Editing Complex Text Elements in Imported XML

As described in the **Importing EML XML Files into ezEML** chapter in this User Guide, ezEML can import EML XML files that were created outside of ezEML.

Because we want to make ezEML simple and easy to use, ezEML implements only a subset of the full EML standard. ezEML's support for importing EML files that were created outside of ezEML, however, necessitates handling some additional complexity beyond what can be created within ezEML itself.

In particular, the EML standard allows for substantial complexity in the specification of text elements such as abstracts and method step descriptions. Such text elements may, for example, contain itemized lists, ordered lists, subscripts, markdown, and so on, and these text element constituents may be nested without limit. Not all EML files employ this kind of complexity, but if you import one that does, ezEML will display a popup dialog that summarizes much of the information below.

In what follows, “complex text elements” will refer to text elements that contain sub-elements beyond what are generated natively by ezEML.

ezEML supports the editing of complex text elements in the following way. Suppose, for example, that the dataset's abstract element consists of the following XML:

```
<abstract>
    <section>
        <title>Hypothetical Abstract</title>
        <para>This is <emphasis>only</emphasis> an example.</para>
        <para>Please note that 1 &lt; 2.</para>
    </section>
</abstract>
```

When you go to the Abstract page in ezEML, what you would see is the following:

The backslash "escape" characters before the < and > in each XML tag enable ezEML to distinguish XML tags from normal text that happens to contain < or > characters. If you edit the XML, you must adhere to this convention.

The text elements that require this kind of special treatment are the ones of type **TextType** as defined in the EML standard's schema. Most text elements are simple strings and don't need this special treatment. Ones that do require it include:

- Abstract,
- Intellectual Rights,
- Maintenance Description,
- Method Step Description,
- Project Abstract, and
- Project Funding.

There are some others, but they don't show up in ezEML.

You don't need to memorize the list. When a dataset contains complex text elements, ezEML's user interface visually distinguishes such text elements in two ways, as shown in the screenshot above:

1) the text is displayed in a different (monospace) font, and
2) a checkmark button is displayed to the right of the element.

Clicking the checkmark button causes the XML validity of the element's content to be checked. A popup will tell you if it's valid or, if there's an error, what the error is and where. As you're editing a text element, it may be useful to click the checkmark button from time to time to catch any errors before you get too far.

Please note that if you import an XML file that contains one or more complex text elements, then ezEML will process the entire file according to the conventions described above. I.e., either *all* of the file's TextType elements will be handled as above, or none of them will. It's not a question of whether a particular text element contains the complexity; it's a question of whether at least *one* such element does in that particular file.

If the file doesn't contain any such complex text elements, then everything will look as it does in normal ezEML usage and you don't need to do any of the special handling described above. In that case, you won't see the monospaced fonts and checkmark buttons. Everything will look as you are used to seeing it in ezEML. And therefore, anyone working only with documents created in ezEML (as opposed to imported from XML) won't see any of what we've been discussing in this chapter.

One minor detail: you may have noticed in the example above that the text includes a "<" symbol as normal text, i.e., not as part of an XML tag. In the XML file, this symbol will have been encoded as "<" as required by the XML standard. ezEML displays it as "<" as a user would expect and then takes care of encoding it again when the XML file is saved. In other words, if you enter "<" or ">" as normal text, just enter them that way. You need to let ezEML do the encoding for you.