

Unicode 2 Feature Reference

Peter S. Baker

November 14, 2021

[Introduction](#)

- [A Case-Related Features](#)
- [B Numbers and Sequencing](#)
- [C Superscripts and Subscripts](#)
- [D Ornaments](#)
- [E Alphabetic Variants](#)
- [F Greek](#)
- [G Punctuation](#)
- [H Abbreviations](#)
- [I Combining Marks](#)
- [J Currency and Weights](#)
- [K Gothic](#)
- [L Runic](#)
- [M Ligatures and Digraphs](#)
- [N Required Features](#)
- [O Non-MUFI Code Points](#)

Introduction

The OpenType features of Junicode version 2 and its variable counterpart (hereafter referred to together as “Junicode”) have two purposes. One is to provide convenient access to the rich character set of the Medieval Unicode Font Initiative (MUFI) recommendation. The other is to enable best practices in the presentation of medieval text, promoting accessibility in electronic texts from PDFs to e-books to web pages.

Each character in the MUFI recommendation has a code point¹ associated with it: either the one assigned by Unicode or, where the character is not recognized by Unicode, in the Private Use Area (PUA) of the Basic Multilingual Plane, a block of codes, running from U+E000 to U+F8FF, that are assigned no value by Unicode but instead are available for font designers to use in any way they please.

The problem with PUA code points is precisely their lack of any value. Consider, as a point of comparison, the letter **a** (U+0061). Your computer, your phone, and probably a good many other devices around the house store a good bit of information about this **a**: that it’s a letter in the Latin script, that it’s lowercase, and that the uppercase equivalent is **A** (U+0041). All this information is available to word processors, browsers, and other applications running on your computer.

Now suppose you’re preparing an electronic text containing what MUFI calls LATIN SMALL LETTER NECKLESS A (**ȁ**). It is assigned to code point U+F215, which belongs to the PUA. Beyond that, your computer knows nothing about it: not that it is a variant of **a**, or that it is lowercase, or a letter in the Latin alphabet, or even a character in a language system. A screen reader cannot read, or even spell out, a word with U+F215 in it; a search engine will not recognize the word as containing the letter **a**.

Junicode offers the full range of MUFI characters—you can enter the PUA code points—but also a solution to the problems posed by those code points. Think of an electronic text (a web page, perhaps, or a PDF) as having two layers: an underlying text, stable and unchanging, and the displayed text, generated by software at the instant it is needed and discarded when it is no longer on the screen. For greatest accessibility the underlying text should contain the plain letter **a** (U+0061) along with markup indicating how it should be displayed. To generate the displayed text, a program called a “layout engine” will (simplifying a bit here) read the markup and apply the OpenType feature

¹A Unicode code point is generally expressed as a four-digit hexadecimal (or base-16) number with a prefix of “U+”. The letter capital “A,” for example, is U+0041 (65 in decimal notation), and lowercase “a” (Middle English yogh) is U+021D.

JUNICODE 2 FEATURE REFERENCE

`cv02[5]`² to the underlying **a**, bypassing the PUA code point, with the result that readers see **a**—the “neckless a.” And yet the letter will still register as **a** with search engines, screen readers, and so on.

This is the Junicode model for text display, but it is not peculiar to Junicode: it is widely considered to be the best practice for displaying text using current font technology.

The full range of OpenType features listed in this document is supported by all major web browsers, LibreOffice, XeTeX, LuaTeX, and (presumably) other document processing applications. All characters listed here are available in Adobe InDesign, though that program supports only a selection of OpenType features. Microsoft Word, unfortunately, supports only Stylistic Sets, ligatures (all but the standard ones in peculiar and probably useless combinations), number variants, and the [Required Features](#). In terms of OpenType support, Word is the most primitive of the major text processing applications.

Many MUFI characters cannot be produced by using the OpenType variants of Junicode. These characters fall into three categories:

- Those with Unicode (non-PUA) code points. MUFI has done valuable work obtaining Unicode code points for medieval characters. All such characters (those with hexadecimal codes that *do not* begin with **E** or **F**) are presumed safe to use in accessible and searchable text. However, some of these are covered by Junicode OpenType features for particular reasons.
- Precomposed characters—those consisting of base character + one or more diacritics. For greatest accessibility, these should be entered not as PUA code points, but rather as sequences consisting of base character + one or more diacritics. For example, instead of MUFI U+E498 LATIN SMALL LETTER E WITH DOT BELOW AND ACUTE, use **e** + U+0323 COMBINING DOT BELOW + U+0301 COMBINING ACUTE ACCENT: **é** (when applying combining marks, start with any marks below the character and work downwards, then continue with any marks above the character

²Many OpenType features produce different outcomes depending on an index passed to an application’s layout engine along with the feature tag. Different applications have different ways of entering this index: consult your application’s documentation. Here, the index is recorded in brackets after the feature tag. Users of fontspec (with Xe_{La}TeX or Lua_{TeX}) should also be aware that fontspec indexes start at zero while OpenType indexes start at one. Therefore all index numbers listed in this document must be reduced by one for use with fontspec.

JUNICODE 2 FEATURE REFERENCE

and work upwards. For example, to make **ő**, place characters in this order: **o**, COMBINING OGONEK U+0328, COMBINING DOT BELOW U+0323, COMBINING MACRON U+0304, COMBINING ACUTE U+0301). Some MUFI characters have marks in unconventional locations, e.g. **ő** LATIN SMALL LETTER O WITH DOT ABOVE AND ACUTE, where the acute appears beside the dot instead of above. This and other characters like it should still be entered as a sequence of base character + marks (here **o**, COMBINING DOT ABOVE U+0307, COMBINING ACUTE U+0301). Junicode will position the marks correctly.

- Characters for which a base character (a Unicode character to which it can be linked) cannot be identified, or for which there may be an inconsistency in the MUFI recommendation. These include:
 - **fl** U+E8AF. This is a ligature of long **s** and **l** with stroke, but there are no base characters with this style of stroke.
 - **űŰ** U+EFD8 and U+EFD9. MUFI lists these as ligatures (corresponding to the historic ligatures **uU**, but they cannot be treated as ligatures in the font because a single diacritic is positioned over the glyphs as if they were digraphs like **aA**.
 - **pp̈P̈** U+EBE7 and U+EBE6, for the same reason.
 - **ð** U+F159 LATIN ABBREVIATION SIGN SMALL DE. Neither a variant of **d** nor an eth (**ð**), this character may be a candidate for Unicode encoding.
- Characters for which OpenType programming is not yet available. These will be added as they are located and studied. [Check: U+EBF1, and smcp version.]

These characters should be avoided, even if you are otherwise using MUFI's PUA characters:

- U+F1C5 COMBINING CURL HIGH POSITION. Use U+1DCE COMBINING OGONEK ABOVE. The positioning problem mentioned in the MUFI recommendation is solved in Junicode (and, to be fair, many other fonts with OpenType programming).
- U+F1CA COMBINING DOT ABOVE HIGH POSITION. Use U+0307 COMBINING DOT ABOVE. It will be positioned correctly on any character.

A. Case-Related Features

1. **smcp** – Small Capitals

Converts lowercase letters to small caps; also several symbols and combining marks. All lower- and uppercase pairs (with exceptions noted below) have a small cap equivalent. Lowercase letters without matching caps may lack matching small caps. fghij → FGHIJ.

Note: Precomposed characters defined by MUFI in the Private Use Area have no small cap equivalents. Instead, compose characters using combining diacritics, as outlined in the introduction. For example, **smcp** applied to the sequence **t** + COMBINING OGONEK (U+0328) + COMBINING ACUTE (U+0301) will change **ť** to **ṭ**.

2. **c2sc** – Small Capitals from Capitals

Use with **smcp** for all-small-cap text. ABCDE → ABCDE.

3. **pcap** – Petite Capitals

Produces small caps in a smaller size than **smcp**. Use these when small caps have to be mixed with lowercase letters. The whole of the basic Latin alphabet is covered, plus several other letters. klmnop → KLMNOP.

4. **case** – Case-Sensitive Forms

Produces combining marks that harmonize with capital letters: \breve{R} , \breve{X} , etc. Use of this feature reduces the likelihood that a combining mark will collide with a glyph in the line above.

B. Numbers and Sequencing

5. **nalt** – Alternate Annotation Forms

Produces letters and numbers circled, in parenthesis, or followed by periods, as follows:

nalt[1], circled letters or numbers: (a) (b) . . . (z); (0) (1) (2) . . . (20).

nalt[2], letter or numbers in parentheses: (a) . . . (z); (0) (1) . . . (20).

nalt[3], double-circled numbers: (0) (1) . . . (10).

nalt[4], white numbers in black circles: 0 1 2 3 . . . 20.

nalt[5], numbers followed by period: ① ② . . . ⑩.

For enclosed figures 10 and higher, **rlig** (Required Ligatures) must also be enabled (as it should be by default: see [Required Features](#) below).

6. **tnum** – Tabular Figures

Fixed-width figures: 0123456789 (default or with **lnum**), 0123456789 (with **onum**).

7. **onum** – Oldstyle Figures

Figures that harmonize with lowercase characters: 0123456789 (default or with **tnum**), 0123456789 (with **pnum**). When combined with **pnum**, this feature also affects subscripts and superscripts.

8. **pnum** – Proportional Figures

Proportionally spaced figures: 0123456789 (default or with **lnum**), 0123456789 (with **onum**). When combined with **onum**, this feature also affects subscripts and superscripts. Most applications (including MS Word) with any support of OpenType features will support this feature and **lnum** in such a way that you don't have to enter them manually.

9. **lnum** – Lining Figures

Figures in a uniform height, harmonizing with uppercase letters: 0123456789 (default or with **tnum**), 0123456789 (with **pnum**).

10. **zero** – Slashed Zero

Produces slashed zero in all number styles: 0 0 0 0. Includes superscripts and subscripts:
0 0 0 0.

C. Superscripts and Subscripts

11. **sup** – Superscripts

Produces superscript numbers and letters. Only affects lining tabular and oldstyle proportional figures. All lowercase letters of the basic Latin alphabet are covered, and most

JUNICODE 2 FEATURE REFERENCE

uppercase letters: ^{0123 4567} abcde ABDEG. Wherever superscripts are needed (e.g. for footnote numbers), use **sup**s instead of the raised and scaled characters generated by some programs. With sups: ⁴⁵⁶⁷. Scaled: ⁴⁵⁶⁷.





12. **subs** – Subscripts





Produces subscript numbers. Only affects lining tabular (the default numbers) and old-style proportional figures (use **pnum** and **onum**): ^{8901 2345}.

D. Ornaments

13. **ornm** – Ornaments

Produces ornaments (fleurons) in either of two ways: as an indexed variant of the bullet character (U+2022) or as variants of a-z, A-C (all fleurons are available by either method):

As a variant of •: 1=, 2=, 3=, 4=, etc., up to 29.

As a variant of a-z, A-C: e=, f=, g=, h=, etc.

The method with letters of the alphabet is easier, but the method with bullets will produce a more satisfactory result when text is displayed in an environment where Junicode is not available or **ornm** is not implemented.

E. Alphabetic Variants

14. **cv01-cv52** – Basic Latin Variants

These features also affect small cap (**smcp**) and underdotted (**ss07**) forms, where available. Where Junicode has no variant for a Basic Latin letter, the expected **cvNN** feature is skipped, being reserved for future development.

Variant of	cvNN	Variants
A	cv01	1=Ⓐ, 2=Ⓐ̇, 3=Ⓐ̈
a	cv02	1=ⓐ, 2=ⓐ̇, 3=ⓐ̈, 4=ⓐ̉, 5=ⓐ̊
B	cv03	No variants available

JUNICODE 2 FEATURE REFERENCE

Variant of	cvNN	Variants
b	cv04	No variants available
C	cv05	1=Ł
c	cv06	1=ċ
D	cv07	1=Ð
d	cv08	1=ð, 2=ᵹ, 3=ᵹ (for 1, see also ss02)
E	cv09	1=€̇, 2=€̈
e	cv10	1=€̇, 2=ė, 3=ë
F	cv11	1=ƒ̇
f	cv12	1=ƒ̇, 2=ƒ̈, 3=ƒ̉, 4=ƒ̊, 5=ḟ, 6=f̈
G	cv13	1=Ǿ̇, 2=Ǿ̈, 3=Ǿ̉
g	cv14	1=Ǿ̇, 2=Ǿ̈, 3=ġ, 4=g̈, 5=g̉, 6=g̊, 7=g̋
H	cv15	1=ħ̇
h	cv16	1=ħ̇, 2=ḧ
I	cv17	1=İ̇, 2=İ̈
i	cv18	1=ı̇, 2=ı̈, 3=ı̉
J	cv19	1=ĵ̇
j	cv20	1=ĵ̇, 2=ĵ̈, 3=j̈
K	cv21	No variants available
k	cv22	1=k̇, 2=k̈, 3=k̉, 4=k̊
L	cv23	No variants available
l	cv24	1=ł̇
M	cv25	1=ℳ̇, 2=ℳ̈, 3=ℳ̉
m	cv26	1=ṁ, 2=m̈, 3=m̉
N	cv27	1=ℕ̇

JUNICODE 2 FEATURE REFERENCE

Variant of	cvNN	Variants
n	cv28	1=ñ, 2=Ñ, 3=ņ, 4=ṇ
O	cv29	1=Ŏ
o	cv30	1=ɔ̌
P	cv31	1=Ɔ
p	cv32	No variants available
Q	cv33	1=Ɓ
q	cv34	1=ɓ
R	cv35	1=ɹ
r	cv36	1=ɽ, 2=ɾ
S	cv37	1=ʂ, 2=ʃ
s	cv38	1=ʃ, 2=ʂ, 3=ʈ, 4=ʉ, 5=ʊ, 6=ʋ
T	cv39	1=Ƨ
t	cv40	1=ɿ, 2=ʈ
U	cv41	No variants available
u	cv42	No variants available
V	cv43	No variants available
v	cv44	1=ʌ, 2=ʋ, 3=ʌ, 4=ʋ
W	cv45	No variants available
w	cv46	No variants available
X	cv47	No variants available
x	cv48	1=x̄, 2=x̂, 3=x̃, 4=x̄
Y	cv49	1=Ȳ
y	cv50	1=ȳ, 2=ȳ
Z	cv51	1=Ƶ

JUNICODE 2 FEATURE REFERENCE

Variant of	cvNN	Variants
z	cv52	1=Ź, 2=Ẑ

15. cv53–cv67 – Other Latin Letters

Some features affect both upper- and lowercase forms. **cv62** also affects combining **e** with ogonek, accessible via **ss10** with the entity reference **&_eogo;**. In this range, **cvNN** features are not reserved for future development, since Unicode already uses or reserves all of the available **cvNN** features.

Variant of	cvNN	Variants
Ą (U+0104)	cv53	1=Ȧ, 2=Ȧ̸, 3=Ȧ̇
ą (U+0105)	cv54	1=ȧ, 2=ȧ̇
æ (U+A733)	cv55	1=Ǽ, 2=Ǽ̇
Æ (U+00C6)	cv56	1=ƆE
æ (U+00E6)	cv57	1=Ȣ, 2=Ȣ̇, 3=Ȣ̈
Ɔ (U+A734)	cv58	1=Ɔ̇
ø (U+A735)	cv59	1=Ɔ̇, 2=Ȣ̇, 3=Ȣ̈
Ȣ (U+A739)	cv60	1=Ȣ̇
đ (U+0111)	cv61	1=đ̇
Ě, ě ... (U+0118, U+0119)	cv62	1=Ě̇, ě̇ ...; 2=Ě̈, ě̈ ...
Ȣ, Ȣ̇ (U+021C, U+021D)	cv63	1=Ȣ̈, Ȣ̈̇
Ɔ (U+A749)	cv64	1=Ɔ̇
ø (U+00F8)	cv65	1=Ɔ̇, 2=Ȣ̇, 3=Ȣ̈, 4=Ȣ̈̇
Ȣ̇, Ȣ̈ (U+A765)	cv66	1=Ȣ̈̇, Ȣ̈̈̇
?	cv67	Reserved for future use

16. ss01 – Alternate thorn and eth

Produces Nordic thorn and eth (þǾþ) when the language is English, and English thorn and eth (þǾþ) with any other language, reversing the font’s usual usage. This also affects small caps, crossed thorn (þ þ̈—see also **cv67**), combining mark eth (U+1DD9, Ǿ̈ Ǿ̈̇),

and enlarged thorn and eth (see [ss06](#)). This feature depends on `loca` (Localized Forms), which in most applications will always be enabled.

17. ss02 – Insular Letter-Forms

Produces insular letter-forms, e.g. $\delta\epsilon\zeta\eta\rho$. Does not affect capitals (except W), as these do not commonly have insular shapes in early manuscripts. For these, enter the Unicode code points or use the Character Variant (**cvNN**) features.

18. ss04 – High Overline

Produces a high overline over letters used as roman numbers: `cdijlmvx` `CDIJLMVXO`.

19. ss05 – Medium-High Overline

Produces a medium-high overline over (or through the ascenders of) letters used as roman numbers, and some others as well: $\overline{\text{bcdhijklmfvxb}}$.

20. ss06 – Enlarged Minusculs

Lowercase letters that match the height of normal ones, but with a higher x-height, e.g. abcdefg. Covers the whole of the basic Latin alphabet and several other letters: consult the MUFI recommendation for details, and if you are using the variable version of the font (JunicodeVF), consider using the [Enlarged axis](#) instead, for reasons of flexibility and accessibility.

21. ss07 – Underdotted Text

Produces underdotted text (indicating deletion in medieval manuscripts) for many letters (including the whole of the basic Latin alphabet and a number of other letters), e.g. `abcdefg HIJKLM`. This also affects small caps, e.g. `ABCDEF` → `AbCdEf`. For letters without corresponding underdotted forms (e.g. `U+A751`, `p`), use `U+0323`, combining dot below (`p̣`).

22. ss08 – Contextual Long s

In English and French text only, varies **s** and **f** according to rules followed by many early printers: fports, effence, fstormy, difheveled, transfusions, flynefs, cliffside. For this

JUNICODE 2 FEATURE REFERENCE

feature to work properly, **calt** “Contextual Alternates” must also be enabled (as it should be by default: see [Required Features](#) below).

23. **ss16** – Contextual r Rotunda

Converts **r** to **ŕ** (lowercase only) following the most common rules of medieval manuscripts: pŕiest, firmer, frost, oŕnament. For this feature to work properly, **calt** “Contextual Alternates” must also be enabled (as it should be by default: see [Required Features](#) below).

24. **cv68** – Variant of ʔ (U+0294, glottal stop)

1=ʔ.

25. **cv99** – revert small cap A to lowercase a

1=a. This features reverts small cap **A** to **a**, enabling it to ligature with small cap **N** or **R** via **hlig**: aN, aR. Be sure to apply **smcp**, **cv99** and **hlig** to both components of the ligature.

F. Greek

JunICODE has two distinct styles of Greek. In the roman face, it is upright and modern, especially designed to harmonize with JunICODE’s Latin letters. In the italic, it is slanted and old-style, being based on the eighteenth-century Greek type designed by Alexander Wilson and used by the Foulis Press in Glasgow. Both Greek styles include the full polytonic and monotonic character sets: αβγδεζ αβγδεζ.

To set Greek properly (especially polytonic text) requires that both **loc1** and **ccmp** be active, as they should be by default in most text processing applications (but in MS Word they must be explicitly enabled by checking the “kerning” box on the “Advanced” tab of the Font Dialog).

Modern monotonic Greek should be set using characters from the Unicode “Greek and Coptic” range (U+0370–U+03FF). When text is set in all caps, JunICODE will suppress accents automatically (except in single-letter words, for which you must substitute accentless forms manually). This does not happen with text using visually identical

letters from the “Greek Extended” range (U+0F00–U+1FFF). Thus when setting polytonic Greek, one should use (for example) **Α** (U+1FBB), not **Α** (U+0386), though they look the same.

One can set polytonic Greek either by entering code points from the Greek Extended range or by entering sequences of base characters and diacritics. When using the latter method, you must first make sure the language in your application is set to Greek, and then enter characters in canonical order (that is, the sequence defined by Unicode as equivalent to the composite character). The order is as follows: 1. base character; 2. diacritics positioned either above or in front of the base character, working from left to right or bottom to top; 3. the *ypogegrammeni* (U+0345), or for capitals, if you prefer, the *prosigrammeni* (U+1FBE).

For example, the sequence ω (U+03C9) ◌̣ (U+0313) ◌̤ (U+0301) ◌̥ (U+0345) will produce ꞑ̣̤̥. Substitute capital Ω (U+03A9) and the result will be "Ω̣̤̥. If you prefer a different result, place U+034F COMBINING GRAPHEME JOINER somewhere in the sequence of combining marks—for example, before the *ypogegrammeni* to make "Ω̣̤̥̦.

G. Punctuation

MUFI encodes nearly twenty marks of punctuation in the PUA. In Junicode these can be accessed in either of two ways: all are indexed variants of `.` (period), and all are associated with the Unicode marks of punctuation they most resemble (but it should not be inferred that the medieval marks are semantically identical with the Unicode marks, or that there is an etymological relationship between them). The first method will be easier for most to use, but the second is more likely to yield acceptable fallbacks in environments where Junicode is not available.

Marks with Unicode encoding are not included here, as they can safely be entered directly. In MUFI 4.0 several marks have PUA encodings, but have since that release been assigned Unicode code points: *paragraphus* (Ÿ U+2E4D), medieval comma (Ꞣ U+2E4C), *punctus elevatus* (ꝛ U+2E4E), *virgula suspensiva* († U+2E4A), triple dagger (‡ U+2E4B).

26. ss18 – Old-Style Punctuation Spacing

Colons, semicolons, parentheses, quotation marks and several other glyphs are spaced as in early printed books.

27. cv69 – Variants of ʝ (U+204A / U+2E52, Tironian nota)

1=ʟ, 2=ʟ̇.

28. cv70 – Variants of . (period)

1=., 2=.,, 3=ʔ, 4=;, 5=.,, 6=:;, 7=;:, 8=;̇, 9=ʹ, 10=ʼ, 11=ʹ, 12=ʹ, 13=ʹ, 14=ʹ, 15=ʹ, 16=ʹ, 17=~, 18=.:, 19=ʹ, 20=/. This feature provides access to all non-Unicode MUFI punctuation marks. Some of them are available via other features (see below).

29. cv71 – Variant of · (U+00B7, middle dot)

1=· (*distinctio*).

30. cv72 – Variants of , (comma)

1=,, 2=ʹ.

31. cv73 – Variants of ; (semicolon)

1=; (*punctus versus*), 2=.,, 3=:;, 4=;:, 5=;̇.

32. cv74 – Variants of ˙ (U+2E4E, *punctus elevatus*)

1=˙, 2=˙, 3=˙, 4=˙ (*punctus flexus*).

33. cv75 – Variant of ! (exclamation mark)

1=! (*punctus exclamativus*).

34. cv76 – Variants of ? (question mark)

1=ʼ, 2=ʼ, 3=ʼ. Shapes of the *punctus interrogativus*.

35. cv77 – Variant of ~ (ASCII tilde)

1=~ (same as MUFI U+F1F9, “wavy line”).

36. cv78 – Variant of * (asterisk)

1=: . MUFI defines U+F1EC as a *signe de renvoi*. Manuscripts employ a number of shapes (of which this is one) for this purpose. Junicon defines it as a variant of the asterisk—the most common modern *signe de renvoi*.

37. cv79 – Variants of / (slash)

1=↯, 2=↱. The first of these is Unicode, U+2E4E.

H. Abbreviations

38. **cv80** – Variant of \mathfrak{z} (U+A75D, rum abbreviation)

 $1=\mathfrak{z}.$

39. **cv81** – Variants of 𐎠 (U+035B, combining zigzag above)

1=𐌿, 2=𐍂, 3=𐍇. Positioning of the zigzag can differ from that of other combining marks, e.g. 𐍃, 𐍆, 𐍇. If **alt** “Contextual Alternates” is enabled (as it is by default in most apps), variant forms of **cv81[2]** will be used with several letters, e.g. 𐍄, 𐍅, 𐍆. Enable **case** for forms that harmonize with capitals (𐍈 𐍉 𐍊 𐍋), **smcp** for forms that harmonize with small caps (𐍌 𐍍 𐍎 𐍏).

40. cv82 – Variants of spacing ' (U+A770)

1=9, 2=9. [cv82\[1\]](#) produces the baseline *-us* abbreviation (same as MUFI U+F1A6). MUFI also has an uppercase baseline *-us* abbreviation (U+F1A5), but as there is no uppercase version of U+A770 to pair it with, it is indexed separately here.

41. cv83 – Variants of \mathfrak{z} (U+A76B, “et” abbreviation)

1=, 2=, [1] is identical in shape to a semicolon, but as it is semantically the same as U+A76B, it is preferable to use that character with this feature. [2] produces a subscript version of the character, a common variant in printed books.

I. Combining Marks

42. **cv84** – MUFI combining marks (variants of U+0304)

MUFI encodes a number of combining marks in the PUA (with code points between E000 and F8FF), but when these characters are entered directly, they can interfere with searching and accessibility, and some important applications fail to position them correctly over their base characters. To avoid these problems, enter U+0304 (̄, COMBINING MACRON) and apply **cv84**, with the appropriate index, to both mark and base character. This collection of marks does not include any Unicode-encoded marks (from the “Combining Diacritical Marks” ranges), as these can safely be entered directly. It does include three marks (**cv84**[36], [37] and [38]) that lack MUFI code points but are used to form MUFI characters.

This feature may often appear to have no effect. When this happens it is because an application replaced a sequence like **a U+0304** with a precomposed character like **ā** (U+0101) before Junicode’s OpenType programming had a chance to work. This process is called normalization, and it usually has the effect of simplifying text processing tasks, but can sometimes prevent the proper functioning of OpenType features. To disable it, place the character U+034F COMBINING GRAPHEME JOINER (don’t waste any time puzzling over the name) between the base character and the combining mark (or the first combining mark). For example, to produce the combination **ū**, enter **u U+034F U+0304**. (Without U+034F, you would get **ū**).

These marks can sometimes be produced by other **cvNN** features, which may be preferable to **cv84** as providing more suitable fallbacks for applications that do not support Character Variant (**cvNN**) features.

For some marks with PUA code points, users may find it easier to use **entities** than this feature.

These marks are not affected by most other features. This is to preserve flexibility, given the rule that the feature that produces them must be applied to both the mark and the base character. For example, if **smcp** “Small Caps” changed **cv84**[11] ^b to [12] ^B, it would be impossible to produce the sequence **N^bAA** with the diacritic properly positioned.

1=̄	6=̇	11=̈	16=̆
2=̆	7=̈	12=̇	17=̇
3=̇	8=̈	13=̈	18=̇
4=̇	9=̈	14=̇	19=̇
5=̈	10=̈	15=̇	20=̇

JUNICODE 2 FEATURE REFERENCE

21=̋	26=̐	31=̗	36=̊
22=̍	27=̑	32=̙	37=̋
23=̎	28=̒	33=̚	37=̌
24=̏	29=̓	34=̛	
25=̐	30=̔	35=̜	

43. ss10 – Entity References for Combining Marks

Instead of [cv84](#) for representing non-Unicode combining marks, some users may wish to use XML/HTML-style entities. When **ss10** is turned on (preferably for the entire text), these entities appear as combining marks and are correctly positioned over base characters. For example, the letter **e** followed by `&_eogo;` will yield **ē**. An advantage of entities is that they are (unlike the PUA code points or the indexes of [cv84](#)) mnemonic and thus easy to use. A disadvantage is that searches cannot ignore combining marks entered by this method as they can using the [cv84](#) method. (Every method of entering non-Unicode combining marks has disadvantages: users should weigh these, choose a method, and stick with it.)

If you use any of these entities in a work intended for print publication, you should call your publisher’s attention to them, since they will likely have their own method of representing them.

<code>&_ansc;</code> → ̑	<code>&_idotl;</code> → ̙	<code>&_orr;</code> → ̛
<code>&_an;</code> → ̍	<code>&_j;</code> → ̑	<code>&_oru;</code> → ̛
<code>&_ar;</code> → ̎	<code>&_jdotl;</code> → ̙	<code>&_q;</code> → ̑
<code>&_arsc;</code> → ̒	<code>&_ksc;</code> → ̋	<code>&_ru;</code> → ̒
<code>&_bsc;</code> → ̐	<code>&_munc;</code> → ̍	<code>&_tsc;</code> → ̓
<code>&_dsc;</code> → ̏	<code>&_oogo;</code> → ̎	<code>&_y;</code> → ̙
<code>&_eogo;</code> → ̋	<code>&_oslash;</code> → ̏	<code>&_thorn;</code> → ̚
<code>&_emac;</code> → ̐	<code>&_omac;</code> → ̏	

44. ss20 – Low Diacritics

The MUFI recommendation includes a number of precomposed characters with base letters b, h, k, þ, ð and ð and combining marks ̑ (U+0363), ̋ (U+0364), ̑ (U+0304/[cv84](#) [18]), ̎ (U+0366), ̏ (U+036C), ̒ (U+1DE2), ̑ (U+036D), ̑ (U+036E), ̑ (U+1DE6) and ̍ (U+0304/[cv84](#) [21]). Instead of being positioned above ascender height as usual (e.g. **h̑**), the MUFI glyphs have the marks positioned above the x-height

(e.g. **ḧ**). Using the MUFI code points for these precomposed glyphs can interfere with searching and drastically reduce accessibility. Users of Junicode should instead use a sequence of base character + combining mark, and apply **ss20** to the two glyphs. A variant shape of eth (**ð**) that accommodates the combining mark will be substituted for the normal base character (but this is not necessary for the other base characters). Examples: **ß**, **ð**, **ḧ**, **k̈**, **þ**, **ð**.

ss20 affects only the diacritics and base characters listed here; other combinations (e.g. **m̈**, **ḧ**) are not affected. It will therefore probably be safe to apply this feature to the whole text if it is needed anywhere.

45. **cv85** – Variant of **◌̆** (U+1DD3, combining open a)

1=◌̆.

46. **cv86** – Variant of **◌̇** (U+1DD8, combining insular d)

1=◌̇.

47. **cv87** – Variant of **◌̈** (U+1DE3, combining r rotunda)

1=◌̈.

48. **cv88** – Variant of combining dieresis (U+0308)

1=◌̈. This also affects precomposed characters on which this variant dieresis may occur, e.g. **ä**.

49. **cv89** – Variant of **◌̄** (U+0305, combining overline)

1=◌̄.

50. **cv91** – Variants of short horizontal stroke (U+0335)

1=◌̣, 2=◌̤, 3=◌̥. This character can be used with letters with ascenders or descenders, e.g. **đ b þ p**. **cv91[1]** widens the stroke, and **cv91[2]** and **[3]** offset the stroke to the right or left. Via **calt** “Contextual Alternates,” this offset is performed automatically for many characters with ascenders and descenders, and so it should rarely be necessary to use an index with **cv91**.

51. cv92 – Variant of breve below (U+032E)

1=◌◌◌◌. Position the mark after the middle of three glyphs, and apply **cv92** to both the mark and (at least) the middle glyph. This mark is not available via **cv84**.

J. Currency and Weights**52. cv93 – Variants of ₪ (U+0044, generic currency sign)**

1=Ⅲ	8=Ꝑ	15=ℳ	22=ⱥ
2=×	9=Φ	16=℔	23=ⱥ
3=¥	10=ⱥ	17=ⱥ	24=℔
4=ⱥ	11=℔	18=ⱥ	25=℔
5=₪	12=℔	19=ⱥ	26=℔
6=ⱥ	13=ⱥ	20=ⱥ	27=ⱥ
7=ⱥ	14=℔	21=ⱥ	

All of MUFI's currency and weight symbols (those that do not have Unicode code points) are gathered here, but some are also variants of other currency signs (see below).

53. cv94 – Variant of ₣ (U+2114)

1=℔. Same as MUFI U+F2EB (French Libra sign).

54. cv95 – Variants of £ (U+00A3, British pound sign)

1=℔, 2=℔, 3=ⱥ, 4=℔, 5=℔, 6=℔. Same as MUFI U+F2EA, F2EB, F2EC, F2ED, F2EE, F2EF, pound signs from various locales.

55. cv96 – Variant of ₭ (U+20B0, German penny sign)

1=ⱥ. Same as MUFI U+F2F5.

56. cv97 – Variant of ₧ (U+0192, florin)

1=ⱥ. Same as MUFI U+F2E8.

57. cv98 – Variant of ƶ (U+2125, Ounce sign)

1=⊙. Same as MUFI U+F2FD, Script ounce sign.

K. Gothic**58. ss19 – Latin to Gothic Transliteration**

Produces Gothic letters from Latin: Warþ þan in dagans jainans → ƿʌʀþ þan in dagans jainans. In web pages, the letters will be searchable as their Latin equivalents.

K. Runic**59. ss12 – Early English Futhorc**

Changes Latin letters to their equivalents in the early English futhorc. Because of the variability of the runic alphabet, this method of transliteration may not produce the result you want. In that case, it may be necessary to manually edit the result. fisc flodu ahof → ƿiƿk ƿiƿðla ƿniƿ.

60. ss13 – Elder Futhark

Changes Latin letters to their equivalents in the Elder Futhark. Because of the variability of the runic alphabet, this method of transliteration may not produce the result you want. In that case, it may be necessary to manually edit the result. ABCDEFG → ƿ<ðmƿx.

61. ss14 – Younger Futhark

Changes Latin letters to their equivalents in the Younger Futhark. Because of the variability of the runic alphabet, this method of transliteration may not produce the result you want. In that case, it may be necessary to manually edit the result. ABCDEFG → ƿʀʀʀ.

62. ss15 – Long Branch to Short Twig

In combination with **ss14**, converts long branch (the default for the Younger Futhark) to short twig runes: ƿʀʀʀ → ƿʀʀʀ.

63. **rtlm** – Right to Left Mirrored Forms

Produces mirrored runes, e.g. 𐀀𐀁𐀂𐀃𐀄𐀅𐀆𐀇𐀈𐀉𐀊𐀋𐀌𐀍𐀎𐀏𐀐𐀑𐀒𐀓𐀔𐀕𐀖𐀗𐀘𐀙𐀚𐀛𐀜𐀝𐀞𐀟𐀠𐀡𐀢𐀣𐀤𐀥𐀦𐀧𐀨𐀩𐀪𐀫𐀬𐀭𐀮𐀯𐀰𐀱𐀲𐀳𐀴𐀵𐀶𐀷𐀸𐀹𐀺𐀻𐀼𐀽𐀾𐀿𐁀𐁁𐁂𐁃𐁄𐁅𐁆𐁇𐁈𐁉𐁊𐁋𐁌𐁍𐁎𐁏𐁐𐁑𐁒𐁓𐁔𐁕𐁖𐁗𐁘𐁙𐁚𐁛𐁜𐁝𐁞𐁟𐁠𐁡𐁢𐁣𐁤𐁥𐁦𐁧𐁨𐁩𐁪𐁫𐁬𐁭𐁮𐁯𐁰𐁱𐁲𐁳𐁴𐁵𐁶𐁷𐁸𐁹𐁺𐁻𐁼𐁽𐁾𐁿𐂀𐂁𐂂𐂃𐂄𐂅𐂆𐂇𐂈𐂉𐂊𐂋𐂌𐂍𐂎𐂏𐂐𐂑𐂒𐂓𐂔𐂕𐂖𐂗𐂘𐂙𐂚𐂛𐂜𐂝𐂞𐂟𐂠𐂡𐂢𐂣𐂤𐂥𐂦𐂧𐂨𐂩𐂪𐂫𐂬𐂭𐂮𐂯𐂰𐂱𐂲𐂳𐂴𐂵𐂶𐂷𐂸𐂹𐂺𐂻𐂼𐂽𐂾𐂿𐃀𐃁𐃂𐃃𐃄𐃅𐃆𐃇𐃈𐃉𐃊𐃋𐃌𐃍𐃎𐃏𐃐𐃑𐃒𐃓𐃔𐃕𐃖𐃗𐃘𐃙𐃚𐃛𐃜𐃝𐃞𐃟𐃠𐃡𐃢𐃣𐃤𐃥𐃦𐃧𐃨𐃩𐃪𐃫𐃬𐃭𐃮𐃯𐃰𐃱𐃲𐃳𐃴𐃵𐃶𐃷𐃸𐃹𐃺𐃻𐃼𐃽𐃾𐃿𐄀𐄁𐄂𐄃𐄄𐄅𐄆𐄇𐄈𐄉𐄊𐄋𐄌𐄍𐄎𐄏𐄐𐄑𐄒𐄓𐄔𐄕𐄖𐄗𐄘𐄙𐄚𐄛𐄜𐄝𐄞𐄟𐄠𐄡𐄢𐄣𐄤𐄥𐄦𐄧𐄨𐄩𐄪𐄫𐄬𐄭𐄮𐄯𐄰𐄱𐄲𐄳𐄴𐄵𐄶𐄷𐄸𐄹𐄺𐄻𐄼𐄽𐄾𐄿𐅀𐅁𐅂𐅃𐅄𐅅𐅆𐅇𐅈𐅉𐅊𐅋𐅌𐅍𐅎𐅏𐅐𐅑𐅒𐅓𐅔𐅕𐅖𐅗𐅘𐅙𐅚𐅛𐅜𐅝𐅞𐅟𐅠𐅡𐅢𐅣𐅤𐅥𐅦𐅧𐅨𐅩𐅪𐅫𐅬𐅭𐅮𐅯𐅰𐅱𐅲𐅳𐅴𐅵𐅶𐅷𐅸𐅹𐅺𐅻𐅼𐅽𐅾𐅿𐆀𐆁𐆂𐆃𐆄𐆅𐆆𐆇𐆈𐆉𐆊𐆋𐆌𐆍𐆎𐆏𐆐𐆑𐆒𐆓𐆔𐆕𐆖𐆗𐆘𐆙𐆚𐆛𐆜𐆝𐆞𐆟𐆠𐆡𐆢𐆣𐆤𐆥𐆦𐆧𐆨𐆩𐆪𐆫𐆬𐆭𐆮𐆯𐆰𐆱𐆲𐆳𐆴𐆵𐆶𐆷𐆸𐆹𐆺𐆻𐆼𐆽𐆾𐆿𐇀𐇁𐇂𐇃𐇄𐇅𐇆𐇇𐇈𐇉𐇊𐇋𐇌𐇍𐇎𐇏𐇐𐇑𐇒𐇓𐇔𐇕𐇖𐇗𐇘𐇙𐇚𐇛𐇜𐇝𐇞𐇟𐇠𐇡𐇢𐇣𐇤𐇥𐇦𐇧𐇨𐇩𐇪𐇫𐇬𐇭𐇮𐇯𐇰𐇱𐇲𐇳𐇴𐇵𐇶𐇷𐇸𐇹𐇺𐇻𐇼𐇽𐇾𐇿𐈀𐈁𐈂𐈃𐈄𐈅𐈆𐈇𐈈𐈉𐈊𐈋𐈌𐈍𐈎𐈏𐈐𐈑𐈒𐈓𐈔𐈕𐈖𐈗𐈘𐈙𐈚𐈛𐈜𐈝𐈞𐈟𐈠𐈡𐈢𐈣𐈤𐈥𐈦𐈧𐈨𐈩𐈪𐈫𐈬𐈭𐈮𐈯𐈰𐈱𐈲𐈳𐈴𐈵𐈶𐈷𐈸𐈹𐈺𐈻𐈼𐈽𐈾𐈿𐉀𐉁𐉂𐉃𐉄𐉅𐉆𐉇𐉈𐉉𐉊𐉋𐉌𐉍𐉎𐉏𐉐𐉑𐉒𐉓𐉔𐉕𐉖𐉗𐉘𐉙𐉚𐉛𐉜𐉝𐉞𐉟𐉠𐉡𐉢𐉣𐉤𐉥𐉦𐉧𐉨𐉩𐉪𐉫𐉬𐉭𐉮𐉯𐉰𐉱𐉲𐉳𐉴𐉵𐉶𐉷𐉸𐉹𐉺𐉻𐉼𐉽𐉾𐉿𐊀𐊁𐊂𐊃𐊄𐊅𐊆𐊇𐊈𐊉𐊊𐊋𐊌𐊍𐊎𐊏𐊐𐊑𐊒𐊓𐊔𐊕𐊖𐊗𐊘𐊙𐊚𐊛𐊜𐊝𐊞𐊟𐊠𐊡𐊢𐊣𐊤𐊥𐊦𐊧𐊨𐊩𐊪𐊫𐊬𐊭𐊮𐊯𐊰𐊱𐊲𐊳𐊴𐊵𐊶𐊷𐊸𐊹𐊺𐊻𐊼𐊽𐊾𐊿𐋀𐋁𐋂𐋃𐋄𐋅𐋆𐋇𐋈𐋉𐋊𐋋𐋌𐋍𐋎𐋏𐋐𐋑𐋒𐋓𐋔𐋕𐋖𐋗𐋘𐋙𐋚𐋛𐋜𐋝𐋞𐋟𐋠𐋡𐋢𐋣𐋤𐋥𐋦𐋧𐋨𐋩𐋪𐋫𐋬𐋭𐋮𐋯𐋰𐋱𐋲𐋳𐋴𐋵𐋶𐋷𐋸𐋹𐋺𐋻𐋼𐋽𐋾𐋿𐌀𐌁𐌂𐌃𐌄𐌅𐌆𐌇𐌈𐌉𐌊𐌋𐌌𐌍𐌎𐌏𐌐𐌑𐌒𐌓𐌔𐌕𐌖𐌗𐌘𐌙𐌚𐌛𐌜𐌝𐌞𐌟𐌠𐌡𐌢𐌣𐌤𐌥𐌦𐌧𐌨𐌩𐌪𐌫𐌬𐌭𐌮𐌯𐌰𐌱𐌲𐌳𐌴𐌵𐌶𐌷𐌸𐌹𐌺𐌻𐌼𐌽𐌾𐌿𐍀𐍁𐍂𐍃𐍄𐍅𐍆𐍇𐍈𐍉𐍊𐍋𐍌𐍍𐍎𐍏𐍐𐍑𐍒𐍓𐍔𐍕𐍖𐍗𐍘𐍙𐍚𐍛𐍜𐍝𐍞𐍟𐍠𐍡𐍢𐍣𐍤𐍥𐍦𐍧𐍨𐍩𐍪𐍫𐍬𐍭𐍮𐍯𐍰𐍱𐍲𐍳𐍴𐍵𐍶𐍷𐍸𐍹𐍺𐍻𐍼𐍽𐍾𐍿𐎀𐎁𐎂𐎃𐎄𐎅𐎆𐎇𐎈𐎉𐎊𐎋𐎌𐎍𐎎𐎏𐎐𐎑𐎒𐎓𐎔𐎕𐎖𐎗𐎘𐎙𐎚𐎛𐎜𐎝𐎞𐎟𐎠𐎡𐎢𐎣𐎤𐎥𐎦𐎧𐎨𐎩𐎪𐎫𐎬𐎭𐎮𐎯𐎰𐎱𐎲𐎳𐎴𐎵𐎶𐎷𐎸𐎹𐎺𐎻𐎼𐎽𐎾𐎿𐏀𐏁𐏂𐏃𐏄𐏅𐏆𐏇𐏈𐏉𐏊𐏋𐏌𐏍𐏎𐏏𐏐𐏑𐏒𐏓𐏔𐏕𐏖𐏗𐏘𐏙𐏚𐏛𐏜𐏝𐏞𐏟𐏠𐏡𐏢𐏣𐏤𐏥𐏦𐏧𐏨𐏩𐏪𐏫𐏬𐏭𐏮𐏯𐏰𐏱𐏲𐏳𐏴𐏵𐏶𐏷𐏸𐏹𐏺𐏻𐏼𐏽𐏾𐏿𐐀𐐁𐐂𐐃𐐄𐐅𐐆𐐇𐐈𐐉𐐊𐐋𐐌𐐍𐐎𐐏𐐐𐐑𐐒𐐓𐐔𐐕𐐖𐐗𐐘𐐙𐐚𐐛𐐜𐐝𐐞𐐟𐐠𐐡𐐢𐐣𐐤𐐥𐐦𐐧𐐨𐐩𐐪𐐫𐐬𐐭𐐮𐐯𐐰𐐱𐐲𐐳𐐴𐐵𐐶𐐷𐐸𐐹𐐺𐐻𐐼𐐽𐐾𐐿𐑀𐑁𐑂𐑃𐑄𐑅𐑆𐑇𐑈𐑉𐑊𐑋𐑌𐑍𐑎𐑏𐑐𐑑𐑒𐑓𐑔𐑕𐑖𐑗𐑘𐑙𐑚𐑛𐑜𐑝𐑞𐑟𐑠𐑡𐑢𐑣𐑤𐑥𐑦𐑧𐑨𐑩𐑪𐑫𐑬𐑭𐑮𐑯𐑰𐑱𐑲𐑳𐑴𐑵𐑶𐑷𐑸𐑹𐑺𐑻𐑼𐑽𐑾𐑿𐒀𐒁𐒂𐒃𐒄𐒅𐒆𐒇𐒈𐒉𐒊𐒋𐒌𐒍𐒎𐒏𐒐𐒑𐒒𐒓𐒔𐒕𐒖𐒗𐒘𐒙𐒚𐒛𐒜𐒝𐒞𐒟𐒠𐒡𐒢𐒣𐒤𐒥𐒦𐒧𐒨𐒩𐒪𐒫𐒬𐒭𐒮𐒯𐒰𐒱𐒲𐒳𐒴𐒵𐒶𐒷𐒸𐒹𐒺𐒻𐒼𐒽𐒾𐒿𐓀𐓁𐓂𐓃𐓄𐓅𐓆𐓇𐓈𐓉𐓊𐓋𐓌𐓍𐓎𐓏𐓐𐓑𐓒𐓓𐓔𐓕𐓖𐓗𐓘𐓙𐓚𐓛𐓜𐓝𐓞𐓟𐓠𐓡𐓢𐓣𐓤𐓥𐓦𐓧𐓨𐓩𐓪𐓫𐓬𐓭𐓮𐓯𐓰𐓱𐓲𐓳𐓴𐓵𐓶𐓷𐓸𐓹𐓺𐓻𐓼𐓽𐓾𐓿𐔀𐔁𐔂𐔃𐔄𐔅𐔆𐔇𐔈𐔉𐔊𐔋𐔌𐔍𐔎𐔏𐔐𐔑𐔒𐔓𐔔𐔕𐔖𐔗𐔘𐔙𐔚𐔛𐔜𐔝𐔞𐔟𐔠𐔡𐔢𐔣𐔤𐔥𐔦𐔧𐔨𐔩𐔪𐔫𐔬𐔭𐔮𐔯𐔰𐔱𐔲𐔳𐔴𐔵𐔶𐔷𐔸𐔹𐔺𐔻𐔼𐔽𐔾𐔿𐕀𐕁𐕂𐕃𐕄𐕅𐕆𐕇𐕈𐕉𐕊𐕋𐕌𐕍𐕎𐕏𐕐𐕑𐕒𐕓𐕔𐕕𐕖𐕗𐕘𐕙𐕚𐕛𐕜𐕝𐕞𐕟𐕠𐕡𐕢𐕣𐕤𐕥𐕦𐕧𐕨𐕩𐕪𐕫𐕬𐕭𐕮𐕯𐕰𐕱𐕲𐕳𐕴𐕵𐕶𐕷𐕸𐕹𐕺𐕻𐕼𐕽𐕾𐕿𐖀𐖁𐖂𐖃𐖄𐖅𐖆𐖇𐖈𐖉𐖊𐖋𐖌𐖍𐖎𐖏𐖐𐖑𐖒𐖓𐖔𐖕𐖖𐖗𐖘𐖙𐖚𐖛𐖜𐖝𐖞𐖟𐖠𐖡𐖢𐖣𐖤𐖥𐖦𐖧𐖨𐖩𐖪𐖫𐖬𐖭𐖮𐖯𐖰𐖱𐖲𐖳𐖴𐖵𐖶𐖷𐖸𐖹𐖺𐖻𐖼𐖽𐖾𐖿𐗀𐗁𐗂𐗃𐗄𐗅𐗆𐗇𐗈𐗉𐗊𐗋𐗌𐗍𐗎𐗏𐗐𐗑𐗒𐗓𐗔𐗕𐗖𐗗𐗘𐗙𐗚𐗛𐗜𐗝𐗞𐗟𐗠𐗡𐗢𐗣𐗤𐗥𐗦𐗧𐗨𐗩𐗪𐗫𐗬𐗭𐗮𐗯𐗰𐗱𐗲𐗳𐗴𐗵𐗶𐗷𐗸𐗹𐗺𐗻𐗼𐗽𐗾𐗿𐘀𐘁𐘂𐘃𐘄𐘅𐘆𐘇𐘈𐘉𐘊𐘋𐘌𐘍𐘎𐘏𐘐𐘑𐘒𐘓𐘔𐘕𐘖𐘗𐘘𐘙𐘚𐘛𐘜𐘝𐘞𐘟𐘠𐘡𐘢𐘣𐘤𐘥𐘦𐘧𐘨𐘩𐘪𐘫𐘬𐘭𐘮𐘯𐘰𐘱𐘲𐘳𐘴𐘵𐘶𐘷𐘸𐘹𐘺𐘻𐘼𐘽𐘾𐘿𐙀𐙁𐙂𐙃𐙄𐙅𐙆𐙇𐙈𐙉𐙊𐙋𐙌𐙍𐙎𐙏𐙐𐙑𐙒𐙓𐙔𐙕𐙖𐙗𐙘𐙙𐙚𐙛𐙜𐙝𐙞𐙟𐙠𐙡𐙢𐙣𐙤𐙥𐙦𐙧𐙨𐙩𐙪𐙫𐙬𐙭𐙮𐙯𐙰𐙱𐙲𐙳𐙴𐙵𐙶𐙷𐙸𐙹𐙺𐙻𐙼𐙽𐙾𐙿𐚀𐚁𐚂𐚃𐚄𐚅𐚆𐚇𐚈𐚉𐚊𐚋𐚌𐚍𐚎𐚏𐚐𐚑𐚒𐚓𐚔𐚕𐚖𐚗𐚘𐚙𐚚𐚛𐚜𐚝𐚞𐚟𐚠𐚡𐚢𐚣𐚤𐚥𐚦𐚧𐚨𐚩𐚪𐚫𐚬𐚭𐚮𐚯𐚰𐚱𐚲𐚳𐚴𐚵𐚶𐚷𐚸𐚹𐚺𐚻𐚼𐚽𐚾𐚿𐛀𐛁𐛂𐛃𐛄𐛅𐛆𐛇𐛈𐛉𐛊𐛋𐛌𐛍𐛎𐛏𐛐𐛑𐛒𐛓𐛔𐛕𐛖𐛗𐛘𐛙𐛚𐛛𐛜𐛝𐛞𐛟𐛠𐛡𐛢𐛣𐛤𐛥𐛦𐛧𐛨𐛩𐛪𐛫𐛬𐛭𐛮𐛯𐛰𐛱𐛲𐛳𐛴𐛵𐛶𐛷𐛸𐛹𐛺𐛻𐛼𐛽𐛾𐛿𐜀𐜁𐜂𐜃𐜄𐜅𐜆𐜇𐜈𐜉𐜊𐜋𐜌𐜍𐜎𐜏𐜐𐜑𐜒𐜓𐜔𐜕𐜖𐜗𐜘𐜙𐜚𐜛𐜜𐜝𐜞𐜟𐜠𐜡𐜢𐜣𐜤𐜥𐜦𐜧𐜨𐜩𐜪𐜫𐜬𐜭𐜮𐜯𐜰𐜱𐜲𐜳𐜴𐜵𐜶𐜷𐜸𐜹𐜺𐜻𐜼𐜽𐜾𐜿𐝀𐝁𐝂𐝃𐝄𐝅𐝆𐝇𐝈𐝉𐝊𐝋𐝌𐝍𐝎𐝏𐝐𐝑𐝒𐝓𐝔𐝕𐝖𐝗𐝘𐝙𐝚𐝛𐝜𐝝𐝞𐝟𐝠𐝡𐝢𐝣𐝤𐝥𐝦𐝧𐝨𐝩𐝪𐝫𐝬𐝭𐝮𐝯𐝰𐝱𐝲𐝳𐝴𐝵𐝶𐝷𐝸𐝹𐝺𐝻𐝼𐝽𐝾𐝿𐞀𐞁𐞂𐞃𐞄𐞅𐞆𐞇𐞈𐞉𐞊𐞋𐞌𐞍𐞎𐞏𐞐𐞑𐞒𐞓𐞔𐞕𐞖𐞗𐞘𐞙𐞚𐞛𐞜𐞝𐞞𐞟𐞠𐞡𐞢𐞣𐞤𐞥𐞦𐞧𐞨𐞩𐞪𐞫𐞬𐞭𐞮𐞯𐞰𐞱𐞲𐞳𐞴𐞵𐞶𐞷𐞸𐞹𐞺𐞻𐞼𐞽𐞾𐞿𐟀𐟁𐟂𐟃𐟄𐟅𐟆𐟇𐟈𐟉𐟊𐟋𐟌𐟍𐟎𐟏𐟐𐟑𐟒𐟓𐟔𐟕𐟖𐟗𐟘𐟙𐟚𐟛𐟜𐟝𐟞𐟟𐟠𐟡𐟢𐟣𐟤𐟥𐟦𐟧𐟨𐟩𐟪𐟫𐟬𐟭𐟮𐟯𐟰𐟱𐟲𐟳𐟴𐟵𐟶𐟷𐟸𐟹𐟺𐟻𐟼𐟽𐟾𐟿𐠀𐠁𐠂𐠃𐠄𐠅𐠆𐠇𐠈𐠉𐠊𐠋𐠌𐠍𐠎𐠏𐠐𐠑𐠒𐠓𐠔𐠕𐠖𐠗𐠘𐠙𐠚𐠛𐠜𐠝𐠞𐠟𐠠𐠡𐠢𐠣𐠤𐠥𐠦𐠧𐠨𐠩𐠪𐠫𐠬𐠭𐠮𐠯𐠰𐠱𐠲𐠳𐠴𐠵𐠶𐠷𐠸𐠹𐠺𐠻𐠼𐠽𐠾𐠿𐡀𐡁𐡂𐡃𐡄𐡅𐡆𐡇𐡈𐡉𐡊𐡋𐡌𐡍𐡎𐡏𐡐𐡑𐡒𐡓𐡔𐡕𐡖𐡗𐡘𐡙𐡚𐡛𐡜𐡝𐡞𐡟𐡠𐡡𐡢𐡣𐡤𐡥𐡦𐡧𐡨𐡩𐡪𐡫𐡬𐡭𐡮𐡯𐡰𐡱𐡲𐡳𐡴𐡵𐡶𐡷𐡸𐡹𐡺𐡻𐡼𐡽𐡾𐡿𐢀𐢁𐢂𐢃𐢄𐢅𐢆𐢇𐢈𐢉𐢊𐢋𐢌𐢍𐢎𐢏𐢐𐢑𐢒𐢓𐢔𐢕𐢖𐢗𐢘𐢙𐢚𐢛𐢜𐢝𐢞𐢟𐢠𐢡𐢢𐢣𐢤𐢥𐢦𐢧𐢨𐢩𐢪𐢫𐢬𐢭𐢮𐢯𐢰𐢱𐢲𐢳𐢴𐢵𐢶𐢷𐢸𐢹𐢺𐢻𐢼𐢽𐢾𐢿𐣀𐣁𐣂𐣃𐣄𐣅𐣆𐣇𐣈𐣉𐣊𐣋𐣌𐣍𐣎𐣏𐣐𐣑𐣒𐣓𐣔𐣕𐣖𐣗𐣘𐣙𐣚𐣛𐣜𐣝𐣞𐣟𐣠𐣡𐣢𐣣𐣤𐣥𐣦𐣧𐣨𐣩𐣪𐣫𐣬𐣭𐣮𐣯𐣰𐣱𐣲𐣳𐣴𐣵𐣶𐣷𐣸𐣹𐣺𐣻𐣼𐣽𐣾𐣿𐤀𐤁𐤂𐤃𐤄𐤅𐤆𐤇𐤈𐤉𐤊𐤋𐤌𐤍𐤎𐤏𐤐𐤑𐤒𐤓𐤔𐤕𐤖𐤗𐤘𐤙𐤚𐤛𐤜𐤝𐤞𐤟𐤠𐤡𐤢𐤣𐤤𐤥𐤦𐤧𐤨𐤩𐤪𐤫𐤬𐤭𐤮𐤯𐤰𐤱𐤲𐤳𐤴𐤵𐤶𐤷𐤸𐤹𐤺𐤻𐤼𐤽𐤾𐤿𐥀𐥁𐥂𐥃𐥄𐥅𐥆𐥇𐥈𐥉𐥊𐥋𐥌𐥍𐥎𐥏𐥐𐥑𐥒𐥓𐥔𐥕𐥖𐥗𐥘𐥙𐥚𐥛𐥜𐥝𐥞𐥟𐥠𐥡𐥢𐥣𐥤𐥥𐥦𐥧𐥨𐥩𐥪𐥫𐥬𐥭𐥮𐥯𐥰𐥱𐥲𐥳𐥴𐥵𐥶𐥷𐥸𐥹𐥺𐥻𐥼𐥽𐥾𐥿𐦀𐦁𐦂𐦃𐦄𐦅𐦆𐦇𐦈𐦉𐦊𐦋𐦌𐦍𐦎𐦏𐦐𐦑𐦒𐦓𐦔𐦕𐦖𐦗𐦘𐦙𐦚𐦛𐦜𐦝𐦞𐦟𐦠𐦡𐦢𐦣𐦤𐦥𐦦𐦧𐦨𐦩𐦪𐦫𐦬𐦭𐦮𐦯𐦰𐦱𐦲𐦳𐦴𐦵𐦶𐦷𐦸𐦹𐦺𐦻𐦼𐦽𐦾𐦿𐧀𐧁𐧂𐧃𐧄𐧅𐧆𐧇𐧈𐧉𐧊𐧋𐧌𐧍𐧎𐧏𐧐𐧑𐧒𐧓𐧔𐧕𐧖𐧗𐧘𐧙𐧚𐧛𐧜𐧝𐧞𐧟𐧠𐧡𐧢𐧣𐧤𐧥𐧦𐧧𐧨𐧩𐧪𐧫𐧬𐧭𐧮𐧯𐧰𐧱𐧲𐧳𐧴𐧵𐧶𐧷𐧸𐧹𐧺𐧻𐧼𐧽𐧾𐧿𐨀𐨁𐨂𐨃𐨄𐨅𐨆𐨇𐨈𐨉𐨊𐨋𐨌𐨍𐨎𐨏𐨐𐨑𐨒𐨓𐨔𐨕𐨖𐨗𐨘𐨙𐨚𐨛𐨜𐨝𐨞𐨟𐨠𐨡𐨢𐨣𐨤𐨥𐨦𐨧𐨨𐨩𐨪𐨫𐨬𐨭𐨮𐨯𐨰𐨱𐨲𐨳𐨴𐨵𐨶𐨷𐨹𐨺𐨸𐨻𐨼𐨽𐨾𐨿𐩀𐩁𐩂𐩃𐩄𐩅𐩆𐩇𐩈𐩉𐩊𐩋𐩌𐩍𐩎𐩏𐩐𐩑𐩒𐩓𐩔𐩕𐩖𐩗𐩘𐩙𐩚𐩛𐩜𐩝𐩞𐩟𐩠𐩡𐩢𐩣𐩤𐩥𐩦𐩧𐩨𐩩𐩪𐩫𐩬𐩭𐩮𐩯𐩰𐩱𐩲𐩳𐩴𐩵𐩶𐩷𐩸𐩹𐩺𐩻𐩼𐩽𐩾𐩿𐪀𐪁𐪂𐪃𐪄𐪅𐪆𐪇𐪈𐪉𐪊𐪋𐪌𐪍𐪎𐪏𐪐𐪑𐪒𐪓𐪔𐪕𐪖𐪗𐪘𐪙𐪚𐪛𐪜𐪝𐪞𐪟𐪠𐪡𐪢𐪣𐪤𐪥𐪦𐪧𐪨𐪩𐪪𐪫𐪬𐪭𐪮𐪯𐪰𐪱𐪲𐪳𐪴𐪵𐪶𐪷𐪸𐪹𐪺𐪻𐪼𐪽𐪾𐪿𐫀𐫁𐫂𐫃𐫄𐫅𐫆𐫇𐫈𐫉𐫊𐫋𐫌𐫍𐫎𐫏𐫐𐫑𐫒𐫓𐫔𐫕𐫖𐫗𐫘𐫙𐫚𐫛𐫜𐫝𐫞𐫟𐫠𐫡𐫢𐫣𐫤𐫦𐫥𐫧𐫨𐫩𐫪𐫫𐫬𐫭𐫮𐫯𐫰𐫱𐫲𐫳𐫴𐫵𐫶𐫷𐫸𐫹𐫺𐫻𐫼𐫽𐫾𐫿𐬀𐬁𐬂𐬃𐬄𐬅𐬆𐬇𐬈𐬉𐬊𐬋𐬌𐬍𐬎𐬏𐬐𐬑𐬒𐬓𐬔𐬕𐬖𐬗𐬘𐬙𐬚𐬛𐬜𐬝𐬞𐬟𐬠𐬡𐬢𐬣𐬤𐬥𐬦𐬧𐬨𐬩𐬪𐬫𐬬𐬭𐬮𐬯𐬰𐬱𐬲𐬳𐬴𐬵𐬶𐬷𐬸𐬹𐬺𐬻𐬼𐬽𐬾𐬿𐭀𐭁𐭂𐭃𐭄𐭅𐭆𐭇𐭈𐭉𐭊𐭋𐭌𐭍𐭎𐭏𐭐𐭑𐭒𐭓𐭔𐭕𐭖𐭗𐭘𐭙𐭚𐭛𐭜𐭝𐭞𐭟𐭠𐭡𐭢𐭣𐭤𐭥𐭦𐭧𐭨𐭩𐭪𐭫𐭬𐭭𐭮𐭯𐭰𐭱𐭲𐭳𐭴𐭵𐭶𐭷𐭸𐭹𐭺𐭻𐭼𐭽𐭾𐭿𐮀𐮁𐮂𐮃𐮄𐮅𐮆𐮇𐮈𐮉𐮊𐮋𐮌𐮍𐮎𐮏𐮐𐮑𐮒𐮓𐮔𐮕𐮖𐮗𐮘𐮙𐮚𐮛𐮜𐮝𐮞𐮟𐮠𐮡𐮢𐮣𐮤𐮥𐮦𐮧𐮨𐮩𐮪𐮫𐮬𐮭𐮮𐮯𐮰𐮱𐮲𐮳𐮴𐮵𐮶𐮷𐮸𐮹𐮺𐮻𐮼𐮽𐮾𐮿𐯀𐯁𐯂𐯃𐯄𐯅𐯆𐯇𐯈𐯉𐯊𐯋𐯌𐯍𐯎𐯏𐯐𐯑𐯒𐯓𐯔𐯕𐯖𐯗𐯘𐯙𐯚𐯛𐯜𐯝𐯞𐯟𐯠𐯡𐯢𐯣𐯤𐯥𐯦𐯧𐯨𐯩𐯪𐯫𐯬𐯭𐯮𐯯𐯰𐯱𐯲𐯳𐯴𐯵

65. **dlig** – Discretionary Ligatures

Produces lesser-used ligatures, but also roman numbers, e.g. ii, II, xi, XI. The lesser-used ligatures: **ct**, **fp**, **str**, **st**, **tr**, **tt**, **ty**.

66. **ss17** – Rare Digraphs

By “digraph” we mean conjoined letters that represent a phonetic value: the most common examples for western languages are **æ** and **œ** (though these, because they are so common, are not included in this feature). Use of this feature in web pages enables easier searches: for example, producing **þau** from **þau** allows the word to be searched as “þau.” The digraphs covered by this feature are **au**, **æu**, **au**, **æu**, **æu**, **æu**, **æu**, **æu**, plus capital and small cap equivalents and digraph + diacritic combinations anticipated in the MUFI recommendation. To produce such a digraph + diacritic combination, either type a letter + diacritic combination as the second element of the digraph or type the diacritic after the second element. For example, **a** + **ú** yields **áu**, and so does **a** + **u** + U+0301 (combining acute accent). To produce a digraph + diacritic combination not covered by MUFI (e.g. **ǣ**), you may have to place U+034F COMBINING GRAPHEME JOINER (see [cv84](#) above) between the second element of the digraph and the combining mark. Without U+034F: **ǣ**. With U+034F: **ǣ**.

N. Required Features

Required features, which provide some of the font’s most basic functionality—ligatures, support for other features, kerning, and more—include **ccmp** (Glyph Composition/Decomposition), **calt** (Contextual Alternates), **liga** (Standard Ligatures), **loca** (Localized Forms), **rlig** (Required Ligatures), **kern** (Horizontal Kerning), and **mark/mkmk** (Mark Positioning). In MS Word these features have to be explicitly enabled on the Advanced tab of the Font dialog (Ctrl-D or Cmd-D: enable Kerning, Standard Ligatures, and Contextual Alternates, and the others will be enabled automatically), but in most other applications they are enabled by default.

O. Non-MUFI Code Points

Characters in Junicode that do not have Unicode code points should be accessed via OpenType features whenever possible. MUFI/PUA code points should be used only in

JUNICODE 2 FEATURE REFERENCE

applications that do not support OpenType, or that support it only partially (for example, MS Word). For certain characters that lack either Unicode or MUFI code points, code points in the Supplementary Private Use Area-A (plane 15) are available.

U+F0000 \mathcal{A}	U+F0008 \bar{d}	U+F0010 \overline{m}	U+F0018 \bar{v}
U+F0001 \mathfrak{a}	U+F0009 \mathbb{E}	U+F0011 \circ	U+F0019 \bar{x}
U+F0002 \mathcal{A}_2	U+F000A \mathfrak{c}	U+F0012 \emptyset	U+F001A \bar{x}
U+F0003 \bar{c}	U+F000B f	U+F0013 \circ	U+F001B \mathfrak{z}
U+F0004 \bar{c}	U+F000C \bar{i}	U+F0014 \circ	U+F001C \mathcal{Z}
U+F0005 \mathfrak{b}	U+F000D j	U+F0015 \mathfrak{q}	U+F001D \mathfrak{z}
U+F0006 \mathfrak{b}	U+F000E \bar{j}	U+F0016 \mathfrak{z}	U+F001E \mathcal{A}
U+F0007 \mathfrak{d}	U+F000F \mathcal{P}	U+F0017 \bar{v}	U+F001F \mathfrak{a}

*This document was set in 12pt Junicode SemiExpanded
using the Xe_{La}TeX typesetting system with fontspec for font management.
The source for the document, Feature_Reference.tex, is available at
<https://github.com/psb1558/Junicode-font>.*