

7 NIFFML: An XML Implementation of the Notation Interchange File Format

Gerd Castan

Höhbergstrasse 16

70327 Stuttgart

Germany

Gerd.Castan@web.de

www.s-line.de/homepages/gerd_castan/compmus/index.html

Abstract

XML provides a useful architecture for resolving ambiguities, contradictions, and redundancies of data content inherent in notation software and embedded in the Notation Interchange File Format (NIFF). It thus offers a useful way to extend the usefulness of NIFF in the interchange of data between notation programs.

COMPUTING IN MUSICOLOGY 12 (1999-2000), 103–112.

7.1 NIFF and XML: A Simple Mapping of the Data Model

¹ A recent Web search showed current use of NIFF in the LIME notation program and in the *PianoScan*, *MIDIScan*, and *SharpEye* recognition programs.

NIFF (the Notation Interchange File Format) is the most comprehensive music-notation interchange format currently in use. Even if only a few music-notation programs use it,¹ there is no other standard format that is actually used as an interchange format. NIFF is built on RIFF (the Resource Interchange File Format), which was developed for MS *Windows*. RIFF suits XML well because it is also hierarchical.

An important difference between RIFF and XML is that elements within a RIFF description can have children, whereas those in an XML description cannot. Figure 1 shows the example (the first few bars of Chopin's Étude Op. 10, No. 3, in E Major) on which the subsequent file samples are based.

Figure 1. Chopin: Étude in E Major, Op. 10, No. 3.



Figure 2 shows an XML representation of a NIFF description of the start of the above example.

Figure 2. An XML representation of a NIFF description (NIFFML) of the pickup and first beat of the Chopin étude shown in Figure 1 in time-slice order (reading notes on the same beat from top to bottom), with full typographical details. The len information is obsolete in XML and is used only for debugging. The NIFF binary file and the completion of this encoding are posted on the Web site complementing this issue [see reverse title-page for details].

```
<?xml version="1.0" encoding="utf-8"?>
<NIFF len="2920">
  <setp len="640">
    <clt len="496" >
      <niffChkIentabEntry
        chunkName="acd1"
        offsetOfFirstTag="1" />
    </clt >
    <nnfo len="14"
      ... [engraving/hardware information omitted]
      absoluteUnits="20"
      midiClocksPerQuarter="16384" />
  </setp >
</NIFF >
```

```

<prts len="26">
  <part
    len="14"
    partID="0"
    name="0"
    abbreviation="0" 47
    numberOfStaves="1"
    ... [MIDI information omitted] />
</prts>
<stbl len="56" >
  ... [font information omitted]
</stbl>
<fnst len="4">
</fnst>
</setp>

<data len="2260">
  <page len="2248">
    <pghd len="8"
      niffWidth="12290"
      niffHeight="4354" />
    <syst len="2220">
      <syhd len="0" />
      <staf len="1996">
        <sthd len="0" />
        <tmsl len="11"
          type="tsEvent"
          startTimeN="0"
          startTimeD="4"
          horizontal="260"
          vertical="2100" />
        <clef len="3"
          shape="clefshapeGclef"
          staffStep="2"
          octaveNumber="clefoctNoNumber" />
        <keys len="1"
          standardCode="4" />
        <time len="2"
          topNumber="2"
          bottomNumber="4" />
        <rest len="16"
          shape="restEighth"
          staffStep="4"
          durationN="1"
          durationD="8"
          niffPartID="8194"
          niffVoiceID="12034" />
        <tmsl len="11"
          type="tsEvent"
          startTimeN="1"
          startTimeD="8"
          horizontal="260"
          vertical="2612" />
        <rest len="16"
          shape="restEighth"
          staffStep="4"
          durationN="1"
          durationD="8"

```

```

niffPartID="8194"
niffVoiceID="12034" />
<tmsl len="11"
  type="tsEvent"
  startTimeN="1"
  startTimeD="4"
  horizontal="260"
  vertical="3122" />
<rest len="16"
  shape="restEighth"
  staffStep="4"
  durationN="1"
  durationD="8"
  niffPartID="8194"
  niffVoiceID="12034" />
<tmsl len="11"
  type="tsEvent"
  startTimeN="3"
  startTimeD="8"
  horizontal="260"
  vertical="3632" />
<stem len="10"
  horizontal="22"
  vertical="logplaceVStemside"
  proximity="logplaceProxDefault"
  niffNumberOfFlags="27" />
<note len="14"
  shape="noteshapeFilled"
  staffStep="-3" [B3]
  durationN="1"
  durationD="8"
  niffPartID="8194"
  niffVoiceID="12034" />
<rest len="16"
  shape="restEighth"
  staffStep="4"
  durationN="1"
  durationD="8"
  niffPartID="8194"
  niffVoiceID="12034" />
<tmsl len="11"
  type="tsEvent"
  startTimeN="2"
  startTimeD="4"
  horizontal="260"
  vertical="4302" />
<barl len="4"
  type="bartypeThin"
  extendsTo="barextThruSpace"
  numberOfStaves="1" />
<tmsl len="5"
  type="tsMeasureStart"
  startTimeN="2"
  startTimeD="4" />
<stem len="6"
  horizontal="22"
  vertical="logplaceVStemside"
  proximity="logplaceProxDefault" />

```

```

<beam len="10"
  beamPartsToLeft="0"
  beamPartsToRight="1"
  niffNumberOfNodes="7170"
  niffID="4610" />
<note len="14"
  shape="noteshapeFilled"
  staffStep="0" [E4]
  durationN="1"
  durationD="8"
  niffPartID="8194"
  niffVoiceID="12034" />
<stem len="6"
  horizontal="22"
  vertical="logplaceVStemside"
  proximity="logplaceProxDefault" />
<beam len="10"
  beamPartsToLeft="0"
  beamPartsToRight="2"
  niffNumberOfNodes="7170"
  niffID="4610" />
<note len="14"
  shape="noteshapeFilled"
  staffStep="-5" [G#3]
  durationN="1"
  durationD="16"
  niffPartID="8194"
  niffVoiceID="12034" />
<tmsl len="11"
  type="tsEvent"
  startTimeN="1"
  startTimeD="16"
  horizontal="260"
  vertical="4700" />
<stem len="6"
  horizontal="22"
  vertical="logplaceVStemside"
  proximity="logplaceProxDefault" />
<beam len="6"
  beamPartsToLeft="2"
  beamPartsToRight="2"
  niffID="4610" />
<note len="14"
  shape="noteshapeFilled"
  staffStep="-3" [B3]
  durationN="1"
  durationD="16"
  niffPartID="8194"
  niffVoiceID="12034" />
<tmsl len="11"
  type="tsEvent"
  startTimeN="1"
  startTimeD="8"
  horizontal="260"
  vertical="5096" />
<stem len="6"
  horizontal="22"
  vertical="logplaceVStemside"

```

```

    proximity="logplaceProxDefault" />
<beam len="6"
  beamPartsToLeft="1"
  beamPartsToRight="2"
  niffID="4610" />
<note len="14"
  shape="noteshapeFilled"
  staffStep="-1" [D#4]
  durationN="1"
  durationD="16"
  niffPartID="8194"
  niffVoiceID="12034" />
<stem len="6"
  horizontal="22"
  vertical="logplaceVStemside"
  proximity="logplaceProxDefault" />
<beam len="6"
  beamPartsToLeft="2"
  beamPartsToRight="2"
  niffID="4610" />
<note len="14"
  shape="noteshapeFilled"
  staffStep="-5" [G#3]
  durationN="1"
  durationD="16"
  niffPartID="8194"
  niffVoiceID="12034" />
<tmsl len="11"
  type="tsEvent"
  startTimeN="3"
  startTimeD="16"
  horizontal="260"
  vertical="5492" />
<stem len="6"
  horizontal="22"
  vertical="logplaceVStemside"
  proximity="logplaceProxDefault" />
<beam len="6"
  beamPartsToLeft="2"
  beamPartsToRight="0"
  niffID="4610" />
<note len="14"
  shape="noteshapeFilled"
  staffStep="0" [E4]
  durationN="1"
  durationD="16"
  niffPartID="8194"
  niffVoiceID="12034" />
<stem len="6"
  horizontal="22"
  vertical="logplaceVStemside"
  proximity="logplaceProxDefault" />
<beam len="6"
  beamPartsToLeft="2"
  beamPartsToRight="0"
  niffID="4610" />

```

```

        <note len="14"
            shape="noteshapeFilled"
            staffStep="-3"
            durationN="1"
            durationD="16"
            niffPartID="8194"
            niffVoiceID="12034" />
        ..
    </staf>
</syst>
</page>
</data>
</NIFF>

```

[B3]

7.1.1 Attributes

It should be stressed that nothing new was invented in Figure 2. It is just a different representation of the NIFF data-model. We call it NIFFML (NIFF Markup Language). However, there are two big benefits in this change of the representation:

- We can print out an example and talk about bugs and issues of the data-model itself.
- We have a formal way to describe and validate the data-model (using a DTD or schema).

A music-notation format needs flexibility and extensibility. To this end, NIFF uses three additional representation mechanisms:

- a fixed binary data structure that is defined for each chunk;
- a length modification of this structure (defined in the `clt` chunk in the beginning) to make it possible to extend the structure in future versions of NIFF; and
- arbitrary tags (the *name tag* in NIFF corresponds to an *attribute* in XML, not to a *tag* in XML); see NIFF documentation for allowable tags.

The binary data structure makes it time-consuming to read object-oriented languages such as Java. The arbitrary tags make it almost impossible for two music-notation programs to talk about the same thing when they use tags in NIFF (the latter is an issue of the NIFF *model*, not the NIFF *representation*).

XML, in comparison, uses only one mechanism; it is inherently flexible, extensible, and self-describing. Suppose that coloration were a desired

attribute of the output. We could add a color specifier for notes in an extended version of the format, e.g. as

```
<note ... color= "red" />
```

For the reading program this is just one more attribute to accommodate. If the reading program has no implementation of color printing, it can ignore the attribute.

7.2 NIFF vs. a Music-Notation Format in XML

Anything that can be expressed in NIFF can be expressed in XML and (with a little more effort) vice versa. It is appropriate to separate the NIFF representation (RIFF and chunk content) from the NIFF data-model.

7.2.1 The NIFF Representation

A NIFF/RIFF approach is more verbose than a pure XML approach in something as complex as musical notation. It is possible that only a few commercial companies have supported NIFF because the market for musical notation is relatively small, while the effort required for implementation is significant.

The value of steering NIFF data towards XML implementations is that the latter adds external schemas and validation. It therefore improves consistency and increases the likelihood of smooth operation, particularly for large projects.

7.2.2 The NIFF Data Model

The NIFF data-model is an invaluable and comprehensive resource of information about musical notation. It should certainly be considered before inventing another format. The only weakness in NIFF is its use of arbitrary tags. This unnecessary flexibility makes it impossible for two music-notation programs to understand each other at the tag level regardless of the fact that they may be able to communicate at other levels. A few suggestions for tag uses for some chunks give a hint of what tags one might expect.²

²The first tag in a chunk may start at an *odd* or *even* offset, whatever the `clt` table defines. All consecutive tags are aligned to *even* offsets.

7.3 Towards an Object-Oriented Notation Interface

One issue which inheres in this data-model is not NIFF-specific. The NIFF 6a specification notes that there are both page-oriented and non-page-oriented music-notation programs. Each way of modeling the data raises issues that are best resolved by the other way.³ Most music-notation formats support only one approach (generally the page-orientation mode). NIFF supports both. Our goal is to work out the issues and try to find a way to resolve them together.

To see the difference, we describe two types of user interface that express the respective models. In both cases we edit a score then print out the score and single parts of it.

7.3.1 The Page-Oriented Approach

First we consider a simple page-oriented user interface that always has the whole score on the screen. The user may define line- and system-breaks or have them implemented automatically. The layout is made by the program for the whole score. The data is stored as it is seen on the screen.

Some programs use the line-breaks of the *score* to print the *parts* without reassigning the horizontal position of the symbols. With this approach the parts take up more horizontal space on the sheet than necessary, which ultimately requires an unnecessarily large number of pages. It also degrades legibility.

7.3.2 The Part-Oriented Approach

Other notation programs depend on a linear input mode; as in a spreadsheet, the input is laid out on one infinitely wide page. After the input is completed, the score and each of the parts is formatted for printout. The difficulty of this solution is that it is difficult to maintain the data after formatting. If there are errata (and we can be sure there will be), we have to go back to the linear mode and do the formatting job again, or else the correction has to be made both in the score and in the part as separate options.⁴

³ See also Good's "time-wise" and "part-wise" approaches (following article).

⁴ It is a common experience of musicians using computer-typeset music to encounter errata corrected in the score but not in the parts or vice versa.

7.4 Conclusion

XML can solve these organizational issues because it enables each item of data to be stored only once, while being called from different routines and multiple approaches to processing. XML promotes above all the separation of content from style, which underlies these layout conflicts and maintenance discrepancies.

URLs

NIFF 6a (1996):

neume.sourceforge.net/niff/ [also in HTML format; 1998].

NIFF download site:

esi23.ESI.UMontreal.CA:80/~belkina/N/

RIFF:

www.ora.com/centers/gff/formats/micriff/index.htm

Web site for notation representation schemes:

www.s-line.de/homepages/gerd_castan/compmus/notationformats_e.html
[also in German at .../notationformats_d.html]