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 PAPER

Subject: MusicXML—the *duration* element

Issue: 5 Reference: TP-MXML_duration-i05.doc

Date: May 18, 2021

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) component of the 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 different and incompatible views as to how the value of that element should be interpreted when generating a MusicXML file or reconstructing the score from such a file. As a result the universal interchange of scores via MusicXML is severely compromised.

This paper describes this situation, and suggests an "understanding" of the meaning of the element *duration*. Finally, a complete protocol for this area is recommended.

1 ADMINISTRATIVE

1.1 Paper not sponsored

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

1.2 Distribution

This paper 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 paper, but makes no representation that it is accurate or useful for any purpose. The reader who relies upon this paper does so at his own risk, and the author cannot be responsible for any result not deemed satisfactory.

2 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.

3 THE MusicXML LANGUAGE

3.1 Introduction

The MusicXML language was developed to provide a standard language for the transport of Musical scores between, most commonly, music notation programs, given that programs from different publishers typically use, natively, parochial formats for the representation of a score.

3.2 The "MusicXML documentation"

What I will refer to here as the "MusicXML documentation" defines the MusicXML language. It comprises in the main three components:

- The syntactic specification. This is done in two forms, each using a specialized language. It describes the various syntactic ingredients of the MusicXML language and how they may and must appear in the MusicXML file.
- The syntactic description. This is in textual form, and illuminates in a more accessible form the implications of the syntactic specification.
- The tutorial. This describes, with the use of sample score excerpts and the corresponding MusicXML code excerpts, the actual use of the MusicXML syntax for many common score situations.

3.3 Critique

The MusicXML documentation describes in exquisite detail and precision the lower-level structure of the MusicXML syntax.

But when we get to the higher syntactic level—perhaps what we could think of as the "grammar" of the language—the document often leaves a number of uncertainties. The result is that implementers of MusicXML in, for example, notation programs, must come to their own conclusions as to the fuller meaning of various parts of the language. The result, of course, is that the transport of a score from one notation program to another, via MusicXML is often significantly

imperfect in one matter or another. One of those matters is the use of the MusicXML element *duration*.

4 TECHNICAL DEFINITIONS

4.1 Durations of a note

4.1.1 *Definitions mine*

These definitions are mine, hopefully comporting with th "official" definitions where such exsit.

4.1.2 *The notational duration*

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 many 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 paper we will encounter other meanings of the term *duration*. To avoid confusion, and for consistency with other terms we will encounter, here I will generally speak of the *time value* of a note as its *notational duration*.

4.1.3 *The musical duration*

This is not often spoken of in musical theory, but for a reason you will see later I must introduce it here). This refers to the amount of musical time "owned" by a note (which is ordinarily, but not always, the same as the *notional duration*). We can think of it as the amount of musical time from the start of the "domain" of "this" note to the start of the "domain" of the next note.

4.1.4 *The play duration*

When we play the score, each note sounds for a clock time that corresponds, given the tempo of play, to a certain amount of musical time, called in much formal writing the *performance duration* of the note. But for convenience and consistency with the common less-formal terminology, here I will use the term *play duration*.

In the simplest situation, the *play duration* corresponds to the *notational duration*. But the scorist might want the *play duration* to be less than the *notational duration*. The usual reason is to avoid the play being "fully legato" when that is not the musical style that is desired.

The *play duration* of notes is often spoken of in terms of "% of face value", where "face value" alludes to the *notational duration* of the note.

5 MUSICXML ELEMENTS AND ATTRIBUTES

5.1 Introduction

The MusicXML language is a form of the XML language, and its details conform to the basic concepts and structures of XML.

5.2 Elements and attributes

The basic ingredient in an XML file (and thus of a MusicXML file) is the *element*. Elements are identified by XML "tags", which bracket the element content. The "opening" form of such a tag, in the classical formation, is: <box>, the "ending" form of such a tag, again in the classical formation, is: </box>. (There is also a short form.)

The properties of an object described by an element may be given in two different ways. One is by subordinate ("child") elements (sometimes spoken of in this context as *subelements*), thus:

```
<box>
   <size>medium
   </size>
</box>
```

The context of the element <box> is the element <size>. The content of the element <size> is medium.

Line breaks between components are optional. Thus this has the same meaning:

```
<box><size>medium</size></box>
```

but is harder to follow to the reader.

The other is by *attributes* of the element. They are embedded in the opening tag of the element to which they pertain, thus:

```
<box size="medium"></box>
```

The rationale for when a property of an element should be defined by a *child element* and when by an *attribute* is mysterious. Various authors espouse various schemas, but then often do not follow them scrupulously.

5.3 Typographical conventions

In this paper I will give the names of elements as if we were seeing their opening tag, thus: "The element <box>"; I will give the names of attributes underlined: "The attribute size."

6 CONTROL OF PLAY DURATION

Most modern notation programs allow the scorist various modes of control over the *play duration* of the notes. In some programs, the scorist can set a default duration

(usually as a fraction of the *notational time*) which deposited notes will initially be given.

Often there is a graphical display on which the *performance durations* of the individual notes can be seen as bars on a musical time scale. Typically the play durations of one or more notes can be changed here, either by numerical input or by "dragging" the end of the bar. In many cases, the data describing the *performance durations* of the notes is spoken of as "MIDI data", since it is in effect the precursor of the MIDI data that will be sent to a sound module during actual play.

In many cases, the scorist can not only control the *play duration* of the notes but as well their *play start time*. For example, for certain notes, the scorist may arrange for the "sounding" in play to commence not at the beginning of the "domain" of the note but perhaps what amounts to the time of a 32nd note later, this being done to attain a certain musical style effect.

7 TWO PARTS OF A MusicXML FILE

The MusicXML documentation states that there are two parts of a MusicXML file:

1. The Notation Part (which does not have that as an actual formal name, but I will call it by that "name"). This describes the actual visible notation.
2. The "MIDI-compatible Part" (its actual formal 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, by way of which a sound module will be caused to perform the notes.

Does the MIDI-compatible Part include provisions for representing, at the pleasure of the composer's notation program, the *play duration* (and possibly offset of the *performance start time*) of a note? Yes. But it is not always fully clear just how that is supposed to work.

8 The unit of musical time

The unit of musical time which applies to the element <duration>, and other creatures we will meet shortly, is the *division*, which is a stated fraction of the *notational duration* of a quarter note. That fraction is declared in a MusicXML file by the element <divisions>, which is usually placed at the beginning of a part (it could vary from part-to-part)[1]. If we have:

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

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

[1] In fact it can change in the middle of a part.

What determines the value of <divisions> used in a certain MusicXML file? For now we can think of it as essentially arbitrary. We will hear more about that in a little while.

9 THE ELEMENT <duration>

9.1 Introduction

The element <duration> seems at first to be clearly defined in the Documentation, but as we get into the broader matter a little deeper, we find that there are many unanswered questions. These are in fact what leads to the adoption of numerous inconsistent protocols for the use of the element <duration>.

9.2 In the syntax

In a properly-formed MusicXML file, each <note> element contains (among many other things) one and only one <duration> element.

9.3 Definition

The MusicXML documentation defines the <duration> element thus (the numbers in brackets are added to allow reference to the various parts of this passage in the subsequent discussion):

> Duration is a positive number specified in division units. [1] This is the intended duration vs. notated duration (for instance, swing eighths vs. even eighths, or differences in dotted notes in Baroque-era music). [2] Differences in duration specific to an interpretation or performance should use the note element's attack and release attributes.

9.4 Meaning

9.4.1 *Is it play duration?*

I conclude that what clause [2] refers to is the *play duration* of a note. Thus <duration> is explicitly not intended to indicate *play duration*.

9.4.2 *So what is "intended duration"*

It seems likely (in this definition) that "intended duration" (clause [1]) is just what is given as examples: an intended departure of the musical durations of the notes from the notated *time values*, not in the sense of a variation in play duration (as for a "more staccato" *vs*. "more legato" rendering), but rather in the sense of a shift in rhythm. But stay tuned.

10 THE ELEMENT <type>

10.1 In the syntax

The element <type> is optional, and may appear once (only) in a <note> element.

10.2 Definition

The MusicXML documentation defines the element <type> thus:

> The note-type type is used for the MusicXML type element and represents the graphic note type, from 1024th (shortest) to maxima (longest).

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

Note that once the note symbol is specified, the note *time value* directly follows.

10.3 But what about this

But we also have in the MusicXML Documentation this passage:

> Given the duration of a note and the divisions attribute, a program can usually infer the symbolic note type (e.g. quarter note, dotted-eighth note). However, it is much easier for notation programs if this is represented explicitly, rather than making the program infer the correct symbolic value. In some cases, the intended note duration does not match what is written, be it some of Bach's dotted notations, notes inégales, or jazz swing rhythms.
>
> The type element is used to indicate the symbolic note type, such as quarter, eighth, or 16th. . .

This suggests that <type> (if present) should provide, in an alternate way, exactly the same information as provided by <duration>: the symbol for the notes, and *ensuite* its *time value*. Or, said in reverse, <duration> should provide, in an alternate way, exactly the same information as provided by <type> (if present).

This latter is of course a little difficult to square with other "definitions" of <duration>.

11 BACK TO <DURATION>

11.1 Two ways to define the same thing?

How is the concept that <type> if present and <duration> define, in different ways, exactly the same information compatible with what I mention in section 9.4.2? It isn't. But that paradox has seemingly been resolved pragmatically (see section 10.3).

11.2 And in another part of the Documentation

In the MusicXML Tutorial (part of the MusicXML Documentation), <duration> is presented as a creature of the MIDI-compatible part of MusicXML, rather than the Notation part. This strongly suggests that <duration> should define the play duration of the note. This is fully in conflict with what I discuss in section 9.4.1.

11.3 The mystery of <duration>

Thus, the MusicXML Documentation seems to define three different meanings of <duration>:

1. A description of the note symbol (and *ensuite,* the *notational duration*) in numerical, rather than verbal, form.
2. A description of an alternate *musical duration* (different from the *notational duration*) of a note in a situation such as *notes inégales*.
3. A description of the intended *play duration* of the note.

It is no wonder that there is confusion in this area.

12 ENTER attack AND release

12.1 Introduction

Suppose we conclude that the real intended meaning of <duration> is either (1) or (2) as listed in section 11.3. How then can we convey the fact that the intended *play duration* of the note differs from its *notational duration*?

That need (and more) is neatly accommodated by two often-ignored optional *attributes* of the <note> element, attack and release. These are defined thus in the MusicXML documentation:

> The attack and release attributes[2] 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", Perhaps it means *notational duration*. Or *musical duration*.

---

[2] Do not confuse these *attributes* of an element with the MusicXML *element* <attributes>, which gives a number of properties of a part (spoken of as "attributes") in the form of child *elements*. The *element* <divisions> is in fact one such "attribute". Aargh!

[3] I suspect that what is meant here would better have been said, "information that is specific to performance", not "**a** performance".

12.2 Usage

As an example, where that ambiguity does not intrude, assume that (a) the value of <divisions> is 120, (b) a half note is being encoded, and (c) we wish the note in the reconstructed score to sound for "90% of face" (and, to be complete, we wish the sounding of the note in play to commence at the nominal start of the domain of the note).

We then make <type> = 'half' and <duration> = 240, both defining the same *notational duration*. And we give the <note> element the attribute release with value -24. This causes the sounding of the note to cease 24/120 of the nominal *musical duration* of a quarter note (10% of the nominal *musical duration* of the half note) before the end of the *musical duration* of the half note.

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 duration* of the note (for a net *play duration* of 85% of face). We again make <type> 'half' and <duration> 240 (again both defining the same *notational duration*). We give the <note> element the attribute attack with value 12, and the attribute release with value -24.

13 SO, WHAT IS THE INTENDED MEANING OF <duration>?

After much pondering of all the above and considering clarifying observations made on various forums by Michael Good, the original developer of the MusicXML language and today its *de facto* "keeper", the author concludes that the intended meaning of <duration> is as (1) in section 11.3:

> A description of the note symbol in numerical, rather than verbal, form.

Does that mean that <type< >, if present, and <duration> tell exactly the same thing? Yes. How odd. Yes.

14 THEN WHAT ABOUT NOTES INÉGALES?

Michael Good, in a forum devoted to MusicXML development, has suggested that this approach to the encoding of *notes inégales* intimated by the MusicXML tutorial (as described here in section 11.3, item 2), is essentially obsolete, suggesting that this should be done directly by way of the attack and release attributes. That works for me.

15 THE <duration> ELEMENT AND *MUSICAL DURATION*

Another interpretation of the meaning of <duration> is found in the protocols used by certain notation programs. They treat <duration> as defining the *musical duration* of the note (see section 4.1.3). But isn't the *musical duration* the same as the *notational duration*? Ordinarily, yes. But one exception is in the case of *notes*

inégales, as described in section 10.3. But, as we see in section 14, that is in actual practice perhaps not a problem.

16 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 giving one of two definitions of the *notational duration* 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" on an eighth note), 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 rule 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* (since such a situation would almost inevitably require a finer precision than suggested just above). This is in fact consistent with the conclusion I have come to above.

But iff attack and release are used to define the starting and ending time of play (and thus the *play duration*), finer precision than suggested just above would be needed.

Some notation programs 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. That is well suited to using <duration> to represent *play duration* to a fine resolution. Except that I don't think it should be used for that.

17 WHAT DO CURRENT NOTATION PROGRAMS DO?

17.1 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).

17.2 Overture

17.2.1 *Version*

This was observed with Overture version 5.6.3-3.

17.2.2 *Receiving a MusicXML file*

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

b. Overture treats <duration> as defining the *play duration* of the note (the performance start time being assumed to be as nominal).

c. The attributes attack and release are not recognized.

17.2.3 *Generating a MusicXML file*

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

b. It makes the element <duration> reflect the *performance 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.

17.3 MuseScore

17.3.1 *Version*

This was observed with MuseScore version 3.5.2.

17.3.2 *Receiving a MusicXML file*

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

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

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

d. The attributes attack and release are not recognized.

e. The total of the <duration> values is taken my MuseScore to be the intended musical time length of the measure. If that does not match the length given by the time signature, MuseScore will give an error message upon loading of the MusicXML file. If we proceed anyway, the notation structure for the measure may be "strange", perhaps involving a gratuitous rest to make up for the seeming discrepancy.

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

17.3.3 *Generating a MusicXML file*

a. The optional element <type> is provided in a <note> element. It indicates (as, for example, 'quarter') the symbol of the note and thus its *notational duration*.

b. The element <duration> gives, in numerical form, in connection with the value of <divisions>, the symbol for the notes consistent with that defined by the element <type>.

c. Departures of the intended play duration (or of the play start or end times) from the nominal are not encoded.

d. The attributes attack and release are not used.

17.4 Transport of scores from Overture to MuseScore via MusicXML

Transport of score from Overture to MuseScore via MusicXML is at this time of special interest to the author and a colleague.

If, as is typical in Overture scores, there are notes whose play duration is set to less than 100% of face" (likely all notes are of this kind), the values of <duration> are less than represent "100% of face".

Suppose, to make the numbers more handy, that in some passage in the Overture score all the notes are quarter notes and all are set for a play duration of "75% of face".

When a MusicXML file generated by Overture is taken into MuseScore, it treats the values of <duration> as meaning the musical duration to be allocated to the notes. Thus in the reconstructed score:

- All the notes are given the quarter note symbol,
- On the "music time timeline", the start of the domains of the notes fall at intervals of 75% of the time of a quarter note (the time of a dotted eight note. That is, they occur at 4/3 the intended rate.
- Bit MuseScore sounds each note for its *notational duration* (in this case, the full time of a quarter note.

This is a truly bizarre result.

And MuseScore adds up all the values of the <duration> elements and sets the working length of the measure to match that total (not the length defined in the MusicXML code as the time signature of the measure). As a corollary, when the MusicXML file is first offered to MuseScore for import, this discrepancy, found during the "intake validation", is reported as a matter of "corruption" of the file.

As to this area, a sensible result in MuseScore will only result if, before generating the MusicXML file from the score in Overture, all note performance durations are set to "100% of face" and any offsets of play start times expunged. Then, the

values of <duration> will match the *notational durations*, MuseScore's intake audit will not produce error messages, and MuseScore will produce a perfectly sensible score.

Of course in some cases these timing subtleties were achieved either by initially recording the work as performed by a human player on a MIDI-connected keyboard, or were tediously crafted by the scorist. In either case, when this score is carried across the River Styx in MusicXML form, all those subtleties must be jettisoned. This is just how it is.

17.5 Finale

17.5.1 *Version*

This was observed with Finale version 26.3.1

17.5.2 *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.

17.5.3 *Receiving a MusicXML file*

a. Finale creates the note symbol based on the value of <type>, and makes the *musical duration* to match.

b. If there are no attack and/or 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 of itself is truly bizarre.[4] But it would be moot for a MusicXML file in which, for each note, the value of <duration> matches the *notational duration* implied by the value of <type> (in accordance with the interpretation of the MusicXML specification I recommend).

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

[4] It is somewhat parallel to the behavior of MuseScore.

attack attribute, the start of note sounding in play is shifted the time it would otherwise have by the value of attack. If there is a release attribute, the end of the note sounding in play is shifted from when it would otherwise be by the value of release.

17.5.4 *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.

18 WITH NO <type> ELEMENT

18.1 Introduction

I suspect it is today rare for a notation program to generate a MusicXML file in 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.

18.2 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> (interpreted in connection with the value of <divisions>, 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 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.

18.3 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 duration* of the last note(s) in a measure can well extend into the realm of the next measure.

18.4 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*.

19 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.

20 MY RECOMMENDATION

20.1 Introduction

I recommend the following protocol be adopted in MusicXML-enabled application programs:

20.2 Encoding into MusicXML

a. The value of <divisions> shall be chosen large enough to accommodate the needed precision of the element <duration> and the attributes attack and release, if used.

b. The <note> element shall be provided with the subelements <type> and <duration>.

c. The <type> element shall describe the symbol of the note (as an enumerative text variable, such as "quarter" or "whole"). This also, in accordance with accepted musical practice, defines the *time value* of the note.

d. The <duration> element shall (interpreted in connection with the value of <divisions>) define the time value of the note (and thus its symbol).

   Comment: Yes, this seems redundant. This situation is a result of the convoluted evolution of this aspect of MusicXML encoding.

e. If it is the intent that the play duration of the note is different than that implied by the time value of the note, and/or that the start of the sounding of the note is displaced from the time instant implied by the location of the note in the measure, then the attributes (of the <note> element) attack and or release shall be included.

f. If the intent is that the starting play time differs from the "nominal", then the attribute attack shall be provided. It specifies the offset of the starting play time from nominal as a signed value in units of divisions.

g. If the intent is that the ending play time differs from the "nominal", then the attribute attack shall be provided. It specifies the offset of the ending play time from nominal as a signed value in units of divisions.

   Note that if the situation is a simple departure of the play duration from nominal, that is treated here as a displacement of the ending play time.

20.3 Response to an imported MusicXML file

a. If the optional element <type> is provided in a <note> element. It indicates (as, for example, "quarter") the symbol of the note and thus its time value.

b. If the optional element <type> is provided, the mandatory element <duration> is ignored.

   Comment: This oddity is a result of the convoluted evolution of this aspect of MusicXML encoding.

c. If the optional element <type> is not provided, the mandatory element <duration> is to be taken as giving (in numerical form, interpreted in connection with the value of <divisions>) the time value of the note and thus its symbol.

d. The “nominal” play duration is to be as implied by the time value of the note. The nominal starting play time is at the instant implied by the position of the note in the measure.

e. If the optional attribute attack is provided, it defines, in units of *division*, the displacement (in either direction–its value is signed) of the starting play time from its nominal value.

f. If the optional attribute release is provided, it defines, in units of *division*, the displacement (in either direction–its value is signed) of the ending play time from its nominal value.

21 ISSUE RECORD

Issue 5, May 18, 2021 (this issue). Title changed from “Technical Report” to “Technical Paper”. Issue number sequences continued. Extensive revisions.

Issue 4, May 12, 2021. Editorial changes.

Issue 3, January 30, 2021. Editorial changes.

Issue 2, January 4, 2021. Editorial changes.

Issue 1, December 22, 2020. Initial issue (as Technical Report).

-#-