `pbrk`, using the values in the table. On return, `pbrk[ich]` is the type of break after the character at index `ich`.

A common optimization in implementation is to determine only the nearest line break opportunity prior to the position of the first character that would cause the line to become

overfull. Such an optimization requires backward traversal of the string instead of forward traversal as shown in the sample code.

7.5 Combining Marks

The implementation of combining marks in the pair table presents an additional complication because rule **LB9** defines a context **X CM*** that is of arbitrary length. There are some similarities to the way contexts of the form **B SP* A** that are involved in indirect breaks are evaluated. However, contexts of the form **SP CM*** or **CM* SP** also need to be handled, while rule **LB10** requires some **CM*** to be treated like **AL**.

Implementing LB10. This rule can be reflected directly in the example pair table in *Table 2* by assigning the same values in the row marked **CM** as in the row marked **AL**. Incidentally, this is equivalent to rewriting the rules **LB11**–**LB31** by duplicating any expression that contains an **AL** on its lefthand side with another expression that contains a **CM**. For example, in **LB22**

AL × IN

would become

AL × IN

CM × IN

Rewriting these rules as indicated here (and then deleting **LB10**) is fully equivalent to the original rules because rule **LB9** already accounts for all **CM**s that are not supposed to be treated like **AL**. For complete prescription see Example 9 in *Section 8.2, Examples of Customization*.

Implementing LB9. Rule **LB9** is implemented in the example pair table in *Table 2* by assigning a special # entry in the column marked **CM** for all rows referring to a line break class that allows a direct or indirect break after itself. (Note that the intersection between the row for class **ZW** and the column for class **CM** must be assigned "_" because of rule **LB8**.) The # corresponds to a `break_action` value of `COMBINING_INDIRECT_BREAK`, which triggers the following code in the sample implementation:

```
else if (brk == COMBINING_INDIRECT_BRK) {    // resolve combining mark break
    pbrk[ich-1] = PROHIBITED_BRK;             // do not break before CM
    if (pcls[ich-1] == SP){
        #ifndef LEGACY_CM                     // new: space is not a base
            pbrk[ich-1] = COMBINING_INDIRECT_BRK;    // apply rule SP ÷
        #else
            pbrk[ich-1] = PROHIBITED_BRK;      // legacy: keep SP CM together
            if (ich > 1)
                pbrk[ich-2] = ((pcls[ich - 2] == SP) ?
                                              INDIRECT_BRK : DIRECT_BRK);
        #endif
    } else                                    // apply rule LB9: X CM * -> X
        continue;                             // do not update cls
}
```

When handling a `COMBINING_INDIRECT_BREAK`, the last remembered line break class in variable `cls` is *not* updated, except for those cases covered by rule **LB10**. A *tailoring* of rule **LB9** that keeps the last SPACE
character preceding a combining mark, if any, and therefore breaks before that SPACE character can easily be implemented as shown in the sample code. (See *Section 9.2, Legacy Support for Space Character as Base for Combining Marks*.)

Any rows in *Table 2* for line break classes that prohibit breaks after must be handled explicitly. In the example pair table, these are assigned a special entry "@", which corresponds to a special break action of `COMBINING_PROHIBITED_BREAK` that triggers the following code:

```
else if (brk == COMBINING_PROHIBITED_BRK) { // this is the case OP SP* CM
    pbrk[ich-1] = COMBINING_PROHIBITED_BRK;  // no break allowed
    if (pcls[ich-1] != SP)
        continue;                           // apply rule LB9: X CM* -> X
}
```

The only line break class that unconditionally prevents breaks across a following **SP** is **OP**. The preceding code fragment ensures that **OP CM** is handled according to rule **LB9** and **OP SP CM** is handled as **OP SP AL** according to rule **LB10**.

7.6 Conjoining Jamos

For Korean Syllable Blocks, the information in rule **LB26** is represented by a simple pair table shown in *Table 3*.

Table 3. Korean Syllable Block Pair Table

	H2	H3	JL	JV	JT
H2	_	_	_	%	%
H3	_	_	_	_	%
JL	%	%	%	%	_
JV	_	_	_	%	%
JT	_	_	_	_	%

When constructing a pair table such as *Table 2*, this pair table for Korean syllable blocks in *Table 3*
is merged with the main pair table for all other line break classes by adding the cells from *Table 3* beyond the lower-right corner of the main pair table. Next, according to rule **LB27**, any empty cells in the new rows are filled with the same values as in the existing row for class **ID**, and any empty cells for the new columns are filled with the same values as in the existing column for class **ID**. The resulting merged table is shown in *Table 2*.

7.7 Explicit Breaks

Handling explicit breaks is straightforward in the driver code, although it does clutter up the loop condition and body of the loop a bit. For completeness, the following sample shows how to change the loop condition and add `if` statements—both before and inside the loop—that handle **BK**, **NL**, **CR**, and **LF**. Because **NL** and **BK** behave identically by default, this code can be simplified in implementations where the character classification is changed so that **BK** will always be substituted for **NL**
when assigning the line break class. Because this optimization does not change the result, it is not considered a tailoring and does not affect conformance.

```
// handle case where input starts with an LF
if (cls == LF)
     cls = BK;

// treat initial NL like BK
if (cls == NL)
     cls = BK;

// loop over all pairs in the string up to a hard break or CRLF pair
for (int ich = 1; (ich < cch) && (cls != BK) && (cls != CR || pcls[ich] == LF); ich++) {

    // handle BK, NL and LF explicitly
    if (pcls[ich] == BK ||pcls[ich] == NL ||  pcls[ich] == LF)
    {
```

```
        pbrk[ich-1] = PROHIBITED_BRK;
        cls = BK;
        continue;
    }

    // handle CR explicitly
    if(pcls[ich] == CR)
    {
        pbrk[ich-1] = PROHIBITED_BRK;
        cls = CR;
        continue;
    }

    // handle spaces explicitly...
```

8 Customization

A real-world line breaking algorithm has to be tailorable to some degree to meet user or document requirements.

In Korean, for example, two distinct line breaking modes occur, which can be summarized as breaking after each character or breaking after spaces (as in Latin text). The former tends to occur when text is set justified; the latter, when ragged margins are used. In that case, even ideographs are broken only at space characters. In Japanese, for example, tighter and looser specifications of prohibited line breaks may be used.

Specialized text or specialized text constructs may need specific line breaking behavior that differs from the default line breaking rules given in this annex. This may require additional tailorings beyond those considered in this section. For example, the rules given here are insufficient for mathematical equations, whether inline or in display format. Likewise, text that commonly contains lengthy URLs might benefit from special tailoring that suppresses **SY** × **NU** from rule **LB25**
within the scope of a URL to allow breaks after a “/” separated segment in the URL regardless of whether the next segment starts with a digit.

The remainder of this section gives an overview of common types of tailorings and examples of how to customize the pair table implementation of the line breaking algorithm for these tailorings.

8.1 Types of Tailoring

There are three principal ways of tailoring the sample pair table implementation of the line breaking algorithm:

1. **Changing the line breaking class assignment for some characters**
   This is useful in cases where the line breaking properties of one class of characters are occasionally lumped together with the properties of another class to achieve a less restrictive line breaking behavior.
2. **Changing the table value assigned to a pair of character classes**
   This is particularly useful if the behavior can be expressed by a change at a limited number of pair intersections. This form of customization is equivalent to permanently overriding some of the rules in *Section 6, Line Breaking Algorithm*.
3. **Changing the interpretation of the line breaking actions**
   This is a dynamic equivalent of the preceding. Instead of changing the values for the pair intersection directly in the table, they are labeled with special values that cause different actions for different customizations. This is most suitable when customizations need to be enabled at run time.

Beyond these three straightforward customization steps, it is always possible to augment the algorithm itself—for example, by providing specialized rules to recognize and break common constructs, such as URLs, numeric expressions, and so on. Such open-ended customizations place no limits on possible changes, other than the requirement that characters with normative line breaking properties be correctly implemented.

Reference [Cedar97] reports on a real-world implementation of a pair table-based implementation of a line breaking algorithm substantially similar to the one presented here, and including the types of customizations presented in this section. That implementation was able to simultaneously meet the requirements of customers in many European and East Asian countries with a single implementation of the algorithm.

8.2 Examples of Customization

Example 1. The exact method of resolving the line break class for characters with class **SA** is not specified in the default algorithm. One method of implementing line breaks for complex scripts is to invoke context-based classification for all runs of characters with class **SA**. For example, a dictionary-based algorithm could return different classes for Thai letters depending on their context: letters at the start of Thai words would become **BB** and other Thai letters would become **AL**. Alternatively, for text consisting of or predominantly containing characters with line breaking class **SA**, it may be useful to instead defer the determination of line breaks to a different algorithm entirely. *Section 7.4, Sample Code*, sketches such approach in which the interface to the dictionary-based algorithm directly reports break opportunities.

Example 2. To implement terminal style line breaks, it would be necessary to allow breaks at fixed positions. These could occur inside a run of spaces or in the middle of words without regard to hyphenation. Such a modification essentially disregards the output of the linebreaking algorithm, and is therefore not a conformant tailoring. For a system that supports both regular line breaking and terminal style line breaks, only some of its line break modes would be conformant.

Example 3.
Depending on the nature of the document, Korean either uses implicit breaking around characters (type 2 as defined above in *Section 3, Introduction*) or uses spaces (type 1). Space-based layout is common in magazines and other informal documents with ragged margins, while books, with both margins justified, use the other type, as it affords more line break opportunities and therefore leads to better justification. Reference [Suign98] shows how the necessary customizations can be elegantly handled by selectively altering the interpretation of the pair entries. Only the intersections of **ID**/**ID**, **AL**/**ID**, and **ID**/**AL** are affected. For alphabetic style line breaking, breaks for these cases require space; for ideographic style line breaking, these cases do not require spaces. Therefore, one defines a pseudo-action, which is then resolved into either direct or indirect break action based on user selection of the preferred behavior for a given text.

Example 4.
Sometimes in a Far Eastern context it is necessary to allow alphabetic characters and digit strings to break anywhere. According to reference [Suign98], this can again be done in the same way as Korean. In this case the intersections of **NU**/**NU**, **NU**/**AL**, **AL**/**AL**, and **AL**/**NU** are affected.

Example 5.
Some users prefer to relax the requirement that Kana syllables be kept together. For example, the syllable *kyu,* spelled with the two kanas *KI* and “small *yu*”, would no longer be kept together as if *KI* and *yu*

were atomic. This customization can be handled via the first method by changing the classification of the Kana small characters from **NS** to **ID** as needed.

Example 6.
Some implementations may wish to tailor the line breaking algorithm to resolve grapheme clusters according to Unicode Standard Annex #29, “Text Boundaries” [Boundaries], as a first stage. Generally, the line breaking algorithm does not create line break opportunities within default grapheme clusters; therefore such a tailoring would be expected to produce results that for most practical cases are close to what are defined by the default algorithm. However, if such a tailoring is chosen, characters that are members of line break class **CM** but not part of the definition of default grapheme clusters must still be handled by rules **LB9** and **LB10**, or by some additional tailoring.

Example 7.
Regular expression-based line breaking engines might get better results using a tailoring that directly implements the following regular expression for numeric expressions:

(**PR** | **PO**) ? (**OP** | **HY**) ? **NU** (**NU** | **SY** | **IS**) * **CL** ? (**PR** | **PO**) ?

This is equivalent to replacing the rule **LB25** by the following tailored rule:

Regex-Number: Do not break numbers.

(PR | PO) × (OP | HY)? NU

(OP | HY) × NU

NU × (NU | SY | IS)

NU (NU | SY | IS)* × (NU | SY | IS | CL)

NU (NU | SY | IS)* CL? × (PO | PR)

This customized rule uses extended contexts that cannot be represented in a pair table. In these tailored rules, (PR | PO) means **PR** or **PO**, the Symbol “?” means 0 or one occurrence and the symbol “*” means 0 or more occurrences. The last two rules can have a left side of any non-zero length.

When the tailored rule is used, **LB13** need to be tailored as follows:

[^NU] × CL

× EX

[^NU] × IS

[^NU] × SY

Otherwise, single digits might be handled by rule **LB13** before being handled in the regular expression. In these tailored rules [^NU] designates any line break class other than **NU**. The symbol ^ is used, instead of !, to avoid confusion with the use of ! to indicate an explicit break.

Example 8. For some implementations it may be difficult to implement **LB9** due to the added complexity of its indefinite length context. Because combining marks are most commonly applied to characters of class **AL**, rule **LB10** by itself generally produces acceptable results for such implementations, but such an approximation is not a conformant tailoring.

9 Implementation Notes

This section provides additional notes on implementation issues.

9.1 Combining Marks in Regular Expression-Based Implementations

For implementations that use regular expressions, it is not possible to directly express rules **LB9** and **LB10**. However, it is possible to make these rules unnecessary by rewriting *all* the rules from **LB11**
on down so that the overall result of the algorithm is unchanged. This restatement of the rules is therefore not a tailoring, but rather an equivalent statement of the algorithm that can be directly expressed as regular expressions.

To replace rule **LB9**, terms of the form

B # A

B SP* # **A**

B #

B SP* #

are replaced by terms of the form

B CM* # **A**

B CM* SP* # **A**

B CM* #

B CM* SP* #

where **B** and **A**
are any line break class or set of alternate line break classes, such as (X |Y), and where # is any of the three operators !, ÷, or ×.

Note that because **sot**, **BK**, **CR**, **LF**, **NL**, and **ZW** are all handled by rules above **LB9**, these classes cannot occur in position **B** in any rule that is rewritten as shown here.

Replace **LB10** by the following rule:

× CM

For each rule containing AL on its left side, add a rule that is identical except for the replacement of AL by CM, but taking care of correctly handling sets of alternate line break classes. For example, for rule

(AL | NU) × OP

add another rule

CM × OP.

These prescriptions for rewriting the rules are, in principle, valid even where the rules have been tailored as permitted in *Section 4, Conformance*. However, for extended context rules such as in Example 7, additional considerations apply. These are described in *Section 6.2,*

Replacing Ignore Rules, of Unicode Standard Annex #29, "Text Boundaries" [Boundaries].

9.2 Legacy Support for Space Character as Base for Combining Marks

As stated in [Unicode5.0], *Section 7.9, Combining Marks*, combining characters are shown in isolation by applying them to U+00A0 NO-BREAK SPACE (NBSP). In earlier versions, this recommendation included the use of U+0020 SPACE. This use of SPACE for this purpose is now deprecated because it has been found to lead to many complications in text processing. Whether using either NBSP or SPACE as the base character, the visual appearance is the same, but the line breaking behavior is different. Under the current rules, **SP CM*** will allow a break between **SP** and **CM***, which could result in a new line starting with a combining mark. Previously, whenever the base character was **SP**, the sequences **CM*** and **SP CM*** were defined to act like indivisible clusters, allowing breaks on either side like **ID**.

Where backward compatibility with documents created under the prior practice is desired, the following tailoring should be applied to those **CM** characters that have a General_Category value of Combining_Mark (M):

*Legacy-CM: In all of the rules following rule **LB8**, if a space is the base character for a combining mark, the space is changed to type **ID**. In other words, break before **SP** in the same cases as one would break before an **ID**.*

Treat **SP CM*** as if it were **ID**.

While this tailoring changes the location of the line break opportunities in the string, it is ordinarily not expected to affect the display of the text. That is because spaces at the end of the line are normally invisible and the recommended display for isolated combining marks is the same as if they were applied to a preceding SPACE or NBSP.

10 Testing

As with the other default specifications, implementations are free to override (tailor) the results to meet the requirements of different environments or particular languages as described in Section 4, Conformance. For those who do implement the default breaks as specified in this annex, and wish to check that that their implementation matches that specification, a test file has been made available in [Tests14].

These tests cannot be exhaustive, because of the large number of possible combinations; but they do provide samples that test all pairs of property values, using a representative character for each value, plus certain other sequences.

A sample HTML file is also available for each that shows various combinations in chart form, in [Charts14]. The header cells of the chart consist of a property value, followed by a representative code point number. The body cells in the chart show the break status: whether a break occurs between the row property value and the column property value. If the browser supports tool-tips, then hovering the mouse over the code point number will show the character name, General_Category and Script property values. Hovering over the break status will display the number of the rule responsible for that status.

> Note: To determine a break it is generally not sufficient to just test the two adjacent characters.

The chart is followed by some test cases. These test cases consist of various strings with the break status between each pair of characters shown by blue lines for breaks and by whitespace for non-breaks. Hovering over each character (with tool-tips enabled) shows the

character name and property value; hovering over the break status shows the number of the rule responsible for that status.

Due to the way they have been mechanically processed for generation, the test rules do not match the rules in this annex precisely. In particular:

1. The rules are cast into a more regex-style.
2. The rules “sot”, “eot”, and “Any” are added mechanically and have artificial numbers.
3. The rules are given decimal numbers without prefix, so rules such as LB14 are given a number using tenths, such as 14.0.
4. Where a rule has multiple parts (lines), each one is numbered using hundredths, such a
   - 13.01) [^NU] × CL
   - 13.02) × EX
   - ...
5. Any “treat as” or “cignore” rules are handled as discussed in this annex, and thus reflected in a transformation of the rules not visible in the tests.

The mapping from the rule numbering in this annex to the numbering for the test rules is summarized in Table 4.

Table 4. Numbering of Rules

Rule in This Annex	Test Rule	Comment
LB2	0.2	start of text
LB3	0.3	end of text
LB12a	12.0	GL ×
LB12b	12.1	[^SP, BA, HY] × GL
LB31	999	÷ any

References

For references for this annex, see Unicode Standard Annex #41, “Common References for Unicode Standard Annexes.”

Acknowledgments

The initial assignments of properties are based on input by Michel Suignard. Mark Davis provided algorithmic verification and formulation of the rules, and detailed suggestions on the algorithm and text. Ken Whistler, Rick McGowan and other members of the editorial committee provided valuable feedback. Tim Partridge enlarged the information on dictionary usage. Sun Gi Hong reviewed the information on Korean and provided copious printed samples. Eric Muller reanalyzed the behavior of the soft hyphen and collected the samples. Adam Twardoch provided the Polish example. António Martins-Tuválkin supplied information about Portuguese. Tomoyuki Sadahiro provided information on use of U+30A0. Christopher Fynn provided the background information on Tibetan line breaking. Andrew West, Kamal Mansour, Andrew Glass, Daniel Yacob, and Peter Kirk suggested improvements for Mongolian, Arabic, Kharoshthi, Ethiopic, and Hebrew punctuation characters, respectively. Kent Karlsson reviewed the line break properties for consistency. Andy Heninger reviewed the rules and provided input on regular expression-based implementations. Jerry Hall reviewed the sample code. Elika J. Etemad (fantasai) reviewed the entire document in an

effort to make it easier to reference from external standards. Many others provided additional review of the rules and property assignments.

Modifications

Rule Numbering Across Versions

Table 5
documents changes in the numbering of line breaking rules. A duplicate number indicates that a rule was subsequently split. (In each version, the rules are applied in their numerical order, not in the order they appear in this table.) Versions prior to 3.0.1 are not documented here.

Table 5. Rule Numbering Across Versions

5.1	5.0.0	4.1.0	4.0.1	4.0.0	3.2.0	3.1.0	3.0.1
LB1	1	1	1	1	1	1	1
LB2	2	2a	2a	2a	2a	2a	2a
LB3	3	2b	2b	2b	2b	2b	3b
LB4	4	3a	3a	3a	3a	3a	3a
LB5	5	3b	3b	3b	3a	3a	3a
LB6	6	3c	3c	3c	3b	3b	3b
LB7	7	4	4	4	4	4	4
LB8	8	5	5	5	5	5	5
		deprecated	7a	7a	7	7	7
LB9	9	7b	7b	7b	6	6	6
LB10	10	7c	7c	7c			
LB11	11	11b	11b	11b	13	13	13
LB12	12	13	11b	11b	13	13	13
LB12a	12	13	11b	11b	13	13	13
LB13	13	8	8	8	8	8	8
LB14	14	9	9	9	9	9	9
LB15	15	10	10	10	10	10	10
LB16	16	11	11	11	11	11	11
LB17	17	11a	11a	11a	11a	11a	
LB18	18	12	12	12	12	12	12
LB19	19	14	14	14	14	14	14
LB20	20	14a	14a	14a			
LB21	21	15	15	15	15	15	15
LB22	22	16	16	16	16	16	16
LB23	23	17	17	17	17	17	17
LB24	24	18	18	18	18	18	18
LB25	25	18	18	18	18	18	18
		removed	18b	18b	15b	15b	15b
LB26	26	18b	6	6	6	6	6
LB27	27	18c	6	6	6	6	6
LB28	28	19	19	19	19	19	19
LB29	29	19b	19b				
LB30	30						
LB31	31	20	20	20	20	20	20

Change History

The following documents the changes introduced by each revision.

Revision 21:

- Draft 2
  - Substantial revisions to the conformance section.
  - 5.7 Word Separators section added
  - 10. Testing section added.
  - Renumber rules for consistency: 12a -> 12; 12b -> 12a
- Draft 1
  - Changed version from 5.0.1 to 5.1
  - Dropped the HL4 proposal for tailoring the behavior of spaces.
  - Added 02DF, MODIFIER LETTER CROSS ACCENT, to class BB.
  - Added change markings to a few differences from rev 19 that weren't marked in rev 20.
  - Added discussion for 202F NARROW NO-BREAK SPACE and 180E MONGOLIAN VOWEL SEPARATOR
  - Corrected typos in LB13 and LB16
  - Added characters introduced with Unicode 5.1 to the lists associated with the line break properties.

Revision 20:

- Added Section 5.2 on the special handling of hyphens. Edited Section 3 for clarity.
- Improved delineation between normative and informative information.
- Changed from **EX** to **IS**
  - 060C (،) ARABIC COMMA
- Changed from **EX** to **PO**
  - 066A (٪) ARABIC PERCENT SIGN
- Changed from **AI** to **OP**.
  - 00A1 (¡) INVERTED EXCLAMATION MARK
  - 00BF (¿) INVERTED QUESTION MARK
- Changed from **BA** to **EX**
  - 1802 (᠂) MONGOLIAN COMMA
  - 1803 (᠃) MONGOLIAN FULL STOP
  - 1808 (᠈) MONGOLIAN MANCHU COMMA
  - 1809 (᠉) MONGOLIAN MANCHU FULL STOP
  - 2CF9 (⸼) COPTIC OLD NUBIAN FULL STOP
  - 2CFE (⸾) COPTIC FULL STOP
- Changed from **BA** to **AL**
  - 1A1E (᨞) BUGINESE PALLAWA
- Changed from **AL** to **BB**
  - 1FFD (´) GREEK OXIA
- Added a note on lack of canonical equivalence for the definition of ambiguous characters.

- Corrected typos in the sample source code.
- Split rule LB12 to accommodate Polish and Portuguese hyphenation.
- Proposed: HL4

Revision 19:

- Changed 000B from **CM** to **BK**, changed 035C from **CM** to **GL**.
- Changed 17D9 from **NS** to **AL**. 203D, 2047..2049 from **AL** to **NS**.
- Corrected listing of **NS** property to match the data file to remove 17D8 and 17DA.
- The data file has been corrected to match the listing of the **BA** property to include 1735 and 1736, also changed 05BE and 103D0 from **AL** to **BA**.
- Changed the brackets 23B4.23B6 to **AL**.
- Updated the **SA** property to make it more generic, includes changing many characters from CM to SA.
- Reflected new characters
- Made several text changes for clarifications, including reworded the intro to Section 6.
- Added Section 9.
- Restated the conformance clauses and reorganized the algorithm into a tailorable and a non-tailorable part; this affects text in Sections 4, 5, and 6.
- Removed redundant term PR x HY from rule 18 and rule into new LB24 and LB26 to provide better granularity for tailoring,
- Moved rule 11b and 13 above rule 8 (new LB13), restating rule 13 (new LB12) to preserve its effect in the new location.
- Added new rule LB30 to handle words like “person(s)”.
- Renumbered the rules.
- Extensive copy-editing as part of Unicode 5.0 publication.

Revision 18 being a proposed update, only changes between revisions 17 and 19 are noted here.

Revision 17:

- Significantly revised the line break classes for Tibetan, as well as Mongolian and Arabic Punctuation.
- Added section 5.6 on Tibetan and section 7.7 on handling explicit breaks.
- Added line break class assignments for Unicode 4.1 characters.
- Significantly revised the line break class assignments for *danda* characters and made it consistent across scripts.
- LB6: Replaced by new rules 18b and 18c, using new classes **JL**, **JV**, **JT**, **H2**, and **H3**.
- LB7a: Deprecated rule 7a because SPACE as base character for standalone combining marks is deprecated.
- LB7b: Revised 7b and section 7.5 as well as Table 2 to match the deprecation of rule 7a.
- LB7b: Clarified that this rule does not apply to SP.
- LB11a: Added a missing **SP** * to make the formula match the rule.
- LB18b: Removed the existing rule 18b because it was redundant.
- Corrected an erratum on revision 14 by splitting **GL** from **WJ** in rule 11b and moving to a new rule 13.
- Updated the pair table and sample code to match the changes in the rules.

- Updated the regular expression for numbers.
- Added several notes on implementation techniques.
- Moved all suggested tailorings from the rules section to the examples in section 8.2.

Revision 16 being a proposed update, only changes between revisions 17 and 15 are noted here.

Revision 15:

- LB19b: Added new rule 19b.
- Changed line breaking class: combining double diacritics from **CM** to **GL**, 037A and 2126 to match their canonical equivalents, 2140 corrected to **AL**, Arabic numerical separators from **AL** to **NU**, many alphabetic characters that are EAW=Ambiguous from **AI** to **AL** to better reflect current practice, remaining circled numbers and letters from **AL** to **AI** for consistency.
- Added a note on the behavior of U+200B and U+3000 when lines are justified.
- Reconciled the data file and description of line breaking classes in section 5.
- Reconciled the rules and pair table implementation of the algorithm.
- Updated the text of the conformance statement in section 4.
- Added section 5.5 on use of double hyphen.
- Updated styles and table formatting.
- Minor edits throughout.

Revision 14:

- Added new line breaking classes **NL** and **WJ** to better support NEL and Word Joiner.
- Deprecated the use of class **SG**.
- Several changes to the rules. Moved rule 15b to 18b, added 14b, moved 13 to 11b. Split rule 6 in to 6a and 7b and split rule 3a into 3a and 3b. Restated rule 7a and added rule 7c.
- Updated the pair table and sample code, adding a special token '#' to account for breaks before **SP** followed by **CM**.
- Clarified the behavior of SHY and MONGOLIAN TODO SOFT HYPHEN, as well as WJ and ZWNBSP.
- Added a new subsection 5.4 on SOFT HYPHEN and a new subsection 7.6 on conjoining jamos.
- Added to the discussion on how to treat combining marks.
- Clarified the conformance requirements in Section 4
- Added a definition of *line breaking class* as synonym for the unwieldy *line breaking property value.*
- Expanded the introduction in Section 3.
- Moved subsections on customization into a new Section 8 and expanded the text.
- Many edits throughout the text to update it for Unicode 4.0.0.

Revision 11 being a proposed update, only changes between revisions 12 and 14 are noted here.

Revision 12:

- Change header for publication of Unicode. Fixed a few additional typos.
- Updated for publication of Unicode, Version 3.2

- Added Word joiner to **GL** and noted that it now is the preferred character instead of FEFF
- Added LB class assignments for the new Unicode 3.2 characters to the data file. Only characters whose LB class differs from those of characters with related General_Category are noted explicitly in this text.

Revision 11, being a proposed update, only changes between revisions 10 and 12 are noted here.]

Revision 10:

- Changed header for publication of Unicode 3.1. Fixed a few additional typos.

Revision 9:

- Fixed several typos, reformatted and sorted some lists by code points
- Reconciled the data file and the description for **BB** (00B4), **XX** (PUA), **AI** (2015,25C8,PUA), **ID** (FE6B), **BA** (00B4)
- Restored PUA to **XX**.
- LB7: Restored the rule, and fixed the note so it matches the rule and Section 7.7 of [Unicode4.0].
- LB11a: added a rule to reconcile the rules against pair table entry B2 ^ B2
- LB19: added an entry to reconcile the rules against pair table entry PR % ID
- Reworked Section 7.5
- Removed two unused definitions (overfull and underfull)

Revision 8:

- New status section, changed format of references. Fixed several typos.
- Added headers to Table 1
- Added a note on use of B and A
- Added mention of PUA to **AI** and removed mention of PUA from **XX** because the data file assigns **AI** to them.
- Clarified the membership and implication of class **CM** and **ID**.
- Updated class **ID** by the new ranges for 3.1.
- LB6: Clarified the description of LB6 to clarify how it affects conjoining Jamo.
- LB7: Fixed the note so it matches the rule.
- LB17: Fixed the regular expression for numbers in the explanation for this rule.
- Reworded Sections 7.6 and 7.7 to clarify the customization process.

Revision 7:

- Fixed several typos.
- New header.

Revision 6:

- Rewrite and reorganization of the text as part of the publication of the Unicode Standard, Version 3.0.

No change history is available for earlier revisions.
