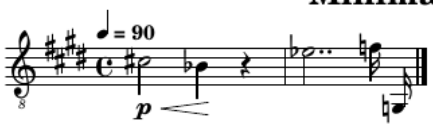
MusicFormats API guide

https://github.com/jacques-menu/musicformats

vv0.9.70 – August 6, 2023

Jacques Menu

Minimal score




```
S_lpsrScore translateMsrToLpsr (
  const S_msrScore&           originalMsrScore,
  const S_msrOahGroup&        msrOpts,
  const S_lpsrOahGroup&       lpsrOpts,
  mfPassIDKind                passIDKind,
  std::string                 passDescription,
  const S_mfcMultiComponent&  multiComponent)
{
  // ... ... ...

  // create an msr2lpsrTranslator
  msr2lpsrTranslator
    translator (
      originalMsrScore);

  // build the LPSR score
  S_lpsrScore
    resultingLpsr =
      translator.translateMsrToLpsr (
        originalMsrScore,
        multiComponent);

  // ... ... ...

  return resultingLpsr;
}
```

Contents

List of Figures

Chapter 1

Acknowledgements

Many thanks to Dominique Fober, the designer and maintainer of the `libmusicxml2` library. This author would not have attempted to work on a MusicXML to LilyPond converter without his work being already available.

In particular, the conversion of MusicXML data to a tree is extremely well done directly from the MusicXML DTD, and that was a necessary step to produce LilyPond code. Dominique also provided a nice way to browse this tree with a two-phase visitor design pattern, which this author uses extensively in his own code. The interested reader can find information about that in `libmusicxml2.pdf`, and more technical details in `MusicFormatsMaintainanceGuide.pdf`.

`xml2ly` and some of the specific examples presented in this document started as this author's contribution to `libmusicxml2`, and was later moved to a separate GitHub repository for practical reasons.

Part I

MusicFormats API principles

Chapter 2

API principles

Part II

The MXSR API

Chapter 3

Creating scores with the MXSR API

Part III

The MSR API

Chapter 4

Creating scores with the MSR API

Part IV

The LPSR API

Chapter 5

Creating scores with the LPSR API

Part V

The BSR API

Chapter 6

Creating scores with the BSR API

Part VI

Indexes

Main index