Encoding formats

The primary notation encoding format used with Verovio is MEI; however, Verovio supports conversion from a number of other formats, including MusicXML. In this tutorial we will look at how we can get Verovio to convert a compressed MusicXML file to MEI.

Saving as MEI

When loading a MusicXML file into Verovio, it converts this internally into MEI, which we will be able to export as MEI.

To do this, and to make our lives easier, we will use a JavaScript library that helps us save a file. In the `<head>` section of your file, add the following `<script>` tag:

HTML / JAVASCRIPT

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/FileSaver.js/2.0.0/FileSaver.min.js"></script>
```

To download the MEI file, we will add a button to our page that will trigger a save of the MEI content from Verovio. Just like in previous tutorials, add a “click” handler for a button:

JAVASCRIPT

```
document.getElementById("saveMEI").addEventListener("click", (event) => {
    let meiContent = tk.getMEI();
    var myBlob = new Blob([meiContent], {type: "application/xml"});
    saveAs(myBlob, "meifile.mei");
});
```

That is, we get the button element (`id="saveMEI"`), and then tell the button what to do when it is clicked. To get the MEI output we can use the `getMEI()` method on the toolkit. This will return a formatted string containing the MEI XML output.

Then we do a few JavaScript things to get the download to work. First we create a new “Blob”, which is just a wrapper around some arbitrary data. Then we call the `saveAs` function from the FileSaver.js library we loaded earlier.

Compressed MusicXML

You may not be aware of it, but there are actually two forms of MusicXML files! Typically, those that end with `.xml` or `.musicxml` are “plain” XML files, and we can load them directly. Here we are going to load a “compressed” MusicXML file, normally ending with a `.mxl` extension. These files are just ZIP files, but have a fixed file-and-folder structure within them. Verovio supports loading these types of files as well, but with a bit of special handling needed.

To display this we follow the same methods as loading previous files, except for two main differences:

- We use `response.arrayBuffer()` instead of `response.text()` to read the initial response;
- We use Verovio’s `loadZipDataBuffer` toolkit method, instead of the regular `loadData` method.

Wrapping up

With Verovio you can easily convert MusicXML files, in both compressed and uncompressed formats, to MEI. There are a number of other formats that Verovio supports as well, but some need to be specially enabled if you wish to use them.

Check out the chapter on Input formats in the Verovio book for more details.

Full example

Open this example in a new window.

HTML / JAVASCRIPT

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width">
        <title>Convert compressed MusicXML</title>
        <script src="https://www.verovio.org/javascript/develop/verovio-toolkit-wasm.js" defer></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/FileSaver.js/2.0.0/FileSaver.min.js"></script
>
    </head>
    <body>
        <button id="saveMEI">Save my MEI!</button>
        <div id="notation"></div>
        <script>
            /**
                We need to wait for the whole page to load before we try to
                work with Verovio.
            **/
            document.addEventListener("DOMContentLoaded", (event) => {
                verovio.module.onRuntimeInitialized = function () {
                    // This line initializes the Verovio toolkit
                    const tk = new verovio.toolkit();

                    document.getElementById("saveMEI").addEventListener("click", (event) => {
                        let meiContent = tk.getMEI();
                        var myBlob = new Blob([meiContent], {type: "application/xml"});
                        saveAs(myBlob, "meifile.mei");
                    });

                    fetch("https://www.verovio.org/examples/musicxml/Schubert_Staendchen_D.923.mxl")
                        .then((response) => response.arrayBuffer())
                        .then((mxlData) => {
                            tk.loadZipDataBuffer(mxlData);
                            let svg = tk.renderToSVG(1);
                            document.getElementById("notation").innerHTML = svg;
                        }).catch(e => {
                            console.log("An error occurred when loading the file!");
                            console.log(e);
                        });
                }
            });
        </script>
    </body>
</html>
```