

# MusicFormats API guide

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

v0.9.64 – June 9, 2022

Jacques Menu

## Minimal score



```
1 S_lpsrScore translateMsrToLpsr (  
2     S_msrScore          originalMsrScore,  
3     S_msrOahGroup       msrOpts,  
4     S_lpsrOahGroup      lpsrOpts,  
5     string              passNumber,  
6     string              passDescription,  
7     S_mfcMultiComponent multiComponent)  
8 {  
9     // ... ..  
10  
11     // create an msr2lpsrTranslator  
12     msr2lpsrTranslator  
13         translator (  
14             originalMsrScore);  
15  
16     // build the LPSR score  
17     S_lpsrScore  
18         resultingLpsr =  
19         translator.translateMsrToLpsr (  
20             originalMsrScore,  
21             multiComponent);  
22  
23     // ... ..  
24  
25     return resultingLpsr;  
26 }
```

# Contents

1	<a href="#">Acknowledgements</a> . . . . .	3
<b>I</b>	<b><a href="#">MusicFormats API principles</a></b>	<b>4</b>
2	<a href="#">API principles</a> . . . . .	5
<b>II</b>	<b><a href="#">The MXSR API</a></b>	<b>6</b>
3	<a href="#">Creating scores with the MXSR API.</a> . . . . .	7
<b>III</b>	<b><a href="#">The MSR API</a></b>	<b>8</b>
4	<a href="#">Creating scores with the MSR API</a> . . . . .	9
<b>IV</b>	<b><a href="#">The LPSR API</a></b>	<b>10</b>
5	<a href="#">Creating scores with the LPSR API</a> . . . . .	11
<b>V</b>	<b><a href="#">The BSR API</a></b>	<b>12</b>
6	<a href="#">Creating scores with the BSR API.</a> . . . . .	13
<b>VI</b>	<b><a href="#">Indexes</a></b>	<b>14</b>

# 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 [MusicFormatsMaintenanceGuide.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

<b>A</b>		
API .....		<a href="#">1</a>
<b>D</b>		
Dominique Fober .....		<a href="#">3</a>
DTD .....		<a href="#">3</a>
<b>L</b>		
libmusicxml2 .....		<a href="#">3</a>
LilyPond .....		<a href="#">3</a>
<b>M</b>		
MusicFormats .....		<a href="#">1</a>
MusicXML .....		<a href="#">3</a>