Saxon-JS - Bug #5074

"XML Parsing Error" when loading SEF file with Saxon-JS 2.3

2021-09-01 14:35 - Martynas Jusevicius

Status:	Resolved	**Start date:**	2021-09-01
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

Not exactly sure that is the reason, but the upgrade from 2.2 to 2.3 coincides with this error message appearing when SEF is being loaded:

```
XML Parsing Error: not well-formed
Location: https://localhost:4443/static/com/atomgraph/linkeddatahub/xsl/client.xsl.sef.json
Line Number 1, Column 1: client.xsl.sef.json:1:1
SEF generated by Saxon-JS 2.3 at 2021-09-01T13:32:15.961+02:00 with -target:JS -relocate:true
```

Execution seems to proceed normally after that. I wonder if there have been changes in this area?

History

#1 - 2021-09-01 14:37 - Martynas Jusevicius

To be clear: I'm using the browser, and the log above is from the console log.

#2 - 2021-09-01 16:46 - Martynas Jusevicius

It looks like 2.3 began using conditional requests (If-Modified-Since/If-None-Match) when requesting SEF? Which makes total sense for performance reasons, but since a 304 Not Modified response does not include a Content-Type header, maybe this messes up the media type recognition somehow?

Saxon-JS 2.3

```
GET /static/com/atomgraph/linkeddatahub/xsl/client.xsl.sef.json HTTP/1.1
Host: localhost:4443
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://localhost:4443/
Connection: keep-alive
Cookie: _ga=GA1.1.828629977.1584086266; LinkedDataHub.first-time-message=true; _octo=GH1.1.128689066.160763774
8
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: no-cors
Sec-Fetch-Site: same-origin
If-Modified-Since: Wed, 01 Sep 2021 14:11:02 GMT
If-None-Match: W/"7084893-1630505462650"
Pragma: no-cache
Cache-Control: no-cache

HTTP/1.1 304
Server: nginx/1.21.0
Date: Wed, 01 Sep 2021 14:17:38 GMT
Connection: keep-alive
ETag: W/"7084893-1630505462650"
```

Saxon-JS 2.2

```
GET /static/com/atomgraph/linkeddatahub/xsl/client.xsl.sef.json HTTP/1.1
Host: localhost:4443
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Referer: https://localhost:4443/?uri=https%3A%2F%2Flocalhost%3A4443%2F
Cookie: _ga=GA1.1.828629977.1584086266; LinkedDataHub.first-time-message=true; _octo=GH1.1.128689066.160763774
8
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

HTTP/1.1 200 
Server: nginx/1.21.0
Date: Wed, 01 Sep 2021 14:30:07 GMT
Content-Type: application/json
Transfer-Encoding: chunked
Connection: keep-alive
Accept-Ranges: bytes
ETag: W/"7070035-1630012354000"
Last-Modified: Thu, 26 Aug 2021 21:12:34 GMT
vary: accept-encoding
Content-Encoding: gzip
```

#3 - 2021-09-01 17:15 - Norm Tovey-Walsh

Interesting. We did some work in 2.3 to improve the way Accept: headers are sent. (Bug #5017)

I suppose this could be an unintended consequence, but it's hard to imagine how we didn't stumble over this ourselves in testing.

Is there any chance you can trim your application down to a test case that demonstrates the problem as your encountering it?

#4 - 2021-09-01 17:20 - Martin Honnen

I see the parsing error with Firefox but not Chrome in https://martin-honnen.github.io/xslt/2021/saxon-js-key-test2.html which uses Saxon-JS 2.3 while neither Chrome nor Firefox show the parsing error with https://martin-honnen.github.io/xslt/2021/saxon-js-key-test1.html which uses Saxon-JS 2.2. Chrome, however, doesn't do me the favour to use the 304 header, it just uses its disk cache. But with Firefox I indeed get an error like

```
XML Parsing Error: not well-formed
Location: https://martin-honnen.github.io/xslt/2021/saxon-js-key-test1.sef.json
Line Number 1, Column 1:
```

for the test case at https://martin-honnen.github.io/xslt/2021/saxon-js-key-test2.html using Saxon-JS 2.3.

#5 - 2021-09-01 17:55 - Martynas Jusevicius

I can confirm that I also observe this behavior in Firefox.

#6 - 2021-09-16 14:20 - Martynas Jusevicius

Any updates on this? The code seems to work fine, but having errors like this in the console is not ideal...

#7 - 2021-09-16 15:15 - Norm Tovey-Walsh

No progress yet, I'm afraid. There's *a lot* going on just at the moment. I'll make sure we get this sorted out as quickly as we can.

#8 - 2021-10-11 09:44 - Martynas Jusevicius

Ping again :) Is Saxon-JS on a lower priority now?

#9 - 2021-10-11 15:18 - Norm Tovey-Walsh

No, not at all. But we're a small team and big releases, like SaxonCS, tend to consume all available cycles. Debbie and I are on the hook for a few other deliverables, short term, but then we'll look at fixing bugs and doing an other Saxon-JS release as soon as possible.

Sincere apologies for the delay. I promise it's a high priority!

#10 - 2021-10-25 14:55 - Norm Tovey-Walsh

- Status changed from New to In Progress

I've recreated the issue locally and I am seeing some really inexplicable behavior from Firefox. In the local test, I have complete control over the server. I can see that Firefox issues the request and I can see that the server responds with 200 and the data. (The test server doesn't have any support for 304 responses.)

What turns up in the Network tab in Firefox's inspector is "304 Not Modified". It's as if the browser detects that the Etags are the same (maybe?) and tells the script it got 304 even when it got 200.

I'm not quite sure how the script is expected to respond to this. It still needs the data.

#11 - 2021-10-25 15:01 - Norm Tovey-Walsh

Indeed. If I hack the server to generate random ETag headers, then the browser's Network tab shows 200. It also still appears to be trying to parse the JSON with an XML parser, though. Bad happens.

#12 - 2021-10-25 15:05 - Martynas Jusevicius

304 Not Modified is a correct response to the conditional request with If-Modified-Since/If-None-Match headers which are sent by Saxon-JS 2.3 but not 2.2 (see my comment #2). https://developer.mozilla.org/en-US/docs/Web/HTTP/Conditional_requests

#13 - 2021-10-25 17:48 - Norm Tovey-Walsh

I have reached the tentative conclusion that this is not an error message generated by SaxonJS. In support of this conclusion:

1. Even though the message is generated, processing continues normally. It doesn't seem to have any impact on the behavior of the application. If this was SaxonJS getting the parsing wrong, I'd expect that to have other impacts.
2. It only occurs on Firefox.
3. The error message begins "XML Parsing Error:" which is not a string in our code base.

Here's what I think happens:

1. SaxonJS wants to load the stylesheet, so it asks for a resourcePromise to do this. It passes the promise the URI and the encoding, but not the (expected?) type.
2. The resourcePromise determines that it's a reasonable request (not a file, etc.) and calls getPromise passing the URI, the headers, the type (undefined in this case), and the encoding.
3. In getPromise, we construct an XMLHttpRequest. Because the type wasn't passed in, it defaults to application/xml.
4. We call overrideMimeType() on the request, specifying application/xml and the encoding.

I *think* Firefox attempts to parse the result as XML because that's what the override said it was. Even though that doesn't work, it still returns the result to SaxonJS which happily carries on. I don't know why Safari and Chrome don't attempt to parse the request.

We know that only a JSON SEF file is an acceptable resource so the fix seems to be simply to specify that it's JSON.

That fixes the bug and I can't imagine it causes any other problems, but I haven't run any of the tests yet.

Next, I'm going to see if I can work out why SaxonJS 2.2 didn't have this problem...

#14 - 2021-10-25 17:50 - Norm Tovey-Walsh

I understood your comments about 304 Not Modified. I was commenting on the fact that the web server I stood up to test this locally *does not* send 304. The browser seems to "inject" 304 into the response because the ETag headers are the same. That surprises me, but doesn't actually seem relevant.

#15 - 2021-10-25 18:09 - Norm Tovey-Walsh

Because I broke it in commit 144a968.

When I added code to support specifying the acceptable encodings, I accidentally forced a default content type where one had not previously been implied. The default content type I enforced was application/xml as described above.

I'm going to fix this by making my changes regarding acceptable encodings dependent on the *caller* having specified *both* a content type and an encoding (since I can't specify the desired encoding without specifying a content type).

Standing by my earlier hypothesis regarding Firefox, I think we didn't catch this because if you aren't watching the browser console, you'd never notice.

#16 - 2021-10-25 18:20 - Norm Tovey-Walsh

- Status changed from In Progress to Resolved

I have committed a fix that I believe preserves the new functionality (provided that a content type is included with the encoding) without breaking any previous functionality (when content type and encoding were not supported). I have added the content type to the request for the stylesheet since it must be JSON.