laboratories dak
scientific and technical undertakings

3102 Thunder Road
Alamogordo, New Mexico 88310

Douglas A. Kerr, P.E. (Ret.)
Principal
575-921-6795
doug@dougkerr.net

TECHNICAL REPORT

Subject: MusicXML—the duration element

Issue: 1 Reference: MusicXML_duration.doc

Date: December 22, 2020

Author: Douglas A. Kerr, P.E. (Ret.)

To: File

ABSTRACT AND INTRODUCTION

MusicXML is a language for transporting musical scores between musical notation programs. It describes the score of interest in terms of its graphical notational symbols and their arrangement on the page. It also conveys further details of how the score is to be "played".

An important (albeit optional) encoding of the MusicXML element for a note is the element <duration>. The MusicXML documentation does not clearly explain the intended meaning of <duration>. As a consequence, different notation program developers have taken drastically different views as to how the value of that element should be interpreted when reconstructing the score. As a result the universal interchange of scores via MusicXML is severely compromised.

This report describes this situation, and suggests an "understanding" of the meaning of the element <duration>.

1 ADMINISTRATIVE

1.1 Report not sponsored

This report, and the research underlying it, was not sponsored by, nor done at the behest of, any external organization.

1.2 Distribution

This report is not "published", but its distribution is not limited.

1.3 Disclaimer

The author has used his best professional skill and available information in the preparation of this report, but makes no representation that it is accurate or useful for any purpose. The reader who relies upon this report does so at his own risk, and the author cannot be responsible for any result not deemed satisfactory.

2 GENERAL

2.1 Reader background

It is hoped that the reader is generally familiar with musical notation, with the concept of music notation programs, and with the basics of the MusicXML language.

But, considerable background pertinent to this topic is given in the companion technical report, "MusicXML—background", by the same author. It is probably available where you got this. I commend it to the reader of this report.

2.2 Report not sponsored

This report, and the research underlying it, was not sponsored by, nor done at the behest of, any external organization.

2.3 Distribution

This report is not "published", but its distribution is not limited.

3 BACKGROUND

3.1 Time

We will be concerned with two kinds of "time":

1. *Musical time* is abstract, and quantifies the "flow" of the musical structure of the notation itself. It is typically denominated in measures, beats, and in some cases, fractions of a beat.
2. *Clock time* is the familiar "time", and may be denominated in hours, minutes, seconds, and in some cases, fractions of a second.

3.2 "Era" vs' "duration"

Often in this report we will hear of some event that starts at a certain time and continues for a certain time, such as the sounding of a note in play.

The length of time occupied by this event is referred to as its *duration*. That term does not properly include consideration of when the event commences.

The span of times occupied by this event is spoken of as its *era*. This term includes consideration of when the event commences as well as its duration (or, alternatively, when it commences and when it finishes).

A homey example of the use of these terms is this: A certain store is open today from 8:00 am through 7:00 pm. The *era* of its operation is 8:00 am through 7:00 pm. The *duration* of its operation is 11 hours.

3.3 The *notational duration* of a note

In conventional musical notation, we have different symbols representing, for example, half notes, quarter notes, eighth notes, and such. These differ in the length of *musical time* they normally occupy. In formal musical texts, the property that differs between these different kinds of note is called their *time value*. But it is quite common, and reasonable, for the property to be spoken of as their "duration". However, in this report we will encounter other meanings of the term *duration*. To avoid confusion, here I will generally speak of the *time value* of a note as its *notational duration*.

3.4 The musical era of a note

Each note occupies (as its "realm") a certain space in musical time, which I call its *musical era*. The length of that I call the *musical duration* of the note.

In the "normal" situation, the *musical duration* of a note (or a rest, actually) is the same as its *notational duration*. But later we will encounter some situations where that it not so.

In any case, keep in mind that properly *musical eras* are contiguous: the beginning of the musical era of a note or rest starts at the same instant as the end of the *musical era* of the previous note or rest.[1]

4 PLAY OF A SCORE

4.1 Introduction

As mentioned earlier, it is very common for a notation program to be able to "play" a score resident in the program. This is typically done by having the program, when play is started by the user, send a MIDI sequence over a MIDI interface (physical or virtual) to a synthesizer.

Nominally, the train of MIDI Note messages in this sequence starts the sounding of each note at a "clock time" that corresponds, in light of the chosen play tempo, to the start of the *musical era* of the note, and continues the sounding of each note

[1] In some case the contiguity of the musical eras is interrupted when we cross the boundary between measures, although that is anomalous.

for a length of time (the *play duration*[2]) that corresponds, in light of the play tempo, to the *notational duration* of the note.

But in many cases, the actual length of time that the note is sounded may differ a bit from the notational duration. A common example is that the scorist may arrange for the *play duration* to be less than the *musical duration* so as to bring about a small gap between the sounding of the individual notes to avoid a "fully legato" effect in play.

4.2 Underdata

As mentioned earlier, in a typical notation program, there are two layers of data held for the score being worked on. One describes its musical components: notes, rests, slur marks, and so forth, and the details of the physical layout of the notation on the score.

The second, which I call the "underdata", describes the details of how the score should be "played". This typically consists mainly of, for each note, the pitch and loudness at which the note is to be sounded, the point in musical time at which its sounding should commence, the and the point in musical time at which its sounding should cease.

In many notation programs, the underdata is called the "MIDI data". While that is not technically precise, it is apt, since the underdata is generally a direct precursor of a MIDI stream. In most case, the properties of the underdata are directly comparable to the parameters of the resulting MIDI stream.

One distinction is that the underdata is in terms of musical time, while the resulting MIDI stream is in terms of clock time. The two are related by the tempo at which the score is being played.

4.3 Control of *play duration*

Absent some intervention by the scorist, the *play start time* and *play duration* in the "underdata layer" should match the *notational duration* values of the note in the "notation layer".

But the scorist might not want it to work out that way. For example, it is quite common in notation programs for the scorist to set, for work on a particular score, the desired default *play duration*, often expressed as a percentage of the *notational duration* of the notes. Setting this to, perhaps, 90% avoids the rendering of all the notes, during play, in what would amount to a *full legato* style.

When speaking of this, it is common to call the musical time given by the *notational duration* of the notes as their "face value". Thus the scorist may set the

---

[2] The term "performance duration" is sometimes used, and is attractive for its generality. Nevertheless, here I use the term "play duration" for its conciseness.

program so that notes deposited are given (in the underdata) *play durations* that are "90% of face value".

Many notation programs allow the scorist, perhaps on a graphic portrayal of the underdata, to adjust the *play start time* and/or *play duration* of individual notes. This is typically done to emulate, when the score is played, the subtleties of times that a human performer would typically give the notes as rendered.

5 TRANSPORT VIA MusicXML—CONSERVATION OF THE UNDERDATA

The introduction to the MusicXML documentation reflects that the intent of MusicXML is to transport both the definition of the score in terms of notation and also what I have called the "underdata", information that would allow the receiving program to play the score. In fact, the extensive tutorial that illuminates the MusicXML language states that there are two "parts" to the language, one devoted to the notation and one, the "MIDI-compatible part", devoted to conveying how the score should sound when played.

The name for the latter part presumably comes from the fact that we can visualize the underdata as a "script" that directs the generation of a MIDI sequence to be sent to a synthesizer for play.

6 TWO PARTS OF A MusicXML FILE

The MusicXML documentation states that there are two part to the MusicXML language:

1. The Notation part (which des not have that as an actual name). This describes the actual visible notation.
2. The MIDI-compatible part (its actual name). This describes how the score should sound, if played. We assume that its name comes from the fact that what it conveys is, in a sense, a script for the generation of a MIDI stream (much as was discussed from the perspective of a notation program in section 4.2).

7 THE ELEMENT <duration>

7.1 In the syntax

The specification makes it clear that there must be one and only one <duration> element in (subordinate to) each <note> element.

7.2 Definition

The MusicXML documentation defines it thus (in part):

> The duration element is an integer that represents a note's duration. This is the intended duration vs. notated duration.

7.3 Meaning

This seems to most credibly suggest (so far) that "intended duration" means "the duration for which we expect the note to sound", recognizing that this may in fact not be the same as the *notational duration* of the note. (But there will be a further twist to this plot.)

The fact that, in the MusicXML tutorial, <duration> is presented as a creature of the MIDI-compatible part of MusicXML, rather than the Notation part, lends further credibility to the interpretation above.

7.4 The unit

The unit of <duration> is the *division*, which is a certain fraction of the ideal *musical duration* of a quarter note. That fraction is declared in a MusicXML file by the element <divisions>, which is usually placed to be global to the score. If we have:

```
<divisions>24</divisions>
```

then one *division* is 1/24 of the ideal *musical duration* of a quarter note.

What determines the value of <divisions> used in a certain MusicXML file? I'll discuss that after we see some of the things that are dominated in *divisions*. For now we can think of it as essentially arbitrary.

8 THE ELEMENT <type>

8.1 In the syntax

The element <type> is optional, and may appear only once in the <note>? element.

8.2 Definition

The MusicXML documentation defines the element <type> thus (in part):

> Type indicates the graphic note type. . .

It does this as an enumerative variable, whose value may be 'quarter", 'eighth', '16th', etc.

8.3 Raison d'etre

The MusicXML documentation introduces the <type> element by observing that the receiving program should be able to deduce the intended note symbols (and thus, notational direction) from the value of <duration>.

> I interrupt the story to note here that in that case there is in fact no opportunity for the "intended" duration of the note to be different from its notated duration.

The discussion continues to say that the (optional) <note element> gives the receiving program an explicit clue in that regard.

Does that in fact also open the door to the possibility that the "intended" duration of the note (given by <duration>) would be different from its notational value (given by <type>)? Seemingly.

8.4 Reality of practice

But we learn that, except in some rather esoteric cases, it is the practice in contemporary use of MusicXML to always have the values of <type> and <<duration> to be consistent. A consequence of that is the we don't, by making the value of <duration> less than consistent with the duration implied by <type>, arrange for the sounding duration of the notes to be, for example, less that the musical duration implied by <type>.

9 ENTER *ATTACK* AND *RELEASE*

9.1 Introduction

But that limitation is neatly overcome by the Music XML specification defining two optional *attributes* of the <note> element, *attack* and *release*. These are defined thus in the MusicXML documentation:

> The attack and release attributes are used to alter the starting and stopping time of the note from when it would otherwise occur based on the flow of durations - information that is specific to a performance[3]. They are expressed in terms of divisions, either positive or negative.

Again here there is an ambiguity, the meaning of "duration". But by triangulating among many passages in the MusicXML documentation, I conclude that "durations" here must mean *musical durations*.

9.2 Application

As an example, assume that the value of <divisions> is 120, a quarter note is being encoded, and we wish the note in the reconstructed score to sound for "90% of face" (and, for completeness, we wish the sounding of the note in play to commence at the nominal instant).

We then make <type> 'quarter' and <duration> '120' (both defining the same *notational duration*). And we give the <note> element the attribute *release* with value "-12". This causes the sounding of the note to cease 12/120 of the notational division of a quarter note before its nominal ending instant.

---

[3] I suspect that what is meant here would better have been said, "information that is specific to performance"

Suppose, in a more subtle definition of the sounding of the note, we want the sounding of the note to commence "5% of face late" and extend until "10% of face" from the end of the musical era of the note.. We again make <type> "quarter" and <duration> "120" (both defining the same notational duration). We give the <note> element the attribute *attack* with value "5", and the attribute *release* with value "-10".

10 CHOICE OF A VALUE OF <DIVISIONS>

What determines the value of <divisions> in MusicXML file. It is essentially arbitrary. But, if in fact we recognize <divisions> as given one of two definitions of the notational value of the note the value of <divisions> must be large enough that the notes to appear in the score can be properly represented. Thus, if the score involves 16th notes (either on their own or as a "dot"), then the value of <divisions> cannot be less than 4 (a 16th note has 1/4 the notational duration of a quarter note).

In fact, a passage of the MusicXML documentation suggests that the value of <divisions> be made no larger than is required as discussed just above.

This seems to rules out the thought that <duration> could be used to represent the musical duration of a note (and thus, basically, its play duration) when the play duration is different from the notational duration.

In fact, it is common for notation programs to uniformly use a value of "divisions" that matches the units used internally for musical time matters. In Overture, for example, the value of <divisions< is uniformly 480.

11 THE CURIOUS MATTER OF *NOTES INÉGALES*

11.1 Introduction

An interesting complication in seeking to interpret the <MusicXML documentation to discern the intent of the <duration> element is the matter of the *notes inégales* construction. The term is French, and means "unequal notes". It is discussed in the MusicXML tutorial.

The most common usage today of this principle is in the "swing eighths[4]" construction, widely used in jazz and other genres. There, in a series of eighth notes, the notes are considered in consecutive pairs. For each pair, in performance, the first note is sounded for greater than its nominal duration, and the second note for less than its nominal duration, the sum of the two performance durations nominally matching the sum of the notes' *notational durations*.

[4] Sometimes called "swung eighths"

11.2 Encoding in MusicXML

11.2.1 *An obvious way*

How might we encode into MusicXML such a structure? There is a very straightforward way. For the first note of the pair, we apply the attribute *release* to delay the ending of play, increasing the *play duration* of that note. For the second note of the pair, we apply the attribute *attack* to delay the *play start time* of that note to match, decreasing its *play duration*. (We may actually use a value of *release* for the first note, and adjust the value of *release* for the second note, to avoid the "full legato" effect.)

11.2.2 *Another way*

But the MusicXML tutorial suggests a different approach, this involving adjusting the values of <duration> for the two notes. It presumes the following:

- The value of <duration> defines the *musical duration* of the note, which may differ from its *notational duration*.
- The *play era* of the note in play would follow the *musical era* of the note.

Then, in the MusicXML encoding, we would give the first note a value of <duration> that is greater than equivalent to the *notational duration* of the note, and the second note a value of <duration> less than equivalent to the *notational duration* of that note, the two values of <duration> adding to the sum of the *notational durations* of the two notes.

Michael Good, the original developer of the MusicXML language and today its *de facto* "keeper", has suggested that this approach to the encoding of *notes inégales* is essentially obsolete.

Harking to that, we can use the technique described in section 11.2.1 to deal with the swing eighth structure (or even other more esoteric *notes inégales* structures.

And then we are free to adopt the outlook, suggested above, that <duration> and <notes> should represent equivalent amounts of musical time.

12 WHAT DO CURRENT NOTATION PROGRAMS DO?

12.1 Introduction

It is instructive to see what current notation programs do in this area. My most detailed investigations pertain to the programs Overture, MuseScore, and Finale. My findings are synopsized here.

In all cases in this section I will discuss the situation in which the <note> element contains both <type> and <duration> elements (almost universally the case for MusicXML code generated by modern notation programs). But see section 13 for information on the situation in which there is no >note> element.

12.2 Overture

12.2.1 *Version*

This was observed with Overture version 5.6.3-3.

12.2.2 *Receiving a MusicXML file*

a. Overture relies on the <type> element to select the note symbol and to establish its notational time.

b. Overture treats <duration> as defining the *play duration* of the note.

c. The attributes *attack* and *release* are not recognized.

12.2.3 *Generating a MusicXML file*

a. Overture makes the element <type> reflect the notational time of the note.

b. It makes the element <duration> reflect the *play duration* of the note. Departure of the play start time from the nominal is not reflected.

c. The attributes *attack* and *release* are not used.

12.3 MuseScore

12.3.1 *Version*

This was observed with MuseScore version 3.5.2.

12.3.2 *Receiving a MusicXML file*

a. MuseScore relies on the <type> element to select the note symbol and to establish its notational time.

b. It uses the value of <duration> to establish the musical time allotted to the note. (This among other things implies the musical time at which the following note commenced its reign.)

c. On Play, the sounding of the note commences at the start of the musical time "reign" of the note. The *play duration* is equal to the notational time of the note ("100% of face").

d. The attributes *attack* and *release* are not recognized.

e. If the values of <duration> do not match the values implied by <type>, MuseScore will generally give an error message upon loading of the MusicXML file, and the notation structure for the measure may be "strange", perhaps involving a gratuitous rest to make up for the seeming discrepancy.

In summary, a sensible result will only be attained if the values of <duration> match the notational durations implied by the values of <type>.

"Doctor, when I make <duration> inconsistent with <type>, my shoulder hurts."

"So don't make <duration> inconsistent with <type>."

12.3.3 *Generating a MusicXML file*

The <type> element comes from the note symbol. The element <duration> has a value corresponding to the notational time implied by the <type> elements. Departures of the established *play duration* or *play start time* from the nominal are not reflected in the encoding The attributes *attack* and *release* are not used.

12.4 Finale

12.4.1 *Version*

This was observed with Finale version 26.3.1

12.4.2 *Special role of Finale?*

We note that, especially given that Michael Good, the original developer of the MusicXML language, and still its *de facto* principal "keeper", is heavily involved in the development of Finale, and that in fact Finale first got its MusicXML capability by way of the plug-in "Dolet", provided by Recordare, the firm within which the MusicXML language was originally developed and promoted, we trend to think of Finale as the arbiter of taste in matters MusicXML.

12.4.3 *Receiving a MusicXML file*

a. Finale creates the note symbol based on the value of <type>, and accords the note an allotment of musical time to match.

b. If there is no *attack* and *release* attributes for a note, the *play duration* of the note will be its notational duration.

c. The values of the <duration> elements for the note of a measure are added up, and Finale treats the sum as the length (in musical time terms) of the measure. After what is deemed to be the length of the measure, play of the first note of the next measure commences forthwith. If the play of the notes in the first measure has not been completed, it continues apace as well.

   This behavior is truly bizarre.

   A sensible result will only in general result if, in the incoming MusicXML code, the value of <duration> matches the *notational duration* implied by the value of <type>.

d. Finale responds appropriately to the attack and release attributes. It is most useful to speak of this behavior in a case where the value of <duration> corresponds to the notational time implied by <type>. Then, if there is an attack attribute, the start of note sounding in play is shifted from its nominal

value by the value of attack. If there is a release attribute, the end of note sounding in play is shifted from its nominal value by the value of release.

12.4.4 *Generating a MusicXML file*

a. The element <type> states the symbol to be used for the note.

b. The element <duration> has a value corresponding to the notational time implied by the <type> attribute.

c. Departures of the established *play duration* or *play start time* from the nominal are not reflected in the encoding.

d. The attributes *attack* and *release* are not used.

13 WITH NO <type> ELEMENT

13.1 Introduction

I suspect it is today rare for a notation program to generate a MusicXML file ion which the <note> elements were not provided with a <type> element. But that possibility, perfectly legitimate according to the MusicXML documentation, plays a role in our effort to divine the overall intended scheme regarding note durations. Accordingly, I have made tests of the response to such a file of several notation programs.

13.2 Overture

Tests were also made with Overture (version 5.6.3-3) of a MusicXML file in which the note elements did not contain the optional <type> element.

a. Overture sets the note symbol (and thus the *notational duration*) based on the value of <duration>, quantized to an increment of a 64th note.

b. If the value of <duration> exactly corresponds to the *notational duration* of the chosen note type (no "rounding" was needed), Overture makes the *musical duration* equal to the notational duration. It makes the *play duration* of the note equal to 90% of the *notational duration* of that note. (This is presumably Overture's default default value to avoid a "fully legato" rendering.)

c. If the value of <duration> does not exactly correspond to the *notational duration* of the chosen note type ("rounding" was involved), Overture makes the *musical duration* equal to the *notational duration*. It makes the *play duration* of the note equal to the *notational duration* of that note.

Perhaps as another part of that "plan", when creating a MusicXML file, Overture makes the value of <duration> one less than the value that would correspond to the notational time of the note as given by the <type> element, which is always included.

13.3 MuseScore

Tests were made with MuseScore (version 3.5.2) of a MusicXML file in which the note elements did not contain the optional <type> element. The <duration> element values corresponded exactly to the notational durations of notes quarter, eighth, 16th, etc., down to 128th.

a. MuseScore makes the note symbol 'quarter' (that also dictates the *notational duration*) regardless of the value of <duration>.

b. The *musical duration* is made according to the value of <duration>

c. The *play duration* is made the same as the value of *notational duration*.

Suffice it to say that this can create a rather bizarre result on both notation and play fronts. The play era of the last note(s) in a measure can well extend into the realm of the next measure.

> "Doctor, when I leave out the<type> element, my shoulder hurts."
>
> "So don't leave out the <type> element."

13.4 Finale

Tests were made with Finale (version 26.3.1) of a MusicXML file in which the note elements did not contain the optional <type> element. The <duration> element values corresponded exactly to the notational durations of notes quarter, eighth, 16th, etc., down to 128th.

a. MuseScore makes the note symbol in accordance with the value of <duration>, seemingly to a resolution of a 2048th note.

b. The *musical durations* are made to essentially match the *notational duration*.

c. The *play durations* are made slightly less than the *musical durations*.

14 INTEROPERATION

Suffice it to say that this "diversity" (to be polite) in dealing with the element <duration> in MusicXML among various notation programs does not bode well for the successful interchange between different notation programs of scores that include *play durations* or *play start times* departing from the nominal.

15 MY RECOMMENDATION

I recommend the following doctrine be adopted in MusicXML-enabled application programs and the like.

15.1 Encoding in the MusicXML file

a. The *notational duration* ("time value") of the note should be encoded in the <type> element

b. The <duration> element should be given a value consistent with the *musical duration* implications of the <type> element.

c. If the established play duration or start time differ from the nominal, that should be encoded using the *attack* and/or *release* attributes, as needed.

15.2 Response to the MusicXML encoding for a note

a. If there is a <note> element

1. Make the note symbol to match

2. Give the note a *musical duration* with the time value of the note symbol.

3. Ignore the <duration> element.

b. If there is no <note> element

1. Make the note symbol (and thus the *notational duration*) that implied by the value of the <duration> element.

2. Give the note a *musical duration* (allotment of musical time) consistent with the *notational duration* of the note symbol.

c. in either case:

1. Make the base play start time as implied by the start of the note's *musical time era*.

2. Make the base play duration that suggested by the *notational duration* of the note symbol.

3. If there is an *attack* attribute and/or a *release* attribute, shift the *play start time* and/or *play end time* (from those of the base play era) according to the values of those attributes, respectively.

16 ISSUE RECORD

Issue 1, December 22, 2020. Initial issue.

-#-