Saxon-JS - Support #4797

Unknown collection error FODC0002

2020-10-19 14:05 - Nik Osvalds

Status:	Closed	**Start date:**	2020-10-19
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0:00 hour
Sprint/Milestone:		**Spent time:**	0:00 hour
Applies to JS Branch:	2	**SEF Generated with:**	
Fix Committed on JS Branch:		**Platforms:**	
Fixed in JS Release:			

Description

I'm attempting to run saxon-js in NodeJS to apply XSL 3 transformations currently working in another system with Saxon HE, because saxon-js looks like it can offer a lot more versatility.

I am essentially brand new to XSL so the learning curve is steep.

I'm trying to run the iati.xslt transform (attached) on a sample IATI file using saxon-js for Node.js. I've converted the .xslt file on the command line to .sef.json and am calling the transformation in Node. I'm receiving the following error which seems to indicate the the collection(...)'s on line 90 of the .xslt cannot be resolved. I've ensured that the files specified in the collection() are available in my project files.

```
            // Applying the XSLT3 Ruleset to IATI Files Using SaxonJS
            let results = await SaxonJS.transform({
                stylesheetFileName: "./rules/iati.sef.json",
                sourceFileName: filePath,
                destination: "file",
                baseOutputURI: "./file_storage/validated/" + xmlIn.md5 + '.xml',
            }, "async")
```

```
message: 'Unknown collection (no collectionFinder supplied)',
code: 'FODC0002',
xsltLineNr: '90',
xsltModule: 'iati.xslt',
```

```
 <xsl:variable name="iati-codelists">
    <codes version="2.03">
      <xsl:apply-templates select="
collection('../lib/schemata/2.03/codelist/?select=*.xml;recurse=yes')" mode="get-codelists"/>
      <xsl:apply-templates select="
collection('../lib/schemata/non-embedded-codelist/?select=*.xml;recurse=yes')" mode="get-codelists
"/>
    </codes>
  </xsl:variable>
```

I see that you can use the collectionFinder parameter in the transform function but it's unclear how this should be implemented and I've only found documentation on implementing a collectionFinder in Java.

Is this something implemented in Saxon HE (which is presently doing the work) and not currently in Saxon-Js?

History

#1 - 2020-10-19 15:08 - Michael Kay

I'm afraid that support for the collection() function in Saxon-JS is minimal: essentially, the function only works if you supply a CollectionFinder as a parameter, and the CollectionFinder has to do all the work. This restriction was in the browser version of Saxon-JS because there's not much else you can usefully do in the browser, and we didn't get around to lifting the restriction for Node.js. It's on our list for a future release.

The CollectionFinder callback is simply a Javascript function that is called supplying the (absolutized) collation URI, and that returns any value (the Javascript value is converted to an XDM value in the usual way); this value is then returned as the result of the collection() function. There's no real

advantage of doing this rather than calling an external Javascript function, other than making your XSLT code a bit more portable.

#2 - 2020-10-20 13:00 - Nik Osvalds

Thanks for the swift response Michael. I've started to write my own collectionFinder but I'm struggling to determine what it should return.

Taking the example XML shared above the original collect() function is finding a directory with several .xml files of codelists. So I've written this into a JS function that's passed to the transform and it's returning an Array of those files in utf8 format. After doing this I'm not seeing the effect of these files being loaded on the transform. Do I need to join these files together and return one object instead?

```
const collectionFinder = async (uri) => {
    let loaded = [];
    let path = uri.split('file://')[1].split('?')[0];
// get the right filepath (remove file:// and after the ?
    let files = await fsPromises.readdir(path)
    if (files) {
        loaded = files.map(async (file) => {
            return await fsPromises.readFile(path + file,'utf8')
        })
    }
    return await Promise.all(loaded)
}
```

#3 - 2020-10-21 18:12 - Martin Honnen

I am not sure the collectionFinder can work asynchronously. As for having a collection of XML documents, I think you need to make sure you pass in nodes, meaning in the context of Saxon-JS 2 it looks like you need to use SaxonJS.getResource to build nodes. Short sample that worked for me:

```
require('saxon-js');

(async () => {
    try {
        let documents = [ await SaxonJS.getResource({ type : 'xml', file : 'files/input1.xml'
 }), await SaxonJS.getResource({ type : 'xml', file : 'files/input2.xml' })];

        let result = await SaxonJS.transform({
            stylesheetFileName: 'sheet.sef.json',
            destination : 'serialized',
            collectionFinder: (url) => {
                if (url === 'http://example.com/col1')
                {
                    return documents;
                }
                else {
                    return [];
                }
            }
        }, 'async');
        console.log('Result: ' + result.principalResult);
    }
    catch (e) {
        console.log(e);
    }
})();
```

Of course given that I build the collection/array of nodes "before" running the transformation it might be easier to just pass them in as a global parameter instead of using the collection function.

#4 - 2020-10-22 12:39 - Nik Osvalds

That did the trick. Thank you so much! I was not able to get the async version of collectionFinder to work as you said, but following your example I was able to get it to pull all the collection files I needed. This is just a proof of concept for us to transition from Saxon HE to Saxon JS so I'm okay with it being a bit "hardcoded" for now.

```
// load codelists since collectionFinder can't be async
            let codelistPaths = [
                "non-embedded-codelist/",
                "2.03/codelist/",
                "2.02/codelist/",
                "2.01/codelist/",
                "1.05/codelist/",
                "1.04/codelist/",
                "1.03/codelist/"
            ];
```

```
// this returns an object of the codelistPaths as Keys and an Array of resolved promises for the Values. these
 promises are grabbing the codelist XML files
            let resources = _.zipObject(codelistPaths, await Promise.all(_.map(codelistPaths, async (path) =>
{
                let files = await fsPromises.readdir("./IATI-Rulesets/lib/schemata/" + path);
                return await Promise.all(files.map(async (file) => {
                    return await SaxonJS.getResource({ type : 'xml', file : "./IATI-Rulesets/lib/schemata/"
 +  path + file })
                }))
            }))

            // this pulls the right array of SaxonJS resources from the resources object
            const collectionFinder = (url) => {
                if (url.includes("codelist")) {
                    let path = url.split('schemata/')[1].split('?')[0];
// get the right filepath (remove file:// and after the ?
                    return resources[path]
                } else {
                    return []
                }
            }

            // results filepath
            let resultsPath = __dirname + "/file_storage/validated/" + xmlIn.md5 + '_results.xml'
            // Applying the XSLT3 Ruleset to IATI Files Using SaxonJS
            let results = await SaxonJS.transform({
                stylesheetFileName: "./IATI-Rulesets/rules/iati.sef.json",
                sourceFileName: filePath,
                destination: "serialized",
                collectionFinder: collectionFinder
            }, "async")
```

```
<xsl:variable name="iati-codelists">
    <codes version="2.03">
      <xsl:apply-templates select="collection('../lib/schemata/2.03/codelist/?select=*.xml;recurse=yes')" mode
="get-codelists"/>
      <xsl:apply-templates select="
collection('../lib/schemata/non-embedded-codelist/?select=*.xml;recurse=yes')" mode="get-codelists"/>
    </codes>
    <codes version="2.02">
      <xsl:apply-templates select="collection('../lib/schemata/2.02/codelist/?select=*.xml;recurse=yes')" mode
="get-codelists"/>
      <xsl:apply-templates select="
collection('../lib/schemata/non-embedded-codelist/?select=*.xml;recurse=yes')" mode="get-codelists"/>
    </codes>
    <codes version="2.01">
      <xsl:apply-templates select="collection('../lib/schemata/2.01/codelist/?select=*.xml;recurse=yes')" mode
="get-codelists"/>
      <xsl:apply-templates select="
collection('../lib/schemata/non-embedded-codelist/?select=*.xml;recurse=yes')" mode="get-codelists"/>
    </codes>
    <codes version="1.05">
      <xsl:apply-templates select="collection('../lib/schemata/1.05/codelist/?select=*.xml;recurse=yes')" mode
="get-codelists"/>
    </codes>
    <codes version="1.04">
      <xsl:apply-templates select="collection('../lib/schemata/1.04/codelist/?select=*.xml;recurse=yes')" mode
="get-codelists"/>
    </codes>
    <codes version="1.03">
      <xsl:apply-templates select="collection('../lib/schemata/1.03/codelist/?select=*.xml;recurse=yes')" mode
="get-codelists"/>
    </codes>
  </xsl:variable>
```

#5 - 2021-01-12 14:10 - Nik Osvalds

So I've realised that this isn't actually working correctly for me.

I've narrowed it down to the fact that my collectionFinder function is returning an Array of nodes, rather than just a single node. When I change my collectionFinder just to load one file using Saxon-JS.getResource() and therefore just return a single node then my transform works as expected.

select=collection() on a directory of .xml files with root elements <codelist name="someUniqueName">

```
select="collection('../lib/schemata/2.03/codelist/?select=*.xml;recurse=yes')"
```

File-structure:

- 2.03/codelist/
  - ActivityDateType.xml
  - ActivityStatus.xml
  - ...

Array returned in JS:

```
(9) [Document$$module$temp$js$source$nodeJS$xmldom$dom, Document$$module$temp$js$source$nodeJS$xmldom$dom, Doc
ument$$module$temp$js$source$nodeJS$xmldom$dom, Document$$module$temp$js$source$nodeJS$xmldom$dom, Document$$m
odule$temp$js$source$nodeJS$xmldom$dom, Document$$module$temp$js$source$nodeJS$xmldom$dom, Document$$module$te
mp$js$source$nodeJS$xmldom$dom, Document$$module$temp$js$source$nodeJS$xmldom$dom, Document$$module$temp$js$so
urce$nodeJS$xmldom$dom]
```

Is there a way to "concatenate" the array of nodes into just one node?

Thanks! Nik

#6 - 2021-01-15 13:19 - Nik Osvalds

I solved this.

When trying to apply iati.xslt transform on an IATI xml file with Saxon-JS the codelists rules were not being applied. I traced this back to the following error: xsl:message evaluation at iati.xslt#-1 failed: XError:Circularity in global variable Q{}iati-codelists; code:XTDE0640

This seems to have occurred because the following templates were declared before the "iati-codelists" variable was defined.

```
  <xsl:template match="codelist" mode="get-codelists">
    <xsl:copy>
      <xsl:copy select="@name"/>
      <xsl:apply-templates select="//code" mode="get-codelists"/>
    </xsl:copy>
  </xsl:template>

  <xsl:template match="code" mode="get-codelists">
    <xsl:copy>{.}</xsl:copy>
  </xsl:template>

  <xsl:variable name="iati-codelists">
    <codes version="2.03">
      <xsl:apply-templates select="collection('../lib/schemata/2.03/codelist/?select=*.xml;recurse=yes')" mode
="get-codelists"/>
      <xsl:apply-templates select="collection('../lib/schemata/non-embedded-codelist/?select=*.xml;recurse=yes
')" mode="get-codelists"/>
    </codes>
    <codes version="2.02">
      <xsl:apply-templates select="collection('../lib/schemata/2.02/codelist/?select=*.xml;recurse=yes')" mode
="get-codelists"/>
      <xsl:apply-templates select="collection('../lib/schemata/non-embedded-codelist/?select=*.xml;recurse=yes
')" mode="get-codelists"/>
    </codes>
...
  </xsl:variable>
```

Swapping the order of these resolved the issue in Saxon-JS.

So looks like it was not a problem with my implementation of collectionFinder after all.

#7 - 2021-02-11 14:08 - Debbie Lockett

- Applies to JS Branch 2 added

#8 - 2021-04-30 15:17 - Norm Tovey-Walsh

- Status changed from New to Closed

It sounds like you've worked through the issues you were having. I'm closing this issue in favor of #4809 which is specifically about the (as yet unresolved) question of asynchrony in collection finder functions.

Files

iati.xslt	5.26 KB	2020-10-19	Nik Osvalds