#DLFteach • #DLFTeach Toolkit: Lesson Plans for Digital Library Instruction

Introduction to the Music Encoding Initiative

Anna Kijas, Raff Viglianti

Published on: Sep 19, 2019

DOI: [10.21428/65a6243c.9fa9b4f7](https://doi.org/10.21428/65a6243c.9fa9b4f7)

Session Specifics

This tutorial can be used for a hands-on workshop or for online learning. Please note that the MEI Guidelines used in this tutorial are version 3. [Work is underway](#) to rewrite the MEI Guidelines and release version 4 soon.

Instructional Partners

- Faculty in music, performing arts, theatre, literature
- Library & information science (LIS) faculty
- Library & archives professionals

Audience

- Digital library professionals
- Music library & archives professionals
- LIS faculty
- Musicology faculty or students
- Undergraduate or graduate students
- Other folks interested in music encoding

Short Introduction

In this tutorial, we provide an introduction to encoding music documents according to the [Music Encoding Initiative](#) (MEI) for people engaged in digital library work. The MEI is a community-driven effort to create an open source representation of music notation in a machine readable structure. While there are a number of different formats in use for marking up or encoding music, including [MusicXML](#), [Humdrum](#), and [Plaine & Easie](#), the focus of this tutorial is on applying the [MEI guidelines](#) (version 3.0.0). These guidelines present a core set of rules for the representation of symbolic, physical, and intellectual aspects of music notation expressed using an XML schema.

Libraries and archives professionals may need to create encoded music files (MusicXML or MEI) in order to ensure preservation or long-term access to music documents. Although both formats are [recommended by the Library of Congress](#), unlike MusicXML, MEI provides greater affordances for encoding semantically and structurally rich metadata and music notation, as well as the ability to identify relationships between music notation and digital images or recordings. Additionally, there

may be interested to use music encoding as part of a digital scholarship project or in training of faculty or students who are interested in encoding as a means of close reading, analysis, or the creation of digital editions. Getting started with music encoding, however, can be intimidating. In this tutorial, we aim to present a straightforward workflow that will enable new encoders to **(a)** generate an MEI file using music notation software and **(b)** create various types of metadata in the MEI header section. Several tools and software (open source and proprietary) will be introduced during the workflow that are necessary for different steps of the process.

Learning Outcomes

1. Learners will be able to generate an MEI file from a music document.
2. Learners will be able to create various metadata for an MEI file.
3. Learners will know where to locate additional MEI resources.
4. Learners will be able to generate an MEI header from MARC XML.
5. Learners will be able to identify and select appropriate tools and software for working with MEI.
6. Learners will be able to render an MEI file.

This tutorial will **not** provide in-depth coverage of the MEI guidelines, applying XSLT (eXtensible Stylesheet Language Transformations) or schemas, optical music recognition (OMR), other encoding standards, or details of the encoding of the music notation itself, which in this case is left to automatic conversion.

The MEI guidelines remain the primary documentation for the header and MEI metadata, although they occasionally only marginally cover some components and the examples tend to be simple and do not showcase more complex situations that could be found in the real world. With this tutorial, we aim at supplementing the guidelines with summaries and examples relevant to digital library work. The best option for reaching out for help is by participating in the MEI community, primarily by writing to the [MEI-L mailing list](#), which welcomes questions from encoders of all levels. Another venue for communicating with MEI maintainers is their [GitHub repository](#). There you can ask questions and file issues if you think something should be handled differently, or if you would like to request a new feature, such as a new attribute or element.

Preparation

- Instructor should review and familiarize themselves with the tutorial in advance, paying attention to the examples and files provided in Additional Instructional Materials below.
- Instructor may also wish to review the [MEI Guidelines](#) in advance.

- Both instructor and participants will need to install software identified in the tutorial used to complete different parts of the workflow.

Materials

Music documents used in this tutorial can be accessed in this [Open Science Foundation repository](#).

Session Outline

Workflow to Generate an MEI File

If this is your first foray into music encoding you might be wondering, how, or where do I get an MEI file? There are different ways to (somewhat easily!) generate an MEI file. A common way is to use music notation software to create a music document or import an existing music document. One thing to note is that if you are extracting music notation from an existing music document in PDF format, as opposed to working with a fully corrected file, there will be some necessary clean-up and editing.

Although we strive to use open source software and tools whenever possible, there are scenarios for which proprietary software may be required or used in combination with open source tools. In this workflow for generating an MEI file we will use [MuseScore](#) (open source) to import a music document, extract and correct the music notation. We present two options for generating the MEI file. For option 1 we will use a web-based tool, [Verovio](#), and for option 2 we will use an advanced method and apply an XSLT transformation. In order to transform MusicXML to MEI you will need to use [Oxygen XML Editor](#) (proprietary) or the command line tool [Saxon](#) PE (also proprietary). The [Atom](#) text editor (open source) does not currently support XSLT transformation for MEI, but this tutorial will show how to set it up to hand code and validate MEI files.

There are other music notation software applications that you may be familiar with, such as [Finale](#) or [Sibelius](#) (both proprietary), which can be used in the process of generating an MEI file. Music notation applications, including MuseScore, Finale, and Sibelius all use [optical music recognition](#) (OMR) software to interpret and extract music notation, text, and other characters from a music document. Sibelius has a plugin ([SibMEI](#)) created by the MEI community that enables users to transform from Sibelius directly into the MEI format; this will be discussed as an alternative option.

MuseScore

In this tutorial, we will be using [MuseScore](#) an open source, professional music notation software that can be installed on Windows, macOS, Linux, and Chromebook. You can input notation with a MIDI keyboard or computer keyboard, as well as import a music document from a PDF file and convert it to the MuseScore default format (.mscz). The import process uses the [Audiveris](#) optical music recognition (OMR) engine.

The images below are hosted in OSF.

Process overview

Install MuseScore

Before we get started you will need to [download the latest stable version](#) of this software for your operating system. You will also need to [create an online MuseScore account](#) where your imported PDFs will be processed by the OMR engine and the extracted notation saved for use with MuseScore.

Materials

[PDF file of “Un certo freddo orrore” by Wilhelmine von Bayreuth](#)

[MusicXML file](#) (cleaned up copy)

Extract Notation from Music Document

We will be extracting notation from a [PDF of an arrangement of the aria “Un certo freddo orrore” from *Argenore*](#), composed by Wilhemine von Bayreuth.

Once you have installed MuseScore and created an account, open the software on your machine and import the PDF so that it can be extracted and opened in MuseScore

- File → Import PDF
- A browser window will open and you will need to log in to MuseScore in order to import the PDF.
- Upload the PDF file
 - MuseScore will process the PDF using OMR software and create a .mscz file. You can open this file directly in MuseScore.

Review & Correct Extracted Notation in MuseScore

If you were successful in importing the PDF you will see it listed in your [MuseScore Import](#) dashboard and will be able to download or choose to open it directly in the MuseScore software on your machine.

- Open each individual .mscz or .musicxml file for side-by-side comparison with the original PDF.
 - Tip: use dual screens if possible to maximize screen size.
- If you find mistakes or missing data from the processed file, make corrections so that it matches the original score. The quality of the extracted notation from the PDF will vary widely depending on the quality of the score.
 - You can save your work-in-progress in the .mscz format or export in the uncompressed .musicxml format.
 - Store your file in a folder or directory where it will be backed up.
- If a PDF can not be imported, there may be a problem with the file quality. Do not upload more than once with the MuseScore Import Tool. It may require that you transcribe the notation manually or use a different OMR software.

As with any software program it is best to save your work at various stages while using MuseScore. The software will save your work in the .mscz native format.

- Select File → Save As
 - Use a standard file naming convention to name your file.
 - Change save location to designated folder.

If you have finished correcting your file, you can export it as an uncompressed .musicxml file

- Select File → Export
 - Use the assigned filename.
 - Change save location to designated folder
 - Change format to Uncompressed MusicXML File (.musicxml)

Useful Tips for Cleaning Up Extracted Notation

Incorrect Duration

1. Select the measure with the incorrect duration by clicking it so that a box appears around the measure.

1. Right click on the measure and a pop-up will appear. Select “Measure Properties” to edit further.

Menu to select Measure Properties

2. A window will pop-up and you can adjust the **Actual** duration in the “Measure Duration” section.

Adjusting actual duration

- Adjust the duration and go back and continue correcting the measure.

Example from [Wilhelmine von Bayreuth's "Un certo freddo orrore" from Argenore," mm. 56-58.](#)

Adding Triplets

When adding triplets, add only the first note of each triplet into the score so that the duration is correct. Once you finish the measure, go back, select the first note of each triplet and select Notes → Tuplets → Triplet to create a triplet. This will preserve the correct duration.

Converting MusicXML to MEI

You will most likely spend the greatest amount of time correcting the music document, depending on the quality of the extracted notation. Once your music document is corrected you will be ready to generate an MEI file.

In this tutorial we present two options for generating an MEI file. The first option is to use [Verovio](#) a lightweight, open source music notation engraving library developed by the [Swiss RISM Office](#). Verovio also offers more advanced methods for converting MusicXML files to MEI via the [command line interface, or Python toolkit](#), as well as the ability to generate SVG and MIDI files from MEI and other supported formats or process MEI via XSLT directly in the browser using the [JavaScript toolkit](#). The second, advanced, option is to run an XSLT script in Oxygen XML Editor, a proprietary software, which also requires using the Saxon PE (professional edition) processor.

Options for converting MusicXML to MEI

Option 1: Verovio

A simple and quick way to convert a MusicXML file to MEI is to upload the file into the [Verovio converter tool](#). This option also enables you to render it in HTML and view it immediately.

- Select a MusicXML file using the “Browse” feature in the Verovio converter. You can use the MusicXML file you just cleaned up or the sample [MusicXML file](#) we provided.

Try it with your file

You can also try to convert your own MusicXML file. If it works, the file will be displayed with Verovio and you will then be able to save the resulting MEI file.

Screenshot of uploading to Verovio converter

- Select your file and choose “Convert to MEI”
- You will now see your MusicXML file rendered in the browser

Un certo freddo orrore
Aria from "Argenore"
an electronic transcription

Wilhelmine von Bayreuth

Adagio

 A screenshot of a musical score for the aria "Un certo freddo orrore" from the opera "Argenore". The score is for a Soprano and an electronic transcription. The tempo is marked "Adagio". The score includes staves for Violins 1, Violins 2, Violas, Soprano, and Violoncellos. The Soprano part has lyrics: "Un cer - to fred - do or - ro - re tut - to mi ge - la il san - gue". The score is marked with a piano (p) dynamic.

Screenshot of Wilhelmine von Bayreuth's "Un certo freddo orrore" from *Argenore*.

- Select the MEI button to convert and export this file in MEI format. Once you download the MEI file you can open up your text editor to enhance the metadata or make any additional corrections.

Option 2: MusicXML to MEI via XSLT Script

You can also convert your MusicXML files to MEI with this [musicxml2mei-3.0.xsl script](#) provided on the MEI GitHub repository. This option requires that you use Oxygen XML Editor with Saxon PE. This

option is best if you will be converting many MusicXML files and/or also need to customize your own version of the XSLT script.

Before using the XSLT script, you should check whether your MusicXML file is in time-wise format. Most music notation software will export to MusicXML in part-wise format, which privileges the hierarchy of parts (rather: staves) over the hierarchy of time (e.g. measures). MEI always privileges the hierarchy of time over parts by nesting staves within measures. MusicXML exports, therefore, will often need to be converted to MusicXML in time-wise format in order to be transformed into MEI.

Converting from Part-wise to Time-wise Format

How do you know if your .musicxml file is part-wise or time-wise? Check the line that begins with “DOCTYPE” for the format. Please note that this option is not required if you use Verovio to generate your MEI file.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE score-partwise PUBLIC "-//Recordare//DTD MusicXML 3.1 Partwise//EN" "http://www.musicxml.org/dtds/partwise.dtd">
3 <score-partwise version="3.1">
4   <work>
5     <work-title>Un certo freddo orrore</work-title>
6   </work>
7   <identification>
8     <creator type="composer">Wilhelmine Von Bayreuth</creator>
9     <creator type="lyricist">Giovanni Andrea Galletti</creator>
10    <encoding>
11      <software>MuseScore 2.2.1</software>
12      <encoding-date>2018-06-07</encoding-date>
13      <supports element="accidental" type="yes"/>
14      <supports element="beam" type="yes"/>
15      <supports element="print" attribute="new-page" type="yes" value="yes"/>
16      <supports element="print" attribute="new-system" type="yes" value="yes"/>
17      <supports element="stem" type="yes"/>
18    </encoding>

```

Example of .musicxml file in part-wise format open in Oxygen XML Editor. Wilhelmine von Bayreuth's "Un certo freddo orrore" from *Argenore*.

If your file is in part-wise format then you will need to apply [the XSLT script provided on the MusicXML website](#). Download or copy the parttime.xsl script and open it in an editor capable of running XSLT scripts, such as OxygenXML Editor or [jEdit](#) (open source) with a [Saxon plugin](#).

● In your XML editor of choice, open the exported .musicxml and the parttime.xsl files. Make sure you are using a current version of [Saxon-HE](#) (the free version of the Saxon processor; licensed versions such as Saxon-PE will also work).

Example of .musicxml file in part-wise format and parttime.xml open in Oxygen XML Editor. Wilhelmine von Bayreuth's "Un certo freddo orrore" from *Argenore*.

- Identify your output location and run the XSLT script. Your "DOCTYPE" will now be identified as time-wise.

Example of .musicxml file in time-wise format open in OxygenXML Editor. Wilhelmine von Bayreuth's "Un certo freddo orrore" from *Argenore*.

Once your file has been converted to time-wise format you are ready to transform your MusicXML file to MEI using this [musicxml2mei-3.0.xsl script](#).

- Open your MusicXML file in time-wise format and the XSLT file in Oxygen XML Editor and switch to the XSLT Debugger (top right). Select the XSLT view to place both files side-by-side. Make sure that Saxon-PE 9.8.0.8 or higher is selected.

Example of .musicxml file in time-wise format and musicxml2mei.xml open in Oxygen XML Editor. Wilhelmine von Bayreuth's "Un certo freddo orrore" from *Argenore*.

- Select the folder icon across from Output to name the MEI output file and identify where you would like to save the file.

Screenshot of save dialogue

- Select run (blue arrow) to transform the MusicXML file to MEI file. The output will appear in a third window and the file will be in the location you identified above.

Screenshot of XML and MEI files open side-by-side

You can validate your MEI file in Oxygen XML Editor and upload it to the Verovio viewer (see steps in Option 1) to see it rendered in HTML.

Sibelius with SibMei

Sibelius is music notation software used by professional typesetters around the world and obtaining a license is expensive. Nonetheless, some academic music departments provide copies of Sibelius to faculty and students; it is worth checking with the music school at your institution.

Sibelius offers a 30-day free trial and even after the trial is expired, you will still be able to use Sibelius to export files to MEI, but you won't be able to save files in the Sibelius format.

Sibelius is a good choice for creating MEI documents because of the SibMei plugin that is able to export Sibelius music scores directly to MEI. Currently, Sibelius is the only music notation software able to export directly to MEI.

Download Sibelius

Get the Sibelius for Windows or Mac from the [Sibelius site](#). Make sure it's the full (or Ultimate) version, as other version may not allow you to install third-party plugins.

Add the MEI Plugin

Download the [MEI plugin](#). To install this plugin, copy the .plg files in the Sibelius Plugin directory on your machine. The specific location depends on your OS and version of Sibelius, see the [instructions](#)

[page](#).

Exporting to MEI

Before exporting, it is worth making sure that the music notation is as semantically correct as possible. For example, make sure that all ties are actually ties and not slurs. It also helps to make sure that curves, dynamics, and other symbols around the staff are anchored to specific notes or rests on the staff whenever possible. It may take a couple of rounds of exporting and checking the MEI to make sure the export is as correct as possible. To export, find the plugin under the “File” ribbon; there will be two SibMei plugins listed, one to export the current file to MEI and another to export an entire folder to MEI.

Finally, a couple more things to keep in mind. The export will be in UTF-16, which may not show correctly in some text editors. Moreover, slurs will be exported using time stamps rather than being associated with specific notes. While this is perfectly correct in MEI, some MEI processors may not work properly with time stamps; Verovio, for example will not render the slurs correctly or at all. There is a simple online tool called [Refine SibMei](#) that changes the export from UTF-16 to the more common UTF-8 and attempts to associate slurs to specific notes on the staff.

Enriching Your MEI File

After obtaining your MEI file following one of the methods described above, you will likely want to edit it and enrich it by hand in an XML editor. There are many reasons for doing so, for example:

- Adding metadata about the file, the physical object it came from (if any), the work of music it encodes, etc.
- Correcting any mistakes resulting from the conversion process.
- Encoding the music notation further with textual phenomena that notation software doesn’t typically handle, such as adding rich editorial markup, variants from multiple sources, non-standard notation, features related to the transcription of manuscripts, etc.

In this tutorial we will focus on adding metadata in the header of the file, which typically needs to be hand encoded with an XML editor. If you have Oxygen XML Editor, feel free to skip straight to the Metadata section; otherwise read on to learn about the free and open source text editor Atom and how to configure it to validate XML.

Using Atom for Hand Coding

Atom is a text editor with support for a number of programming and markup languages, including XML. It is free and open source. Through a plugin, it can be used to validate XML files against a schema — for example to make sure the file being edited follows MEI rules. The same plugin also offers autocomplete suggestions, which makes it easier to figure out which MEI elements and attributes to use.

This section will guide you through a number of steps to install and configure Atom.

Download Atom

Atom can be downloaded at the [Atom site](#). Versions are available for Windows, Mac, and Linux. Select and install the appropriate version for your operating platform, as you would any other application.

Add XML schema plugin to Atom

- Open Atom and access its settings by hitting Ctrl + , (comma) or on Mac, ⌘ + , (comma).
- Select “Install”.
- In the search bar, enter “linter-autocomplete-jing”.
- When the corresponding result appears, click on “Install”.
- If prompted to “install dependencies,” click “Yes”.
- When the installation is complete, quit and restart Atom.

Install Java Development Kit (JDK)

The plugin to validate XML requires Java code. Your machine will likely already be equipped to use Java, but if you had trouble installing the plugin, you may want to install the latest Java Development Kit (JDK) and try again. The JDK can be downloaded [from the Oracle website](#). Make sure to select the correct platform (Windows, Mac OS, etc.) and follow the instructions to install it.

Test XML Schema Plugin

- In Atom, open a new file from the main menu bar: “File” → “New File.”
- Copy and paste the MEI template below. If copying from a PDF version of this tutorial, make sure the lines in grey are over only three lines, each starting with “<?xml”

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="https://music-encoding.org/schema/3.0.0/mei-
CMN.rng" type="application/xml"
schematypens="http://relaxng.org/ns/structure/1.0"?>
<?xml-model href="https://music-encoding.org/schema/3.0.0/mei-
CMN.rng" type="application/xml"
schematypens="http://purl.oclc.org/dsdl/schematron"?>
<mei xmlns="http://www.music-encoding.org/ns/mei">
  <meiHead>
    <fileDesc>
      <titleStmt>
        <title></title>
      </titleStmt>
      <pubStmt></pubStmt>
    </fileDesc>
  </meiHead>
  <music></music>
</mei>
```

The first lines (those that begin with `<?xml ...`) (1) indicate that this is an XML file, and (2) associate this file with the latest MEI schema available online.

- Navigate to “File” → “Save As...” to save the file with an “.xml” extension to activate the plugin (e.g. “test.xml”). Perform the operations below to make sure the plugin is working properly.
- Check for error messages (indicated by a red-colored circle along the left-hand side) where the XML is not “well formed”, e.g. by removing a closing tag.

Screenshot of checking error messages in Atom

- Start typing a new attribute or element to get autocomplete suggestions.

Screenshot of autocomplete suggestions in Atom

Metadata

Materials

Your own MEI file or the sample MEI file of “Un certo freddo orrore” — available in Additional Instructional Materials below, under Data

MEI files for *The Wreckers* by Ethel Smyth — available in Additional Instructional Materials below, under Data

MEI file for Piano Trio in D minor by Ethel Smyth — available in Additional Instructional Materials below, under Data

Introduction to the MEI Header

In this section we will focus on a selection of components and elements metadata in the MEI header `<meiHead>` that will be most useful to people engaged in digital library work. The MEI header contains the bibliographic description and other documentation about the encoded work. This tutorial is meant to summarize and supplement the MEI guidelines rather than provide an in-depth overview of the full MEI header. If you are interested, Section 2 of the [MEI guidelines](#) (currently 3.0.0) provides an in-depth outline and explanation of each section of the MEI header. We will focus on the following sections of the MEI Header:

- 1. Description of the File
- 1. Description of the Source
- 1. Description of several aspects of the Work
- 1. Description of Revisions

1. Description of the File

The file description `<fileDesc>` contains the full bibliographic description of the encoded work as well as the source or sources from which the encoded work is derived. The file description is the only mandatory section in the entire MEI header and it contains two mandatory and six optional elements. A complete file description may look like this:

```
<meiHead>  
  <fileDesc>  
    <titleStmt>
```

```
        <title><!-- title of the resource -->
    </title>
</titleStmt>
<editionStmt>
    <!--information about the edition of the resource --
>
</editionStmt>
<extent>
    <!--description of the size of the resource -->
</extent>
<pubStmt>
    <!--information about the publication and
distribution of the resource -->
</pubStmt>
<seriesStmt>
    <!--information about any series to which the
resource belongs -->
</seriesStmt>
<notesStmt>
    <annot>
        <!--notes on other aspects of the resource -->
    </annot>
</notesStmt>
<sourceDesc>
    <!-- information about the source(s) from
which the resource was derived -->
</sourceDesc>
</fileDesc>
</meiHead>
```

Title Statement

The title statement [<titleStmt>](#) element is **mandatory** and it contains the title [<title>](#) for the encoded work, which should be distinguishable from the original source title. For example if we created a title statement for an encoded Wilhelmine von Bayreuth's Aria "Un certo freddo orrore" from *Argenore* it would look like this:

```
<titleStmt>
  <title>Un certo freddo orrore</title>
  <title type="subtitle">Aria from "Argenore"</title>
  <title type="subtitle">an electronic transcription</title>
</titleStmt>
```

Within this element you will normally include statements of [primary responsibility](#) for the person or people who created the source from which the encoded work is derived (e.g. composer, editor, lyricist, arranger). Secondary responsibility can be encoded within a [<respStmt>](#) element which identifies the name of the individual [<persName>](#) and the nature of the responsibility with the @role attribute. You can use values from the Library of Congress' MARC [relator code list](#) or [term list](#), where applicable. For example if we expand the title statement for the same work, but want to add the librettist and the name of the encoder it would look like this:

```
<titleStmt>
  <title>Un certo freddo orrore</title>
  <title type="subtitle">Aria from "Argenore"</title>
  <title type="subtitle">an electronic transcription</title>
  <respStmt>
    <persName role="composer">Wilhelmine von
Bayreuth</persName>
    <persName role="librettist">Giovanni Andrea
Galletti</persName>
    <persName role="encoder">Anna Kijas</persName>
  </respStmt>
</titleStmt>
```

Publication Statement

The publication statement [<pubStmt>](#) is a **mandatory** component of the <fileDesc> element. This statement should include information about the publisher of and access to the encoded work. For example a publication statement for the encoded Wilhelmine von Bayreuth's Aria "Un certo freddo orrore" from *Argenore* might look like this:

```
<pubStmt>
  <publisher>Music Theory Examples by Women</publisher>
  <date>August 30, 2018</date>
  <availability>
    <useRestrict>To the best of our knowledge, the full
compositions on this site are in the public domain, the excerpts
are in the public domain or are allowable under fair-use, and
the few compositions that are still under copyright are used by
permission. These scores, and recordings are provided for
educational use only and are not to be used commercially.
    </useRestrict>
  </availability>
</pubStmt>
```

Notes Statement

The notes statement [<notesStmt>](#) is an optional component of the <fileDesc> element. Additional information that may not have a specific element can be included in the notes statement.

Example of two of the seven components of the <fileDesc> element:

```
<titleStmt>
  <title>Un certo freddo orrore</title>
  <title type="subtitle">Aria from "Argenore"</title>
  <title type="subtitle">an electronic transcription</title>
</titleStmt>
<respStmt>
  <persName role="composer">
```

```

        Wilhelmine von Bayreuth
    </persName>
    <persName role="librettist">
        Giovanni Andrea Galletti
    </persName>
    <persName role="encoder">Anna Kijas</persName>
</respStmt>
</titleStmt>
<pubStmt>
    <publisher>Music Theory Examples by Women</publisher>
    <date>August 30, 2018</date>
    <availability>
        <useRestrict>To the best of our knowledge, the full
compositions on this site are in the public domain, the excerpts
are in the public domain or are allowable under fair-use, and
the few compositions that are still under copyright are used by
permission. These scores, and recordings are provided for
educational use only and are not to be used commercially.
        </useRestrict>
    </availability>
</pubStmt>

```

2. Description of the Source

The source description [<sourceDesc>](#) is an optional component of the [<fileDesc>](#) element. It contains a number of [<source>](#) elements, each identifying a source document from which the MEI file is derived. A source could be a manuscript, a printed score, or even a digital object. The source description may be omitted altogether if the MEI file authored directly and not derived from other documents.

When MEI is used to encode a document (e.g. a printed score held in a music library), one [<source>](#) element should be used. Often, however, MEI is used to create new editions of music works, in which case there will be more than one source document involved.

The [<source>](#) element contains elements to describe a document's title and publication information, but also provide information about its physical location, physical condition, and history. We will be considering as an example an MEI transcription of the 1916 edition of the piano-vocal score of Ethel Smyth's opera *The Wreckers*.

Identifiers

```
<sourceDesc>
  <source xml:id="wreckers1916">
    <identifier>ES1</identifier>
  </source>
</sourceDesc>
```

There are two optional ways of identifying the source being described. The first is by using an XML ID (e.g. wreckers1916), which can be used within the MEI document to refer to this specific source. This is essential for encoding variants in critical editions, for example. XML IDs need to follow two simple rules: 1) being unique within the XML file and 2) always starting with a letter or an underscore. The child element [<identifier>](#), on the other hand, provides a place for an unrestricted form of identifier that you may want to use for this source.

Title, Edition, Publication

To encode a source's title, edition, and publication data, [<source>](#) uses the same elements described above. This time, instead of referring to the MEI file and its creation, they refer to a specific object from which the MEI is derived.

```
<sourceDesc>
  <source xml:id="wreckers1916">
    <identifier>ES1</identifier>
    <titleStmt>
      <title>The Wreckers</title>
      <title type="subtitle">
        Lyrical Drama in three Acts
      </title>
    <respStmt>
      <name>Ethel Smyth</name>
```



```

        </respStmt>
    </titleStmt>
    <pubStmt>
        <pubPlace>Wien</pubPlace>
        <respStmt>
            <corpName>Universal-edition A.G.</corpName>
        </respStmt>
        <date>1916</date>
    </pubStmt>
</source>
</sourceDesc>

```

Physical Description & Location

The child element [<physDesc>](#) describes several bibliographical aspects of the object, including dimensions and physical conditions. This is done in a semi-structured manner: prose can be used, but specific data, such as numbers, units, dates, etc., can be tagged explicitly.

```

<source xml:id="wreckers1916">
    <!-- ... -->
    <physDesc>
        <scoreFormat>Piano-vocal score</scoreFormat>
        <dimensions>
            <num unit="page">277 p.</num>
            <height unit="cm">31 cm</height>
        </dimensions>
        <condition>Cover pages removed.</condition>
    </physDesc>
</source>

```

The child element [<physLoc>](#) describes the repository and provenance of the object in a semi-structured manner.

```
<source xml:id="wreckers1916">
  <!-- ... -->
  <physLoc>
    <repository>
      <settlement>
        University of North Carolina at Chapel Hill
      </settlement>
      <identifier>782 S667nx</identifier>
    </repository>
    <provenance>
      Donated via a generous grant by <name>...</name>.
    </provenance>
    <annot>(Fictional example)</annot>
  </physLoc>
</source>
```

History

The history of the source document can be encoded using the [<history>](#) element, again in a semi-structured manner. This history has to be pertinent to the source object being described and different from both its provenance (encoded within [<physLoc>](#)) and the history of the abstract work (encoded in [<workDesc>](#)).

Contents

The source's contents can be described in detail with the [<contents>](#) element. This is particularly useful when the object contains multiple movements or is an anthology.

```
<source xml:id="wreckers1916">
  <!-- ... -->
  <contents>
    <contentItem>Act I</contentItem>
    <contentItem>Act II</contentItem>
```

```

    <contentItem>Act III</contentItem>
  </contents>
</source>

```

Classification

It is also possible to record classification terms about the source document. The [MEI guidelines](#) explain in detail how to refer to specific classification codes, such as the ones provided by the Library of Congress. It is important to remember that this classification should be specific to the source being described rather than the abstract work, which is instead described in [<workDesc>](#). Inevitably, there will be common terms that can describe both the source and the more abstract work of music and it will be up to the encoder to decide whether to duplicate terms in both [<source>](#) and [<workDesc>](#) or if it is preferable to separate metadata more strictly.

```

<source xml:id="wreckers1916">
  <!-- ... -->
  <classification>
    <termList>
      <term>Opera</term>
    </termList>
  </classification>
</source>

```

Describing Manuscript Sources

The description of manuscript sources often includes details not usually described for printed materials, such as providing a list of scribes and their writing media (ink, pencil) and more details about the manuscript's physical attributes.

The [<physDesc>](#) element can be used to describe and encode these features; for example let us consider the autograph manuscript of Ethel Smyth's Piano Trio in D minor, held at Durham University Library.¹

```

<source xml:id="A">
  <editionStmt>
    <edition>

```

```
Autograph copy full score. Working draft. Dated at
end
    <date isodate="1880-08-25">
        Aug. 25th. <expan>18</expan>80
    </date>
</edition>
</editionStmt>
<physDesc>
    <dimensions>340 x 265 mm</dimensions>
    <extent>26 leaves</extent>
    <condition>The whole of the original p.4 and parts of
p.6 and p.13 have been cancelled, and a different version pasted
over.
    </condition>
    <handList>
        <hand xml:id="ES" medium="pencil">Ethel
Smyth</hand>
        <hand xml:id="pg" medium="pencil">
            Library pagination
        </hand>
    </handList>
</physDesc>
<contents>
    <contentItem>Allegro</contentItem>
    <contentItem>
        <supplied>slow movement</supplied>
    </contentItem>
    <contentItem>Scherzo and trio</contentItem>
    <contentItem>Allegro vivace</contentItem>
</contents>
</source>
```

The [<editionStmt>](#) element can be used to indicate that the source is an autograph manuscript and a working draft. We also use [<date>](#) to both transcribe the date present at the end of the manuscript, and provide a standardized form with the `isodate` attribute. Note the use of the [<expn>](#) element to expand the abbreviated date “80” into “1880”.

As described previously, [<physDesc>](#) can be used to record information about the physical object; in the example above we also added information about scribes using [<handList>](#). We give an ID to each [<hand>](#) because it may be useful to refer back to this list from the transcription in order to identify shifts in writing medium or scribe. Finally, note the use of the [<supplied>](#) element in the second [<contentItem>](#). This indicated that the title “slow movement” has been provided by the curator because a tempo marking is missing in the manuscript text. There are also other editorial elements available to transcribe, for example, authorial deletions and additions.

3. Description of Several Aspects of the Work

The element [<workDesc>](#) can be used to describe several aspects pertinent to the work of music of which the MEI file is an edition, or a transcription of one of the work’s sources. This element can capture metadata related to instrumentation, key, tempo, meter, but also paratext such as the work’s history, a description of its creation, etc. The second chapter of the [MEI guidelines, section 2](#), provides a detailed description of the various components. In this tutorial, we supplement the Guidelines by showing as an example the encoding of instrumentation and performance history of Ethel Smyth’s opera *The Wreckers*.

```
<workDesc>
  <work>
    <creation>Composition history in prose form can be
    encoded here. A number of elements are available to mark-up
    entities such as names and dates.
  </creation>
  <history>
    <eventList type="performances">
      <event>
        <p>First performance in Leipzig,
        Stadttheater,
        <date isodate="1906-11-11">11 November
        1906</date>
```

</p>

</event>

<event>

<p>Prague, Neues Deutsches Theater,

<date isodate="1906-12-12">12 December

1906</date>

</p>

</event>

<event>

<p>London, Her Majesty's Theatre,

<date isodate="1909-06-22">22 June

1909</date>

</p>

</event>

<event>

<p>London, Covent Garden,

<date isodate="1910">1910</date>

</p>

</event>

<event>

<p>London, Royal Albert Hall,

<date isodate="1994">1994</date>

</p>

</event>

<event>

<p>New York, Avery Fisher Hall,

<date isodate="2007-09-30">30 September

2007</date>

</p>

</event>

<event>

<p>Gießen, Stadttheater,

```

                                <date isodate="2007">2007</date>
                                </p>
                                </event>
                                <event>
                                <p>Annandale-On-Hudson, The Fisher Center,
                                <date isodate="2017-07-24">24 July
                                2017</date>
                                </p>
                                </event>
                                </eventList>
                                </history>
                                <!-- etc. -->
                                </work>
                                </workDesc>

```

Providing a history of performance as shown above is common, though more information about the history of the work can be encoded in semi-structured prose (prose with interspersed markup), as well as list of other types of events. As recommended by the MEI guidelines, we use the type attribute to indicate that by “event” in this case we mean “performance”. Each [<event>](#) element can contain more detail in prose.

Below is an example of how cast roles and instruments can be encoded.

```

<workDesc>
  <work>
    <!-- ... -->
    <perfMedium>
      <castList>
        <castItem>
          <role>Pascoe</role>
          <roleDesc>the local preacher</roleDesc>
          <perfRes>Baritone</perfRes>
        </castItem>
        <castItem>

```

```
        <role>Thirza</role>
        <roleDesc>his wife</roleDesc>
        <perfRes>Mezzo-soprano</perfRes>
    </castItem>
    <castItem>
        <role>Lawrence</role>
        <roleDesc>the Lighthouse keeper</roleDesc>
        <perfRes>Tenor</perfRes>
    </castItem>
    <castItem>
        <role>Mark</role>
        <roleDesc>a young fisherman</roleDesc>
        <perfRes>Tenor</perfRes>
    </castItem>
    <castItem>
        <role>Avis</role>
        <roleDesc>Lawrence's daughter</roleDesc>
        <perfRes>Soprano</perfRes>
    </castItem>
    <castItem>
        <role>Chorus</role>
        <roleDesc>villagers and fishermen</roleDesc>
        <perfRes>Chorus</perfRes>
    </castItem>
</castList>
<perfResList>
    <perfRes>Piccolo</perfRes>
    <perfRes>Flauti</perfRes>
    <perfRes>Oboi</perfRes>
    <!-- etc. -->
</perfResList>
</perfMedium>
```



```

    </work>
  </workDesc>

```

4. Description of Revisions

The [<revisionDesc>](#) is used to document changes made to the MEI file. It is useful to document details concerning how the original file was generated and transformed, especially for preservation purposes and to inform others of your process and corrections to the file. Within this sub-element you can identify the person responsible for modifications or updates to the file, as well as the date when it occurred. Each change is documented within a <change> element and identified with a number.

```

<revisionDesc>
  <change n="1">
    <respStmt>
      <persName resp="#AK"/>
    <respStmt>
    <changeDesc>
      <p>The original MusicXML file was generated using
MuseScore 2.</p>
    </changeDesc>
    <date isodate="2018-06-29"/>
  </change>
  <change n="2">
    <respStmt>
      <persName resp="#AK"/>
    <respStmt>
    <changeDesc>
      <p>Transcoded from a MusicXML version 3.1 file
using an XSLT stylesheet (musicxml2mei v. 3.0).</p>
    </changeDesc>
    <date isodate="2018-06-29"/>
  </change>
</revisionDesc>

```

Converting Existing MARC Fields

Materials

Available in data folder under Additional Instructional Materials below:

- MARC XML record for *Turbulence*, for piano solo. Op. 17, no. 2 by Marion Bauer
- Sample MEI Header file for *Turbulence*, for piano solo. Op. 17, no. 2 by Marion Bauer

Digital library work often includes cross-walking or converting metadata records into other formats. If you will be generating MEI files for music documents for which metadata records already exist in MARC XML format then you can convert those records into MEI files with this [XSL script](#) on the MEI GitHub repository. Converting existing MARC XML files to MEI header XML will make your workflow more efficient and ensure consistency across records and collections.²

We will use this MARC XML record for the composition, *Turbulence*, for piano solo. Op. 17, no. 2 by Marion Bauer as an example for transforming the metadata in the MARC XML to an MEI header. The [catalog record](#) is available online and displays the following fields:

Details	
Title	<i>Turbulence, for piano solo. Op. 17, no. 2.</i>
Author	Bauer, Marion >
Subjects	Piano music >
Related Titles	Series: : Contemporary American composers.
Publisher	Radio City, N.Y., E.B. Marks Music Corp.
Format	6 pages 30 cm..
Creation Date	1942
Notes	Cover title.
Source	ALMA_BOSU1
Record #	21609606680001161

Image of catalog record from Boston University.

The public display only shows us a snapshot of what is contained in the entire MARC record. Here is a representation of the above display using MARC code fields with the corresponding display field with the corresponding MEI header component and element. This will be discussed in greater detail below.

MARC Code Fields	Display Field & Tag	MEI Header Component	MEI Element
Title and Title-Related	Title (245)	<titleStmt>	<title>

Title and Title-Related	Publisher (including location) (260)	<pubStmt>	<publisher> <pubPlace>
Title and Title-Related	Creation Date (260)	<pubStmt>	<date>
Main Entry	Author (100)	<respStmt>	<persName>
Subject Access	Subjects (650)	<classification>	<termList>
Series Statement	Series (490)	<seriesStmt>	<title>
Physical Description	Format (300)	<physDesc>	<extent> <dimensions>
Note	Notes (500)	<notesStmt>	<annot>

Open up the MARC XML record in the XML-aware text editor that you have been using for this tutorial and you will see fields and tags that are not visible in the public display. These fields include control numbers and other coded information that is necessary for the processing of machine-readable bibliographic records. These fields also include standard numbers, classification numbers, and other data about the bibliographic record for this music document for which the MEI file is being created.

```
<?xml version="1.0" encoding="UTF-8" ?>
<marc:collection xmlns:marc="http://www.loc.gov/MARC21/slim"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.loc.gov/MARC21/slim http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd">
  <marc:record>
    <marc:leader>00826ccm a2200265K 4500</marc:leader>
    <marc:controlfield tag="001">ocm09597834 </marc:controlfield>
    <marc:controlfield tag="003">0CoLC</marc:controlfield>
    <marc:controlfield tag="005">20180927085418.0</marc:controlfield>
    <marc:controlfield tag="008">830610s1942 nyuzzz n zxx d</marc:controlfield>
    <marc:datafield tag="040" ind1=" " ind2=" " >
      <marc:subfield code="a">BOS</marc:subfield>
      <marc:subfield code="b">eng</marc:subfield>
      <marc:subfield code="c">BOS</marc:subfield>
      <marc:subfield code="d">OCLCQ</marc:subfield>
      <marc:subfield code="d">OCLCG</marc:subfield>
      <marc:subfield code="d">UKMGB</marc:subfield>
      <marc:subfield code="d">OCLCQ</marc:subfield>
      <marc:subfield code="d">OCLCF</marc:subfield>
      <marc:subfield code="d">OCLCO</marc:subfield>
      <marc:subfield code="d">OCLCQ</marc:subfield>
    </marc:datafield>
    <marc:datafield tag="016" ind1="7" ind2=" " >
      <marc:subfield code="a">004199695</marc:subfield>
      <marc:subfield code="2">Uk</marc:subfield>
    </marc:datafield>
    <marc:datafield tag="050" ind1=" " ind2="4" >
      <marc:subfield code="a">M25.B39</marc:subfield>
      <marc:subfield code="b">T9 1942M</marc:subfield>
    </marc:datafield>
    <marc:datafield tag="049" ind1=" " ind2=" " >
      <marc:subfield code="a">BXMM</marc:subfield>
    </marc:datafield>
  </marc:record>
</marc:collection>
```

Screenshot of selected bibliographic metadata in the MARC XML file.

Transform Metadata from MARC XML to MEI Header

To transform the metadata in the MARC XML record into an MEI header file you will need to use this [marc2mei.xsl script](#). Please note that this script also references this [marc2mei59x.xsl script](#). You may want to download both scripts from the MEI GitHub repository and put them in a folder along with the MARC XML file so that Oxygen XML Editor can find the files and validate correctly.

Once you have all of the files in a single folder or directory, launch Oxygen XML Editor and open the [MARC XML file](#) of Marion Bauer's Turbulence. The next few steps may be familiar if you walked through Option 2 above and converted a MusicXML file to MEI with XSLT.

Screenshot of MARC XML record for Marion Bauer's Turbulence.

Next, you will open the [marc2mei.xml script](#) and switch to the XSLT debugger to view the two files side-by-side. The XML and XSL files will be listed in your top menu. Make sure you are using a current version of [Saxon-HE](#) (the free version of the Saxon processor; licensed versions such as Saxon-PE will also work).

Screenshot of MARC XML record for Marion Bauer's Turbulence side-by-side with marc2mei.xml script.

You will now need to identify the name of the output MEI file. Select the folder icon across from Output to name the MEI output file and identify where you would like to save the file.

Screen shot of Save dialogue

Select run (blue arrow) to transform the MARC XML to MEI file. The output will appear in a third window and the file will be in the location you identified above. You will now see the MEI file with your MARC metadata. You will notice that in the MEI header within the <fileDesc> each of the fields transformed from the MARC XML to MEI will contain the corresponding MARC tags and subfields. The two components: <titleStmt> and <pubStmt> will contain minimal metadata from the MARC record and will require that you enhance these sections based on information about the electronic music document that is being created. For a review of the <fileDesc> component, go back to the section in Metadata: Description of the File.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <meiHead xmlns="http://www.music-encoding.org/ns/mei" meiversion="3.0.0" xml:lang="eng">
3   <fileDesc>
4     <titleStmt>
5       <title type="proper" analog="marc:245|a-b">Turbulence, for piano solo. Op. 17, no.
6         2.</title>
7     <respStmt>
8       <persName role="creator" analog="marc:100|a">Bauer, Marion, <date analog="marc:100|d"
9         >1882-1955</date>
10      </persName>
11    </respStmt>
12  </titleStmt>
13  <pubStmt>
14    <publisher/>
15    <date>[2018]</date>
16  </pubStmt>
17  <sourceDesc>
18    <source xml:id="ocm09597834"> [44 lines]
63  </sourceDesc>
64 </fileDesc>
65 <encodingDesc>
66   <appInfo> [4 lines]
71 </encodingDesc>
72 <workDesc>
73   <work> [10 lines]
84 </workDesc>
85 <revisionDesc>
86   <change> [6 lines]
93 </revisionDesc>
94 </meiHead>

```

Screenshot of MEI header in MEI file generated from MARC XML metadata record.

The <sourceDesc> contains bibliographic metadata about the source which is being used as the basis for the electronic music document. For a review of the <sourceDesc>, go back to the section in Metadata: Description of the Source.

```

<sourceDesc>
  <source xml:id="ocm09597834">
    <identifier analog="marc:040|a">BOS</identifier>
    <identifier analog="marc:040|c">BOS</identifier>
    <identifier analog="marc:040|d">OCLCQ</identifier>
    <identifier analog="marc:040|d">OCLCG</identifier>
    <identifier analog="marc:040|d">UKMGB</identifier>
    <identifier analog="marc:040|d">OCLCQ</identifier>
    <identifier analog="marc:040|d">OCLCF</identifier>
    <identifier analog="marc:040|d">OCLCO</identifier>
    <identifier analog="marc:040|d">OCLCQ</identifier>
    <titleStmt>
      <title type="diplomatic">
        <title analog="marc:245|a">Turbulence, for piano solo. Op. 17, no. 2.</title>
      </title>
      <respStmt>
        <persName role="creator" analog="marc:100|a">Bauer, Marion, <date
          analog="marc:100|d">1882-1955</date>
        </persName>
      </respStmt>
    </titleStmt>
    <pubStmt>
      <pubPlace analog="marc:260|a">Radio City, N.Y.,</pubPlace>
      <publisher analog="marc:260|b">E.B. Marks Music Corp.</publisher>
      <date analog="marc:260|c">[1942]</date>
    </pubStmt>
    <physDesc>
      <extent analog="marc:300|a">6 pages</extent>
      <dimensions analog="marc:300|c">30 cm.</dimensions>
    </physDesc>
    <seriesStmt analog="marc:490">
      <title>Contemporary American composers</title>
    </seriesStmt>
    <notesStmt>
      <annot type="general" analog="marc:500">Cover title.</annot>
    </notesStmt>
    <classification>
      <classCode xml:id="LCCN">Library of Congress Classification Number</classCode>
      <classCode xml:id="LCSH">Library of Congress Subject Headings</classCode>
      <classCode xml:id="fast">fast</classCode>
      <termList>
        <term label="callNum" classcode="#LCCN" analog="marc:050">M25.B39 T9 1942M</term>
        <term label="topic" classcode="#LCSH" analog="marc:650">Piano music</term>
      </termList>
    </classification>
  </source>
</sourceDesc>

```

Screenshot of <sourceDesc> in MEI file generated from MARC XML metadata record.

Conclusion & Further Resources

With this brief tutorial, we hope to have provided practical examples of the kind of metadata that can be encoded in the MEI header, both about specific physical sources and the more abstract musical work. Of course more than what has been shown above can be encoded, for example the [Functional](#)

[Requirements for Bibliographic Records](#) (FRBR) module can be used in MEI to provide a more detailed description for an encoded work, especially when it comes to encoding variants and relationships between two or more sources. The module uses four elements that correspond to the group 1 entities of the FRBR model: [<work>](#), [<expression>](#), [<source>](#), and [<item>](#). One thing to note is that in the FRBR MEI module *source* is the equivalent to the *manifestation* entity.

A number of libraries and institutions are using music encoding or other music information retrieval processes in their projects and workflow. The Music Encoding Initiative maintains a short list of [projects](#) and [open source tools](#) that have been created by members of the community. One tool in particular that may be useful for digital library professionals is the [Metadata Editor and Repository for MEI Data](#) (MerMEId) at the Danish Centre for Music Editing (DCM). Although it is still under development it aims to provide tools that might be useful for editing comprehensive metadata in MEI files. A demo installation is available for testing and several online catalogs from the DCM have been produced using this tool. Another resource with rich examples of metadata MEI headers is the [Catalogue of the Works of Frederick Delius](#).

Assessment

- If this tutorial is used as part of a course or workshop, comprehension will be assessed through hands-on work, discussion, and assignments.
 - If the tutorial is used in an online environment, an individual can self-assess their understanding of the process and concepts as they progress through the tutorial. They can further assess their comprehension when they have to create their own MEI files and metadata for the MEI header.
 - MEI files created following this tutorial can be validated against the MEI schema to determine whether metadata header and body of musical work is valid.
 - A metadata header created following this tutorial can be compared against an existing MARC or other record for the original musical work from which encoding is prepared.
 - The body of the encoded musical work created following this tutorial can be compared against the original musical work to ensure that it was transcribed correctly.
 - Proper use of elements and attributes in MEI files can be reviewed against the [MEI Guidelines](#).
-

Notes

Additional Instructional Materials

Handout

[Example sheet music for encoding](#)

Data

[Data folder](#), includes xml files and encodings for pieces referenced in this lesson

Footnotes

1. The example is based on [this catalog entry](#). ↩
2. See [MARC 21 format for Bibliographic Data from the Library of Congress](#)
↩