OpenType Features in JuniusX / JuniusVF

Introduction

The OpenType features of JuniusX 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 point[1] 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.

1 A 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 “ȝ” (Middle English yogh) is U+021D.

The problem with PUA code points is precisely their lack of any value. For example, if you encode the PUA character U+F215 latin small letter neckless a (ꭺ) in your text, the software that displays the text has no notion 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**.

JuniusX 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 a display text (perhaps generated from an XML document) as having two layers: an underlying text and the one your readers see. In the case of U+F215 the underlying text should contain the plain letter **a** (U+0061); a layout engine applies OpenType feature `cv01[5]`[2] to this **a**, bypassing the PUA code point, and the result is that your readers see **ꭺ**—the "neckless a"—while searches and screen readers operate on the underlying text, which still contains a plain **a**.

The full range of OpenType features listed in this document is supported by 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 JuniusX. These characters fall into three categories:

- Those with 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 safe to use in accessible and searchable text. As a convenience, however, many of these are covered by JuniusX OpenType features.

2 Many 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.

- Precomposed characters—those consisting of base character + one or more diacritic. 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: ẹ́ (when applying combining marks, start with any marks below the character and work downwards, then continue with any marks above the character and work upwards. For example, to make ǭ̣́, place characters in this order: o, ogonek U+0328, dot below U+0323, macron U+0304, acute U+0301). Some MUFI characters have marks in unconventional locations, e.g. ȯ́ 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, dot above U+0307, acute U+0301). JuniusX 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:
  - **** U+E8AF. This is a ligature of **ſ** and **l** with stroke, but there are no base characters with this style of stroke.
  - **** U+EFD8 and U+EFD9. MUFI lists these as ligatures (corresponding to the ligatures U+E8C7 and U+E8C6, 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 **ꜳꜲ**.
  - **** U+EBE7 and U+EBE6, for the same reason.
  - **** 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 JuniusX.

- **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 + ogonek (U+0328) + acute (U+0301)** will change ţ́ to ᴛ̨́.

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. klmnoþ → KLMNOÞ.

4. `case` – Case-Sensitive Forms

Produces combining marks that harmonize with capital letters: Ř, X̉, 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: ⓐ ⓑ . . . ⓩ; ⓪ ① ② . . . ⑳.
`nalt[2]`, letter or numbers in parentheses: (a) . . . (z); (0) (1) . . . (20).
`nalt[3]`, double-circled numbers: ◎ ⓵ . . . ⓾.
`nalt[4]`, white numbers in black circles: ⓿ ❶ ❷ ❸ . . . ⓴
`nalt[5]`, numbers followed by period: 0. 1. . . . 20.

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` so 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. `sups` – 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 uppercase letters: $^{0123\ 4567\ abcde\ ABDEG}$.

12. `subs` – Subscripts

Produces subscript numbers. Only affects lining tabular and oldstyle 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 a variant 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 JuniusX is not available or `ornm` is not implemented.

E. Alphabetic Variants

For features where one or more case-groups are listed (in the order lowercase-uppercase-small cap), missing case forms should be assumed to be the default. For example, for `cv01` "Variants of rR," these forms are given:

1=ꞃꞂꞃ, 2=ɼ.

Understand this string as shorthand for the following:

1=ꞃꞂꞃ, 2=ɼRʀ.

That is, `cv24[2]` changes only the lowercase form, but `cv24[1]` changes uppercase, lowercase and small cap.

For a number of especially complex features, the collection of variants is displayed in a table with the indices on the x-axis and the characters affected on the y-axis.

14. `ss02` – Insular Letter-Forms

Produces insular letter-forms, e.g. ꝺꝼᵹꞃꞅƿ. Does not affect capitals (except W), as these do not not commonly have insular shapes in early manuscripts. For these, enter the Unicode code points or use the Character Variant (`cvNN`) features.

15. `ss04` – High Overline

Produces a high overline over letters used as roman numbers: c̅d̅i̅j̅l̅m̅v̅x̅ C̅D̅I̅J̅L̅M̅V̅X̅Ↄ̅.

16. `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: b̄c̄d̄h̄ı̄ȷ̄k̄l̄m̄ſ̄v̄x̄ƿ̄.

17. `ss06` – Enlarged Minuscules

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.

18. `ss07` – Underdotted Text

Produces underdotted text (a standard way of indicating deletion in medieval manuscripts) for many letters (including the whole of the basic Latin alphabet and a number of other letters), e.g. ạḅc̣ḍẹf̣g̣ Ḥ Ị J̣ Ḳ Ḷ Ṃ. This also affects small caps, e.g. ᴀʙᴄᴅᴇꜰ → ᴀ̣ʙ̣ᴄ̣ᴅ̣ᴇ̣ꜰ̣. For letters without corresponding underdotted forms (e.g. U+A751, ꝑ), use U+0323, combining dot below (ꝑ̣).

19. `cv01` – Variants of aA

With an index of 1–7, this feature affects aA and certain digraphs with aA as a component (boxes are blank where the feature has no effect). See also `cv34`–`cv37`.

	1	2	3	4	5
a	ɑ	a	ɑ	a	a
A	Ɑ	A	A		
ᴀ	ɑ				

20. cv38 – Variant of ä

1=ä.

21. `cv02` – Variants of ąĄ

	1	2	3
ą	ą	ą	
Ą	Ą	Ą	Ą
ᴀ̨	ᴀ̨	ᴀ̨	

`cv03`

There are no variants of bB.

22. `cv04` – Variant of cC

1=c, 2=C.

23. `cv05` – Variants of dD

	1	2	3
d	ꝺ	ꝺ	d
D	Ꝺ		
ᴅ	ꝺ		

24. `cv06` – Variant of đ (U+0111, d with stroke)

1=ď

25. `cv07` – Variants of eE

To produce the small cap variant at `cv07[2]`, you must also apply `c2sc` (Capitals to Small Caps) to capital E. Applying `smcp` (Small Caps) will not work.

	1	2	3
e	ϵ	e	e
E	Є	Є	
ᴇ	ϵ	ϵ	

26. `cv08` – Variants of ęĘ

The default forms of ę are appropriate for modern languages, the forms produced by `cv08[1]` for medieval Latin and other older languages. The forms produced by `cv08[2]` are used in some older Polish printed books. Note that this feature also affects combining ę (but only when produced by the entity reference `&_eogo;` with `ss10`): and *remember that a feature that affects a combining mark must also be applied to the base character*).

Note that forms with dot and acute accents must be produced by typing ę + dot accent (U+0307) + acute accent (U+0301). The MUFI precomposed character (U+E4EC) will not work. Similarly, the “Enlarged minuscule” form must have been produced by applying `ss07` (“Enlarged Minuscules”) to ę.

	1	2
ę	ę	ę
Ę	Ę	Ę
ᴇ̨	ᴇ̨	ᴇ̨
ę̣	ę̣	
Ę̣	Ę̣	
ᴇ̨̣	ᴇ̨̣	
ę̇́	ę̇́	
Ę̇́	Ę̇́	
ᴇ̨̇́	ᴇ̨̇́	
ę	ę	
◌ᷤ	◌ᷤ	

27. `cv09` – Variants of f F

`cv09[1]` also affects the underdotted form of f (produced by `ss07`):

	1	2	3	4	5	6
f	ꝼ	ꝼ	ꝓ	ꝥ	f	fc
F	Ꝼ					
ꜰ	ꝼ					
f̣	ꝼ̣					
F̣	Ꝼ̣					
ꜰ̣	ꝼ̣					

28. `cv10` – Variants of Gg

	1	2	3	4	5	6
g	ᵹ	ɡ	g	g	ɡ	gꝛ
G	Ᵹ	Ɡ				
ɢ	ᵹ	ɢ				

29. `cv11` – Variants of ȝȜ (Yogh)

1=ȝȜᴣ. This feature also affects the yogh with dot below: 1=ȝ̣Ȝ̣ᴣ̣.

30. `cv12` – Variants of hH

1=ꜧꜦ, 2=ꞕ.

31. `cv13` – Variants of iI

`cv13[1]` toggles the dot on lowercase, capital and small cap i: where the dot is present it is deleted, and where the dot is missing it is added.

	1	2	3
i	ı	ɩ	ɨ̇
ı	i		
I	İ	I	
İ	I		
ɪ	ɪ̇		
ɪ̇	ɪ		

32. `cv14` – Variants of jJ

1=ȷJ̇j, 2=ɉ.

33. `cv15` – Variants of k

1=k, 2=k, 3=ꝁ, 4=ꝁ.

34. `cv16` – Variant of l

1=ɭ.

35. `cv17` – Variant of ꝉ (U+A749, l with high stroke)

1=ƚ.

36. `cv18` – Variants of mM

	1	2	3
m	m	ɱ	ɱ
M	Ɱ	Ɱ	*M*
ᴍ	m	ɱ	

37. `cv19` – Variants of nN

	1	2	3	4
n	ŋ	ɴ	ꞥ	n
N	Ŋ			
ɴ	ŋ	ɴ		

`cv20`

There are no variants of oO.

38. `cv21` – Variants of ø

1=ø, 2=ø, 3=o, 4=o.

39. `cv39` – Variant of ö

1=ö.

40. `cv52` – Variants of ꝥꝥ (U+A765)

1=ꝥ in English text, ꝥ elsewhere. See also `ss01`.

41. cv22 – Variant of P

1=ꟼ.

42. cv23 – Variants of qQ

1=ꝗ, 2=ꟼ.

43. cv24 – Variants of rR

1=ɼꞃɼ, 2=r̉.

44. ss11 – r Rotunda

In lowercase and small caps, substitutes ꝛ rotunda (ꝛꝛ) for r. This features does not affect capital R: the uncommon Ꝛ (U+A75A) must be entered manually. See also ss16.

45. ss16 – Contextual r Rotunda

Converts r to ꝛ (lowercase only) following the most common rules of medieval manuscripts: pꝛiest, firmer, frost, oꝛnament. For this feature to work properly, `calt` “Contextual Alternates” must also be enabled (as it should be by default: see Required Features below). See also ss11.

46. cv25 – Variants of sS

	1	2	3	4	5	6
s	ꞅ	ȿ	ſ	ʃ	ẜ	ẝ
S	Ꞅ	Ȿ				
s	ꞅ	ȿ				

47. hist – Historical Forms

Changes s to ſ (longs).

48. ss03 – Long s

Changes s to ſ (duplicating `hist`). see also `ss08`.

49. `ss08` – Contextual Long s

In English and French text only, varies s and ſ according to rules followed by many early printers: ſports, eſſence, ſtormy, diſheveled, transfuſions, flyneſs, cliffside. For this feature to work properly, `calt` "Contextual Alternates" must also be enabled (as it should be by default: see Required Features below).

50. `cv26` – Variants of tT

1=ꞇƬꞇ, 2=ꝷ.

`cv27`

There are no variants of uU.

51. `cv28` – Variants of vV

1=ꝩ, 2=v̄, 3=v̿, 4=ꝟ.

`cv29`

There are no variants of wW.

52. `cv30` – Variants of x

1=x, 2=x, 3=x, 4=x, 5=x.

53. `cv31` – Variants of y

1=ỿ, 2=ɏ, 3=Y.

54. `cv32` – Variants of zZ

1=ʒƷꝫ, 2=ƕ.

55. `ss01` – Alternate thorn and eth

Produces Nordic thorn and eth (þðÞ) when the language is English, and English thorn and eth (þðꝥ) with any other language. This also affects small caps, crossed thorn (ꝥ ꝥ—see also `cv52`), combining mark eth (U+1DD9, ◌ᷙ ◌ᷙ), and enlarged thorn and eth (see `ss06`). This feature depends on `loca` (Localized Forms), which in most applications will always be enabled.

56. `cv33` – ᴀ to a

1=a. This features reverts small cap ᴀ to a, enabling it to ligature with small cap ɴ or ʀ via `hlig`: ᴀɴ, ᴀʀ. Be sure to apply `smcp`, `cv33` and `hlig` to both components of the ligature.

57. `cv34` – Variants of ꜳ (U+A733)

1=ꜳ, 2=ꜳ.

58. `cv35` – Variants of æÆ

1=æÆᴁ, 2=æ, 3=æ.

59. `cv36` – Variants of ꜵꜴ (U+A735, A734)

1=ꜵꜴꜵ, 2=ꜵ. `cv36[1] also` produces a variant of the “enlarged minuscule” ꜵ: ꜵ.

60. cv37 – Variant of ꜹ (U+A739)

1=ꜹ.

61. `cv50` – Variant of ʔ (U+0294, glottal stop)

1=ˀ.

F. Punctuation

MUFI encodes nearly twenty marks of punctuation in the PUA. In JuniusX 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 JuniusX 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* (⹍ U+2E4D), medieval comma (⹌ U+2E4C), *punctus elevatus* (⹎ U+2E4E), *virgula suspensiva* (⹊ U+2E4A), triple dagger (⹋ U+2E4B).

62. `ss18` – Old-Style Punctuation Spacing

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

63. `cv40` – Variants of ⁊⹒ (U+204A / U+2E52, Tironian nota)

1=ꝣꝢ, 2=ꝣ̵Ꝣ̵.

64. `cv79` – Variants of . (period)

1=˙, 2=„, 3=’, 4=;, 5=.„, 6=:„, 7=·;·, 8=⁖;, 9=⹎, 10=⸮, 11=.⸍, 12=ˀ, 13=!, 14=⸮̃, 15=∻, 16=⸪, 17=~, 18=∵, 19=⹊, 20=/. See also `cv70`–`cv78`.

65. `cv70` – Variant of · (U+00B7, middle dot)

1=˙ (*distinctio*).

66. `cv71` – Variants of , (comma)

1=„, 2=’.

67. `cv72` – Variants of ; (semicolon)

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

68. `cv73` – Variants of ⹎ (U+2E4E, *punctus elevatus*)

1=⹎, 2=⸮, 3=.⸍, 4=ˀ (*punctus flexus*).

69. `cv74` – Variant of ! (exclamation mark)

1=! (*punctus exclamativus*).

70. cv75 – Variants of ? (question mark)

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

71. cv76 – Variant of ~ (ASCII tilde)

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

72. cv77 – 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. JuniusX defines it as a variant of the asterisk—the most common modern *signe de renvoi*.

73. cv78 – Variants of / (slash)

1=⹎, 2=/. The first of these is Unicode, U+2E4E.

G. Abbreviations

74. cv41 – Variant of ꝝ (U+A75D, rum abbreviation)

1=ꝝ.

75. cv42 – Variants of ◌͛ (U+035B, combining zigzag above)

1=◌͛, 2=◌͛, 3=◌͛. Positioning of the zigzag can differ from that of other combining marks, e.g. b͛, f͛, d͛. If `calt` “Contextual Alternates” is enabled (as it is by default in most apps), variant forms of `cv42[2]` will be used with several letters, e.g. d͛, f͛, k͛. Enable `case` for forms that harmonize with capitals (A͛ B͛ C͛ D͛), `smcp` for forms that harmonize with small caps (ᴇ͛ ꜰ͛ ɢ͛ ʜ͛).

76. cv53 – Variants of spacing ꝰ (U+A770)

1=ꝰ, 2=ꝰ. `cv53[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.

77. cv54 – Variant of ꝫ (U+A76B, “et” abbreviation)

1=;. 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.

H. Combining Marks

78. cv48 – MUFI combining marks (variants of U+1DD1)

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+1DD1 (◌᷑, combining ur above) and apply `cv48`, 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 (`cv48[35]`, `[36]` and `[37]`) that lack MUFI code points but are used to form MUFI characters.

These marks can sometimes be produced by other features (see `cv42`–`cv46`), which may be preferable to `cv48` as providing 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. Note that entities beginning **&_** (ampersand + underscore) are for combining diacritics.

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 `cv48[11]` $\overset{\text{b}}{◌}$ to `[12]` $\overset{\text{ʙ}}{◌}$, it would be impossible to produce the sequence ɴ$\overset{\text{b}}{\text{ᴀ}}$ᴀ with the diacritic properly positioned.

1=◌᷑, 2=$\overset{\dot{-}}{◌◌}$, 3=$\overset{\text{⁊}}{◌}$, 4=◌ᷓ, 5=$\overset{\text{ʾ}}{◌}$, 6=$\overset{\bar{u}}{◌}$, 7=$\overset{\text{an}}{◌}$, 8=$\overset{\text{aɴ}}{◌}$, 9=$\overset{\text{ar}}{◌}$, 10=$\overset{\text{aʀ}}{◌}$, 11=$\overset{\text{b}}{◌}$, 12=$\overset{\text{ʙ}}{◌}$, 13=$\overset{\text{ᴅ}}{◌}$, 14=$\overset{\text{ē}}{◌}$, 15=$\overset{\text{ę}}{◌}$, 16=$\overset{\text{f}}{◌}$, 17=$\overset{\text{ı}}{◌}$, 18=$\overset{\text{j}}{◌}$, 19=$\overset{\text{ȷ}}{◌}$, 20=$\overset{\text{ᴋ}}{◌}$, 21=$\overset{\text{m}}{◌}$, 22=$\overset{\text{ō}}{◌}$, 23=$\overset{\text{ǫ}}{◌}$, 24=$\overset{\text{ø}}{◌}$, 25=$\overset{\text{p}}{◌}$, 26=$\overset{\text{q}}{◌}$, 27=$\overset{\text{ꝝ}}{◌}$, 28=$\overset{\text{ᴛ}}{◌}$, 29=$\overset{\text{ꞇ}}{◌}$, 30=$\overset{\text{w}}{◌}$, 31=$\overset{\text{y}}{◌}$, 32=$\overset{\text{þ}}{◌}$, 33=$\overset{\text{oꝛ}}{◌}$, 34=$\overset{\text{oꝝ}}{◌}$, 35=$\overset{\text{.:}}{◌}$, 36=$\overset{\text{·′}}{◌}$, 37=$\underset{\text{¿}}{◌}$.

79. ss20 – Low Diacritics

The MUFI recommendation includes a number of precomposed characters with base letters b, h, k, þ, ꝺ and ð and combining marks ◌ͣ (U+0363), ◌ͤ (U+0364), ◌᷑ (U+1DD1/cv48[17]), ◌ͦ (U+0366), ◌ͬ (U+036C), ◌ᷢ U+1DE2), ◌ͭ (U+036D), ◌ͮ (U+036E), ◌ᷦ (U+1DE6) and ◌᷑ (U+1DD1/cv48[21]). Instead of being positioned above ascender height as usual (e.g. hͣ), the MUFI glyphs have the marks positioned above the x-height (e.g. hͣ). Using the MUFI code points for these precomposed glyphs can interfere with searching and drastically reduce accessibility. Users of JuniusX 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: bͤ, ðᷦ, hͫ, kͤ, þͭ, ðᷢ.

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

80. `cv43` – Variant of ◌ᷓ (U+1DD3, combining open a)

1=◌ᷓ.

81. `cv44` – Variant of ◌ᷣ (U+1DE3, combining r rotunda)

1=◌ᷣ.

82. `cv45` – Variant of ◌̅◌ (U+0305, two-letter overline)

1=◌̅◌.

83. `cv46` – Variant of ◌̃ (U+0303, combining tilde)

1=◌̃.

84. `cv47` – Variants of short horizontal stroke (U+0335)

1= ◌̵, 2= ◌̵, 3= ◌̵
This character can be used with letters with ascenders or descenders, e.g. đ ƀ ꝥ ꝑ.
`cv47[1]` widens the stroke, and `cv47[2]` and `[3]` offset the stroke to the right or left.

Via `calt` “Contextual Alternates,” this offset is performed automatically for certain characters with ascenders, e.g. ƀ đ ꝥ ꝧ. Thus it should rarely be necessary to use an index with `cv47`.

85. cv61 – Variant of breve below (U+032E)

1=◌◌̮◌. Position the mark after the middle of three glyphs, and apply `cv61` to both the mark and (at least) the middle glyph. This mark is not available via `cv48`.

I. Currency and Weights

86. `cv55` – Variants of ¤ (U+0044, generic currency sign)

1=ΠΙ, 2=x̶, 3=ɏ, 4=ᵽ, 5=ʒ, 6=v̈, 7=ƒ, 8=gꝶ, 9=ɸ, 10=ꝟ, 11=ℓb, 12=ꝉt, 13=ℒ̶, 14=ℒ, 15=Ƚ, 16=£, 17=ꝉ, 18=ꝉ, 19=mꝉ, 20=ꝟ, 21=Ꞓ, 22=ꝟr, 23=ꝑ, 24=ß, 25=ßℓ, 26=ʒ, 27=℈. 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).

87. `cv56` – Variant of ℔ (U+2114)

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

88. `cv57` – Variants of £ (U+00A3, British pound sign)

1=ℓb, 2=ꝉt, 3=ℒ̶, 4=ℒ, 5=Ƚ, 6=£. Same as MUFI U+F2EA, F2EB, F2EC, F2ED, F2EE, F2EF, pound signs from various locales.

89. `cv58` – Variant of ₰ (U+20B0, German penny sign)

1=ꝟr. Same as MUFI U+F2F5.

90. `cv59` – Variant of ƒ (U+0192, florin)

1=ƒ. Same as MUFI U+F2E8.

91. `cv60` – Variant of ℥ (U+2125, Ounce sign)

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

J. Gothic

92. `ss19` – Latin to Gothic Transliteration

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

K. Runic

93. `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 → ᚠᛁᛋᚳ ᚠᛚᚩᛞᚢ ᚪᚻᚩᚠ.

94. `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 → ᚨᛒᚲᛞᛖᚠᚷ.

95. `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 → ᛅᛒᚴᛏᛁᚠᚴ.

96. `ss15` – Long Branch to Short Twig

In combination with `ss14`, converts long branch to short twig runes: ᛅᛒᚴᛏᛁᚠᚴ → ᛆᛓᚴᛐᛁᚠᚴ.

97. `rtlm` – Right to Left Mirrored Forms

Produces mirrored runes, e.g. ᚨᛒᚲᛞᛖᚠᚷ → ᚨᛒᚲᛞᛖᚠᚷ.

L. Ligatures and Digraphs

98. `hlig` – Historic Ligatures

Produces ligatures for combinations that should not ordinarily be rendered as ligatures in modern text.[3] Most of these are from the MUFI recommendation, ranges B.1.1(b) and B.1.4. This feature does not produce digraphs (which have a phonetic value), for which see ss17. The ligatures:

af→af	ck→ck	hr→hr	PP→PP	ſtr→ſtr
aꝼ→aꝼ	ꝺꝺ→ꝺꝺ	hſ→hſ	pp→pp	ſƿ→ſƿ
ag→ag	ey→ey	hf→hf	ꝓp→ꝓp	ꞇꞇ→ꞇꞇ
al→al	fä→fä	kr→kr	Ps→Ps	UE→UE
an→an	gd→gd	kſ→kſ	ps→ps	ue→ue
aɴ→aɴ	gð→gð	kf→kf	Psi→Psi	UU→UU
ap→ap	gꝺ→gꝺ	ll→ll	psi→psi	uu→uu
ar→ar	gg→gg	ɴſ→ɴſ	qƿ→qƿ	þr→þr
aʀ→aʀ	ꝿg→ꝿg	oc→oc	qʒ→qʒ	þſ→þſ
aþ→aþ	go→go	OꝚ→OꝚ	QꝚ→QꝚ	þf→þf.
bb→bb	gp→gp	oꝛ→oꝛ	qꝛ→qꝛ	
bg→bg	gr→gr	OꝜ→OꝜ	ſä→ſä	
ch→ch	Hr→Hr	oꝝ→oꝝ	ſch→ſch	

Notes: For **aɴ** and **aʀ** see `cv33` above. For the ligature **ɴſ** to work properly, U+017F **ſ** must be entered directly, not by applying an OpenType feature to **s**.

99. `dlig` – Discretionary Ligatures

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

100. `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

3 Some fonts define `hlig` differently, as including all ligatures in which at least one element is an archaic character, e.g. those involving **long s** (ſ). However, there is no circumstance in which a ligature of, say, **ſ** and **l** (as in “ſlick”) would be inappropriate, and the JuniusX definition of `hlig` reflects that position.

so common, are not included in this feature). Use of this feature in web pages enables easier searches: for example, producing **þꜷ** from **þau** allows the word to be searched as "þau." The digraphs covered by this feature are ꜳ, ꜵ, ꜷ, ꜹ, ꜽ, ꝏ, ꝡ, 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 **ꜷ́**, and so does a + u + U+0301 (combining acute accent). To produce a digraph + diacritic combination not covered by MUFI (e.g. **ꜵ̈**), you may have to enter the digraph directly and not via `ss17` (rare diacritics, however, are usually safe to use with this feature—e.g. **ꜳ̀̊**). Note that `ss17` must be applied to the two elements of the digraph *and* any following diacritic.

M. 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.

N. Entities

101. `ss10` – Character Entity References

In XML and HTML, characters that can't easily be typed on a keyboard can be expressed as entity references consisting of an ampersand, a name, and a semicolon. The XML standard defines a few entities and HTML many more, but document authors can define as many as they like in a DTD. JuniusX anticipates the need of medievalists for entities in addition to those defined in XML and HTML and defines more than a hundred of them. A few of these overlap with the collection of HTML entities, but most are peculiar to JuniusX, emphasizing characters that are widely used in medieval texts and those that have only PUA code points (especially combining marks, which present special technical difficulties).

In applications that support Stylistic Sets, `ss10` makes these entities appear as the characters they represent. But as such entities can interfere with searching and accessibility when embedded in a web page, you should think of them as a convenient set of mnemonics to be used when typing, to be replaced with their Unicode equivalents before publication. If you use any non-Unicode characters (highlighted in the list below), you should call your publisher's attention to them, since they will present difficulties to typesetters.

&aa; → ꜳ	&q1; → ꝗ	&_aelig; → ◌ᷔ
&AA; → Ꜳ	&q2; → ꝙ	&_b; → ◌ᷨ
&aelig; → æ	&rr; → ꝛ	&_bsc; → $\overset{\text{ʙ}}{◌}$
&AElig; → Æ	&ru; → ꝝ	&_c; → ◌ͨ
&ao; → ꜵ	&is; → ꝭ	&_ccedil; → ◌ᷗ
&AO; → Ꜵ	&sd; → ẝ	&_d; → ◌ͩ
&au; → ꜷ	&US; → ꝰ	&_dsc; → $\overset{\text{ᴅ}}{◌}$
&AU; → Ꜷ	&vy; → ꝡ	&_dins; → $\overset{\text{ꝺ}}{◌}$
&av; → ꜹ	&VY; → Ꝡ	&_eth; → ◌ᷙ
&AV; → Ꜹ	&wn; → ƿ	&_e; → ◌ͤ
&ay; → ꜽ	&WN; → Ƿ	&_eogo; → $\overset{\text{ę}}{◌}$
&AY; → Ꜽ	&thorn; → þ	&_emac; → $\overset{\text{ē}}{◌}$
&co; → ꝯ	&THORN; → Þ	&_f; → ◌ᷫ
&eth; → ð	&ct; → ꝥ	&_g; → ◌ᷚ
&ETH; → Ð	&_ZZ; → ◌᷏	&_gsc; → ◌ᷛ
&et; → ꝫ	&_zz; → ◌͛	&_h; → ◌ͪ
&ti; → ⁊	&_us; → ◌᷒	&_i; → ◌ͥ
&yo; → ȝ	&_ol; → ◌︦◌	&_idotl; → $\overset{\text{ı}}{◌}$
&YO; → Ȝ	&_a; → ◌ͣ	&_j; → $\overset{\text{j}}{◌}$
&kl; → ꝁ	&_oa; → ◌ᷓ	&_jdotl; → $\overset{\text{ȷ}}{◌}$
&ob; → ꝋ	&_ansc; → $\overset{\text{ᴀɴ}}{◌}$	&_k; → ◌ᷜ
&OB; → Ꝋ	&_an; → $\overset{\text{an}}{◌}$	&_ksc; → $\overset{\text{ᴋ}}{◌}$
&OO; → Ꝏ	&_ar; → $\overset{\text{ar}}{◌}$	&_l; → ◌ᷝ
&oo; → ꝏ	&_arsc; → $\overset{\text{ᴀʀ}}{◌}$	&_lsc; → ◌ᷞ
&pr; → ꝑ	&_ao; → ◌ᷕ	&_m; → ◌ͫ
&po; → ꝓ	&_av; → ◌ᷖ	&_msc; → ◌ᷟ

&_munc; → ◌ͫ
&_n; → ◌ᷠ
&_nsc; → ◌ᷡ
&_o; → ◌ͦ
&_oogo; → $\overset{ǫ}{◌}$
&_oslash; → $\overset{ø}{◌}$
&_omac; → $\overset{ō}{◌}$
&_orr; → $\overset{oꝛ}{◌}$
&_oru; → $\overset{oꝝ}{◌}$
&_p; → ◌ᷮ

&_q; → $\overset{q}{◌}$
&_r; → ◌ͬ
&_rr; → ◌ᷣ
&_rsc; → ◌ᷢ
&_ur; → $\overset{ꝰ}{◌}$
&_ru; → $\overset{ꝝ}{◌}$
&_s; → ◌ᷤ
&_longs; → ◌ᷥ
&_t; → ◌ͭ
&_tins; → $\overset{ꞇ}{◌}$

&_tsc; → $\overset{ᴛ}{◌}$
&_u; → ◌ͧ
&_v; → ◌ͮ
&_w; → ◌ᷱ
&_x; → ◌ͯ
&_y; → $\overset{y}{◌}$
&_thorn; → $\overset{þ}{◌}$
&_z; → ◌ᷦ

O. Non-MUFI Code Points

Characters in JuniusX that do not have Unicode code points should be accessed via OpenType features whenever possible. MUFI/PUA code points should be used only in 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 A̷	U+F0008 d̅	U+F0010 m̅	U+F0018 v̅
U+F0001 a̷	U+F0009 E̷	U+F0011 o̷	U+F0019 x̄
U+F0002 A̢	U+F000A ę̸	U+F0012 ø	U+F001A x̅
U+F0003 c̄	U+F000B f	U+F0013 o̦	U+F001B ˀ
U+F0004 c̅	U+F000C i̅	U+F0014 o	U+F001C Ꝛ
U+F0005 ꝺ	U+F000D j	U+F0015 q	U+F001D ꝛ
U+F0006 ꝺ̷	U+F000E j̅	U+F0016 ꝫ	U+F001E A̸
U+F0007 dʼ	U+F000F lʼ	U+F0017 v̄	U+F001F a̸

Index