pua2ot: A Utility for Junicode

Peter S. Baker

January 25, 2026

1 The scripts

Modern best practice, when typesetting documents or creating web pages, is to avoid the use of characters from the Unicode Private Use Area (PUA) whenever possible, instead preferring standard Unicode characters styled with OpenType features.[1] The Medieval Unicode Font Initiative (MUFI), founded before OpenType was widely supported by software, defines more than 850 PUA characters, and Junicode includes all of them. But the font also offers more-accessible alternatives to all but a handful of these characters.

OpenType is a powerful technology, but it is undeniably more difficult to work with than MUFI's PUA characters and entities. To copy any character (or its corresponding entity) from the MUFI website and paste it into your text is a straightforward procedure. To do the same job with OpenType features is more complex, requiring users to locate the correct Unicode characters and features and to figure out how to make them work together in various software programs.

pua2ot automates this task. For each PUA character or MUFI entity in a text or HTML document, the script substitutes a **base** consisting of one or more standard Unicode characters and, if necessary, applies one or more **methods** (grounded in OpenType technology) to style this base. For many characters, more than one base is available, and more than one method.

The methods employed by pua2ot are these:

otag (OpenType tag). One or more OpenType features is turned on for a character or sequence of characters in much the same way that, say, italics can be applied to a run of text in a word processor: it is a modal method, which has the

[1] For an explanation, see the *Junicode Manual*, § 4.1

advantage of being legible and widely understood but can greatly swell the size of a text containing many PUA characters.

utag (Unicode tag). An alternative to the otag method is one (exclusive to Junicode) that uses characters from the Unicode "Tags" range (U+E0020–E007E) as a modeless style of markup in which a single base character is modified by two following tag characters. When used in a browser, tag characters are invisible and do not interfere with searches, but their invisibility can make them difficult to work with. pua2ot can use entities for tag characters to make them visible: the entities are resolved to tag characters when the text is displayed.

zwj (Zero-width joiner). The zwj character (U+200D) can be used to signal that two characters should be joined in a ligature: it provides an alternative to otag-style markup. For example, the PUA code point U+EBAC is a ligature of long **s** (ſ) and insular **v** (ꝩ): to obtain an equivalent one can apply the OpenType feature hlig (Historic Ligatures) to the base **ſꝩ**—or one can place a zwj between the two characters. This method is similar to the utag method in that it is modeless and employs a character that is typically invisible.

entity (Junicode entities). For technical reasons, combining marks encoded in the PUA should not be used, since software cannot position them correctly over their bases. One alternative is to use Junicode entities, which resemble HTML entities but begin with an underscore. For example, in the sequence a&_upmc;, the entity &_upmc; represents an upward-curving macron, which will be correctly positioned over the preceding a: **a̍**.

enla (Enlarge axis). For some forty-five of MUFI's "enlarged" letters, used to represent the *litterae notabiliores* of medieval manuscripts, one can use Junicode's Enlarge axis, which allows one to control the size of the letters precisely (on a scale from 0 to 100, where 0 produces a letter the size of other lowercase letters, and 100 produces a lowercase letter the size of a capital—the default value is 32) without using either MUFI's PUA characters or Junicode's sso6 "Enlarged minusucles" feature.

Alternate bases are grouped as follows (their names are followed by the identifiers to be used with the "Alternate bases" option, discussed below):

Insular (insular). By default, pua2ot replaces Unicode characters described as "insular" because these characters are often not searchable as their alphabetic

equivalents. However, since some projects have have good reasons to retain these characters, pua2ot's default behavior can be overridden (see "Options" below for details).

Small caps (smallcap). MUFI PUA characters described as "small capitals" are by default replaced by normal alphabetic characters with OpenType markup applied (typically pcap "Petite capitals"). But Unicode has characters labeled "small capital." These are not intended as variants of alphabetic characters (they are instead phonetic characters), and in some software they can't be searched by their apparent alphabetic equivalents. But if you prefer, the Unicode small caps can be used as a base.

Punctuation (punctuation). MUFI encodes a number of punctuation marks in the PUA. Often it is difficult to find a plausible Unicode base for these marks; instead, Junicode uses the period as a base for all of them. Where plausible bases are available, pua2ot uses them by default; but you can use the period as a base if you prefer.

Marks (mark). As with punctuation, so with marks. All of MUFI's combining marks can be handled in Junicode as variants of the macron (U+0304), but where better bases are available, they are used by default. If you prefer, you can use the macron for all PUA-encoded punctuation marks.

Currency (currency). All of MUFI's PUA-encoded currency symbols are treated by Junicode as variants of U+00A4, Unicode's generic currency sign, but where plausible Unicode bases are available, those are used instead. If you prefer, it is possible to use U+00A4 as the base for all PUA-encoded currency symbols.

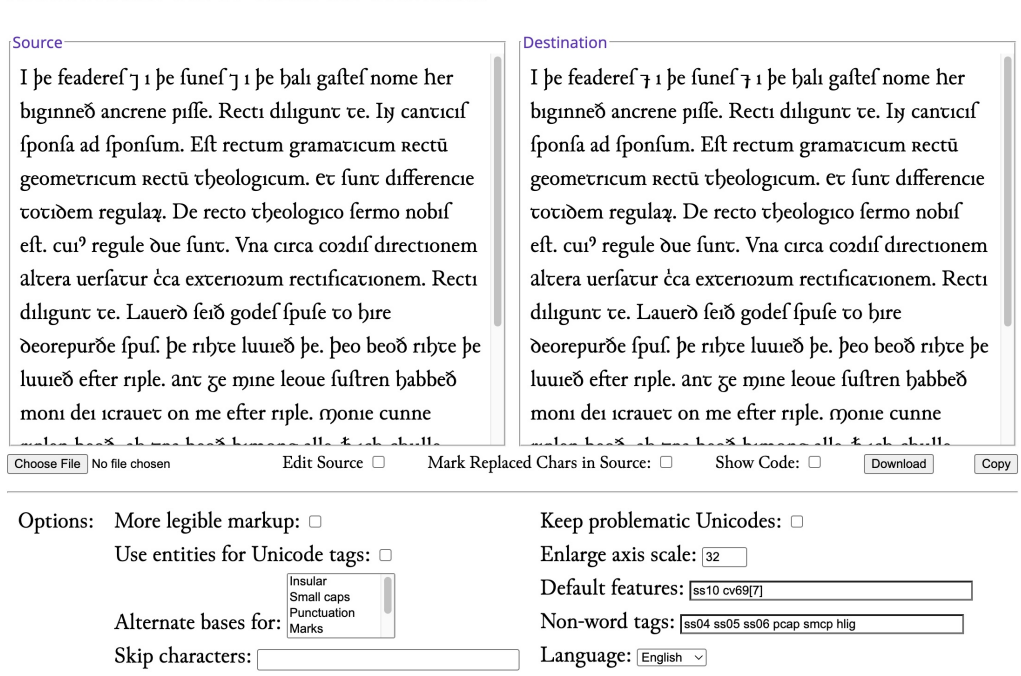
Miscellaneous alphabetic (alpha). For a few alphabetic characters (U+E7E4 ɼ, U+EEFD ı, U+EEFE ȷ, U+F0A7 ꝺ, U+F0A8 ꝺ, U+F127 ſ) more than one base can be used. The defaults here are chosen to promote accessibility, but these defaults can be overridden.

For both methods and bases, pua2ot consults a priority list: methods or base classes are listed in order of preference: if more than one method or base class is listed for a PUA character, the one that comes earliest in the priority list is preferred. These lists are discussed further in the "Options" section below.

Before you do anything else with pua2ot, copy the two font files JunicodeVF-Roman.woff2 and JunicodeVF-Italic.woff2 into the pua2ot/fonts directory in the unzipped Junicode file tree.

1.1 With Graphical User Interface

To use pua2ot with a GUI, find the file pua2ot-gui.html in the pua2ot/gui directory and double-click or drag it into an up-to-date web browser. The program will look like this:



The large text-box on the left (initially displaying an excerpt from the Middle English *Ancrene Wisse*) is the **Source Box**; it shows the text with PUA characters or entities. The text-box on the right is the **Destination Box**: it displays the result of the conversion. These two boxes should look the same: their sameness assures you that the conversion has gone well.

Immediately below the text boxes are options that affect their operation. To load a text or html file, click the "Choose File" button ("Browse" in some browsers) or drag the file into the Source Box. To edit the text, check the "Edit Source" box: you can type or paste in text when editing is enabled. To see which characters in the source are being replaced, check the "Mark Replaced Chars in Source" box. To view the code that underlies the text displayed in the Destination Box, check the "Show Code" box. To export the converted text, click the "Download" or the "Copy" button.

To skip the conversion of any element of an HTML file, add the class noconv to the element. You can add a with class noconv to skip an arbitrary stretch of text.

The other options on the page affect the conversion. These are explained below, in the "Options" section.

1.2 Embedded in an HTML file

The second version of the script converts the <body> element of any HTML page before it is displayed, with the result that characters that could not be searched will become searchable, and the accessibility of the page will be improved. If the HTML file contains static text, make sure the script file, pua2ot-embed-min.js, is in the same folder as your HTML file. Then include these lines at the very end of the <body>:

```
<script src="pua2ot-embed-min.js"></script>
<script class="noconv">
    // options.utagEntities = false;
    // options.utagLookup = UTAG_DICT;
    // options.nonWordTags = ["ss04", "ss05", "ss06", "pcap", "smcp", "hlig"];
    // options.keepUnicode = false;
    // options.methodPriority = "compact";
    // options.prefList = ["utag", "zwj", "otag", "entity"];
    // options.defaultTags = {"ss10": 1, "cv69": 7};
    // options.language = "en";
    // options.basePreferences = [];
    // options.enlargedScale = 32;
    // options.charSkip = [];
    convertAll();
</script>
```

If your web page loads text dynamically, it won't work simply to include the command convertAll() at the end of the HTML file, as it will likely be executed before the text is loaded. Instead, have the process that loads the text trigger an event when the loading is done and call convertAll() from inside the event handler.

In the html file, include noconv in the class list of any element whose content you don't want converted: the script will skip over such elements.

In the example above, the commented lines before the convertAll() command are options, explained in a later section.

1.3 Node module

The file pua2ot-common.js includes instructions for using it as a Node module, but if you are not making changes in the programming you can use node/pua2ot-node-min.js, which packages the pua2ot database and program code in a single minified file.

The file exports the object options (the options it supplies are listed in the next section) and two functions: pua2otNode.cleanup_string(s), which collapses all sequences of whitespace characters into single spaces, and pua2otNode.convert(text_buffer, repl_ent), which converts and returns the text text_buffer, first replacing any MUFI entities if repl_ent is true.

For an example of how the Node module can be used, consult and run the file test2.js. Note especially the require statement in line 3, the options object in line 13, and the invocation of convert() in line 15.

1.4 Reference page

In the src folder is a file charlist.html. Drag this into a web browser to view a list of all the transformations that pua2ot is capable of. Originally intended as a tool for proofing the database that powers pua2ot, the page can also be used to look up the methods for individual PUA codes. Type the hexadecimal code for any of MUFI's PUA codes into the "Filter" box. Alternatively, type any word or phrase (e.g. "combining," "small capital") to search for in the "description" column. To view the hexadecimal codes for the Unicode base for any conversion, click on a cell in the "base" column. To view the underlying code for any character, click on any of the last five columns of the row for the character you are interested in. You can copy any of these methods onto the clipboard and paste them into a document: just click the "Copy" button in the dialog the appears when you click on one of the last five cells in a row.

2 Options

In addition to the checkboxes, menus, and text-entry blanks in the web app, the JavaScript program has an options object whose properties can be manipulated to affect the program's output.

2.1 More legible markup

Program default: options.methodPriority="compact".
By default, pua2ot uses the **utag** method for most of its conversions of single characters and the **zwj** method for most ligatures: these two methods occur first in the "compact" preference list. Because the Unicode tags and zero-width joiner employed by these methods are zero-width and invisible, they can make it difficult to diagnose

problems in your code. If you set this option to "legible," pua2ot will prefer the **otag** and (where appropriate) **enla** methods, which insert HTML tags into the text so that you can see clearly what the code is doing. If you choose the "legible" option to generate a web page, the HTML file for the page will be noticeably larger than with the "compact" option: you should consider running the output of this script through a utility (e.g. the Javascript HTML Style Converter) that converts inline styles to a CSS stylesheet.

2.2 options.prefList

Program default: options.prefList = ["utag", "zwj", "enla", "otag", "entity"].
Not available in the GUI flavor. Use this option to install a custom priority list instead of the two predefined lists labeled "compact" and "legible." Methods that come earlier in the array are preferred over those that come later.

2.3 Use entities for Unicode tags

Program default: options.utagEntities=false.
If true, pua2ot uses entities to represent Unicode tags. (These are not HTML entities, but unique to Junicode.) This makes the code produced by this script more legible, but the entities make the page somewhat larger and can interfere with searching. The best use of this feature may be to help you spot problems with your Unicode tags and turn it off before publishing your text.

2.4 Keep problematic Unicodes

Program default: options.keepUnicode=false.
"Problematic Unicodes" are characters that are standard Unicode (not PUA) and look like variants of common characters, but that many applications do not recognize as variants. Examples include dotless i (ı, U+0131) and the "insular" letter-shapes (e.g. ꝺ, U+A77A). By default, this script replaces these characters with ones that can be searched as letters of the alphabet (e.g. ꝺ as d). To retain the "problematic Unicodes," likely simplifying your markup and making your file smaller, but making your text less accessible in some applications, check this box in the GUI version or set options.keepUnicode=true.

2.5 Alternate bases

Program example: options.basePreferences=["insular", "punctuation"].
As a large font with many OpenType features, Junicode often provides more than one way to achieve the outcome you want. This option affects the character classes mentioned above, pp. 2–3. Select one or more of the items in the GUI list (use the Shift or Control/Command keys to make multiple selections) or assign an array containing the identifiers for these classes to options.basePreferences: pua2ot will use alternate bases for the classes you select.

2.6 options.replaceMUFIEntities

Program default: options.replaceMUFIEntities=true.
If true, MUFI's entities (those that resolve to PUA code points) are replaced by those code points before any other processing takes place. Set this to false if the text you want to convert contains no entities and you want to save a little time processing a large text. Not available in the GUI, where entities are always converted.

2.7 Default features

Program default: options.defaultTags={"ss10": 1, "cv69": 7}.
GUI default: ss10 cv69[7].
You may wish some OpenType features to be on for the whole document. For example, ss10 "Entities and Tags" must be on in any document that uses Junicode entities or Unicode tags. To manage your text's OpenType features, this script needs to know about these default features; and when you download or copy your text, they will be recorded in an enclosing . Note the formats: In the GUI this is a space-separated list of four-letter OpenType tags; if you need to add an index, do so in square brackets immediately after the tag (the format used in the *Junicode Manual*). Elsewhere it is a comma-separated list of key-value pairs, where the keys (four-letter tags identifying OpenType features) are in double quotation marks, followed by a colon and a number (1 means "on," but some features (especially the cvnn "Character Variant" features) require a numerical index).

2.8 Non-word tags

Program default: options.nonWordTags=["sso4","sso5","sso6","pcap","smcp","hlig","enla"].
GUI default: sso4 sso5 sso6 pcap smcp hlig enla.

When "More legible markup" is checked, the script applies most tags to whole words, but a few to individual characters. This distinction is based on an educated guess as to which OpenType features are appropriate to apply to whole words. Sometimes this guess will be wrong. For example, the word *rectificationem* (about halfway through the sample *Ancrene Wisse* text) contains two different styles of **t**, but when cv40 (which transforms default **t** into insular **ꞇ**) is applied to the whole word, that difference is erased. If you add cv40 to this list, the feature cv40 will always be applied to single letters rather than whole words. In the GUI version of pua2ot, this is a space-separated list of four-letter tags (only tags used by this program are accepted here); elsewhere these tags must be enclosed in quotation marks and separated by commas, and the whole list must be enclosed in square brackets.

2.9 Language

Program default: options.language="en".
It is important to set the language properly, both here and in your document, since this script and some of Junicode's OpenType features behave differently depending on the language. If your language is not in the drop-down list (available only in the GUI), choose "Other" and type in the two-letter code for the language (you can look it up here.).

2.10 Enlarge axis scale

Program default: options.enlargedScale=32.
When options.methodPriority="legible", or when you use a custom priority list with "enla" near the top, pua2ot will invoke Junicode's Enlarge Axis for "enlarged minuscules." One advantage of this method is that you can precisely control their size on a scale from 0 (the letter is the same size as other lowercase letters) to 100 (the letter is the same size and weight as a capital). The default is 32, which makes it the same size as the enlarged minuscules produced by other means.

2.11 Skip character

Program example: options.charSkip=["0131", "A787", "A77A", "F0AE"].
GUI example: ı A787 A77A ð.
To cause pua2ot to skip over all instances of a character, enter it here. In the GUI, you can type or paste a space-separated list of characters and hexadecimal codes

in the blank. In programming, only hexadecimal codes are allowed; they must be enclosed in quotation marks and separated by commas, and the whole list must be enclosed in square brackets.