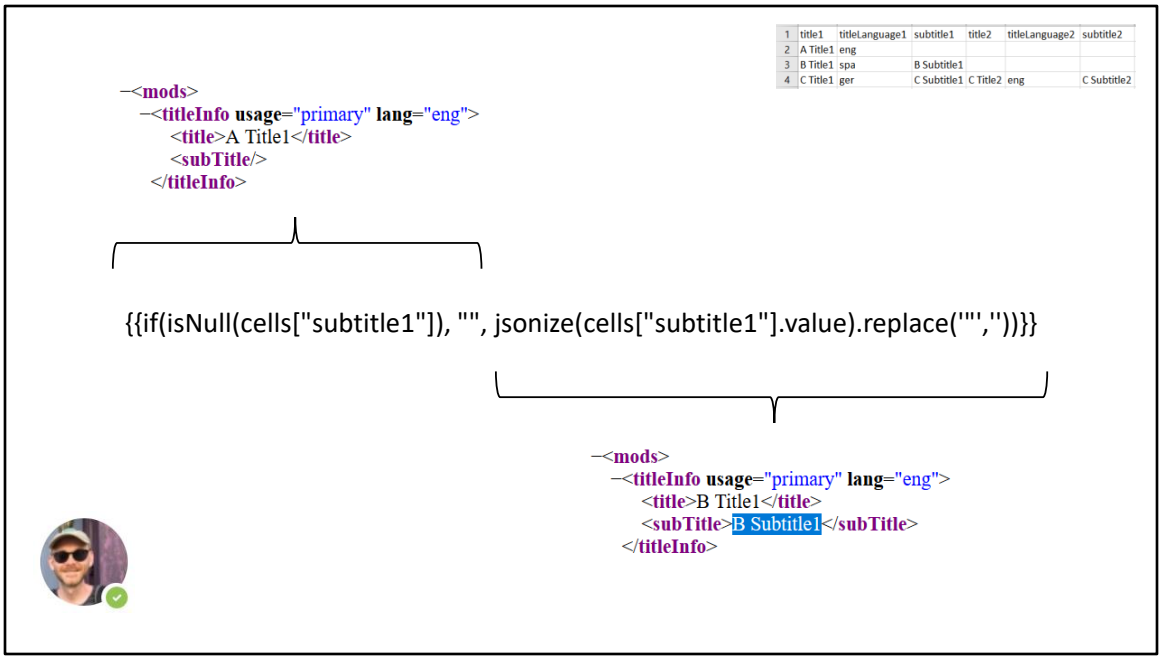
A revised approach to DAMS MODS XML

Each of us have varying levels of experience with entering metadata into the DAMS. There's two basic approaches: manual or bulk editing within the DAMS interface or batch creating MODS XML for batch ingesting large numbers of objects. Batch creating MODS XML currently requires meticulous template and spreadsheet encoding in order to avoid syntax and processing errors downstream. This also makes the current approach difficult to distribute among staff involved with metadata ingest. I wanted to see if we could revise the MODS XML batch creation approach. Where coding MODS templates with JSONIZE expressions and keying in custom spreadsheet headers, plus trying to escape reserved XML syntax so it doesn't clobber the next transformation step, could be avoided as much as possible, and save time, be free of headaches and be re-useable by anybody.

	title1	titleLanguage1	subtitle1	title2	titleLanguage2	subtitle2	description1	contributorName1	roleTerm1	dateCreated	subjectTopicTerm1	subjectTopicTerm2	subjectTopicTerm3	identifierNumberOrCode1
2	A Title1	eng					A Description1	A ContributorName1	A RoleTerm1	2020-04-29	A SubjectTopicTerm1			1234567
3	B Title1	spa	B Subtitle1				B Description1	B ContributorName1	B RoleTerm1	undated	B SubjectTopicTerm1	B SubjectTopicTerm2		2345678
4	C Title1	ger	C Subtitle1	C Title2	eng	C Subtitle2	C Description1	C ContributorName1	C RoleTerm1	1960s	C SubjectTopicTerm1	C SubjectTopicTerm2	C SubjectTopicTerm3	3456789

But before I talk about a new way to transform metadata, I'll start at the beginning, the source, our object metadata. Often times it's kept in spreadsheets like this example. A single collection can have varying metadata. Some objects have multiple titles or multiple subject terms, while others do not. This means we're in a conditional case, where an if-then statement written in a computer programming expression will help create the metadata we need (and leave out what we don't need) when collections do have varying metadata.



Thanks to Mirko, we have a conditional statement we can use in our MODS transformations. (He'll try to tell you it's just from the OpenRefine (G)eneral (R)efine (E)xpression L()anguage instead of taking credit… try not to fall asleep) In this example, our first object doesn't have a subtitle (take a look at the metadata in the upper right-hand corner), so our conditional resolves to creating what's known as an "empty-element tag" < />. Our second object does have a subtitle so our conditional statement resolves by inserting the subtitle text into the MODS.

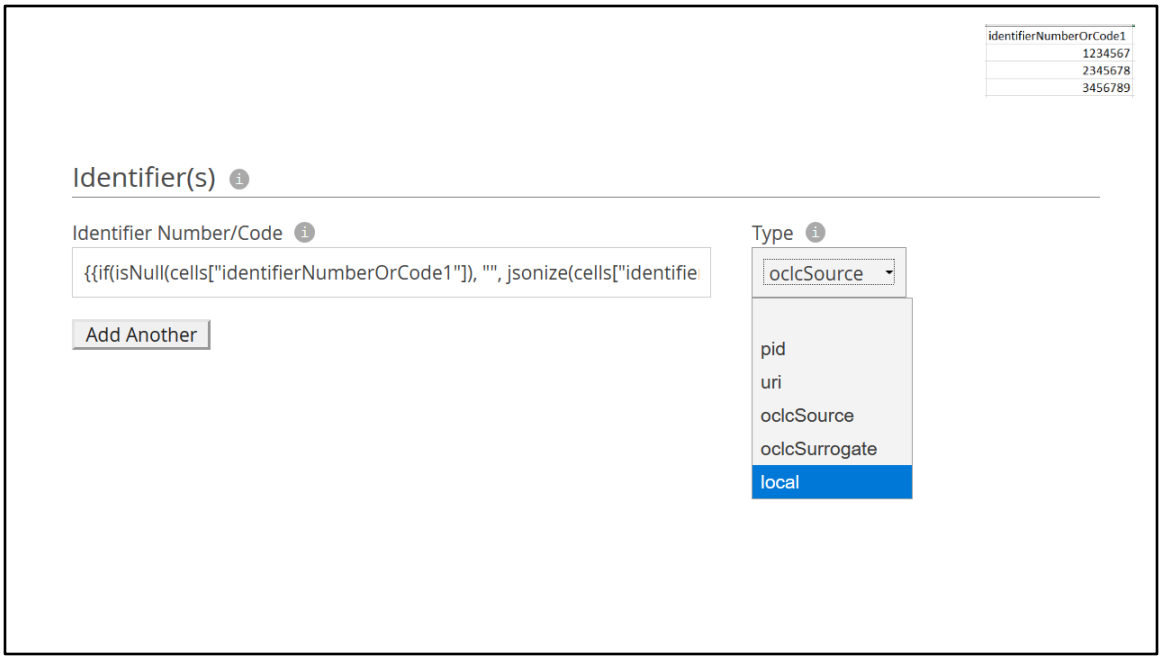
	DAMS MODS Field	Conditional Statement
2	Title *	{{if(isNull(cells["title1"]), "", jsonize(cells["title1"].value).replace('"',''))}}
3	Title Language *	{{if(isNull(cells["titleLanguage1"]), "", jsonize(cells["titleLanguage1"].value).replace('"',''))}}
4	Subtitle	{{if(isNull(cells["subtitle1"]), "", jsonize(cells["subtitle1"].value).replace('"',''))}}
5	Description *	{{if(isNull(cells["description1"]), "", jsonize(cells["description1"].value).replace('"',''))}}
6	Description Language *	{{if(isNull(cells["descriptionLanguage1"]), "", jsonize(cells["descriptionLanguage1"].value).replace('"',''))}}
7	Contributor Name *	{{if(isNull(cells["contributorName1"]), "", jsonize(cells["contributorName1"].value).replace('"',''))}}
8	Contributor Name Authority URI	{{if(isNull(cells["contributorNameAuthorityURI1"]), "", jsonize(cells["contributorNameAuthorityURI1"].value).replace('"',''))}}
9	Role Term *	{{if(isNull(cells["roleTerm1"]), "", jsonize(cells["roleTerm1"].value).replace('"',''))}}
10	Role Term Authority URI	{{if(isNull(cells["roleTermAuthorityURI1"]), "", jsonize(cells["roleTermAuthorityURI1"].value).replace('"',''))}}
11	Role Term Language *	{{if(isNull(cells["roleTermLanguage1"]), "", jsonize(cells["roleTermLanguage1"].value).replace('"',''))}}
12	Date Created/Date Created Start *	{{if(isNull(cells["dateCreated"]), "", jsonize(cells["dateCreated"].value).replace('"',''))}}
13	Date Created End (if applicable)	{{if(isNull(cells["dateCreatedEnd"]), "", jsonize(cells["dateCreatedEnd"].value).replace('"',''))}}
14	Publisher Name/Statement	{{if(isNull(cells["publisherNameOrStatement1"]), "", jsonize(cells["publisherNameOrStatement1"].value).replace('"',''))}}
15	Publisher Language (if Statement)	{{if(isNull(cells["publisherLanguageIfStatement1"]), "", jsonize(cells["publisherLanguageIfStatement1"].value).replace('"',''))}}
16	Place Name	{{if(isNull(cells["placeName1"]), "", jsonize(cells["placeName1"].value).replace('"',''))}}
17	Language of Place Name	{{if(isNull(cells["languageOfPlaceName1"]), "", jsonize(cells["languageOfPlaceName1"].value).replace('"',''))}}
18	Coverage Period, Date, or Date Range	{{if(isNull(cells["coveragePeriodDateOrDateRange1"]), "", jsonize(cells["coveragePeriodDateOrDateRange1"].value).replace('"',''))}}
19	Coverage Language (if Period)	{{if(isNull(cells["coverageLanguageIfPeriod1"]), "", jsonize(cells["coverageLanguageIfPeriod1"].value).replace('"',''))}}
20	Extent	{{if(isNull(cells["extent"]), "", jsonize(cells["extent"].value).replace('"',''))}}
21	Form/Medium Term	{{if(isNull(cells["formOrMediumTerm1"]), "", jsonize(cells["formOrMediumTerm1"].value).replace('"',''))}}
22	Form/Medium Authority URI	{{if(isNull(cells["formOrMediumAuthorityURI1"]), "", jsonize(cells["formOrMediumAuthorityURI1"].value).replace('"',''))}}
23	Language of Form/Medium Term	{{if(isNull(cells["languageOfFormOrMediumTerm1"]), "", jsonize(cells["languageOfFormOrMediumTerm1"].value).replace('"',''))}}
24	Rights – Use and Reproduction *	{{if(isNull(cells["rightsUseAndReproduction"]), "", jsonize(cells["rightsUseAndReproduction"].value).replace('"',''))}}
25	Rights – Restriction on Access *	{{if(isNull(cells["rightsRestrictionOnAccess"]), "", jsonize(cells["rightsRestrictionOnAccess"].value).replace('"',''))}}
26	Language Name *	{{if(isNull(cells["languageName1"]), "", jsonize(cells["languageName1"].value).replace('"',''))}}
27	Language Code *	{{if(isNull(cells["languageCode1"]), "", jsonize(cells["languageCode1"].value).replace('"',''))}}
28	Identifier Number/Code	{{if(isNull(cells["identifierNumberOrCode1"]), "", jsonize(cells["identifierNumberOrCode1"].value).replace('"',''))}}
29	Identifier Number/Code Type	{{if(isNull(cells["identifierNumberOrCodeType1"]), "", jsonize(cells["identifierNumberOrCodeType1"].value).replace('"',''))}}
30	Source Collection Name	{{if(isNull(cells["sourceCollectionName"]), "", jsonize(cells["sourceCollectionName"].value).replace('"',''))}}

I wondered if I could take it to the next step by pairing up the conditional statements we potentially want to see for every corresponding field available in the DAMS. That should mean we can create a MODS template rather quickly.

Here is the new approach. Enter a conditional statement for the maximum number of repeated fields for a given set of objects for ingest. Use the companion handout 'DAMS MODS Conditional Statements.xlsx' for easy cut and paste. Back in our sample metadata, shown in the upper right-hand corner, we have a third object with a title in German and an alternate title in English, so we enter a conditional statement into two title form fields, making sure to distinguish the second title and attributes with a number '2'

Added bonus, if your browser form field history is enabled (which by default it is) you will always have the option to select the last conditional statement you entered for that field from your browser form field history. So the next time you need to create a MODS template, it's going to be a lot quicker.

Again, using conditional statements in the form editor will create a ready to use MODS template.



A note about the field Identifiers. This will sound familiar to those using the current approach to creating MODS XML as documented in the DAMS wiki. This note also applies to the new approach I'm speaking about today. The Identifiers field is optional in the DAMS form (there's no red asterisk hovering next it), but Identifiers **is required** if using a script that renames files based on the Identifier XML node in order to create matching XML and object filenames. Our objects are uniquely named, oftentimes with OCLC or local identifiers embedded in the filename. Here in the upper right-hand corner, our spreadsheet has OCLC number identifiers but it can also represent local identifiers (we could also leave the attribute type blank.) When we get to file renaming later, we'll use a script, which is available for download in the DAMS wiki. It looks for MODS:Identifier nodes and copies the text values into the XML filenames so that they match the object filenames. But renaming talk is for another time. Let's get back to finishing our MODS with conditional statements.

OK we're done. I'm not showing all the fields, but let's assume we're done with filling in all our conditional statements (as well as any fixed metadata) and this is now what our MODS template looks like. After downloading it from the Manage Datastreams tab in the DAMS, we'll need to make one or two edits in a text editor, depending on if you used conditional statements for attributes instead of hard coding text. In the example, we used conditional statements for the titleLanguage attribute, so we will need to unescape the characters in the selection between the double quotes (") in order to get the conditional statement to function in OpenRefine. If you use Notepad++ and you have the highly recommended XML Tools plugin installed, you can find the menu command for unescaped characters under Plugins>XML Tools>"Unescape characters in selection…"
Our MODS template is almost ready for use in OpenRefine.

```
      <languageTerm type="code" authority="iso639-2b" authorityURI="http://id.loc.gov/vocabulary/iso639-2">eng</languageTerm>
    </languageOfCataloging>
  </recordInfo>
  <subject lang="eng">
    <topic>
      {{if(isNull(cells["subjectTopicTerm1"]), "", jsonize(cells["subjectTopicTerm1"].value).replace("","))}}
    </topic>
  </subject>
  <subject lang="eng">
    <topic>
      {{if(isNull(cells["subjectTopicTerm2"]), "", jsonize(cells["subjectTopicTerm2"].value).replace("","))}}
    </topic>
  </subject>
  <subject lang="eng">
    <topic>
      {{if(isNull(cells["subjectTopicTerm3"]), "", jsonize(cells["subjectTopicTerm3"].value).replace("","))}}
    </topic>
  </subject>
  <identifier type="utldamsPID">utlmisc:4b8c15d0-d670-4647-a826-2353cbdb2247</identifier>
  <identifier type="utldamsURI">
    dams-t01-rh7.lib.utexas.edu/islandora/object/utlmisc:4b8c15d0-d670-4647-a826-2353cbdb2247
  </identifier>
  <identifier type="fileName">Conditional Statements.tiff</identifier>
  <relatedItem displayLabel="UTLDAMS Digital collection" type="host">
    <identifier type="utldamsPID">utlmisc:15ccb766-d041-443b-9331-5089341627ac</identifier>
    <identifier type="utldamsURI">
      dams-t01-rh7/islandora/object/utlmisc:15ccb766-d041-443b-9331-5089341627ac
    </identifier>
    <titleInfo displayLabel="UTLDAMS Digital collection name">
      <title>How to Create a MODS template</title>
    </titleInfo>
  </relatedItem>
</mods>
```

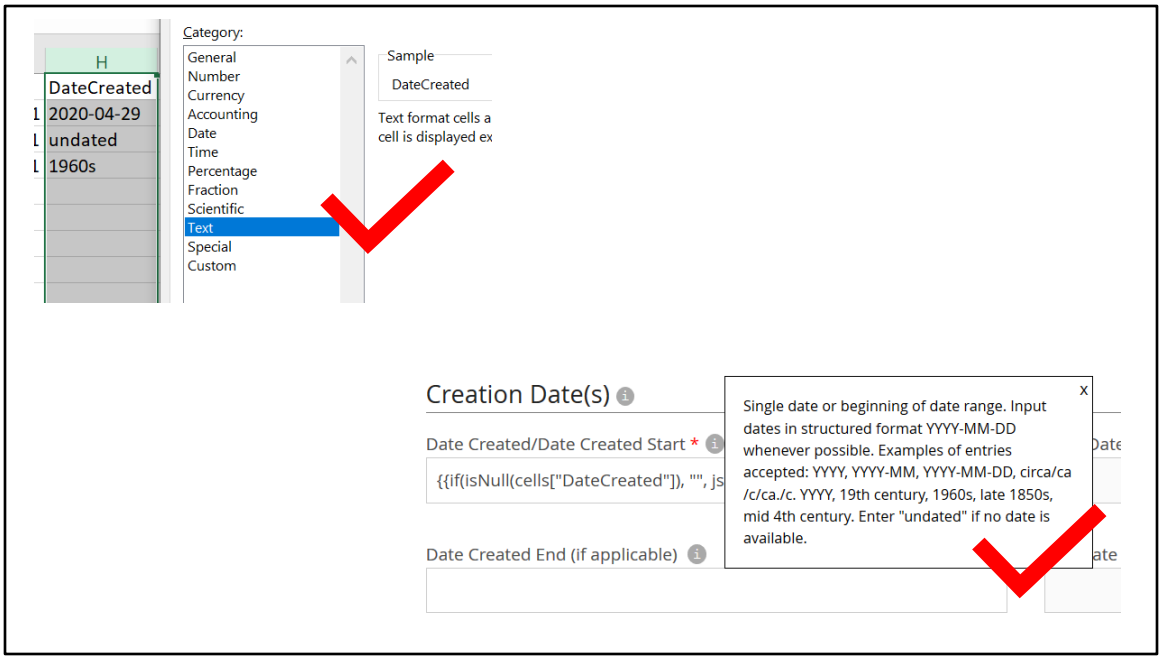
We just need to delete the machine generated info.

```
−<mods>
 −<titleInfo usage="primary" lang="{{if(isNull(cells["titleLanguage1"]), "", jsonize(cells["titleLanguage1"].value).replace("","))}}">
  −<title>
     {{if(isNull(cells["title1"]), "", jsonize(cells["title1"].value).replace("","))}}
   </title>
  −<subTitle>
     {{if(isNull(cells["subtitle1"]), "", jsonize(cells["subtitle1"].value).replace("","))}}
   </subTitle>
 </titleInfo>
 −<titleInfo lang="{{if(isNull(cells["titleLanguage2"]), "", jsonize(cells["titleLanguage2"].value).replace("","))}}">
  −<title>
     {{if(isNull(cells["title2"]), "", jsonize(cells["title2"].value).replace("","))}}
   </title>
  −<subTitle>
     {{if(isNull(cells["subtitle2"]), "", jsonize(cells["subtitle2"].value).replace("","))}}
   </subTitle>
 </titleInfo>
 −<abstract displayLabel="Description" lang="eng">
    {{if(isNull(cells["description1"]), "", jsonize(cells["description1"].value).replace("","))}}
 </abstract>
 −<name displayLabel="Contributor name" type="personal">
  −<namePart>
     {{if(isNull(cells["contributorName1"]), "", jsonize(cells["contributorName1"].value).replace("","))}}
   </namePart>
  −<role>
```

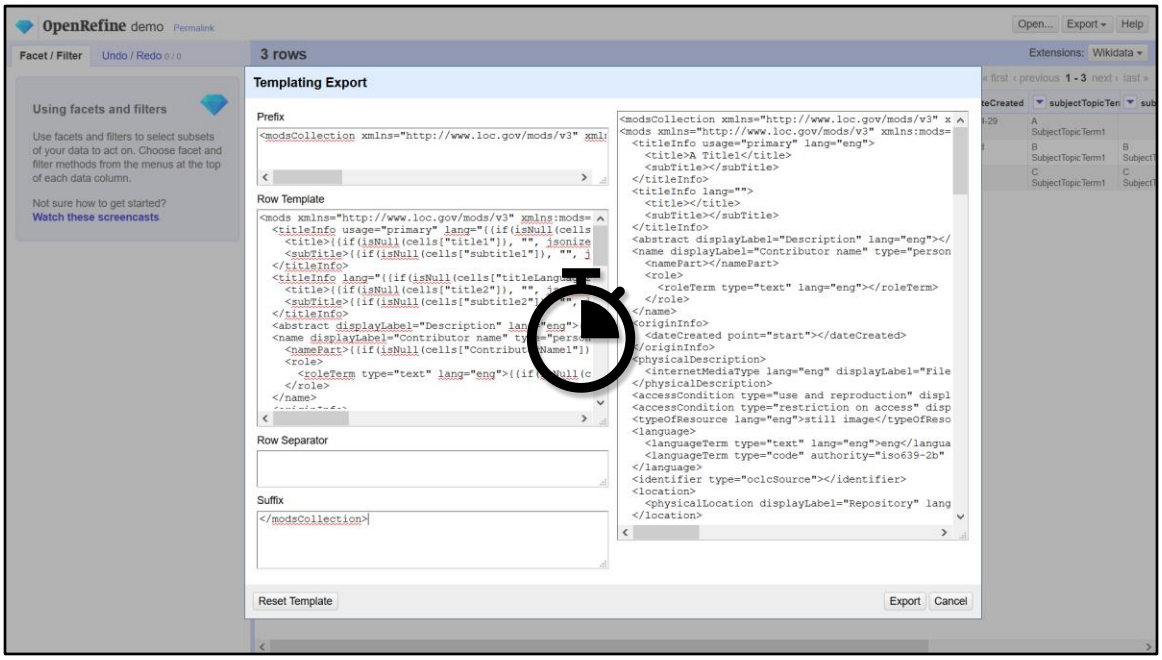
	title1	titleLanguage1	subtitle1	title2	titleLanguage2	subtitle2	description1	contributorName1
2	A Title1	eng					A Description1	A ContributorName1
3	B Title1	spa	B Subtitle1				B Description1	B ContributorName1
4	C Title1	ger	C Subtitle1	C Title2	eng	C Subtitle2	C Description1	C ContributorName1

Our conditional statement “values” were created with “camelCase” (no separators like hyphens and underscores) to facilitate quick copying and pasting into a DAMS specific spreadsheet.

Yes, this new approach involves creating a DAMS specific spreadsheet. (Feels like more work) But I think the tradeoff is worth it. Common MODS template coding errors in the old approach are avoided. Downstream processing in OpenRefine is simplified. Plus, if it matters, original source metadata spreadsheets can retain their original usage, while a specific spreadsheet of DAMS specific metadata is obtained.



Also, don't forget to change your DateCreated cell format to 'Text' to avoid "auto-mangling" cell data and accommodate the recommended input provided in the DAMS information box.



Time spent in OpenRefine was only on uploading the DAMS specific metadata sheet and pasting in my MODS template. No more coding errors encountered and troubleshooting syntax. I imagine even for large numbers of records, time spent in OpenRefine would just be minutes, not longer like it can be in the current approach that can involve troubleshooting unexpected output.

Reference

https://dams-t01-rh7.lib.utexas.edu/islandora/object/utlmisc%3A15ccb766-d041-443b-9331-5089341627ac/btitle

A before and after look of DAMS MODS metadata is available in DAMS-T01 if you follow the link. Please be aware that the link contains a lot of MODS conditional testing that I did and continue to do. Ideally there will be a reference spot in the future, if widely adopted, where things are left untouched and are ideal representations. I also recommend making your own test collection with conditional values filled into the MODS form. Let me know if you try this new approach to generate a MODS template. The ultimate goal is to simplify processes and increase our ingest rates, and I think this new approach takes significantly less time (I'm talking minutes, instead of hours or days) once things are setup. I hope this helps so please let me know if you try it out. I can send you the MODS form field to conditional statement pairings shown earlier.

Considerations:

If we are creating “empty-element tags” in the DAMS,

e.g. <title/> <subTitle/>

what’s the impact on the database and storage systems (backend)? My guess is it’s insignificant. Each character occupies a byte. If there are very large numbers of empty-element tags stored in the DAMS, that eventually leads to undesirable storage consumption. (It’ll be awhile, 1 million characters is only 1MB) Left totally unchecked in the long term it could be an impact. DAMS-IT consult would be a good course of action before producing significant numbers of empty-element tags. If the opinion is negative that would call for producing minimal “empty-element tags” within reason.

Are empty attributes, e.g. <titleInfo lang=""> a problem? Doesn’t seem to be invalid XML syntax. What about schema validity.

Testing publishing to the public facing side should be performed.

What else is wrong with this approach?