

Author	Bertrand ÉMERIT
Title	Proposal for Encoding Common Musical Notation in Unicode
Date	September 30, 2020

## Table of contents

1) Is the proposal compliant with Unicode aims and guiding principles? .....	2
a) Stability: .....	3
b) Convertibility: .....	3
c) Dynamic Composition: .....	3
d) Unification: .....	3
e) Semantics: .....	3
f) Plain text: .....	4
g) Characters, not glyphs: .....	5
h) Efficiency: .....	5
i) Logical order: .....	6
j) Universality: .....	6
2) Is the proposal useful? .....	8
3) Is this proposal feasible? .....	11
a) The musical word .....	11
b) The musigram description and the general algorithm .....	13
c) Musical script: creating a first measure .....	17
d) Lyrics .....	20
e) Directions, more about lyrics, more about tags .....	21
f) Staff groups and the position algorithm .....	27
4) Is the proposal reasonable? .....	38
a) Comments on existing characters .....	38
b) Code points in the 1D100 block .....	41
c) Combining beat characters .....	45
d) Combining staff characters .....	46
e) Combining pitch characters .....	47
f) Musical context control characters .....	48
g) Group control characters .....	51
h) Combining stem characters .....	52
i) Staff characters .....	52
j) Fingering characters .....	53
k) Character count .....	54
l) Collation .....	54
5) Annex A: Example encoding .....	55

This proposal is about being able to write musical scores with Unicode.

I assume that, when reviewing a proposal, you check for four key points:

- 1) Is the proposal compliant with Unicode aims and guiding principles?  
Well, the short answer is “Yes”, but I guess that, for the moment, you are thinking the exact opposite, so I will have to write a lot to prove that point.
- 2) Is the proposal useful?  
Section 2) will tell where the proposal is useful and why there is still a need distinct from MusicXML, SMuFL and other important assets.
- 3) Is the proposal feasible?  
In this section we will review in detail the script, to fully understand how it is possible to be compliant with Unicode design principles.  
Moreover, you will find attached a whole score (Nocturne Op 9 No 2 by Chopin, see annex A), both as a step-by-step (xlsx) and as a string (txt).
- 4) Is the proposal reasonable?  
We’re talking about 175 code points, so the short answer would be yes. We will use this section to describe all the proposed code points with their semantics.

### 1) Is the proposal compliant with Unicode aims and guiding principles?

As stated in the [Core Specification document](#), paragraph 2.2, the Unicode design principles are:

Principle	Statement
Universality	The Unicode Standard provides a single, universal repertoire.
Efficiency	Unicode text is simple to parse and process.
Characters, not glyphs	The Unicode Standard encodes characters, not glyphs.
Semantics	Characters have well-defined semantics.
Plain text	Unicode characters represent plain text.
Logical order	The default for memory representation is logical order.
Unification	The Unicode Standard unifies duplicate characters within scripts across languages.
Dynamic composition	Accented forms can be dynamically composed.
Stability	Characters, once assigned, cannot be reassigned and key properties are immutable.
Convertibility	Accurate convertibility is guaranteed between the Unicode Standard and other widely accepted standards.

We will have to pass through all these principles, but not in the above order. We will keep Universality as the last, but not least, one.

a) Stability:

Let us emphasize that the current proposal provides a full backward compatibility.

Any Unicode sequence will still be treated with the same meaning and the same rendering. In particular, a note in the 1D100 block will continue to behave as a spacing character.

b) Convertibility:

Widely accepted standards hold music representation as well as (when relevant) graphical representation. E.g. most standards will tell to draw a C4.

This proposal does not base on music as heard, but on sole written instructions. We will not draw a C4 but a note at a certain position.

Therefore, getting from the music representation to the current proposal will be straightforward, but going the other way around might need some reading ability, even if automated.

There will be a discussion about converting MusicXML (a de facto standard) to Unicode and conversely, in the part 2) about usefulness.

In any case, this convertibility is considered accurate, as the principle is about preserving the character identity, which is the case.

c) Dynamic Composition:

The current proposal makes use of this principle. It also does not limit the use of dynamic composition.

d) Unification:

All usable code points are not duplicated. In particular, the 1D100 block musical symbols will be reused (with the same semantics).

As for now, the code points used for music are:

- The range 1D000-1D0FF "Byzantine Musical Symbols"
- The range 1D100-1D1FF "Musical Symbols"
- The range 1D200-1D24F "Ancient Greek Musical Notation"
- 266D "Musical Flat Sign", 266E "Music Natural Sign", 266F "Musical Sharp Sign"

(for a grand total of 549 code points.)

The 1D100-1D1FF, 266D, 266E, 266F will be extensively used, as the musical scores which we are discussing are part of the Common Musical Notation (CMN).

e) Semantics:

All the previously-existing characters keep their semantics.

All the added code points will be given a precise semantics.

The additional code points similar to the existing ones will be given similar properties, for consistency reasons.

f) Plain text:

No room will be made for bold, italics and so forth, which are often seen in musical scores, but are not part of Unicode.

Furthermore, the fact that we have several ways to draw the musical components, for the same meaning, is style, not script. For instance:



We see the beams can be horizontal or diagonal (or more or less sloped). There are some conventions about this (nothing universal), and this certainly helps the reader and makes the score beautiful, but, like italics or bold, it does not semantically distinguish the characters, and will not be encoded in the script. The proposal contains “enough information to permit the text to be rendered legibly, and nothing more”.

It might be the font designer, or the text renderer designer, decide it wants to elaborate per default (same as *h* is nicely rendered by the font by putting the acute diacritic at just the right place, which is higher as in the *é*), but it is whatever not Unicode to tell about it.

Similarly, metadata (such as the author name) will not be part of the proposal.

To be precise, I will use Beethoven’s 7<sup>th</sup> symphony (available [here](#) thanks to Stanford’s CCARH centre) to distinguish what is the script and what is the layout, metadata or style, being clear that the script belongs to Unicode, but the rest to a higher-level protocol:

Symphony No. 7

Poco sostenuto. (♩ = 60) I

Beethoven: Symphony No. 7 in A major, Op. 92  
<http://www.musdata.org/beethoven/sym-7>  
 © 2006 Center for Computer Assisted Research in the Humanities (CCARH)

page 2  
21 Sep 2008

What is outside the blue rectangle, is metadata, not script. What is inside is the musical script, that can be described by the current proposal. However, there will be no italics for instrument names, and no bold for the “zu 2” direction in the Corni part. The dynamics will be kept somewhat with the same appearance, as the 1D18F and 1D191 code points will be reused.

We can see that the musical script key elements are in the 1D100 block but also other blocks, as Latin alphabet characters.

g) Characters, not glyphs:

The general architecture is kept identical: the text renderer will use both the Unicode characters and a font to produce a musical score. This proposal is only about characters, and not glyphs.

h) Efficiency:

Efficiency is what makes this proposal useful, and will be further discussed in 2). In 3) we will describe the algorithm for text rendering, to show that it is, indeed, simple.

i) Logical order:

Feasibility of a logical order (especially for many-staff scores) will seem dubious until the part 3), but the renderer will be able to render the musical script in the same way it renders other left-to-right scripts: left to right, then top to bottom.

The characters will be put in the logical order, which means that the renderer will write the character flow as they occur.

There will be similar issues as for other scripts, that previous characters in a line sometimes need to be redrawn (as for spacing in a justified line).

j) Universality:

This is probably the controversial point. Let me paste here how it is described in the General Structure chapter:

*“The Unicode Standard encodes a single, very large set of characters, encompassing all the characters needed for worldwide use. This single repertoire is intended to be universal in coverage, containing all the characters for textual representation in all modern writing systems, in most historic writing systems, and for symbols used in plain text.*

*The Unicode Standard is designed to meet the needs of diverse user communities within each language, serving business, educational, liturgical and scientific users, and covering the needs of both modern and historical texts.*

*Despite its aim of universality, the Unicode Standard considers the following to be outside its scope: writing systems for which insufficient information is available to enable reliable encoding of characters, writing systems that have not become standardized through use, and writing systems that are nontextual in nature.”*

I suppose no one will contradict that musical score is needed for worldwide use, that sufficient information is available to reliably encode characters and that it has become standardized through use. What will however not be accepted as self-evident, is that this writing system is textual.

There is no clear definition of text (even in the Unicode Glossary), but we may hazard to combine the text definition in the *Nouveau dictionnaire encyclopédique des sciences du langage* by Oswald Ducrot and Jean-Marie Schaeffer, and the Wikipedia definition in [https://en.wikipedia.org/wiki/Writing\\_system#Text,\\_writing,\\_reading\\_and\\_orthography](https://en.wikipedia.org/wiki/Writing_system#Text,_writing,_reading_and_orthography).

- Text is “A linguistic chain, spoken or written, forming a communication unit”
- Text refers to “an instance of written or spoken material with the latter having been transcribed in some way. The act of composing and recording a text may be referred to as writing, and the act of viewing and interpreting the text as reading.”

So, a textual writing system shall have an acoustic counterpart, with a two-way correspondence by the means of a set of symbols and rules permitting to convert acoustic information to written information, and a set of rules permitting to get back from the written information to a similar acoustic production; it shall also target to convey semantics.

Such a definition of course holds for natural languages, but it also holds for music (acoustic part) and musical score (written part): one can convert heard music (but not any sound) to a written score, and one can read the score and produce the music back.

Hopefully this observation is conclusive about the first design principle of Unicode.

We have passed through all ten principles. I am cognizant that section 3) is necessary to be final about the Efficiency, the Logical order, and also somewhat the section 4) about Semantics, but suffices for this part to tell that we are compliant. Part 3) will tell *how* this is possible to use Unicode for musical scores, while being compliant.

## 2) Is the proposal useful?

The second question is hopefully much easier to answer.

The [Proposal for Encoding Western Music Symbols in ISO/IEC 10646](#) tries to estimate the community of the users. The number of users for just the theoretical books has been considered sufficient to create the 1D100 block.

As for musical score users, I know of no global statistics; we can dare a ballpark by using the [Enquête des pratiques culturelles](#) from the French Ministry of Culture, which states that, for the past 35 years, 10 to 15% of French people have practised music, which makes about 7 million people. Other sources hint that the ratio is similar in many other countries... it might not be worth knowing the exact number of users anyway, as it is in the order of magnitude of tens (maybe hundreds) of millions.

Now, we are not as if no score could be written (with the help of computers) nowadays. We have to tell in which situations it will be useful to have a Unicode representation of the score, and also when it will not.

Let us start with the current pitfall of musical score writing, and see where this proposal helps. Then let us discuss how this proposal interacts with MusicXML and SMuFL, for better clarity.

Scorewriting applications help writing music. Which really means, they permit to create either musical documents (MusicXML) or score representations (PDFs).

However, a huge number of documents (in a broad sense) have both natural language and scores; one range of such documents, “theoretical works, pedagogical texts” etc., is the one which triggered the creation of the 1D100 Musical Symbols block.

Since the development of the world-wide web, this use has skyrocketed: A quick search on easy words like “musical staff”, “musical score” and the like, on any web search engine, will return over one billion results, most of which indeed include a staff among other languages.

There is currently only one (common) way to create such documents: inserting images (as in this very document). A huge use case is then: 1) create a score or a musical example in a scorewriter 2) screen-copy the score (or the created pdf) 3) include as a picture in the intended document.

But images can only be read. They cannot be edited as musical scores, they cannot be searched, copy-pasted back in a scorewriter, etc.

In these pictures, the interpretation of the music does not count (they are no MIDI files), solely the visual representation. The fine visual rendering also seldom counts (images are blurred, hardly scaled, etc.), as if they do, the final PDF is usually stored; semantics is what counts in these documents, which is, plain text—which does not mean that higher-level styling will not be useful—.



There is for this reason a need that is considered important, to be able to let scores be text in textual documents and rendered by the text renderers of common platforms. The need is important, it is also distinct from being able to issue whole-musical documents, which is usually done by PDF or directly within diverse score readers.

As for the integration with MusicXML and SMuFL.

First, there should be no collision between this proposal and SMuFL; on the contrary, the over 2440 characters defined by SMuFL in the Private Use Area (U+E000 and following) will fit transparently as base characters.

Second, the integration with MusicXML should enrich it.

Users should continue to write scores in a scorewriter (I personally encoded this proposal using a spreadsheet, and would not expect anyone to endure it), and they should certainly, when possible, continue to save them as MusicXML, which includes both the visual rendering and the playback information. Even for visual rendering, MusicXML includes the finest details (which means, style), while this proposal stays at plain-text level.

The intended use is therefore: 1) create a score in a scorewriter 2) copy the score as text 3) paste in the intended document.

Any MusicXML document can be rendered as Unicode string with this proposal. There will be a loss of playback, hopefully the style can be conveyed in a style sheet if needed, but let us assume that the style is lost as well. The result will be the minimal (semantic) string of music, and will be easily stored in any document, whatever the type. This document will display the score, without the need for the word processor or the web browser to understand MusicXML directly. (This string will also be included in a way that is independent to the device (i.e. somewhat responsive), mainly because there are no line breaks.) The quality of the rendered score will depend on the renderer of course, and it looks unavoidable that the many complex rules of fine score engraving are unattainable by any software but a specialized one, but all representations will be semantically equivalent.

The opposite —Unicode to MusicXML— would require some intelligence (similar as an OCR). For instance, voices (musical parts sharing a single staff) are not encoded per se in a graphical representation. The opus title and author would have to be recognized as such from the text around the score. The instruments would have to be mapped with technical codes. MusicXML also encodes the playback, which means some interpretation of the music (actual note duration for instance, even for fermatas), that are not included in the graphical score. However, between a conversion from a picture and a conversion from a plain-text score, the latter will be both easier and more precise.

To be complete, some protocols do, indeed, target document parts, for instance the Score Extension in MediaWiki. (As a matter of fact, this extension uses one of the LilyPond or the ABC notations.) But they push the scores as pictures, and there is, for this reason, the same need for a Unicode musical score ability.

As a conclusion, this proposal should help interchange and process musical scores or musical score excerpts, especially in composite documents, which amount to tens of millions of documents for tens of millions of people.

### 3) Is this proposal feasible?

Here is the section where we explain which characters we need, with which semantics, to be able to write scores.

A good part of this section will take as an example an old but very well-known French song: "Au clair de la lune". The first verse reads:



(The scores for this song are my own work.)

#### a) The musical word

First, as stated in the Stability principle, all existing characters keep their meaning: we can still write: ♪♪♪♪♪♪♪♪♪♪♪ – and we can also write the 5-line staff character: ≡.

This does not (and will not) make a score. This is because, in the 1D100-1D1FF "Musical Symbols" range (as well as in the 1D000-1D0FF "Byzantine Musical Symbols" range, in the 1D200-1D24F "Ancient Greek Musical Notation" range and in the 3 characters 266D "Musical Flat Sign", 266E "Music Natural Sign", 266F "Musical Sharp Sign"), musical notation is considered a set of independent symbols, and not a script.

As stated in the Unicode 3.1 release notes, chapter 12.11 "Musical Symbols", the current aim of these symbols is to be used "in contexts such as theoretical works, pedagogical texts, terminological dictionaries, bibliographic databases, thematic catalogues, and databases of musical data".

The intended use is therefore writing sentences like: "a quarter note is notated with a black note head and a straight beam with no flag: ♪". They have not been intended to create scores.

Continuing, "Because of the complexities of layout and of pitch representation in general, the encoding of musical pitch is intentionally outside the scope of the Unicode Standard. The Musical Symbol block provides a common set of elements for interchange and processing. Encoding of pitch, and layout of resulting musical structure, involves not only specifications for the vertical relationship between multiple notes simultaneously, but in multiple staves, between instrumental parts, and so forth. These musical features are expected to be handled entirely in higher-level protocols making use of the proposed graphical elements. Lack of pitch encoding is not a shortcoming, but is a necessary feature of the encoding."

Well, so was it in 2001.

Indeed, a number of formats have been designed to cope with pitch representation, including NIFF, LilyPond and ABC. All these formats are textual instructions for a higher-level processor, which issues a picture representing a musical score.

Then in 2004 came MusicXML, which aims to let applications share sheet music. Again, it does not describe a score, it provides document metadata, playback instructions (like MIDI instructions) as well as sheet music drawing instructions.

Among many breakthroughs of this format, one is key for our purpose: the basic “word” of music is not a note, but a measure. By word I do not mean a part of a musical phrase; I rather speak about the smallest significant element when drawing music. Complexity (the complexity that was considered to be handled by higher-level protocols) was due to thinking that the key element was a note, whereas it is a measure (and indeed, for instance, try to draw a whole rest alone and you will have to rely on higher-level protocols to centre it at the middle of a measure, if your score is a multi-part one).

Instead, MusicXML (following Humdrum representation) splits music in a series of measures: either the score is a list of parts, each part being a list of measures, or it is a list of measures, each measure being a list of parts. The notes and rests and directions are contained within a measure.

The first way, called the “partwise” representation (the score is a list of parts, each part being composed of measures) describes music “staff by staff”, adding one below the other. The second way, called the “timewise” representation, describes music measure by measure, adding one next to the other. In this “timewise” representation, we describe the first measure (the part for instrument 1 in this first measure is blah blah, the part for instrument 2 in this first measure is blah blah, etc.), then get to the next measure.

Both representations are equivalent as proven by MusicXML, which can transform partwise to timewise and timewise to partwise.

Now, they might be equivalent for an XML representation. But, as for our purpose, they are not. The so-called “partwise” representation makes it that distinct threads of characters must be read and vertically aligned (how?) so as to write a score; this is exactly what was told about in the excerpt “Encoding of pitch [...] involves not only specifications for the vertical relationship between multiple notes simultaneously, but in multiple staves, between instrumental parts, and so forth”.

However, as for the second “timewise” representation, the algorithm is like writing a word with a string of characters, then getting to the next word on its right. And if we get to the end of a line, we cut at the end of the word (a measure) and start a new line: the score is written left-to-right and top-to-bottom. Only, the unit of writing is the measure with all its parts. This unit is the musical word – and we see through the line breaks, that the barlines are kind of spaces that delimit these words. (We will get more precise on Unicode Properties and line break opportunities, further.)

Just to be clear: the unit for rendering the musical script is the musical word, which is one-measure-wide, and many-staff-tall (all the staves of the staff group, one below the other, are part of one single word). A musical score is typically larger than tall for most scores, which have one or two staves, but in case of many-part music, one such word can get very tall.

We see here that the word in question is not unlike Chinese compound characters or Hangul syllables. You have notes, lyrics, rests, etc. getting to a specific position to make the whole have a meaning for the reader. That position we need to tell the text renderer. Then the renderer is able to create a “musigram”, which is to say, a measure. (Note that the renderer needs the help of a variable font, for some glyphs such as ties and beams.)

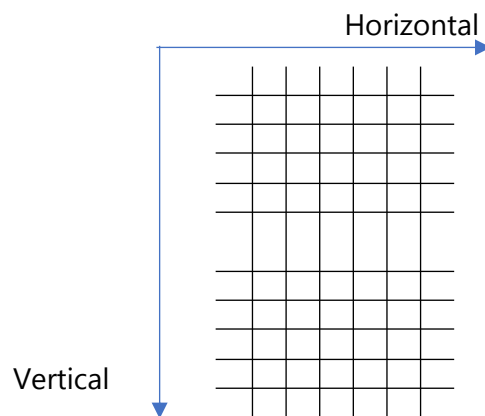
In short, the proposal is to enrich Unicode to describe musical compound words, which are score measures. Base components are already at hand in particular in the 1D100 block.

I will stress one trait: we do not wish to ask the text renderer to understand music and to convert musical instructions into graphical elements. We will tell the graphical elements directly.

So, we will not tell “I have a flat B”, because B is a musical semantic, but “I have a U+1D15F (♯) in the middle of the staff”, because this is a score representation (and that could be a B in a treble clef or a C in a bass clef, the text renderer does not need to know). The script will tell nothing about the fact that this character makes four beats, because that is irrelevant to a graphical representation of music. The characters remain spacing characters, and if the renderer wants to add more space after half notes than after quarter notes, that is not Unicode to dictate. In other words: the script tells the score, not the music.

b) The musigram description and the general algorithm

So how is a “musigram” built?



Basically, it is a grid of anchor points for base components.

So, say, a note, will be said to be horizontally at place 3 and vertically at place 2.

Is this new to Unicode? No, that principle is the one of combining characters: ‘ê’ is a ‘e’ that is put a ‘^’ above; the U+0302 tells to put ‘^’ at the position *above*, whereas U+032D tells to put it at the position *below* (̣). Only here there are more diacritical marks. Another characteristic

is that, whereas for diacritics the anchors are typically handled by fonts, here, in this compound character, the anchors are handled by the text renderer instead. However, this is transparent to Unicode.

Now, how many horizontal and vertical positions do we need? In theory, an infinity. In practice, infinitely less.

Horizontally: that is probably the most difficult. Let us start by that.

A quick review of many scores tells us that you have regularly 4 times 4 semiquavers, but 16 is not enough.

Take the first three staves of the first measure of [Festo Visitationis Mariae](#) by Bach (well-known for the "Jesus Bleibet meine Freude" part):



(this picture is an excerpt issued from imslp – public domain music sharing site.)

The measure will have one horizontal position (let us call them beats, even if we speak about graphical alignment) for the clefs, one for the key signature (only one sharp), one for the time signature... and here we see that Bach chose different signatures, two distinct triple meters ( $\frac{9}{8}$  and  $\frac{3}{4}$ ), which makes that the second staff's 16<sup>th</sup> (second and fourth notes) are not aligned with any of the first staff's quavers:



It may be easier to count 36 beats (least common multiple of  $9 \times 8^{\text{ths}}$  and  $3 \times 4 \times 16^{\text{ths}}$ ) per measure here.

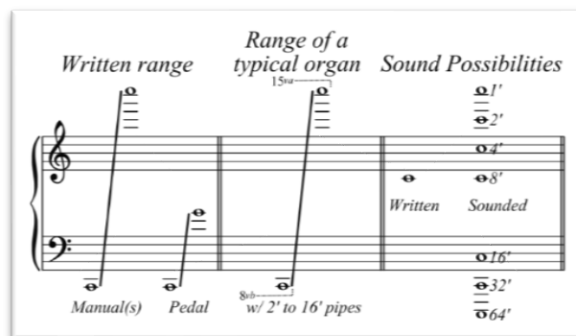
This extreme case shows us that 40 beats (36 + clef + signatures + 1 for the barline) should not be unseen, and, even if an invisible word separation will help with a few cases (we will see one

in Chopin in annex A), it looks like 48 beats will do. We will also need 2 additional positions, to denote the start and the end of the measure.

Vertically:

You first have to count staves. A full score holds 19 staves, this looks like an absolute maximum. So, let's make it 24. There should also be some special positions for directions and other elements, but we will not elaborate on this for the moment, and come back to the topic when developing our "Au clair de la lune" example.

Then come positions in a given staff (pitch), which is, either a place on a line, or between lines. Staves can have one to six lines, plus some optional ledger lines. In theory any number of ledger lines can be added, but the musical script is about being able to write scores that will be *read*, and then there are not so many positions: after "a few" ledger lines the score becomes unreadable, whence the 8va (or 15ma, or 8vb or 15mb) notation. How many is "a few"? Well, instrument ranges sum it up, and organ is excellent to show the extent (from Wikipedia [https://en.wikipedia.org/wiki/File:Organ\\_Range.svg](https://en.wikipedia.org/wiki/File:Organ_Range.svg)):



Which makes 10 ledger lines = 15 lines in the whole, and 15 + 1 spaces (1 below the lowest line + 1 above each other lines), so a total of 5 standard lines + 10 ledger lines + (15 + 1) spaces = 31 positions at most. We will see that the base position is the space above the lowest line (especially for clefs), but it is easier to place an element relative to the middle of a staff, whatever its number of lines. Then we can tell that graphically an element can be positioned 16 steps lower to 16 steps higher than the staff center. Let us call these positions "itches", even if they have to do with vertical placing and not actual pitch.

As a conclusion: to draw a "musigram", the general algorithm is simple. We need to:

- 1) Place all elements of a given measure on a 50-beat × (24-staff × 33-pitch) anchor grid
- 2) Kern, i.e. determine the width and height of each inter-anchor row or column, depending on the size of the attached elements (which incidentally trims all unused anchor lines)
- 3) Ask a (hopefully) variable font to give glyphs to all the elements.

To be plain, again, a redrawing of former musigrams is to be expected, because further measures could have more vertical space needs when adding directions, dynamics and the like – text renderers are used to this (justified text, combining characters, etc.).

Actual musical applications will of course add a step 4), which is a higher-level protocol, to make the result graphically optimal: space the notes relative to their duration, maybe trim unused staves, attach the ties finely so as they aim the centre of the noteheads, etc. But that is semantically equal, so it is not relevant to the script itself.

Let us come back to our Beethoven example, and, just because I like it a lot, go to the part 2:

The image shows a page of a musical score for Beethoven's Symphony No. 5, Part 2, measures 12-24. The tempo is marked 'Allegretto. (♩ = 1/6)'. The score is arranged in a standard orchestral format with staves for woodwinds, brasses, timpani, and strings. The woodwinds and brasses play chords with dynamics like *f* and *pp*. The strings play a rhythmic pattern with dynamics like *p* and *pp*. The page number '12' is visible at the top left of the string section.

This page is made out of 24 “musigrams” (measures). There is no carriage return after the 11<sup>th</sup> one.

The first musigram holds the instrument names and the “Allegretto” direction, as well as the staff groupings (Woodwinds, Brasses, Strings, and within strings the Violino and Violoncello braces), the clefs and signatures. It also has 8 notes, 8 rests, 4 fortes, 4 decrescendos. All these are probably encoded top to bottom for convenience, but this is not mandatory, as it suffices



to tell where they are in the musigram (staff and pitch). It is however necessary that they be encoded before to put a measure-breaking character. Then the second measure is encoded (8 notes and 8 rests, and we will see why there is nothing about decrescendos in the second measure later). Then the third, etc.

Side notes:

- You see that the renderer has suppressed the Wind staves in measures 12-24, composed of nothing, but if they had been added, the music would have been the same: semantically equivalent.
- The renderer has had to redraw the first two musigrams when at measure 3 the viola and violoncello made use of dynamics, resulting in more inter-staff space than necessary until then.

Now we know what the architecture of our writing system is, we will use it to write “Au clair de la lune”. Let us just make a liminal comment.

In many scripts, Unicode proposes both decomposed forms and precomposed forms. However, with musigrams, the complete enumeration is impossible: it would be of the order of magnitude of the number of base elements (notes and rests at the very least) power  $50 \times 24 \times 33 = 39600$ . We will have to describe the word composition and let the text renderer make the word compound glyph. This will require:

- base elements (the 1D100 block mainly, but any Unicode characters is legitimate)
- combining characters (horizontal “beats”, vertical “staves” and “itches”)
- We will also need some few control characters, some of which already being in the 1D100 block, as U+1D173 MUSICAL SYMBOL BEGIN BEAM.

c) Musical script: creating a first measure

So. Let us start encoding “Au clair de la lune”:



Let us start by the first measure. We already know that “ $\text{C}\text{J}\text{J}\text{J}\text{J}$ ” will draw... well, “ $\text{C}\text{J}\text{J}\text{J}\text{J}$ ”. The clef and the common time signature (C) are correctly placed, but the four quarters should not be at this position. To put the notes at the right place, we need to tell how far they are from the middle line, counting both spaces and lines. The first three are at position -6 (one space (the A4 one) + one line (the G4 one) + one space + one line + one ledger space + one ledger line). The last one is at position -5.

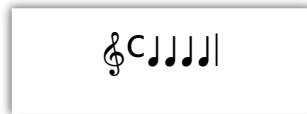
We already have all the constituents, apart from 3: two position combining characters and something to tell that we have finished our measure (our word).

To make things concrete, I will base the new characters on the 1D200 block, starting from 1D250 (immediately following the Ancient Greek Musical Notation 1D200-1D24F, that is), with the range 1D299 to 1D2BF for pitch combining characters, from -16 to -1 then 0 (1D2AF) then +1 to +16. Thus, we need the 1D2A9 (P-6) and 1D2AA (P-5) code points.

As for finishing our word, the technical discussion will occur later, but we need to reuse characters with line break properties ZW or SP, because no script may alter their behaviour. We will use the zero-width space 200B (a standard space could do, but would add some space).

The first measure would therefore be encoded as: 1D11E (♩) 1D134 (C) 1D15F (J) 1D2A9 (P-6) 1D15F (J) 1D2A9 (P-6) 1D15F (J) 1D2A9 (P-6) 1D15F (J) 1D2AA (P-5) 1D100 (l) 200B (ZWSP).

Which gives the following result:



Well, it obviously does not work. We have no overlying staff, and, worse, noteheads are still at F4 level (write  $\text{≡}\text{♩}\text{≡}$  and you see that clearly). Why that? Well, telling “6 positions below the centre of the staff”, when you have no staff, means nothing.

We definitely miss a staff. Two choices:

- Either we tell that there is a 5-line staff, spanning from the beginning beat to the end beat – we will discuss the spanning later
- Or we tell the text renderer we are in a musical context.

Contexts are not what we expect when defining a script, but the musical script is not the only one that needs some context. Contexts are not unknown to Unicode. One of the most known is the BIDI algorithm, that makes use of control characters. We are in the same situation: the musical script has a rare feature, that when splitting a phrase after a word (a measure), which is not foreseeable at encoding, the line should start with the clef and the (here absent) key signature, and repeat all staves with possible brackets, before to push the new measure components. So, the text renderer needs some context, and the staff being used is part of it.

We need to define rules as to when the musical context starts, and when the musical context ends. Given design principles, we need to make them unambiguous and efficient:

- The musical context starts with one control character specifically marked to be the beginning of a musical context. (In this proposal, we define 4 of them.)
- The musical context ends when the measure is marked as being the last one (explicit ending)
- If a measure counts more than 48 beats, the musical context ends at its beginning (implicit ending). 1D100 block elements are not treated differently from any other spacing character in Unicode, so if this sentence was put in a musical context, the renderer would understand after the string “1D100 block elements are not treated differently” (48 characters) that the musical context was over before “1D100”.

We will therefore set 8 code points for musical context, from 1D2C0 to 1D2C7, starting with 1D2C0 BEGIN STAFF and 1D2C1 END STAFF. (Other ones are for handling staff groups, lyrics and chords.)

If we are to keep characters in logical order (unlike Lilypond or MusicXML which define documents and not document parts) —and we do want to keep the characters in a logical order as we want the text renderer to draw the musigrams one by one with little if any correction, then we need to use the BEGIN and END characters one next to the other —the exceptions being the 1D173 and 1D174 BEGIN and END BEAM, and we will add similar ones for appoggiaturas. By “one next to the other”, I mean that the BEGIN character needs to be combined with the relevant horizontal and vertical combining characters, then followed by what begins, then followed by the END character combined with the relevant horizontal and vertical combining characters.

We will not write: “BEGIN-staff some notes END-staff”, but “BEGIN staff END some notes”. In general, we will not write: BEGIN-something blah-at-1 blah-at-2 END, but BEGIN-at-1 something END-at-2 blah-at-1 blah-at-2. This way, before blah-at-1 is reached, the renderer knows what to do. It does not have to wait until the END character is reached, to start its work. For instance, a slur can be drawn independently of slurred notes, before (or after) the notes are described. This is relevant for a graphical representation (we add a slur to the score, we do not slur notes), and efficient as for the text rendering.

In the present case, the BEGIN character is at the right place (at the beginning of the first measure), so no combining character is needed. The END character however needs to be placed at the end of the score, which is, in this case, at the end of the musical context, which means at the measure named “end”. How do we name a measure? We need a set of characters that represent standard characters (digits and letters), but that would not appear. I propose to make use of the TAG block, E0001 to E007F. We will reserve 2 tags composed of only one character: E007F to mean the last measure (which ends the context) and E002B to mean the next measure (especially needed in slurs).

Now we can correct our encoding as: 1D2C0 (BEGIN STAFF) 1D11A (≡) 1D11E (♩) 1D2C1 (END STAFF) E007F (CANCEL TAG) 1D134 (C) 1D15F (♯) 1D2A9 (P-6) 1D15F (♯) 1D2A9 (P-6) 1D15F (♯) 1D2A9 (P-6) 1D15F (♯) 1D2AA (P-5) 1D100 (l) 200B (ZWSP) E007F (CANCEL TAG). Which gives the following result (give or take some space between the notes):



Incidentally, if the text renderer or the font is not compatible with musical scores, it will be displayed as garbled text. Here is the whole score until the last whole C4 (+ final barline); as your renderer is not (yet) compatible with this proposal, it will be garbled. However, note that you can copy and paste a part, search for the treble clef, and put the score in a middle of a text line: ≡♩C♯♯♯♯lZWSP||



And this gives:



OK, that was our first goal. Let us now look at the rest 𐀀.

e) Directions, more about lyrics, more about tags

Now, what if we want the full lyrics, say, the first two verses?

If you do not know "Au clair de la lune", the song is quite simple, each verse is of the form A-A-B-A, and there is no chorus. In details, the song is twice the above phrase A ("Au clair de la lune, mon ami Pierrot", repeated with lyrics "Prête-moi ta plume pour écrire un mot"), then an intermediate phrase B ("Ma chandelle est morte, je n'ai plus de feu" with the tune:



), and then one last time the phrase A ("Ouvre-moi ta porte pour l'amour de Dieu").

We could use a repetition bar and a Dal capo al fine scheme. We will indeed do so when adding an instrumental part. But for now, we will keep it simple and develop the melody with no repetition. If we used repetitions, the lyrics would be unreadable (imagine 6 lyrics lines for the first, second and *fourth* rhymes of the two verses below the A phrase, and 2 lyrics lines for the third rhymes of the two verses below the B phrase!)

So, what we will write is:

A complex musical score for the song "Au clair de la lune". It features a tempo marking of  $\text{♩} = 80$  and a common time signature (C). The score is divided into three systems of musical notation, each with lyrics underneath. The first system (measures 1-4) contains two verses: "1. Au clair de la lu - ne, mon a - mi Pier - rot," and "2. Au clair de la lu - ne, Pier - rot ré - pon - dit :". The second system (measures 5-8) contains the lyrics for the intermediate phrase B: "Prê - te - moi ta plu - me pour é - cri - re un mot. Ma chan - del - le est mor - te, 'Je n'ai pas de plu - me, je suis dans mon lit. Va chez la voi - si - ne,". The third system (measures 9-12) contains the lyrics for the final phrase A: "je n'ai plus de feu. Ou - vre - moi ta por - te pour l'a - mour de Dieu ! je crois qu'elle y est, Car dans sa cui - si - ne on bat le bri - quet."

This more complex score adds several complications:

- i) A tempo direction ("♩ = 80")
- ii) Two verses

- iii) Verse numbers before the first word ("1." and "2.")
- iv) Syllables that span over two words (second staff, first verse: "re un" and "le est". We also need to take care of the "dit :" on the first staff second verse, and of the "Dieu !" on the third staff, first verse, as in French you need a space before colons and exclamation points.)
- v) Measure numbers at the start of each lines ("5" and "11"... but had the page been narrower or wider, that would be other measures)

i. A tempo direction ("♩ = 80")

The tempo direction is simply handled. It is the text "♩ = 80" put above the staff, at the beginning of the first measure. Note that conventions vary as for the precise horizontal place of the tempo directions; furthermore, it is usually in boldface... But both these aspects have no place in the encoding as they are semantically equivalent, so what we need is to tell that the text is above the staff, at the beginning of the first measure. The 1D2D6 is the BEGIN DIRECTION code point (1D2D7 for the END DIRECTION), the 1D29A means "above the staff", and is a combining pitch character as the ones we have already used.

(Note: there is therefore no distinction between staff group direction, staff direction, note direction etc. All these are simply directions, placed accordingly.)

So, before the <sup>♩</sup> we will add: 1D2D6 (BEGIN DIRECTION) 1D250 (BEAT START OF MEASURE) 1D29A (P+STAFF ABOVE) 1D15F (♩) = 80 1D2D7 (END DIRECTION).

ii. Two verses

We will of course have to push two BEGIN LYRICS / END LYRICS groups in the musical context. As for distinguishing between them, the first LYRICS calling element for one given measure element (note, usually) will use the first LYRICS group, the second will use the second LYRICS group, etc.

That is, the lyrics context will be:

1D2C4 (BEGIN LYRICS)  
 Au clair de la lune, mon ami Pierrot,  
 Prête-moi ta plume pour écrire un mot.  
 Ma chandelle est morte, je n'ai plus de feu.  
 Ouvre-moi ta porte, pour l'amour de Dieu !  
 1D2C5 (END LYRICS)  
 1D2C4 (BEGIN LYRICS)  
 Au clair de la lune, Pierrot répondit :  
 "Je n'ai pas de plume, je suis dans mon lit.  
 Va chez la voisine, je crois qu'elle y est,  
 Car dans sa cuisine on bat le briquet".  
 1D2C5 (END LYRICS)

And, where in the previous version we had one 1D2C8 LYRICS SYLLABLE after each note (and some BEGIN/END SYLLABLES), in this version we will have two per note: 1D15F (♩) 1D2A9 (P-6)

1D2C8 (LYRICS SYLLABLE) *1D2C8 (LYRICS SYLLABLE)* 1D15F (♪) 1D2A9 (P-6) 1D2C8 (LYRICS SYLLABLE) *1D2C8 (LYRICS SYLLABLE)* etc.

iii. Verse numbers before the first word

The text renderer keeps track of the current index in the lyrics. If the BEGIN SYLLABLE skips some (non-whitespace) characters, then they are added to the syllable, preferably aligned to the right at the left of the beat.

To make things clear, the lyrics should be:

1D2C4 (BEGIN LYRICS)

1. Au clair de la lune, mon ami Pierrot,  
Prête-moi ta plume pour écrire un mot.  
Ma chandelle est morte, je n'ai plus de feu.  
Ouvre-moi ta porte, pour l'amour de Dieu !

1D2C5 (END LYRICS)

1D2C4 (BEGIN LYRICS)

2. Au clair de la lune, Pierrot répondit :  
"Je n'ai pas de plume, je suis dans mon lit.  
Va chez la voisine, je crois qu'elle y est,  
Car dans sa cuisine on bat le briquet".

1D2C5 (END LYRICS)

and the first note should be: 1D15F (♪) 1D2A9 (P-6) *1D2CC (LYRICS BEGIN SYLLABLE) Au 1D2CD (LYRICS END SYLLABLE) 1D2CC (LYRICS BEGIN SYLLABLE) Au 1D2CD (LYRICS END SYLLABLE)*.

This tells the renderer to put "Au", twice, below the first note (C4), but as the lyrics start respectively with "1." and "2." which are left over, these will be put at the left of "Au", with a right alignment:

1.	Au
2.	Au

(Which is a similar alignment rule as appoggiaturas and accidents, by the way.)

iv. Syllables that span over two words

As hinted before, to handle the "re un" and "le est", it suffices to use the LYRICS BEGIN/END SYLLABLE. Only, as space may break lines and we do not want that, we will include a no-break space (U+00A0) in the LYRICS BEGIN/END SYLLABLE. All whitespaces (spaces and line breaks – all characters with property WSpace="Y") are equivalently matched by any whitespace in the LYRICS BEGIN/END SYLLABLE.

The 7<sup>th</sup> measure ("pour écrire un" / "je suis dans mon") therefore reads:

1D15F (♪) 1D2A9 (P-6) 1D2C8 (LYRICS SYLLABLE) 1D2C8 (LYRICS SYLLABLE)

1D15F (J) 1D2AB (P-4) 1D2CC (LYRICS BEGIN SYLLABLE) é 1D2CD (LYRICS END SYLLABLE) 1D2C8 (LYRICS SYLLABLE)

1D15F (J) 1D2AA (P-5) 1D2CC (LYRICS BEGIN SYLLABLE) cri 1D2CD (LYRICS END SYLLABLE) 1D2C8 (LYRICS SYLLABLE)

1D15F (J) 1D2AA (P-5) *1D2CC (LYRICS BEGIN SYLLABLE) re un 1D2CD (LYRICS END SYLLABLE)* 1D2C8 (LYRICS SYLLABLE)

1D100 (SINGLE BARLINE)

The “dit :” and “Dieu !” could be handled similarly. Though, in correct French typography, the space between “dit” and “:” should already be a no-break one, and as the “dit” and the “:” no longer break, they will stay together with the usual LYRICS SYLLABLE. The same holds for “Dieu !”, even if typographers will object that before an exclamation point (in French, still), a narrow no-break space should be used instead, which is, U+202F and not U+00A0. Of course as long as the space is a no-break one (narrow or not), it all comes to the same as for the fact that the exclamation point will remain with the previous word.

v. Measure numbers

The display of measure numbers is well-spread in the musical scores, and is usually done at line breaks. It is sometimes put every few measures instead, as an alternative. It is essentially a direction placed above the staff at the measure start: if the writer wants to add a number every three measures, then she just has to add a direction (like the “♩ = 80” one) at measures 1, 4, 7, 10, 13 and 16.

However, line breaks are not anticipable at encoding time. But we have seen that the measures can be tagged, so what we have to do is to call this tag with a 1D2CF MEASURE NAME, which works the same as the LYRICS SYLLABLE, but decodes the measure tag (U+E0040 → U+0040 etc.).

By putting this character in the BEGIN STAFF context, we will make sure that the measure name be added at every line break:

instead of 1D2C0 (BEGIN STAFF) 1D11A (≡) 1D11E (♩) 1D2C1 (END STAFF) E007F (CANCEL TAG),

we will tell: 1D2C0 (BEGIN STAFF) 1D11A (≡) *1D2CF (MEASURE NAME) 1D250 (COMBINING BEAT START OF MEASURE)* 1D11E (♩) 1D2C1 (END STAFF) E007F (CANCEL TAG)

(1D2CF defaults to the P+STAFF ABOVE pitch.)

We will see there is another way to achieve the same result, but until then, we will make sure *not* to give a tag to the first measure, so as no name be written at the start of the score. (The other way permits to tag the first measure and still not to write its name.)

Let us talk a bit further about tags.

What can we tag (= give name to)?

- Lyrics by adding a tag after the 1D2C4 BEGIN LYRICS
- Chords by adding a tag after the 1D2C6 BEGIN CHORDS



- Measures, by adding a tag after any space (unusual), the usual measure-breaking ZWSP 200B, sometimes ZWNJ 200C (to restore the beat count without breaking a line), or, as for the first measure, by adding a tag either after the 1D2C0 BEGIN STAFF or the 1D2C2 BEGIN STAFF GROUP (the former precedes in case both are used)

All other use of tags means that the corresponding tag is called for:

- 1D2C8 LYRICS SYLLABLE, 1D2C9 LYRICS MELISMA, 1D2CA LYRICS DASH, 1D2CB LYRICS SYNALÉPHA, 1D2CC LYRICS BEGIN SYLLABLE call for a lyrics tag, which means, pop some characters in the relevant LYRICS group.
- 1D2CE CHORD calls for a chord tag, which also pops a word in the relevant CHORD group.
- 1D2CF MEASURE NAME, as seen, decodes the current measure tag.
- 1D2C1 END STAFF, 1D2C3 END STAFF GROUP, 1D2D1 END SPAN, 1D176 END TIE, 1D178 END SLUR, 1D17A END PHRASE call for a measure tag. They use the tag as a horizontal displacement instruction: the staff ends at the given measure, the slur ends at the given measure, etc. (Without further combining character, ending at a given measure means ending at the beat end of measure position.)

This is why, in the Beethoven example above, there was no decrescendo in the second measure: the first measure states 1D2D0 (BEGIN SPAN) 1D283 (COMBINING STAFF 02) 1D251 (COMBINING BEAT 1) 1D29B (P-STAFF BELOW) 1D193 (̣) 1D2D1 (END SPAN) E0033 (TAG DIGIT THREE) 1D283 (COMBINING STAFF 02) 1D251 (COMBINING BEAT 1) 1D29B (P-STAFF BELOW): all required information is known from measure 1, there is nothing more to tell in measure 2 (nor in measure 3, apart that the “pp” is at the same anchor). It is however correct that the renderer will have to draw three times the measure 1: one time when the first measure barline is encoded, assuming the next measure will be named “3” and will end the decrescendo, one time when the second barline is found, assuming the next measure will be named “3” and trying to guess where the beat 1 will be, and a third and final time at the third barline, when the beat 1 of the measure “3” is placed. There could be other occurrences, at line breaks, as musical score is typically justified, which could displace the starting and ending points.

Summary of the paragraph e):

In order to have i) A tempo direction (“♩ = 80”) ii) Two verses iii) Verse numbers before the first word (“1.” and “2.”) iv) Syllables that span over two words v) Measure numbers at the start of each lines, we need to write the score as follows:

1. Au clair de la lune, mon ami Pierrot,  
 Prête-moi ta plume pour écrire un mot.  
 Ma chandelle est morte, je n'ai plus de feu.  
 Ouvre-moi ta porte, pour l'amour de Dieu !

2. Au clair de la lune, Pierrot répondit :  
 "Je n'ai pas de plume, je suis dans mon lit.  
 Va chez la voisine, je crois qu'elle y est,  
 Car dans sa cuisine on bat le briquet".

In the case your renderer is compatible with this proposal, you will see the same score as at the beginning of this section. In the case your renderer is not, you will see garbled text. I am not sure any of the cases will be really of interest, so here is the Unicode string with code points (and lyrics and directions rendered as normal text), segmented in rhymes, and with comments in italics:

*Lyrics context:*

U+1D2C4 1. Au clair de la lune, mon ami Pierrot,  
 Prête-moi ta plume pour écrire un mot.  
 Ma chandelle est morte, je n'ai plus de feu.  
 Ouvre-moi ta porte, pour l'amour de Dieu ! U+1D2C5  
 U+1D2C4 2. Au clair de la lune, Pierrot répondit :  
 "Je n'ai pas de plume, je suis dans mon lit.  
 Va chez la voisine, je crois qu'elle y est,  
 Car dans sa cuisine on bat le briquet". U+1D2C5

*Staff context:*

U+1D2C0 U+1D11A U+1D2CF U+1D250 U+1D29A U+1D11E U+1D2C1 U+E007F

*First rhyme:*

U+1D2D6 U+1D29A J = 80 U+1D2D7 U+1D134 U+1D15F U+1D2A9 U+1D2CC Au U+1D2CD U+1D2CC  
 Au U+1D2CD U+1D15F U+1D2A9 U+1D2C8 U+1D2C8 U+1D15F U+1D2A9 U+1D2C8 U+1D2C8  
 U+1D15F U+1D2AA U+1D2C8 U+1D2C8 U+1D100 U+200B U+E0032 U+1D15E U+1D2AB U+1D2CC lu  
 U+1D2CD U+1D2CC lu U+1D2CD U+1D15E U+1D2AA U+1D2C8 U+1D2C8 U+1D100 U+200B U+E0033  
 U+1D15F U+1D2A9 U+1D2C8 U+1D2CC Pier U+1D2CD U+1D15F U+1D2AB U+1D2CC a U+1D2CD  
 U+1D2C8 U+1D15F U+1D2AA U+1D2C8 U+1D2CC ré U+1D2CD U+1D15F U+1D2AA U+1D2CC Pier  
 U+1D2CD U+1D2CC pon U+1D2CD U+1D100 U+200B U+E0034 U+1D15D U+1D2A9 U+1D2C8  
 U+1D2C8 U+1D100

*Second rhyme:*

U+200B U+E0035 U+1D15F U+1D2A9 U+1D2CC Prê U+1D2CD U+1D2C8 U+1D15F U+1D2A9  
 U+1D2CC te- U+1D2CD U+1D2C8 U+1D15F U+1D2A9 U+1D2C8 U+1D2C8 U+1D15F U+1D2AA  
 U+1D2C8 U+1D2C8 U+1D100 U+200B U+E0036 U+1D15E U+1D2AB U+1D2CC plu U+1D2CD  
 U+1D2CC plu U+1D2CD U+1D15E U+1D2AA U+1D2C8 U+1D2C8 U+1D100 U+200B U+E0037  
 U+1D15F U+1D2A9 U+1D2C8 U+1D2C8 U+1D15F U+1D2AB U+1D2CC é U+1D2CD U+1D2C8  
 U+1D15F U+1D2AA U+1D2CC cri U+1D2CD U+1D2C8 U+1D15F U+1D2AA U+1D2CC re un U+1D2CD  
 U+1D2C8 U+1D100 U+200B U+E0038 U+1D15D U+1D2A9 U+1D2C8 U+1D2C8 U+1D100

*Third rhyme (phrase B):*

U+200B U+E0039 U+1D15F U+1D2AA U+1D2C8 U+1D2C8 U+1D15F U+1D2AA U+1D2CC chan  
 U+1D2CD U+1D2C8 U+1D15F U+1D2AA U+1D2CC del U+1D2CD U+1D2C8 U+1D15F U+1D2AA  
 U+1D2CC le est U+1D2CD U+1D2CC voi U+1D2CD U+1D100 U+200B U+E0031 U+E0030 U+1D15E  
 U+1D2A7 U+1D2CC mor U+1D2CD U+1D2CC si U+1D2CD U+1D15E U+1D2A7 U+1D2C8 U+1D2C8  
 U+1D100 U+200B U+E0031 U+E0031 U+1D15F U+1D2AA U+1D2C8 U+1D2C8 U+1D15F U+1D2A9  
 U+1D2C8 U+1D2C8 U+1D15F U+1D2A8 U+1D2C8 U+1D2C8 U+1D15F U+1D2A7 U+1D2C8 U+1D2C8  
 U+1D100 U+200B U+E0031 U+E0032 U+1D15D U+1D2A6 U+1D2C8 U+1D2C8 U+1D100

*Forth rhyme (phrase A):*

U+200B U+E0031 U+E0033 U+1D15F U+1D2A9 U+1D2CC Ou U+1D2CD U+1D2C8 U+1D15F U+1D2A9  
 U+1D2CC vre- U+1D2CD U+1D2C8 U+1D15F U+1D2A9 U+1D2C8 U+1D2C8 U+1D15F U+1D2AA  
 U+1D2C8 U+1D2CC cui U+1D2CD U+1D100 U+200B U+E0031 U+E0034 U+1D15E U+1D2AB U+1D2CC  
 por U+1D2CD U+1D2CC si U+1D2CD U+1D15E U+1D2AA U+1D2C8 U+1D2C8 U+1D100 U+200B  
 U+E0031 U+E0035 U+1D15F U+1D2A9 U+1D2C8 U+1D2C8 U+1D15F U+1D2AB U+1D2CC l'a

U+1D2CD U+1D2C8 U+1D15F U+1D2AA U+1D2C8 U+1D2C8 U+1D15F U+1D2AA U+1D2C8 U+1D2CC  
 bri U+1D2CD U+1D100 U+200B U+E0031 U+E0036 U+1D15D U+1D2A9 U+1D2C8 U+1D2C8 U+1D102

*Explicit end of context:*  
 U+200B U+E007F

f) Staff groups and the position algorithm

Before to go through all the proposed characters, let us transform "Au clair de la lune" to a piano part and see how to cope with staff groups.

In order to make it sure the score is my own invention, you will see that I wrote a really poor adaptation. Please consider the fact that I could not do any better as immaterial.

We will write this score:



The additional directions will no longer raise any question, apart about the layout of  $\text{Ⓢ}$ , but both directions are set to be above the staff, one at measure end, one at measure start.

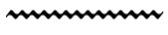

Remains to see i) how to create a staff group and multi-staff barlines, ii) how to create chords and beams (hint: as today) and iii) how to place notes in two staves.

### i. Staff groups

Again, there are two ways to create a staff group. If we just want the brace with a height of two staves, then BEGIN SPAN / END SPAN does the trick.

BEGIN SPAN / END SPAN asks the renderer to extend the last character before the END character, to cover the necessary span. The way this character is extended is not relevant for our purpose, but it looks logical to make use of variable fonts. Variable glyphs will include these characters, without doubt:

(The complete list is given in section 4 )

- Vertically:
  - staff brackets: 1D114 (brace), 1D115 (bracket), and it looks like we should add 1D1E9 (square bracket which looks like [ ]) and 1D1EA (line bracket which looks like | ).  
(I do propose to have them in the 1D100 block, to keep the blocks efficiently organized.)
  - measure barlines: 1D100 to 1D107 and 1D1F6 (left and right repeat sign)
- Horizontally:
  - Crescendos and Decrescendos (◁ and ▷)
  - Staves (1D116 — to 1D11B ≡), which usually span across measures (but not always: codas for instance are sometimes put after a staff interruption)
  - Possibly the multiple measure rest (1D129 ≡), depending on how the rendered designer considers it should be displayed
  - Trill: the 1D188 smear, even if a variable font will not be enough: the spanning should be  (the repetition of the character) and not  (the elongation of the glyph). Note that 1D188 is a combining character. 1D100 proposal on page 10 states we need to encode 1D196 (<sup>tr</sup>) 1D188 1D188 1D188.
  - Ties and phrases. These are currently encoded with the 1D175 BEGIN TIE and 1D176 END TIE, and with the 1D179 BEGIN PHRASE and 1D17A END PHRASE characters, but using control characters to display a simple spacing character (␣) does not look like normal Unicode so I propose to have those characters added per se: 1D1FB MUSICAL SYMBOL TIE and 1D1FD MUSICAL SYMBOL PHRASE.
- Both vertically and horizontally:
  - Beams and slurs, and for the same reasons as ties and phrases, I propose 1D1FA MUSICAL SYMBOL BEAM (just the filled parallelogram without stems) and 1D1FC MUSICAL SYMBOL SLUR

So, the first way is to input a staff brace with a BEGIN SPAN staff 01 MUSICAL SYMBOL BRACE END SPAN staff 02.

The other way is to declare a staff group context with 1D2C2 BEGIN STAFF GROUP and 1D2C3 END STAFF GROUP. The syntax is of course the same (BEGIN STAFF GROUP, staff 01, MUSICAL SYMBOL BRACE, END STAFF GROUP). The sole difference is that there can be one or two

spanned characters: the group symbol is either the last or the last-but-one, and optionally the last one is the 1D2DF MULTI-STAFF BARLINE, which indicates that for each barline, an implicit span is added between those staves. Without this character, each staff has its barline and they are not connected.

As is the case in the BEGIN SPAN/END SPAN, the first characters (all but one or two in the STAFF GROUP) do not span. Therefore, if one adds a text after the BEGIN STAFF GROUP and before the group symbol, this text is added before the group symbol at each line break, centred, like this:



The same holds when the begin staff is the same as the end staff: as there is no symbol to extend, any text placed in the staff group context will be centred against the staff, which can be used to give instrument names.

The start of the piano system for Au clair de la lune is:

1D2C2 (BEGIN STAFF GROUP) 1D282 (COMBINING STAFF 01) 1D114 (f) 1D2DF (MULTI-STAFF BARLINE) 1D2C3 (END STAFF GROUP) 1D283 (COMBINING STAFF 02)

1D2C0 (BEGIN STAFF) 1D282 (COMBINING STAFF 01) 1D11A (≡) 1D11E (♩) 1D2C1 (END STAFF) E007F (CANCEL TAG)

1D2C0 (BEGIN STAFF) 1D283 (COMBINING STAFF 02) 1D11A (≡) 1D122 (♯) 1D2C1 (END STAFF) E007F (CANCEL TAG)

What about more complex systems?

Let us look again at our Beethoven piece:

Allegretto. (♩ = 76)

The system will be described as such, left to right:

1D2C2 (BEGIN STAFF GROUP) 1D282 (STAFF 01) Flauti 1, 2 1D2C3 (END STAFF GROUP)  
 1D2C2 (BEGIN STAFF GROUP) 1D283 (STAFF 02) Oboi 1, 2 1D2C3 (END STAFF GROUP)  
 1D2C2 (BEGIN STAFF GROUP) 1D284 (STAFF 03) Clarinetti 1, 2 in A 1D2C3 (END STAFF GROUP)  
 1D2C2 (BEGIN STAFF GROUP) 1D285 (STAFF 04) Fagotti 1, 2 1D2C3 (END STAFF GROUP)  
 1D2C2 (BEGIN STAFF GROUP) 1D286 (STAFF 05) Corni 1, 2 in E 1D2C3 (END STAFF GROUP)  
 1D2C2 (BEGIN STAFF GROUP) 1D287 (STAFF 06) Tombe 1, 2 in D 1D2C3 (END STAFF GROUP)  
 1D2C2 (BEGIN STAFF GROUP) 1D288 (STAFF 07) Timpani in A, E 1D2C3 (END STAFF GROUP)  
 1D2C2 (BEGIN STAFF GROUP) 1D289 (STAFF 08) Violino 2160 ( I ) 1D2C3 (END STAFF GROUP)  
 1D2C2 (BEGIN STAFF GROUP) 1D28A (STAFF 09) Violino 2161 ( II ) 1D2C3 (END STAFF GROUP)  
 1D2C2 (BEGIN STAFF GROUP) 1D28B (STAFF 10) Viola 1D2C3 (END STAFF GROUP)  
 1D2C2 (BEGIN STAFF GROUP) 1D28C (STAFF 11) Violoncello 2160 ( I ) 1D2C3 (END STAFF GROUP)  
 1D2C2 (BEGIN STAFF GROUP) 1D28D (STAFF 12) Violoncello 2161 ( II ) e Contrabasso 1D2C3 (END STAFF GROUP)

1D2C2 (BEGIN STAFF GROUP) 1D282 (STAFF 01) 1D115 (ℓ) 1D2DF (MULTI-STAFF BARLINE) 1D2C3 (END STAFF GROUP) 1D285 (STAFF 04)  
 1D2C2 (BEGIN STAFF GROUP) 1D286 (STAFF 05) 1D115 (ℓ) 1D2DF (MULTI-STAFF BARLINE) 1D2C3 (END STAFF GROUP) 1D287 (STAFF 06)

1D2C2 (BEGIN STAFF GROUP) 1D289 (STAFF 08) 1D114 (f) 1D2C3 (END STAFF GROUP) 1D28A (STAFF 09)  
 1D2C2 (BEGIN STAFF GROUP) 1D28C (STAFF 11) 1D114 (f) 1D2C3 (END STAFF GROUP) 1D28D (STAFF 12)  
 1D2C2 (BEGIN STAFF GROUP) 1D289 (STAFF 08) 1D115 (l) 1D2DF (MULTI-STAFF BARLINE) 1D2C3 (END STAFF  
 GROUP) 1D28D (STAFF 12)

1D2C0 (BEGIN STAFF) 1D282 (STAFF 01) 1D11A (≡) 1D11E (♩) 1D2C1 (END STAFF) E007F (CANCEL TAG)  
 1D2C0 (BEGIN STAFF) 1D283 (STAFF 02) 1D11A (≡) 1D11E (♩) 1D2C1 (END STAFF) E007F (CANCEL TAG)  
 1D2C0 (BEGIN STAFF) 1D284 (STAFF 03) 1D11A (≡) 1D11E (♩) 1D2C1 (END STAFF) E007F (CANCEL TAG)  
 1D2C0 (BEGIN STAFF) 1D285 (STAFF 04) 1D11A (≡) 1D122 (♯) 1D2C1 (END STAFF) E007F (CANCEL TAG)  
 1D2C0 (BEGIN STAFF) 1D286 (STAFF 05) 1D11A (≡) 1D11E (♩) 1D2C1 (END STAFF) E007F (CANCEL TAG)  
 1D2C0 (BEGIN STAFF) 1D287 (STAFF 06) 1D11A (≡) 1D11E (♩) 1D2C1 (END STAFF) E007F (CANCEL TAG)  
 1D2C0 (BEGIN STAFF) 1D288 (STAFF 07) 1D11A (≡) 1D122 (♯) 1D2C1 (END STAFF) E007F (CANCEL TAG)  
 1D2C0 (BEGIN STAFF) 1D289 (STAFF 08) 1D11A (≡) 1D11E (♩) 1D2C1 (END STAFF) E007F (CANCEL TAG)  
 1D2C0 (BEGIN STAFF) 1D28A (STAFF 09) 1D11A (≡) 1D11E (♩) 1D2C1 (END STAFF) E007F (CANCEL TAG)  
 1D2C0 (BEGIN STAFF) 1D28B (STAFF 10) 1D11A (≡) 1D121 (♯) 1D2C1 (END STAFF) E007F (CANCEL TAG)  
 1D2C0 (BEGIN STAFF) 1D28C (STAFF 11) 1D11A (≡) 1D122 (♯) 1D2C1 (END STAFF) E007F (CANCEL TAG)  
 1D2C0 (BEGIN STAFF) 1D28C (STAFF 12) 1D11A (≡) 1D122 (♯) 1D2C1 (END STAFF) E007F (CANCEL TAG)

This done, a question will undoubtedly occur: Where do we store the abbreviated instrument names, that are used for next systems (i.e. after line breaks)? Well of course, we do not store them, because we are not giving document metadata, we are writing scores.

However, the musical context for staff and staff groups is only written by the renderer at two places: with the first measure, and at line breaks. Apart from these two cases, the musical context is not written, it is "in memory". So, if a new BEGIN STAFF GROUP or a new BEGIN STAFF occurs, and a STAFF GROUP or a STAFF is already existing in the context, the context is simply replaced.

Therefore, adding *1D2C2 (BEGIN STAFF GROUP) 1D28D (COMBINING STAFF 12) Vc 2 e Cb 1D2C3 (END STAFF GROUP)* after the first measure results in next lines being prefixed by "Vc 2 e Cb" whereas the first was "Violoncello II e Contrabasso". By the same means, we can avoid that the measure name be put at the first measure, even if we give a tag to it.

By "already existing in the context", we mean that:

- The context has not ended (the END STAFF with a tag different from E007F for instance, and the measure has been reached)
- The staff already has a context. There can be 24 staff contexts (one per staff) and 24 single-staff staff group contexts, 23 two-staff staff group contexts, 22 three-staff staff group contexts... 1 single 24-staff staff group context, so 300 staff group contexts overall.

ii. Chords and beams

How do we write this?



As for the chord, we are willing to combine three notes in one. Because of the Unification principle, this will be encoded with the zero-width joiner U+200D, as in many other scripts (and emojis).

So, the first beat is: 1D15E (♩) 1D2AE (P-1) 200D (ZWJ) 1D15E (♩) 1D2B0 (P+1) 200D (ZWJ) 1D15E (♩) 1D2B2 (P+3)

You will remark that the stem is directed to the bottom, whereas until now all stems have been directed to the top. There should be no semantics about it (so no need for Unicode to be directive about this) but impacts may be so huge e.g. in case of beams that we will set a rule: notes below the middle line are up-stemmed, notes above the middle line and on the middle line are down-stemmed. When in chords, the majority wins and default is up-stemmed.

(Usual conventions differ about the middle line, if the part is vocal, or when adjacent notes agree on a different direction.)

However, if only to show that in didactic books, one will need to be able to force the stem direction as needed. We therefore propose two code points 1D2DC COMBINING STEM UP and 1D2DD COMBINING STEM DOWN. This should also reverse the BEAMS, TIES, SLURS and PHRASES, as well as lines.

As for beams, there is no description in the [1D100 proposal](#) (adopted *verbatim* in the release notes) of the exact syntax of the BEGIN BEAM / END BEAM structure, only that the notes are enclosed between the two control characters.

As a consequence, we propose that the correct encoding for the dotted 8<sup>th</sup>-16<sup>th</sup> rhythm is: 1D173 (BEGIN BEAM) 1D160 (♩) 1D16D (COMBINING AUGMENTATION DOT) 1D161 (♩) 1D174 (END BEAM). The number of flags gives the number of beams.

We also propose that BEGIN BEAM / END BEAM can be nested, so as:



can be encoded as: 1D173 (BEGIN BEAM) 1D173 (BEGIN BEAM) 1D161 (♩) 1D161 (♩) 1D174 (END BEAM) 1D173 (BEGIN BEAM) 1D161 (♩) 1D161 (♩) 1D174 (END BEAM) 1D174 (END BEAM)

Null noteheads will help coding reiterations (tremolos) and generally any beam exception, such as if two beams are needed between 32<sup>nd</sup> groups: putting 1D159 (null notehead) feathered with a 1D16F (⦿) will give the desired result.



iii. How to place notes in two staves?

As you know, the basics is to use the combining characters 1D282 (staff 01) to 1D299 (staff 24). I will add a beat-positioning character for the 8<sup>th</sup>, and explain it after.

Let us encode the first measure (after the staff context):

1D134 (C) 1D15F (♩) 1D2A9 (P-6) 1D15F (♩) 1D2A9 (P-6) 1D15F (♩) 1D2A9 (P-6) 1D15F (♩) 1D2AA (P-5)

1D134 (C) 1D283 (staff 02) 1D15E (♩) 1D2AE (P-1) 200D (ZWJ) 1D15E (♩) 1D2B0 (P+1) 200D (ZWJ) 1D15E (♩) 1D2B2 (P+3) 1D173 (BEGIN BEAM) 1D160 (♩) 1D16D (COMBINING AUGMENTATION DOT) 1D254 (BEAT 4) 1D2AE (P-1) 200D (ZWJ) 1D160 (♩) 1D16D (COMBINING AUGMENTATION DOT) 1D2B0 (P+1) 200D (ZWJ) 1D160 (♩) 1D16D (COMBINING AUGMENTATION DOT) 1D2B2 (P+3) 1D161 (♩) 1D2AE (P-1) 200D (ZWJ) 1D161 (♩) 1D2B0 (P+1) 200D (ZWJ) 1D161 (♩) 1D2B2 (P+3) 1D174 (END BEAM) 1D13D (‡)

This gives the following result:




which is not too bad, but you will remark that the last quarter of the first staff aligns with the 16<sup>th</sup> of the second staff, whereas it should align with the quarter rest.

This is because we are dealing with normal spacing characters, though in an anchored compound word.

In order to handle spacing, the renderer needs to keep track of the current "caret" location, and as we are in a compound word, it needs to keep track of it, both horizontally and vertically, within the musigram. The algorithm is much less complex than the BIDI algorithm, but let us state it whatever.

Position algorithm:

- Any element (which mean, all except BEGIN/END pairs) may force its position as follows:
  - The position combining characters must follow this order: first the staff, then the beat, then the pitch.
  - If an element is missing, the current staff and beat indexes are used. If the staff is included, the beat defaults to 1.
  - The pitch must always be given for elements which have semantic difference depending on their pitch position (i.e. notes): any note must have a pitch. This pitch is not maintained to the next character unless a BEGIN character. The default is 1D2AC PITCH -3 character (≡≡ should not alter position because of the proposal); when characters have a different default pitch, this is made explicit in the section 4): 1D2C8, 1D2C9, 1D2CC, 1D2CE, 1D2CF, 1D2DA.
  - Exceptions:
 

Combining characters outside the 1D250-1D2BF range only combine and cannot be forced a position (e.g. COMBINING AUGMENTATION DOT). Which means that glyph combining characters need be encoded before the first positioning combining character.
- After the element (unless the following is a BEGIN character), the staff is maintained and the beat is increased by 1 if the character is a spacing one.
  - At each whitespace character, and at the musical context beginning, the staff index is set to one.
  - At each staff change (which subsumes measure breaks and musical context beginning), the beat is set to the 1<sup>st</sup> position.
  - Unless included in a BEGIN/END PAIR, after a space (SP line-break category), a ZWSP (ZW category) or a ZWNJ, the beat is set to the 1<sup>st</sup> position.

Note that the following are not spacing characters: tags and combining characters. Elements joined by a zero-width joiner space only if one of the elements do, and only once.
- Special pair: Beamed tuplets follow the positioning of the BEGIN/END BEAM group they are in, and cannot be independently positioned. A beamed triplet will be coded as 1D173 (BEGIN BEAM) 1D2D4 (BEGIN TUPLET) 3 1D2D5 (END TUPLET) [3 notes] 1D174 (END BEAM).
- BEGIN characters may force their position.
  - The position combining characters must follow this order: staff, beat and pitch.
  - If an element is missing, all indexes are used: staff, beat and pitch.
  - A horizontal position means the first anchored character at this beat if the pitch is P-16 to P+16, the last otherwise.
  - A staff vertical position means the top of the staff (unless stated otherwise by pitch) and beat to 0 (begin of measure, unless stated otherwise).
  - After a BEGIN character, the beat does not change unless forced. (BEGIN does not space.)
- END characters may force their position.
  - The position combining characters must follow this order: measure tag, staff, beat and pitch.
  - Measure tag is allowed for the above-mentioned 1D2C1 END STAFF, 1D2C3 END STAFF GROUP, 1D2D1 END SPAN, 1D176 END TIE, 1D178 END SLUR, 1D17A END PHRASE

- If an element is missing, the indexes of the BEGIN pair are used: staff, beat and pitch.
- The measure tag defaults the staff to 01 and the beat to end of measure.
- A horizontal position means the last anchored character at this beat if pitch is between P-16 and P+16, the first otherwise: a span from a given beat to itself can have a width of more than zero; such cases cover appoggiatura slurs.
- A staff vertical position means the bottom of the staff (unless stated otherwise by pitch). Such a case could be spanning measure barlines.
- After an END character, the beat is the one of the BEGIN character if forced, and the previous one if not (so in a sequence note1 at beat 5, BEGIN (...) END, note2 will have BEGIN and END at beat 5 and note2 at beat 6).
- Exceptions:  
 BEGIN BEAM / END BEAM, BEGIN TIE / END TIE, BEGIN SLUR / END SLUR, BEGIN PHRASE / END PHRASE, and the new BEGIN FANNED BEAM / END FANNED BEAM.  
 If some characters are included in these pairs, they update the index. (E.g. if a BEGIN TIE / END TIE contains three notes, the index will be incremented by 3. However, a BEGIN TIE END TIE with no included note works like any other pair.)
- The STAFF GROUP context and the STAFF context have their own beat count.  
 Because the line breaks are not anticipable, the contents of the BEGIN STAFF/END STAFF, and the contents of the BEGIN STAFF GROUP/END STAFF GROUP, are not counted in the beat index (one would not know if a note is preceded by a clef or not, so would not be able to tell the note beat).  
 Within a STAFF GROUP or STAFF CONTEXT, elements and BEGIN characters can be given a "+" (E002B) measure tag, to mean the measure that starts the line.  
 The chord diagrams (to be seen in 4) ) have their own beat and pitch count.
- There can be any number of elements attached to a given anchor, in which case they generally stack left-to-right, with two exceptions:
  - If the element is a direction (enclosed in a BEGIN DIRECTION / END DIRECTION) or any element pitched at the 1D29C tablature, 1D29D lyrics or 1D29E chords, then it stacks according to the multiple combining character algorithm (Unicode general structure, 2.11 Combining Characters), which is, from the staff outward (middle is considered above): two staff-above directions, encoded as D1 and D2 in this order, will be rendered D2 above D1 and conversely, two staff-below directions encoded as D3 and D4 in this order will be rendered D4 below D3 (this is why lyrics verses stack according to their context creation).  
 Two such elements linked with a ZWJ are placed left-to-right.
  - Usually the first elements anchored to a specific beat will align. This is not the case if the first elements are accidentals (266D to 266F, 1D12A to 1D133), appoggiaturas (contained within 1D2D8 BEGIN APPOGGIATURA and 1D2D9 END APPOGGIATURA) or grace notes (1D194 and 1D195).  
 These are still stacked left-to-right, but it is recommended they be right-aligned at the left of the anchor (so as the first following element will align with the others of same beat). This is the same rule as for skipped lyrics (verse numbers). If no other element is present at the beat, then the elements should be right-aligned at the left of the next anchor.

Playing this algorithm on behalf of the renderer:

- BEGIN STAFF GROUP puts a brace from staff 01 to staff 02, beat 1, of the staff group context pseudo-measure, which is really the first measure, but with an independent beat index, and initializes the (measure) beat index to 1.
- BEGIN STAFF puts the treble clef at staff 01, beat 1 of the staff pseudo-measure, and (re-)initializes the measure beat to 1.
- BEGIN STAFF puts the bass clef at staff 02, beat 1, and (re-re-)initializes the measure beat to 1.
- The direction-above puts the tempo direction where it is told to, but the indexes remain staff=01 beat=01, because directions do not space.
- <sup>c</sup> is put at staff 01, beat 1, and this spacing character increments the beat index to 2
- the three C4s go at beats 2, 3 and 4, and the D4 at *beat 5*.
- Then the staff 02 combining character for the <sup>c</sup> changes the staff index to 02, the <sup>c</sup> is put at beat 1, and the beat index incremented
- The C3 half note is put staff 02, beat 2, but as it is followed by a zero-width joiner, the E3 is also put staff 02, beat 2, and the same holds for the G3. Then the beat index is set to 3.
- However, the next element (the dotted eighth) is explicitly put at beat 4, not 3. The index is therefore put at beat 4 and the C3 placed there. The two other notes of the chord are linked by the ZWJ and therefore also are at beat 4. Then the index is set to 4+1 = 5.
- The eighth is put at *beat 5* and the rest at beat 6.

Which is why the D4, at beat 5, aligns with the eighth, also at beat 5.

What we forgot is that the renderer has no musical knowledge. A dotted 8<sup>th</sup>-16<sup>th</sup> rhythm does not take one beat, it takes two spacing characters, that is, two horizontal anchors that we called "beats". The word is the measure and we cannot encode the different parts within one given measure independently. We should therefore inform the renderer that, at first staff, the D4 should be at beat 6, not 5:

1D134 (C) 1D15F (J) 1D2A9 (P-6) 1D15F (J) 1D2A9 (P-6) 1D15F (J) 1D2A9 (P-6) 1D15F (J) *1D257 (BEAT 6)* 1D2AA (P-5)

1D134 (C) 1D283 (staff 02) 1D15E (J) 1D2AE (P-1) 200D (ZWJ) 1D15E (J) 1D2B0 (P+1) 200D (ZWJ) 1D15E (J) 1D2B2 (P+3) 1D173 (BEGIN BEAM) 1D160 (J) 1D16D (COMBINING AUGMENTATION DOT) 1D255 (BEAT 4) 1D2AE (P-1) 200D (ZWJ) 1D160 (J) 1D16D (COMBINING AUGMENTATION DOT) 1D2B0 (P+1) 200D (ZWJ) 1D160 (J) 1D16D (COMBINING AUGMENTATION DOT) 1D2B2 (P+3) 1D161 (J) 1D2AE (P-1) 200D (ZWJ) 1D161 (J) 1D2B0 (P+1) 200D (ZWJ) 1D161 (J) 1D2B2 (P+3) 1D174 (END BEAM) 1D13D (‡)

The next measure space (1D100 (I) *200B (ZWSP)* E0032 (TAG 2) ) results in staff index being back to 1 and beat index back to one, which means that the second measure is simply:

1D15E (J) 1D2AB (P-4) 1D15E (J) 1D2AA (P-5) 1D15D (C) 1D283 (staff 02) 1D2AE (P-1) 200D (ZWJ) 1D15D (C) 1D2B0 (P+1) 200D (ZWJ) 1D15D (C) 1D2B2 (P+3) 1D100 (I) *200B (ZWSP)* E0033 (TAG 3)

And the third measure is the same as the first one, without the <sup>c</sup>'s and with different pitches:

1D15F (J) 1D2A9 (P-6) 1D15F (J) 1D2AB (P-4) 1D15F (J) 1D2AA (P-5) 1D15F (J) 1D256 (BEAT 5)  
1D2AA (P-5)

1D15E (J) 1D283 (staff 02) 1D2AE (P-1) 200D (ZWJ) 1D15E (J) 1D2B0 (P+1) 200D (ZWJ) 1D15E  
(J) 1D2B2 (P+3) 1D173 (BEGIN BEAM) 1D160 (J) 1D16D (COMBINING AUGMENTATION DOT)  
1D254 (BEAT 3) 1D2AE (P-1) 200D (ZWJ) 1D160 (J) 1D16D (COMBINING AUGMENTATION  
DOT)1D2B0 (P+1) 200D (ZWJ) 1D160 (J) 1D16D (COMBINING AUGMENTATION DOT)1D2B2  
(P+3) 1D161 (J) 1D2AE (P-1) 200D (ZWJ) 1D161 (J) 1D2B0 (P+1) 200D (ZWJ) 1D161 (J) 1D2B2  
(P+3) 1D174 (END BEAM) 1D13D (‡)

#### 4) Is the proposal reasonable?

We will now review, range by range, the different proposed code points.

##### a) Comments on existing characters

As stated previously, a number of precisions have been made on existing characters, in the 1D100 block. This does not alter their semantics but they are worth mentioning.

The precisions are:

- Many characters should be able to span:
  - Vertically:
    - Measure bars: 1D100, 1D101, 1D102, 1D103, 1D104, 1D105
    - Codas: 1D106, 1D107
    - Staff brackets: 1D114, 1D115
    - Notes: 1D15E, 1D15F, 1D160, 1D161, 1D162, 1D163, 1D164
    - Stems: 1D165, 1D166
    - Articulation: 1D183, 1D184
    - Mensural notes: 1D1B6, 1D1B7, 1D1BB, 1D1BC, 1D1BE, 1D1BF, 1D1C0
  - Horizontally:
    - Staves: 1D116, 1D117, 1D118, 1D119, 1D11A, 1D11B
    - Rest: 1D129
    - Dynamics: 1D192, 1D193
    - Ornaments: 1D19D even if in that case, it is rather a character repetition
    - Pedals: 1D1B0
  - Both horizontally and vertically:
    - Beams and slurs, encoded with the characters: 1D173 and 1D174, 1D175 and 1D176, 1D177 and 1D178, 1D179 and 1D17A
    - Miscellaneous symbols: 1D1B1, 1D1B2
- Many characters should be able to combine:
  - With the beat, staff and pitch combining characters: All Unicode characters.
  - With 1D2DC COMBINING STEM UP and 1D2DD COMBINING STEM DOWN:
    - (A number of characters, including holds and pauses 1D110 and 1D111, will not be decomposed because addition of decomposition is prohibited by the Stability design principle)
    - Notes: 1D15E, 1D15F, 1D160, 1D161, 1D162, 1D163, 1D164
    - Stems: 1D165, 1D166
    - Beams and slurs: 1D175, 1D177 and 1D179 do not down-stem or up-stem enclosed notes, but the tie, slur or phrase is put below and upside-down (☺). 1D173 is not included in this list: the beam pair should put the beam(s) according to the stems of inner notes.
  - With 1D1EB COMBINING NUT:
    - Tablature: 1D11C, 1D11D
- Control characters 1D173 to 1D17A have been added a syntax: either there are enclosed notes, in which case they are beamed (with number of beams equal to number of flags), tied, slurred or phrased, or they are empty but with positioned begin and end

- characters, in which case they draw the underlying symbol. In the case of beams, a nesting pair creates a beam between nested pairs, but does not add them a beam.
- The zero-width non-joiner (200C) is used to reinitialize the beat count without breaking a measure.
  - The zero-width joiner (200D) is used:
    - To create chords (elements aligned vertically and, if relevant, sharing a single stem)
    - Conversely, to stack left-to-right the elements that should stack vertically: directions (enclosed in a BEGIN DIRECTION / END DIRECTION) or any element pitched at 1D29C tablature, 1D29D lyrics or 1D29E chords (1D29A staff above, 1D29B staff below are explicitly not in this list)
  - The fraction slash (2044) is used to create the time signature fractions (the semantics is really a fraction). However, the text renderer should, in a musical context, kern this character with 0 width and just display both sequence of adjacent digits one above the other.
  - Tags are being used to name measure, and E007F and E002B have special meanings: last and next measure.

Precision on the line breaks:

As shortly discussed early in the section 3), once the musical context is opened, there would be no opportunity for line breaks apart after a measure barline. The CMN looks like a fourth style of context analysis to determine line break opportunities, with a straightforward algorithm: do not break until 1D100 to 1D107 character.

However, the [Unicode annex #14 line breaking algorithm](#) makes it clear that:

- Whatever the script, Mandatory breaks (the combination of BK, CR, LF and NL character classes) are mandatory. If such a break occurs, the line must break at the current beat because that is the user expectation.  
Note: The editable character should be a beat group within a musigram and the editable word, a measure: subsequent characters at the same beat should still be added to the previous line, the break is really after the beat, not the character.

This means that no direction can contain such characters, but that is not a problem as directions are stacked vertically. The same goes for lyrics syllables.

Musical contexts (staff, staff group, lyrics and chords) are not displayed until used, so there is no line to break in these groups even if they contain mandatory breaks. Especially, lyrics rhymes can be separated by line breaks, as the lyrics syllable algorithm searches for whitespaces without displaying them.

If a mandatory break is found within a staff or staff group context creation, the user will probably expect the instrument name to be segmented in two "lines" (rows really), but this contradicts the annex, because a musical score line is one-musigram-tall. This means that no instrument name can contain such characters, which does not impact semantics but may contradict the user expectation. The two "lines" may, for this need, be encapsulated in two DIRECTION groups.

- Similarly, SP and ZW classes (0020 and 200B as of now) are not tailorable, but whitespaces may be included in directions or lyrics, and of course line must not break there.

Therefore, it is highly recommended to use only no-break spaces (and zero-width non-joiners) wherever a space is needed in directions and lyrics syllables. (The contents that is pushed is the lyrics syllable enclosed characters, which means a no-break space will be pushed even if the lyrics context contains a standard space.)

If this rule is not followed, then on some occasions the user will see an unexpected line break. These occasions are when such direction of lyrics is nearing the end of line, which is hopefully not frequent, but unfortunately not foreseeable.

(There is of course an "emergency mode" (= no other choice), which is, in the worst case the line may break at any beat, and if not sufficient (many elements at the same anchor), at any grapheme. Inter-beat compression and extension is however usually more than sufficient to break lines only at normal break line opportunities.)

Note that technically a line break is *after* a space, so the space will be attached to the *previous* measure, but it will be optionally tagged with the name of the *following* measure.

\* \* \*



b) Code points in the 1D100 block

The 1D100 Musical *Symbols* block has 25 free code points, and therefore it seems logical to use them to store new *symbols*.

The proposed symbols are not *necessary* for this proposal of encoding scores, but these are very well-spread symbols, used both in scores and educational material. They look like missing for both the original 1D100 aim and the score-writing.

We should bear in mind that many symbols (>2400) are needed to cover all musical needs (SMuFL makes it absolutely clear), and keep the additions to the 1D100 block to a minimum. Apart from the above-mentioned rule of original aim, the added characters have a strong affinity, if not similarity, with others already in the 1D100 block.

The proposed new code points are:

- Clefs (continued from 1D11E-1D126):

Code point	Glyph	Name	Comments
1D127	T A B	MUSICAL SYMBOL TAB CLEF	Many scores have a tablature staff (whence the 1D11B SIX-LINE STAFF), but it misses the clef for it.

- Staff brackets:

These two brackets are standard notation and come in addition to 1D114 and 1D115:

Code point	Glyph	Name	Comments
1D1E9	[	MUSICAL SYMBOL SQUARE BRACKET	
1D1EA		MUSICAL SYMBOL BRACKET LINE	

- Chord diagrams:

The chord diagrams (usually placed at the tablature pitch level) already have the 1D11C and 1D11D fretboard characters.

The common symbols usually put on such diagrams (filled circle, empty circle, cross, circled numbers) already exist but two: the fretboard nut and barre chord symbols, below.

In order to encode such a diagram in a score, the syntax is:

- 1D2DA BEGIN CHORD DIAGRAM, at the correct beat. The pitch defaults to P+TABLATURE
- The wanted fretboard character, possibly nuted
- All wanted elements, which can be anything, but are usually one of:
  - 26AA (○) medium white circle and 26AB (●) medium black circle



- 00D7 (×) multiplication sign
- Letter T (thumb) and digits (1 to 4 for fingers, but can also be different for fret numbering)
- Alternately, 24C9 ① and 2460 ① to 2463 ④, in replacement for the medium black circle
- A barre symbol (could also be a tie)

The elements need be correctly placed on the fretboard, with the following remarks:



- The chord diagrams have their own beat and pitch indexes but the algorithm is the same.
- Beats are still horizontal and pitch vertical, which can be confusing from a fretboard point of view, but is not if we recall beats and pitches are really about anchoring.
- All pitches are negative and 0 is the top fret, which can also be confusing from a fretboard point of view.



○ 1D2DB END CHORD DIAGRAM



The Dm chord (  ) is therefore encoded as: 1D2DA (BEGIN CHORD DIAGRAM) 1D11C (  ) 1D1EB (COMBINING FRETBOARD NUT) 00D7 (×) 1D29A (P+STAFF ABOVE) 00D7 (×) 1D29A (P+STAFF ABOVE) 26AA (○) 1D29A (P+STAFF ABOVE) 26AB (●) 1D2AC (P-3) 26AB (●) 1D2AA (P-5) 26AB (●) 1D2AE (P-1) 1D2DB (END CHORD DIAGRAM)



The B chord (  ) is encoded as: 1D2DA (BEGIN CHORD DIAGRAM) 1D11C (  ) 2 1D250 (BEAT START OF MEASURE) 1D2AE (P-1) 00D7 (×) 1D29A (P+STAFF ABOVE) 1D2D0 (BEGIN SPAN) 1D2AE (P-1) 1D1EC (BARRE CHORD) 1D2D1 (END SPAN) 1D256 (BEAT 6) 1D2AE (P-1) 26AB (●) 1D253 (BEAT 3) 1D2AA (P-5) 26AB (●) 1D2AA (P-5) 26AB (●) 1D2AA (P-5) 1D2DB (END CHORD DIAGRAM)

Code point	Glyph	Name	Comments
1D1EB		MUSICAL SYMBOL COMBINING FRETBOARD NUT	If a nuted precomposed fretboard be proposed, then 1D11C (or D) + 1D1EB should be a canonical decomposition.
1D1EC		MUSICAL SYMBOL FRETBOARD BARRE	Variable width



• Lines:

In this range we propose to store different lines that are useful for many scores (and educational books), namely full lines (used as volta brackets especially), dashed lines and dotted lines (used with the <sup>8va</sup> for instance) and an invisible line. The similar characters in the 1D100 block are the analytics ones: 1D1A6 <sup>h</sup> 1D1A7 <sup>n</sup> and 1D1A8 <sup>l</sup>.

All these four lines exist with or without a start ( <sup>r</sup> ), and with or without an end ( <sup>l</sup> ). The start and end are therefore provided as combining characters.

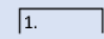
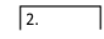
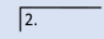

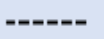



All these four lines symbols are intended to combine with 1D2DC COMBINING STEM UP and 1D2DD COMBINING STEM DOWN (e.g. the 8<sup>va</sup> can be placed below a staff). They are also intended to be of variable widths.

Note for voltas:


Voltas are used around repeat barlines, to tell which measures are to be played at the n<sup>th</sup> repetition: , , etc. The numbers are a DIRECTION BELOW, so the correct encoding is to have a SPAN followed by a DIRECTION.

A "first" volta is then: 1D2D0 (BEGIN SPAN) 1D250 (BEAT 0) 1D29A (P+STAFF ABOVE) 1D1F0 (SOLID LINE) 1D1F4 (COMBINING LINE START) 1D1F5 (COMBINING LINE END) 1D2D1 (END SPAN) 1D281 (BEAT END OF MEASURE) 1D29A (P+STAFF ABOVE) BEGIN DIRECTION BELOW 1. END DIRECTION BELOW.

However, as the first volta and the closed- and open-ending second volte appear everywhere, especially in educational texts, precomposed characters are proposed.

Code point	Glyph	Name	Comments
1D1ED		MUSICAL SYMBOL VOLTA 1. WITH START AND END OF LINE	With the compatibility decomposition as stated above, which will only work in a musical context (because of positioning characters). This precomposed form may be used in a BEGIN SPAN/END SPAN pair, placed above the staff.
1D1EE		MUSICAL SYMBOL VOLTA 2. WITH START AND END OF LINE	Idem
1D1EF		MUSICAL SYMBOL VOLTA 2. WITH START OF LINE	Idem
1D1F0		MUSICAL SYMBOL SOLID LINE	
1D1F1		MUSICAL SYMBOL DASHED LINE	
1D1F2		MUSICAL SYMBOL DOTTED LINE	
1D1F3	(none)	MUSICAL SYMBOL INVISIBLE LINE	Especially used for stime and tuplets, with combining start or end.
1D1F4		MUSICAL SYMBOL COMBINING START OF LINE	Looks similar to U+2E22
1D1F5		MUSICAL SYMBOL COMBINING END OF LINE	Looks similar to U+2E23 and U+1D1A8 (MUSICAL SYMBOL END OF STIMME)





- Bars:

Code point	Glyph	Name	Comments
1D1F6		MUSICAL SYMBOL LEFT AND RIGHT REPEAT SIGN	This useful character cannot be composed with currently-available characters in the 1D100 block. Note that, when a line breaks at this measure barline, the renderer must use the 1D106 (⋮) glyph at the end of the line and the 1D107 (⋮) glyph at the beginning of the next one, after the staff group and staff contexts.

- Beams and slurs:

All these symbols but the two last ones are intended to combine with 1D2DC COMBINING STEM UP and 1D2DD COMBINING STEM DOWN.

The last two control characters are intended in addition to the 1D173 BEGIN BEAM / 1D174 END BEAM, to add the capability of fanned (aka feathered) beams. They have become common notations for accelerandi and rallentandi. The number of lines at start is given by the number of feathers of the first included element, and the number of lines at end is given by the number of feathers of the last included element (number of feathers of other elements are not considered). In order to keep consistency with the 1D173/1D174 pair, the included elements space in the measure anchors.

Code point	Glyph	Name	Comments
1D1FA		MUSICAL SYMBOL BEAM	From a variable-font point of view, this is a parallelogram (in case of sloped beam) and the length adjusts to cover the inter-note distance or a part of it (dotted 8 <sup>th</sup> -16 <sup>th</sup> rhythm case).
1D1FB		MUSICAL SYMBOL TIE	Variable width
1D1FC		MUSICAL SYMBOL SLUR	Variable width and height
1D1FD		MUSICAL SYMBOL PHRASE	Variable width
1D1FE	(none)	MUSICAL SYMBOL BEGIN FANNED BEAM	Control character, the included elements space
1D1FF	(none)	MUSICAL SYMBOL END FANNED BEAM	Control character, the included elements space

\* \* \*

For the next characters, as stated before, it is intended to use the 1D2xx block starting at 1D250. However, the characters could be placed everywhere: the codes should be of the form 1xxyy, with yy ranging over at most 176 positions, which means that yy could also be from 1xx00 to 1xxAF.

The compromise will be to display the proposed code points as 1xx50 to 1xxFF.

c) Combining beat characters

Code point	Glyph	Name	Comments
1xx50	(none)	MUSICAL SCRIPT COMBINING BEAT START OF MEASURE	Alias MUSICAL SCRIPT COMBINING BEAT 0
1xx51	(none)	MUSICAL SCRIPT COMBINING BEAT 1	
1xx52	(none)	MUSICAL SCRIPT COMBINING BEAT 2	
1xx53	(none)	MUSICAL SCRIPT COMBINING BEAT 3	
1xx54	(none)	MUSICAL SCRIPT COMBINING BEAT 4	
1xx55	(none)	MUSICAL SCRIPT COMBINING BEAT 5	
1xx56	(none)	MUSICAL SCRIPT COMBINING BEAT 6	
1xx57	(none)	MUSICAL SCRIPT COMBINING BEAT 7	
1xx58	(none)	MUSICAL SCRIPT COMBINING BEAT 8	
1xx59	(none)	MUSICAL SCRIPT COMBINING BEAT 9	
1xx5A	(none)	MUSICAL SCRIPT COMBINING BEAT 10	
1xx5B	(none)	MUSICAL SCRIPT COMBINING BEAT 11	
1xx5C	(none)	MUSICAL SCRIPT COMBINING BEAT 12	
1xx5D	(none)	MUSICAL SCRIPT COMBINING BEAT 13	
1xx5E	(none)	MUSICAL SCRIPT COMBINING BEAT 14	
1xx5F	(none)	MUSICAL SCRIPT COMBINING BEAT 15	
1xx60	(none)	MUSICAL SCRIPT COMBINING BEAT 16	
1xx61	(none)	MUSICAL SCRIPT COMBINING BEAT 17	
1xx62	(none)	MUSICAL SCRIPT COMBINING BEAT 18	
1xx63	(none)	MUSICAL SCRIPT COMBINING BEAT 19	
1xx64	(none)	MUSICAL SCRIPT COMBINING BEAT 20	
1xx65	(none)	MUSICAL SCRIPT COMBINING BEAT 21	
1xx66	(none)	MUSICAL SCRIPT COMBINING BEAT 22	
1xx67	(none)	MUSICAL SCRIPT COMBINING BEAT 23	
1xx68	(none)	MUSICAL SCRIPT COMBINING BEAT 24	
1xx69	(none)	MUSICAL SCRIPT COMBINING BEAT 25	
1xx6A	(none)	MUSICAL SCRIPT COMBINING BEAT 26	
1xx6B	(none)	MUSICAL SCRIPT COMBINING BEAT 27	
1xx6C	(none)	MUSICAL SCRIPT COMBINING BEAT 28	
1xx6D	(none)	MUSICAL SCRIPT COMBINING BEAT 29	
1xx6E	(none)	MUSICAL SCRIPT COMBINING BEAT 30	
1xx6F	(none)	MUSICAL SCRIPT COMBINING BEAT 31	
1xx70	(none)	MUSICAL SCRIPT COMBINING BEAT 32	
1xx71	(none)	MUSICAL SCRIPT COMBINING BEAT 33	
1xx72	(none)	MUSICAL SCRIPT COMBINING BEAT 34	
1xx73	(none)	MUSICAL SCRIPT COMBINING BEAT 35	
1xx74	(none)	MUSICAL SCRIPT COMBINING BEAT 36	
1xx75	(none)	MUSICAL SCRIPT COMBINING BEAT 37	

1xx76	(none)	MUSICAL SCRIPT COMBINING BEAT 38	
1xx77	(none)	MUSICAL SCRIPT COMBINING BEAT 39	
1xx78	(none)	MUSICAL SCRIPT COMBINING BEAT 40	
1xx79	(none)	MUSICAL SCRIPT COMBINING BEAT 41	
1xx7A	(none)	MUSICAL SCRIPT COMBINING BEAT 42	
1xx7B	(none)	MUSICAL SCRIPT COMBINING BEAT 43	
1xx7C	(none)	MUSICAL SCRIPT COMBINING BEAT 44	
1xx7D	(none)	MUSICAL SCRIPT COMBINING BEAT 45	
1xx7E	(none)	MUSICAL SCRIPT COMBINING BEAT 46	
1xx7F	(none)	MUSICAL SCRIPT COMBINING BEAT 47	
1xx80	(none)	MUSICAL SCRIPT COMBINING BEAT 48	
1xx81	(none)	MUSICAL SCRIPT COMBINING BEAT END OF MEASURE	

d) Combining staff characters

Code point	Glyph	Name	Comments
1xx82	(none)	MUSICAL SCRIPT COMBINING STAFF 01	
1xx83	(none)	MUSICAL SCRIPT COMBINING STAFF 02	
1xx84	(none)	MUSICAL SCRIPT COMBINING STAFF 03	
1xx85	(none)	MUSICAL SCRIPT COMBINING STAFF 04	
1xx86	(none)	MUSICAL SCRIPT COMBINING STAFF 05	
1xx87	(none)	MUSICAL SCRIPT COMBINING STAFF 06	
1xx88	(none)	MUSICAL SCRIPT COMBINING STAFF 07	
1xx89	(none)	MUSICAL SCRIPT COMBINING STAFF 08	
1xx8A	(none)	MUSICAL SCRIPT COMBINING STAFF 09	
1xx8B	(none)	MUSICAL SCRIPT COMBINING STAFF 10	
1xx8C	(none)	MUSICAL SCRIPT COMBINING STAFF 11	
1xx8D	(none)	MUSICAL SCRIPT COMBINING STAFF 12	
1xx8E	(none)	MUSICAL SCRIPT COMBINING STAFF 13	
1xx8F	(none)	MUSICAL SCRIPT COMBINING STAFF 14	
1xx90	(none)	MUSICAL SCRIPT COMBINING STAFF 15	
1xx91	(none)	MUSICAL SCRIPT COMBINING STAFF 16	
1xx92	(none)	MUSICAL SCRIPT COMBINING STAFF 17	
1xx93	(none)	MUSICAL SCRIPT COMBINING STAFF 18	
1xx94	(none)	MUSICAL SCRIPT COMBINING STAFF 19	
1xx95	(none)	MUSICAL SCRIPT COMBINING STAFF 20	
1xx96	(none)	MUSICAL SCRIPT COMBINING STAFF 21	
1xx97	(none)	MUSICAL SCRIPT COMBINING STAFF 22	
1xx98	(none)	MUSICAL SCRIPT COMBINING STAFF 23	
1xx99	(none)	MUSICAL SCRIPT COMBINING STAFF 24	

e) Combining pitch characters

Code point	Glyph	Name	Comments
1xx9A	(none)	MUSICAL SCRIPT COMBINING PITCH ABOVE THE STAFF	
1xx9B	(none)	MUSICAL SCRIPT COMBINING PITCH BELOW THE STAFF	
1xx9C	(none)	MUSICAL SCRIPT COMBINING PITCH TABLATURE POSITION	
1xx9D	(none)	MUSICAL SCRIPT COMBINING PITCH LYRICS POSITION	
1xx9E	(none)	MUSICAL SCRIPT COMBINING PITCH CHORDS POSITION	
1xx9F	(none)	MUSICAL SCRIPT COMBINING PITCH 16 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -16
1xxA0	(none)	MUSICAL SCRIPT COMBINING PITCH 15 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -15
1xxA1	(none)	MUSICAL SCRIPT COMBINING PITCH 14 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -14
1xxA2	(none)	MUSICAL SCRIPT COMBINING PITCH 13 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -13
1xxA3	(none)	MUSICAL SCRIPT COMBINING PITCH 12 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -12
1xxA4	(none)	MUSICAL SCRIPT COMBINING PITCH 11 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -11
1xxA5	(none)	MUSICAL SCRIPT COMBINING PITCH 10 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -10
1xxA6	(none)	MUSICAL SCRIPT COMBINING PITCH 9 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -9
1xxA7	(none)	MUSICAL SCRIPT COMBINING PITCH 8 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -8
1xxA8	(none)	MUSICAL SCRIPT COMBINING PITCH 7 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -7
1xxA9	(none)	MUSICAL SCRIPT COMBINING PITCH 6 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -6
1xxAA	(none)	MUSICAL SCRIPT COMBINING PITCH 5 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -5
1xxAB	(none)	MUSICAL SCRIPT COMBINING PITCH 4 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -4
1xxAC	(none)	MUSICAL SCRIPT COMBINING PITCH 3 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -3
1xxAD	(none)	MUSICAL SCRIPT COMBINING PITCH 2 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -2
1xxAE	(none)	MUSICAL SCRIPT COMBINING PITCH 1 BELOW	Alias MUSICAL SCRIPT COMBINING PITCH -1
1xxAF	(none)	MUSICAL SCRIPT COMBINING PITCH CENTER OF THE STAFF	Alias MUSICAL SCRIPT COMBINING PITCH 0
1xxB0	(none)	MUSICAL SCRIPT COMBINING PITCH 1 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +1
1xxB1	(none)	MUSICAL SCRIPT COMBINING PITCH 2 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +2

<b>1xxB2</b>	(none)	MUSICAL SCRIPT COMBINING PITCH 3 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +3
<b>1xxB3</b>	(none)	MUSICAL SCRIPT COMBINING PITCH 4 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +4
<b>1xxB4</b>	(none)	MUSICAL SCRIPT COMBINING PITCH 5 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +5
<b>1xxB5</b>	(none)	MUSICAL SCRIPT COMBINING PITCH 6 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +6
<b>1xxB6</b>	(none)	MUSICAL SCRIPT COMBINING PITCH 7 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +7
<b>1xxB7</b>	(none)	MUSICAL SCRIPT COMBINING PITCH 8 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +8
<b>1xxB8</b>	(none)	MUSICAL SCRIPT COMBINING PITCH 9 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +9
<b>1xxB9</b>	(none)	MUSICAL SCRIPT COMBINING PITCH 10 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +10
<b>1xxBA</b>	(none)	MUSICAL SCRIPT COMBINING PITCH 11 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +11
<b>1xxBB</b>	(none)	MUSICAL SCRIPT COMBINING PITCH 12 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +12
<b>1xxBC</b>	(none)	MUSICAL SCRIPT COMBINING PITCH 13 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +13
<b>1xxBD</b>	(none)	MUSICAL SCRIPT COMBINING PITCH 14 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +14
<b>1xxBE</b>	(none)	MUSICAL SCRIPT COMBINING PITCH 15 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +15
<b>1xxBF</b>	(none)	MUSICAL SCRIPT COMBINING PITCH 16 ABOVE	Alias MUSICAL SCRIPT COMBINING PITCH +16

f) Musical context control characters

The musical context (first 8 characters) is not printed when found, but when used, either by a first measure (STAFF and STAFF GROUP), a line break (STAFF and STAFF GROUP), or a direct call (LYRICS and CHORDS).

Code point	Glyph	Name	Comments
<b>1xxC0</b>	(none)	MUSICAL SYMBOL BEGIN STAFF CONTEXT	This character opens a musical context. The first character (usually a staff symbol) is expended for all the measures while the context remains valid. All centres must align within a line, which means redrawing a musigram if additional vertical elements (e.g. lyrics) are found at following measures. A second use of a BEGIN STAFF / END STAFF for a still-valid staff replaces the staff context.
<b>1xxC1</b>	(none)	MUSICAL SYMBOL END STAFF CONTEXT	



<b>1xxC2</b>	(none)	MUSICAL SYMBOL BEGIN STAFF GROUP CONTEXT	This character opens a musical context. The last character (last but one in the case the last is 1xxDF MULTI-STAFF BARLINES) expands vertically between staves, if the end staff is different from the begin staff. A second use of a BEGIN STAFF GROUP / END STAFF GROUP replaces the staff group context.
<b>1xxC3</b>	(none)	MUSICAL SYMBOL END STAFF GROUP CONTEXT	
<b>1xxC4</b>	(none)	MUSICAL SYMBOL BEGIN LYRICS CONTEXT	This character opens a musical context. A second use of a BEGIN LYRICS / END LYRICS adds a new lyrics context, which can be called for either by another lyrics call at one given horizontal anchor, or by means of a tag.
<b>1xxC5</b>	(none)	MUSICAL SYMBOL END LYRICS CONTEXT	
<b>1xxC6</b>	(none)	MUSICAL SYMBOL BEGIN CHORDS CONTEXT	This character opens a musical context. A second use of a BEGIN CHORDS / END CHORDS adds a new chords context, which can be called for either by another chords call at one given horizontal anchor, or by means of a tag.
<b>1xxC7</b>	(none)	MUSICAL SYMBOL END CHORDS CONTEXT	
<b>1xxC8</b>	(none)	MUSICAL SYMBOL NEXT LYRICS SYLLABLE	Calls for the lyrics context and returns the next word, defined as the string enclosed between two characters with property Mandatory Break or SP or ZW (sot/eot included). Defaults at PITCH LYRICS POSITION. The character (whatever the resulting return) does not space (shifts the indexes).
<b>1xxC9</b>	-	MUSICAL SYMBOL LYRICS MELISMA	This is a control character which does not space. It spans horizontally. The melisma is ZWJ with the next syllable, or can be put at each necessary lyrics place if a verse placeholder is needed, in which case two consecutive melismas should join. Defaults at PITCH LYRICS POSITION.
<b>1xxCA</b>	-	MUSICAL SYMBOL LYRICS DASH	This character is implied when a BEGIN SYLLABLE/END SYLLABLE matches a string of characters that are not the next ones. It is a spacing character.

<b>1xxCB</b>	◡	MUSICAL SYMBOL LYRICS SYNALEPHA	Looks like 203F ◡ UNDERTIE. It is a spacing character. It is also a space and will match any word boundary in the lyrics syllable process.
<b>1xxCC</b>	(none)	MUSICAL SYMBOL BEGIN NEXT LYRICS SYLLABLE	Defines the next syllable to be produced, and updates the lyrics context accordingly. It tries to match the enclosed text and shifts the lyrics index to the next occurrence. While matching, whitespaces are all considered equal. If some characters are left over while matching, the missed text is returned as a prefix to next anchor. If no match is found, the missed text is returned as well. The pair does not space, whatever the included characters. It defaults at the PITCH LYRICS POSITION.
<b>1xxCD</b>	(none)	MUSICAL SYMBOL END NEXT LYRICS SYLLABLE	
<b>1xxCE</b>	(none)	MUSICAL SYMBOL NEXT CHORD	Calls for the chords context, and does not space. It defaults at the PITCH CHORD POSITION.
<b>1xxCF</b>	(none)	MUSICAL MEASURE NAME	Calls for the current measure tag and decodes it by mapping E0021-E007E to 0021-007E and E0020 to 00A0 (no-break space). Does not return anything if no tag has been given to the specific measure. It does not space. It defaults at the PITCH ABOVE THE STAFF POSITION.

g) Group control characters

Notes for directions:

Directions permit to insert elements in the score relative to other elements:

- The default positioning of a direction is at the preceding beat but also at the preceding pitch.
- A free place is searched for the element, upwards or downwards, depending on if above or below the middle line. Places occupied by anchored elements, other directions, beams, ties, slurs, etc. are not free for this respect.
- If a direction is zero-width-joined with an element (before or after this element), the direction is put at the told position, stacking left-to-right.

Directions are mainly used to cover actual directions at staff-level (tempo like Allegretto, dynamics like piano, pedal marks, ...) as well as note notations and especially fingering.

Code point	Glyph	Name	Comments
1xxD0	(none)	MUSICAL SYMBOL BEGIN SPAN	Permits to include any other string of characters with two consequences: <ul style="list-style-type: none"> <li>• The last character is expanded horizontally and vertically from the BEGIN SPAN position (after the other enclosed characters) to the END SPAN position</li> <li>• The indexes are not altered.</li> </ul> In case of line breaking in the span, the new line should start again with the starting characters. The spanned character might be complicated to split in two, but this process is for the renderer to handle.
1xxD1	(none)	MUSICAL SYMBOL END SPAN	
1xxD2	(none)	MUSICAL SYMBOL BEGIN BRACKET TUPLET	A bracket with a gap to fit the text set between the BEGIN and the END. E.g. "2" in this bracket tuplet: $\lrcorner^2\llcorner$ . The line spans from the BEGIN position to the END position.
1xxD3	(none)	MUSICAL SYMBOL END BRACKET TUPLET	
1xxD4	(none)	MUSICAL SYMBOL BEGIN BEAM TUPLET	The enclosed characters are added as a tuplet indication to the including beam group, next to the beam, without bracket. The position cannot be altered, it is the one of the beam group.
1xxD5	(none)	MUSICAL SYMBOL END BEAM TUPLET	
1xxD6	(none)	MUSICAL SYMBOL BEGIN DIRECTION	The directions do not space, nor do their contents.

<b>1xxD7</b>	(none)	MUSICAL SYMBOL END DIRECTION	
<b>1xxD8</b>	(none)	MUSICAL SYMBOL BEGIN APPOGGIATURA	The enclosed characters are displayed smaller than usual (and usually aligned right) The characters are not slurred, an additional slur must be added. Appoggiaturas are significant, not style, and are therefore included. Even if small notes are also used for cue notes, this group is not intended to cover cue notes, which are style.
<b>1xxD9</b>	(none)	MUSICAL SYMBOL END APPOGIATURA	
<b>1xxDA</b>	(none)	MUSICAL SYMBOL BEGIN CHORD DIAGRAM	The default position is at PITCH TABLATURE POSITION. It does not space, nor does its contents.
<b>1xxDB</b>	(none)	MUSICAL SYMBOL END CHORD DIAGRAM	

h) Combining stem characters

Code point	Glyph	Name	Comments
<b>1xxDC</b>	(none)	MUSICAL SYMBOL COMBINING STEM UP	Forces the stemmed notes to be up-stemmed. Forces the ties, slurs, phrases and bracket tuplets to have ends lower than the centre. The start and end of lines are lower than the line.
<b>1xxDD</b>	(none)	MUSICAL SYMBOL COMBINING STEM DOWN	Forces the stemmed notes to be down-stemmed Forces the ties, slurs, phrases and bracket tuplets to have centre lower than the ends. The start and end of lines are above the line.

i) Staff characters

Code point	Glyph	Name	Comments
<b>1xxDE</b>	(none)	MUSICAL SYMBOL STAFF NONE	This symbol is used when positioning is necessary but not the printing of the staff. Ledger lines, starting at $P\pm 6$ , are still printed.
<b>1xxDF</b>	(none)	MUSICAL SYMBOL MULTI-STAFF BARLINES	This control character, when placed in a STAFF GROUP context, just before the END character, induces a span of all measure bars (1D100 to 1D107 and 1D1F6) in this staff group.

j) Fingering characters

Fingering is a semantic indication (with a huge presence in theoretical book, educational websites, etc.); it uses small letters and numbers, but because of different semantics, it should not be covered by style alone. This is consistent with the recommendation of the W3C for the use subscript and superscript: “when super and sub-scripts are to reflect semantic distinctions, it is easier to work with these meanings encoded in text rather than markup”.

Main characters for fingering are therefore available: 0 to 5 and p, i, m, a.

Note that the left hand of a guitar (0 to 4) is typically fingered at the left of the notehead (which means direction-zwj-chord), but right hand of a guitar (p, i, m, a) and piano fingerings (1 to 5) are usually stacked just above the staff (Pitch+5).

Code point	Glyph	Name	Comments
1xxE0	0	MUSICAL SYMBOL FINGERING 0	Looks identical to 0
1xxE1	1	MUSICAL SYMBOL FINGERING 1	Looks identical to 1
1xxE2	2	MUSICAL SYMBOL FINGERING 2	Looks identical to 2
1xxE3	3	MUSICAL SYMBOL FINGERING 3	Looks identical to 3
1xxE4	4	MUSICAL SYMBOL FINGERING 4	Looks identical to 4
1xxE5	5	MUSICAL SYMBOL FINGERING 5	Looks identical to 5
1xxE6	p	MUSICAL SYMBOL FINGERING p	Looks identical to p
1xxE7	i	MUSICAL SYMBOL FINGERING i	Looks identical to i
1xxE8	m	MUSICAL SYMBOL FINGERING m	Looks identical to m
1xxE9	a	MUSICAL SYMBOL FINGERING a	Looks identical to a

k) Character count

Range	Name	Count
<b>1D127</b>	Clefs	1
(1D128)	(unassigned)	(1)
<b>1D1E9-1D1EA</b>	Staff brackets	2
<b>1D1EB-1D1EC</b>	Chord diagrams	2
<b>1D1ED-1D1F5</b>	Lines	9
<b>1D1F6</b>	Bars	1
(1D1F6-1D1F9)	(unassigned)	(3)
<b>1D1FA-1D1FF</b>	Beams and slurs	6
<i>subtotal</i>		<i>21</i>
<b>1xx50-1xx81</b>	Combining beat characters	50
<b>1xx82-1xx99</b>	Combining staff characters	24
<b>1xx9A-1xxBF</b>	Combining pitch characters	38
<b>1xxC0-1xxCF</b>	Musical context control characters	16
<b>1xxD0-1xxDB</b>	Group control characters	12
<b>1xxDC-1xxDD</b>	Combining stem characters	2
<b>1xxDE-1xxDF</b>	Staff characters	2
<b>1xxE0-1xxE9</b>	Fingering characters	10
(1xxEA-1xxFF)	(unassigned)	(22)
<i>subtotal</i>		<i>154</i>
<b>Total</b>		<b>175</b>

l) Collation

It is of course not relevant to sort musigrams. However, as it is the case for emojis, an order in symbols (1D100 block) may be useful.

It is proposed to keep the code point order except when ranges have been split:

- 1D1F6 (left and right repeat sign) inserted between 1D107 and 1D108
- Volta signs 1D1ED to 1D1EF inserted before 1D10D (figure repetitions)
- Staff brackets 1D1E9 and 1D1EA inserted after 1D115
- Fretboards the 1D1EB and 1D1EC inserted after 1D11D
- Accidentals 266D to 266F inserted before 1D12A (these symbols existed prior to this proposal)
- Fanned beam 1D1FE and 1D1FF inserted after 1D174 (begin / end beam)
- Beam and slur symbols 1D1FA to 1D1FD inserted after 1D17A (beam and slur control characters)

## 5) Annex A: Example encoding

In order to assess whether a “normal” score (which is not really the intended use case) may be encoded with this proposal, a proof of concept has been made.

Nocturne Op. 9 No.2 by Chopin has been chosen, for a number of reasons:

- It is available in MusicXML format (encoded by Rodila, [here](#))
- It has been verified by peers
- It is of reasonable length (2 staves, 34 measures, 1259 notes and rests, 4 pages)
- It includes many features (ornaments, pedals, ottave, appoggiature, fingering indications, senza tempo measure, etc.).

The encoding from the MusicXML format has been automated with the following caveats:

- It bases on the xml but the code is made to comply with the linked pdf. Which means that, for instance, the  $\downarrow=60$  and  $p$  directions in measure 0, that do not appear in the pdf, have been removed manually
- A few tweaks, mainly, the context has been encoded manually, and the direction positioned manually (no positioning was available in the xml).

The encoded string is 23KB long, which compares to the PDF (115KB) and the xml (498KB, or 21KB in the mxl zipped format).