Checking Pointers Using your TEI ODD Customization

Syd would like to see TCW32 published (as TCUxx, or just Uxx?) in October, if possible. Reviewers (which should include you, James, as you are assigned to this ticket) should:

- Read the prose, make any obvious changes necessary, check right back into master branch.
- Read the prose, inform Syd or post any controversial suggested changes. Use a new branch and PR if you make extensive changes.
- Try out the examples, make sure they do what they say they will do. Since there are 17 of them, and some are quite complex, and in most cases you will have to make a test file, it might be nicer to divvy them up among several people.

Here is a list of the 19 snippets of ODD in TCW32 that need to be tested. If folks reading this could each grab one or two at random and test, that would be lovely.

P.S. If you want to generate HTML to read, one way is to use the TEI Stylesheets, in which case you probably want to set the $numberBackFigures parameter to “true”.

Example	Initials	Comments
#eg01: PureODD to limit attribute to one URI	MDH	This doesn’t seem to work as shown. I get this in my schema under <element name="s">: `<optional>` `<attribute name="corresp">` `<a:documentation xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">(corresponds) points to elements that correspond to the current element in some way.</a:documentation>` `<data type="anyURI"/>` `</attribute>` `</optional>` but when I validate this: `<s corresp="tei:thing tei:thing2">thing</s>` I don’t get an error. I must be missing something obvious. SB: MDH is on to a bigger kettle of fish, here. See ticket 2185. The fix, I think, is to the *Guidelines*, not here.
#eg02: Ensure ref of g is a shorthand pointer	HBS	It works as expected. No suggestions.
#eg03: Ensure ref of g is a shorthand pointer, PureODD	MS	It works as described. The remark of the element spec could possibly be changed to “This attribute must be a single shorthand pointer to an element in the same XML file, i.e. must look like” SB: Wording change made, not checked in yet.
#eg04: Check that ref of g points to something	HBS	Tested in combination with #eg02. It works as expected. No suggestions.
#eg05: Check that ref of g points to a char or a glyph	HBS	Tested in combination with #eg02 and #eg04. It works as expected. The documentation of this examples says that it should be used in combination with something like #eg02 or #eg03, but #eg04 is not mentioned. If the constraint of #eg04 is not explicitly recommended in the documentation, I wonder if the error message of #eg05 shouldn‘t include a condition like the one used in #eg14 so we get the word “nothing” if the pointer does not point to an existing element. If the constraint of #eg05 is used with the one in #eg04, it wouldn’t be necessary, because, if the pointer does not exist, we will get the error message of #eg04. However, without the constraint of #eg04, the error message of #eg05 would look “weird”. SB: Good point. Two possible solutions. Given that the structure of the document is to try to organize the examples from easier to more complex, I think I lean towards solution #2. 1. Fix the error message by changing `points to a '<sch:value-of select="local-name( id( $ref ) )"/>'` to `points to <sch:value-of select="if ( id( $ref ) ) then concat('a '', local-name( id( $ref ) ),'' element') else 'nothing'"/>'.`

		2. Add a sentence or paragraph of prose “We also presume that the <att>ref</att> of <gi>g</gi> actually points to <emph>something</emph> in the current document. Thus this constraint should be used in conjunction with something like <ptr target="#eg04"/>”. HBS: I also prefer solution #2 (and the change in the prose is consistent with the mentions to previous constraints) SB: OK, prose changed (although not exactly like above; I combined the 2 sentences into one). Fixed in 3c8c46d.
#eg06: Require the url of moduleRef to refer to a particular file	MS	Works.
#eg07: Check that url of moduleRef refers to an RNG file in the same directory	MS	Tested. This also works when the referenced RNG file is not in the same directory as the ODD file.
#eg08: Ensure file is readable XML	LB	Tested. As noted by MDH below, if the filename supplied uses a relative path, it is understood to be relative to the location of the schema file. Maybe supplying the path explicitly in the error message would help: like this: “No valid XML content was found at the supplied URL (out/wibble.xml)” SB: Oy vey. Thanks for catching the *same* boo-boo I made in #eg12 (and #eg09). Sigh. Fixed (the same way, adding `resolve-uri()`) in cf6e9d5.
#eg09: Ensure file is readable RELAX NG grammar	MS	Tested. This does not work for me. Error message: Module ./out/tester.rng is not readable, well-formed XML. I wonder if this has something to do with the Schematron preferences in Oxygen (b/c #eg08 doesn’t work for me either). SB & MS played a bit, and it seems that (at least in MS’s oXygen setup) the schema is being looked for at the URL *relative to the schema, not the instance*, despite the use of `resolve-uri()`.
#eg10: Require persName to refer to a person in the local personography	EBB	There's an outright error in defining the sch:variable $file, since it's pointing to the substring-after(., '#'). SB: I do not understand what is going on here, EBB, but in both the version that is checked into GitHub (line 786) and the one on my local repo (line 792) $file is defined as substring-before(.,'#'). [Note: I have since pushed my prose changes in, so it is now line 792 in the repo, too, but that link is a permalink to the previous version]. But the $file variable isn't easy to correct, because later there is this: <sch:let name="element_found" value="doc($file)//id($ID)"/> doc($file) simply doesn't work if $file is defined as a filename or filepath. Evidently we need to define $file as the whole doc() function like so: <sch:let name="file" value="doc('../persons.xml')"/>

		Then you can define $element_found thus: <sch:let name="element_found" value="$file//id($ID)"/> NOTE: (as I think MS and SB found earlier), if people generate RNG in an out/ directory, below the ODD, the file being validated, and person.xml, the path to person.xml must be doc('../person.xml') not doc('person.xml'). SB: Sigh. Will have to look into this more thoroughly. After redefining these variables, all works as expected. However, I think we can do better: What if we have multiple space separated values of @ref? SB: This example is in the section “Direct reference by single value”. There are examples of how to handle multiple values in the section “Direct access by multiple values”. (I just noticed now those two headings are not parallel — which is better?) SUGGESTED REVISION: <sch:let name="IDs" value="for $i in tokenize(@ref, '\s+') return substring-after($i,'#') "/> Then test those tokens like you do in eg15, maybe something like this: <sch:let name="IDs_pt_to_person" value="for $id in $IDs return exists($file//person[@xml:id = $id]"/> <sch:report test="$IDs_pt_to_person = false()"> At least one of the values of @ref in "<sch:value-of select="normalize-space(.)"/>" does not end up pointing to a 'person' element. </sch:report> CAUTION: I have not tested this construction, but I have something similar running in the Digital Mitford ODD to test multiple space-separated @ref values. Suggestion 2: Since it seems you've arranged these examples from easier to more complex, how about we create two versions of eg10, one to test singleton values of `@ref` and one to test a sequence of `@ref` values?
#eg11: Ensure uri of equiv is present and refers to an item on a particular page	RV	Tested. It works and only allows NMTOKEN after # (if I understand the regex correctly). So this is considered valid: http://www.wwp.neu.edu/markup_taxonomy.xhtml#_asd But this is not: http://www.wwp.neu.edu/markup_taxonomy.xhtml#1asd It *may* refer to an item on a particular page, but it doesn’t stop the user from referring to a non-existing id.

		SB: I *think* I understand your concerns, here: 1) The regex requires that the shorthand reference be an xs:Name, and 2) the test only checks that we are pointing to an item on a particular web page, but does not check that the item exists. 1) Right. The target is XHTML, so “1asd” is not a valid identifier. The prose in the introduction is quite clear that TCW32 only discusses XML documents (including XHTML), not HTML or JPEG documents. 2) Absolutely true, by design — it is only *supposed* to check the syntax. But a) the prose does not explain that well, and b) there probably should be an example that shows how to check that there is something retrievable there. So I think there is some work for me to do here re-writing the prose around #11 and adding a new example. RV: good idea making this more clear in the prose. SB: Prose in subsection “remote pointers” updated. RV: Not sure we need an example that shows how to check that there is something retrievable there: I think that’s out of scope of schematron and I’m not sure we should make a recommendation on which tools to use to retrieve, parse, and check the target document. SB: I disagree in that I think we *do* need an example that shows how to check there is something retrievable there (it is in scope of Schematron, IMHO, and part of point of this document is to demonstrate how to do so); I agree in that I no longer think we need *another* example. I am thinking now that #eg11 is sufficient (since it does retrieve the document) with an added note that points this out.
#eg12: Require a filter on equiv that points to an XSLT program	MDH	Tested this one, and although it works, there is one caveat: a relative path to the XSLT file must be relative to the *schema*, not to the XML instance file. SB: Good catch! I also forgot normalize-space(). Fixed in 72dcf89.
#eg13: rendition points to 1 or 2 renditions	RV	Tested, all assertions work.
#eg14: Check that each pointer in wit points to witness	HBS	Tested and I couldn’t find any issues. The error message is very detailed and it is extremely helpful how it handles multivalued cases. The first time I read the error message, though, I wondered for a second if the pronoun “they” referred to all the pointers or only to the ones that were incorrect. My suggestion to make it less ambiguous is a bit longer than the original: One (or more) of the pointers of this @wit does (do) not point to a 'witness' element~~;~~ ~~they~~. The pointers in this @wit point to the following items, in the order specified:

		SB: Sounds reasonable (if verbose) to me. Updated at 0734820.
#eg15: Required ref of persName eventually refers to person	EBB	
#eg16: Required ref of persName eventually refers to person using abstract patterns	LB	Tested. The simple case (value of who points to a person) works. But more complex ones did not. E.g. it accepted the following as valid `<body><p><persName ref="#who"></persName></p><p><alt xml:id="who" type="person" target="#smith #jones"/></p></body>` (when smith and jones are not defined). But maybe this is tag abuse of <alt>? SB: Doesn't seem to be tag abuse of <alt> to me, at least not when smith and jones are properly defined. But you have definitely caught an erroneous case, here. Code seems to work fine if smith and jones are mis-defined (i.e., are <place> or <emph> elements), but fails if they are simply pointers to nowhere. Sigh. Working on it. Fixed in e5af56a.
#eg17: Resolve prefixDef for references to people	MS	
#eg18: XIncluded yet? — one size fits all	HBS	I am confident it works as expected, but I am not sure how to play with this one to make it fire: the editors I use perform the XInclude processing very, very quickly. I tried looking into the settings of Oxygen to see if I could disable the XInclude processing, but I only saw it as an XSL option. SB: Not sure what is going on here, HBS. In oXygen 24.0 this is controlled by the "Enable XInclude Processing" checkbox in the Options > Preferences > XML > XML Parser pane. Can be found by searching for either "XInclude", or more precisely "fix-up" in the preferences search box. HBS: Found it! I have an older version, but the location of that checkbox was the same. So indeed now I can confirm that if you disable that option, the error fires.
#eg19: XIncluded yet? — per-test reporting	HBS	All the rules worked. However, the expected syntax of @target does not seem to be compliant with the URI generic syntax (fragments should begin with '#', right?). If in this case @target should only include one URI (for some reason), then I think that the variable $target should be defined as: `<sch:let name="target" value="normalize-space(@target) => substring-after('#')"/>`

		However, if the example supposes that @target can contain more than one URI, then the definition should be something similar to this: <sch:let name="target" value="for $x in tokenize(@target, '\s+') return substring-after($x, '#')"/> SB: Good catch! Right you are about the missing #. We know this example is for only 1 pointer because the caption paragraph immediately below the example says “the @target of <catRef> is a single shorthand pointer”. Will fix shortly. (Although I am going to use the XSLT 2 syntax, substring-after(normalize-space(@target),’#’), as some Schematron processors still don’t do XSLT 3 binding.) Fixed at **759623d.**

MS: line 162: and then cases where the an item is being referred to indirectly
SB: Fixed, 2021-11-10, checked in at 535953ed…404a8016, I think.