Saxon - Bug #4433

Saxon 9.9.1-5 outputs malformed XML due to duplicate default namespace declaration

2020-01-19 22:51 - Andreas Sewe

Status:	Closed	**Start date:**	2020-01-19
Priority:	Normal	**Due date:**	
Assignee:	Michael Kay	**% Done:**	100%
Category:	XQuery Java API	**Estimated time:**	0:00 hour
Sprint/Milestone:		**Spent time:**	0:00 hour
Legacy ID:		**Fix Committed on Branch:**	9.9
Applies to branch:	9.9	**Fixed in Maintenance Release:**	9.9.1.7

Description

This bug occurs with Saxon 9.9.1-5 and -6 using the XQJ API. (I have previously also mention it on saxon-help, but unfortunately got no response there.)

I have attached a simple Maven project with which you can *reproduce*. Just run mvn clean verify exec:java -DsaxonVersion=9.9.1-6. This outputs malformed XML:

```
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <title>Title</title>
   </head>
   <body>
      <article>
         <header>
            <h1>Title</h1>
         </header>
         <section xmlns="http://www.w3.org/1999/xhtml" xmlns="http://www.w3.org/2005/Atom">
            <p>Some content</p>
         </section>
      </article>
   </body>
</html>
```

Note the duplicate default namespace declaration of the <section> element.

This bug does not occur with **older versions** of Saxon, as showcased by mvn clean verify exec:java -DsaxonVersion=9.9.1-4.

```
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <title>Title</title>
   </head>
   <body>
      <article>
         <header>
            <h1>Title</h1>
         </header>
         <section xmlns="http://www.w3.org/1999/xhtml">
            <p>Some content</p>
         </section>
      </article>
   </body>
</html>
```

It also does not occur with any version when using the **command-line interface**, as showcased by java -cp ~/.m2/repository/net/sf/saxon/Saxon-HE/9.9.1-6/Saxon-HE-9.9.1-6.jar net.sf.saxon.Query -q:src/main/resources/org/example/saxonbug/query.xq -s:src/main/resources/org/example/saxonbug/input.atom. Command-line Saxon interestingly places the namespace declarations differently again – and IMHO best (output piped through xmllint -format for readability):

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Title</title>
  </head>
  <body>
    <article>
      <header>
        <h1>Title</h1>
      </header>
      <section>
        <p>Some content</p>
      </section>
    </article>
  </body>
</html>
```

This bug may hence be specific to invoking Saxon through XQJ, but it prevents us from upgrading. Also, I am really curious why using XQJ produces different results than using the command-line interface in **both** 9.9.1-4 and 9.9.1-6.

FYI, this might be related to issue #4319, which also sprung up with 9.9.1-5 and is related to namespace declarations as well.

I hope this information is useful in locating and fixing the bug.

History

#1 - 2020-01-20 00:15 - Michael Kay

- Status changed from New to In Progress

- Assignee set to Michael Kay

- Priority changed from Low to Normal

- Applies to branch 9.9 added

- Applies to branch deleted (9.5)

#2 - 2020-01-20 01:22 - Michael Kay

In my first attempt to run this, I can't reproduce the bug. I'm getting the same output with XQJ as with the command line (that is, no redundant namespace declarations).

The fact that you're seeing incorrect output in 9.9.1.4 as well (the redundant namespace declaration should not be there) means, I think, that the problem is already present with earlier releases, but merely has a different external manifestation. The different manifestation might well be related to bug #4319, where the explanation is that Saxon has reduced the amount of checking that it applies to the input coming from the XML parser: that is, it trusts the parser more than it used to.

I suspect that the reason you're seeing different results from mine is something to do with JDK and/or parser versions.

If you add the line

```
((SaxonXQDataSource)dataSource).getConfiguration().setConfigurationProperty(Feature.TIMING, true);
```

then it will give you output (on System.err) showing which SAX parser is in use: I get

```
Using parser org.apache.xerces.jaxp.SAXParserImpl$JAXPSAXParser
```

If you're using the built-in JDK parser (rather than Apache Xerces) then it would be useful to know which JDK version you are on.

#3 - 2020-01-20 01:43 - Michael Kay

As to the question, why is XQJ different from the command line, there can be a number of possibilities. Both environments will be setting up a pipeline containing parser, query engine, and serializer, but there can be many differences in the exact detail of this pipeline. In particular the pipeline will at a number of points contain a NamespaceReducer whose task is to eliminate redundant namespace declarations and to insert namespace declarations where they are actually needed. Historically we've been inclined to add too many NamespaceReducers into the pipeline (the extra ones do no harm but serve no useful function) and we've been trying to identify cases where we can avoid adding them. This accounts for the observation in bug #4319 that we have become less resilient to bad input from the XML parser, but it might also account for differences between one execution pipeline and another.

A specific difference between the two cases is that with the command line, the processor streams the query result directly to the serializer in push mode. With XQJ, because the mechanism for delivering results is pull-based, this is not possible, so the result tree is actually materialized in memory.

#4 - 2020-01-20 09:40 - Andreas Sewe

Using Feature.TIMING, I determine com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser as parser. JRE and OS is as follows (mvn -version):

```
Apache Maven 3.6.1 (d66c9c0b3152b2e69ee9bac180bb8fcc8e6af555; 2019-04-04T21:00:29+02:00)
Maven home: /opt/apache-maven
Java version: 1.8.0_201, vendor: Oracle Corporation, runtime: /usr/lib/jvm/java-8-oracle/jre
Default locale: en_CA, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-74-generic", arch: "amd64", family: "unix"
```

Same symptoms with a Java 11 JRE (JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64 mvn -version):

```
Apache Maven 3.6.1 (d66c9c0b3152b2e69ee9bac180bb8fcc8e6af555; 2019-04-04T21:00:29+02:00)
Maven home: /opt/apache-maven
Java version: 11.0.5, vendor: Private Build, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: en_CA, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-74-generic", arch: "amd64", family: "unix"
```

JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64 mvn clean verify exec:java -DsaxonVersion=9.9.1-6 still produces the same malformed XML.

Is there any way to isolate Saxon (using XQJ) from the used JRE some more? I am currently developing an xquery-maven-plugin, which supports (not just) Saxon via XQJ and I would really like to avoid this kind of instability.

Finally, thanks for explaining the difference between CLI and XQJ “modes.” I was under the misconception that XQJ was just a thin wrapper around net.sf.saxon.Query (just without the String argument processing), but I wasn’t aware of the push vs. pull issue at all.

#5 - 2020-01-20 09:48 - Michael Kay

OK, I'll try running with the built-in JDK parser and see if that's the significant difference.

In the past I always recommended running with the Apache parser because the JDK parser had some serious bugs and Oracle seemed to be doing nothing to fix them. Since JDK 8, however, the JDK parser has been a lot more stable. Mostly from habit, however, the majority of our testing is still done using the Apache parser.

#6 - 2020-01-20 11:06 - Michael Kay

I have run using the JDK parser and I'm still not seeing the problem. It's a slightly different version of the JDK - jdk1.8.0_121.jdk on Mac OS, but that doesn't seem likely to be a signficant difference.

I'm now studying exactly how this query works under XQJ (in my environment), in case that yields any insights as to what factors might prevent it working.

#7 - 2020-01-20 12:12 - Michael Kay

These notes capture my investigation and are largely for my own benefit.

Under XQJ we start evaluation in pull mode using XQueryExpression.iterator() which calls expression.iterate(). The function call to local:atom-entry-content() has been inlined. When we get to the Choose that's the body of this function, we're executing in push mode and select the first branch; there's no explicit copy-of operation in the expression tree, so we evaluate the path expression $entry/atom:content/div/* in push mode. This leads to a call on TreeReceiver.append() with the argument being the section element in the source tree.

Here's a diagnostic dump of the input TinyTree at this point (the section element is node 9):

```
Tree size: 17 nodes, 17 characters, 1 attributes
    node    kind   depth    next   alpha    beta    name    type
       0       9       0      -1       0       0      -1     630
       1       1       1       0      -1       1    1024     630 Q{http://www.w3.org/2005/Atom}entry
       2       4       2       303233024       0      -1     630
       3      17       2       4       0       5    1025     630 Q{http://www.w3.org/2005/Atom}title
       4       4       2       503233024       0      -1     630
       5       1       2      15       0      -1    1026     630 Q{http://www.w3.org/2005/Atom}content
       6       4       3       703364096       0      -1     630
       7       1       3      14      -1       2    1028     630 Q{http://www.w3.org/1999/xhtml}div
       8       4       4       903495168       0      -1     630
       9       1       4      13      -1      -1    1036     630 Q{http://www.w3.org/1999/xhtml}section
      10       4       5      1103626240       0      -1     630
      11      17       5      12       5      12    1037     630 Q{http://www.w3.org/1999/xhtml}p
      12       4       5       903495168       0      -1     630
      13       4       4       703364096       0      -1     630
      14       4       3       503233024       0      -1     630
      15       4       2       190519040       0      -1     630
      16      11       0      -1       0       0      -1     630
    attr  parent    name    value
       0       5    1027    xhtml
      ns  parent  prefix     uri
       0       0xml=http://www.w3.org/XML/1998/namespace
       1       1=http://www.w3.org/2005/Atom
       2       7=http://www.w3.org/1999/xhtml
```

This all looks perfectly correct. TreeReceiver.append(item) goes on to call item.copy(...) with copyOptions set to "all namespaces". TinyElementImpl.copy() tests to see if it can do a "bulk copy" and decides not - this is a potential danger point, bulk copy is a "fast path" that's relatively new in the product and therefore worth treating with caution. The reason it doesn't do a bulk copy is because the current output destination is a TreeReceiver, not a SequenceWriter or ComplexContentOutputter.

It then sends a startElement() event for the section element down the pipeline, followed by sending all the in-scope namespaces. The CCO adds the binding (""=>XHTML) to the pendingNSList. CCO.startContent() is called, and this sends a startElement() event to the next thing in the pipeline, which is a NamespaceReducer; this sends it on to a TinyBuilder. The CCO then send the namespace binding event to the NamespaceReducer, which decides that the binding is already in scope, and so ignores it.

On thing that surprises me in this is that the TinyElementImpl.copy() operation, when sending the in-scope namespaces of the section element, didn't send the binding (""=>Atom). I would have expected this to be sent down the pipeline, and then eliminated as a duplicate. However, if I add a non-duplicate namespace xmlns:x="x.com" to the root element, it does get copied through to the output, so perhaps my worries are misplaced.

#8 - 2020-01-20 12:31 - Michael Kay

I think it would be useful to establish whether the source tree is the same in the failing environment from the non-failing environment. It's not particularly easy to do this with XQJ, but it can be done like this. I'd be grateful if you can run this and let me know the result.

```
            XQDataSource dataSource = new SaxonXQDataSource();
            ((SaxonXQDataSource) dataSource).getConfiguration().setConfigurationProperty(Feature.TIMING, true)
;
            XQConnection connection = dataSource.getConnection();
            try  InputStream input = new FileInputStream("...../input.atom")) {
                XQPreparedExpression expression = connection.prepareExpression(".");
                expression.bindDocument(XQConstants.CONTEXT_ITEM, input, null, null);
                XQResultSequence results = expression.executeQuery();
                results.next();
                XQItem result = results.getItem();
                NodeInfo it = (NodeInfo)((SaxonXQItem)result).getSaxonItem();
                ((TinyDocumentImpl)it).getTree().diagnosticDump();
                results.close();
            } catch (Exception e) {
                fail(e.getMessage());
            }
```

If the tree is different, that tells us we need to focus on how the input is processed (including details of the XML parser configuration). If it's the same, then we need to look at the query processing and serialization. So this will help to inform the next step in the investigation.

#9 - 2020-01-20 14:58 - Andreas Sewe

Here you go:

```
Using parser com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser
Building tree for null using class net.sf.saxon.tree.tiny.TinyBuilder
Tree built in 9.365797ms
Tree size: 17 nodes, 17 characters, 1 attributes
    node    kind   depth    next   alpha    beta    name    type
       0       9       0      -1       0       0      -1     630
       1       1       1       0      -1       1    1024     630 Q{http://www.w3.org/2005/Atom}entry
       2       4       2       303233024       0      -1     630
       3      17       2       4       0       5    1025     630 Q{http://www.w3.org/2005/Atom}title
       4       4       2       503233024       0      -1     630
       5       1       2      15       0      -1    1026     630 Q{http://www.w3.org/2005/Atom}content
       6       4       3       703364096       0      -1     630
```

```
      7       1       3      14      -1       2    1028     630 Q{http://www.w3.org/1999/xhtml}div
      8       4       4       903495168       0      -1     630
      9       1       4      13      -1      -1    1029     630 Q{http://www.w3.org/1999/xhtml}section
     10       4       5      1103626240      0      -1     630
     11      17       5      12       5      12    1030     630 Q{http://www.w3.org/1999/xhtml}p
     12       4       5       903495168       0      -1     630
     13       4       4       703364096       0      -1     630
     14       4       3       503233024       0      -1     630
     15       4       2       190519040       0      -1     630
     16      11       0      -1       0       0      -1     630
   attr  parent    name    value
      0       5    1027    xhtml
     ns  parent  prefix     uri
      0       0xml=http://www.w3.org/XML/1998/namespace
      1       1=http://www.w3.org/2005/Atom
      2       7=http://www.w3.org/1999/xhtml
```

#10 - 2020-01-20 15:38 - Michael Kay

OK thanks. That's the same as mine, so we now know it's related to the way the tree is processed, not the way it is built. I'll try to devise some more diagnostics.

#11 - 2020-01-20 15:54 - Michael Kay

I would like to eliminate the possibility that in your environment, the new "bulk copy" feature is being invoked. This is controlled by a static variable: could you please try setting

```
net.sf.saxon.tree.tiny.TinyTree.useBulkCopy = false;
```

just to see if it makes any difference to the output.

The other thing that might be useful is a diagnostic dump of the result tree. You can get that in the same way as the previous diagnostic dump, but this time running the real query rather than the dummy query ".". The purpose of this is to establish whether the problem occurs while constructing the result tree, or while serializing it. It's almost certainly the former, but we need to be systematic.

I have absolutely no idea now why you should be getting different results from mine. I'll have access to a Linux environment tomorrow and it may be worth us trying it on that, but it's a very long shot. The other thing I will try is running the product as issued, outside the development and debugging environment where we normally conduct all our tests.

#12 - 2020-01-20 16:57 - Andreas Sewe

I am afraid the static variable is final, so I can't easily set it to false:

```
    /**
     * Temporary flag introduced in Saxon 9.8.0.5 to enable or disable fast-path code for copying
     * element nodes from one TinyTree to another. This code is currently disabled by default until
     * it has been more thoroughly tested.
     */
    public final static boolean useBulkCopy = true;  // Until bug 4089 is resolved
```

#13 - 2020-01-20 17:00 - Andreas Sewe

As for your other request: Using the real query rather than "." resulted in a ClassCastException: net.sf.saxon.tree.tiny.TinyElementImpl cannot be cast to net.sf.saxon.tree.tiny.TinyDocumentImpl, so I changed the diagnosticDump line accordingly:

```
results.next();
XQItem result = results.getItem();
NodeInfo it = (NodeInfo)((SaxonXQItem)result).getSaxonItem();
((net.sf.saxon.tree.tiny.TinyElementImpl)it).getTree().diagnosticDump();
```

This gives me

```
Using parser com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXParser
Building tree for null using class net.sf.saxon.tree.tiny.TinyBuilder
Tree built in 6.523431ms
Tree size: 17 nodes, 17 characters, 1 attributes
    node    kind   depth    next   alpha    beta    name    type
       0       1       0      -1      -1       1    1029     630 Q{http://www.w3.org/1999/xhtml}html
       1       1       1       3      -1      -1    1030     630 Q{http://www.w3.org/1999/xhtml}head
       2      17       2       1       0       5    1031     630 Q{http://www.w3.org/1999/xhtml}title
       3       1       1       0      -1      -1    1032     630 Q{http://www.w3.org/1999/xhtml}body
       4       1       2       3      -1      -1    1033     630 Q{http://www.w3.org/1999/xhtml}article
       5       1       3       7      -1      -1    1034     630 Q{http://www.w3.org/1999/xhtml}header
       6      17       4       5       5       5    1035     630 Q{http://www.w3.org/1999/xhtml}h1
       7       1       3       4      -1       2    1036     630 Q{http://www.w3.org/1999/xhtml}section
       8       4       4       903626240       0      -1     630
       9      17       4      10      10      12    1037     630 Q{http://www.w3.org/1999/xhtml}p
      10       4       4       703495168       0      -1     630
      11      11       0      -1       0       0      -1     630
    attr  parent    name    value
      ns  parent  prefix     uri
       0       0xml=http://www.w3.org/XML/1998/namespace
       1       0=http://www.w3.org/1999/xhtml
       2       7=http://www.w3.org/1999/xhtml
       3       7=http://www.w3.org/2005/Atom
```

#14 - 2020-01-20 17:20 - Michael Kay

Thanks. This establishes that the problem is in the result tree construction rather than the serialization.

#15 - 2020-01-20 18:31 - Michael Kay

I've now reproduced the problem, by running against the Saxon-9.9.1.5 PE jar file, rather than with the EE code as previously. That should make it fairly easy to pin down.

#16 - 2020-01-20 18:44 - Michael Kay

In PE, the function is no longer inlined, but it is still executed in push mode. The pipeline in this case sends the result directly to a SequenceOutputter, and this triggers use of the bulkCopy mechanism; although I haven't probed any further, I can be 90% confident that bulk copy is causing the problem, by failing to handle namespaces correctly. In fact, although I wasn't aware of specific problems, I was sufficiently uncomfortable with the handling of namespaces in bulk copy that I rewrote it completely for 10.0.

Disabling bulk copy solves the problem, and this is probably what I shall do as a patch.

With that, I will turn my attention to the 10.0 code, to establish (a) that the rewritten bulk copy handles this use case correctly, and (b) that bulk copy is actually invoked in both the PE and EE paths. The reason it's not happening on the EE path is because the pipeline is a bit more complicated because of the need to handle schema-validated data, but that complexity ought not to appear unless the query is actually schema-aware.

#17 - 2020-01-20 18:56 - Michael Kay

- Fix Committed on Branch 9.9 added

I have suppressed bulk copy completely on the 9.9 branch, which resolves the bug. But I will leave it open pending further investigations on 10.0. Currently the test is working but bulk copy is not being triggered, and I need to explore whether this decision is correct.

#18 - 2020-01-20 19:02 - Michael Kay

See also bug #4273 for some of the history of bulk copy and namespace handling.

#19 - 2020-01-21 01:35 - Michael Kay

- Status changed from In Progress to Resolved

I have decided to scrap the "bulk copy" and "tree grafting" optimizations from Saxon 10.0. Both aim to reduce the cost of copying subtrees, in one case by doing the copy at a lower level by manipulating the physical data structure, and in the other case by enhancing the data structure to allow nodes to be shared between trees. Both optimizations give excellent performance results, but only affect a small minority of applications; and both have proved more trouble than they are worth in terms of bugs introduced by the extra complexity.

With that I'm marking this bug resolved.

#20 - 2020-01-23 21:35 - Andreas Sewe

Thanks for addressing this issue so promptly.

#21 - 2020-03-05 11:09 - O'Neil Delpratt

- Status changed from Resolved to Closed

- % Done changed from 0 to 100

- Fixed in Maintenance Release 9.9.1.7 added

Patch applied in the 9.9.1.7 maintenance release.

Files

saxonbug.zip	2.12 KB	2020-01-19	Andreas Sewe