# Saxon - Bug #5102

## System.ArgumentNullException from message listener (unknown location URI?)

2021-09-22 13:49 - Emanuel Wlaschitz

| | | | |
|---|---|---|---|
| **Status:** | Resolved | **Start date:** | 2021-09-22 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | Michael Kay | **% Done:** | 0% |
| **Category:** | Internals | **Estimated time:** | 0:00 hour |
| **Sprint/Milestone:** | | **Spent time:** | 0:00 hour |
| **Legacy ID:** | | **Fixed in Maintenance Release:** | |
| **Applies to branch:** | 10 | **Platforms:** | |
| **Fix Committed on Branch:** | 10 | | |

**Description**

We're currently in the process of updating to Saxon-HE 10.6N and one of our stylesheets started to blow up for no apparent reason. Deep down we see this one:

```
System.ArgumentNullException
  HResult=0x80004003
  Message=Value cannot be null.
Parameter name: uriString
  Source=System
  StackTrace:
   at System.Uri..ctor(String uriString)

  This exception was originally thrown at this call stack:
    System.Uri.Uri(string)
```

...which is at:

```
  System.dll!System.Uri.Uri(string uriString) Unknown
  saxon-he-api-10.6.dll!Saxon.Api.MessageListenerProxy2.message(net.sf.saxon.s9api.XdmNode xn, net.sf.saxon.s9api.QName qn, bool b, javax.xml.transform.SourceLocator sl) Unknown
  saxon-he-10.6.dll!net.sf.saxon.s9api.MessageListener2Proxy.write(net.sf.saxon.om.Item value) Unknown
  saxon-he-10.6.dll!net.sf.saxon.event.SequenceWriter.append(net.sf.saxon.om.Item item, net.sf.saxon.s9api.Location locationId, int copyNamespaces) Unknown
  saxon-he-10.6.dll!net.sf.saxon.s9api.MessageListener2Proxy.append(net.sf.saxon.om.Item value, net.sf.saxon.s9api.Location value, int value) Unknown
  saxon-he-10.6.dll!net.sf.saxon.event.SequenceWriter.endDocument() Unknown
  saxon-he-10.6.dll!net.sf.saxon.event.TreeReceiver.endDocument() Unknown
  saxon-he-10.6.dll!net.sf.saxon.event.ProxyReceiver.endDocument() Unknown
  saxon-he-10.6.dll!net.sf.saxon.event.ComplexContentOutputter.endDocument() Unknown
  saxon-he-10.6.dll!net.sf.saxon.expr.instruct.Message.processLeavingTail(net.sf.saxon.event.Outputter output, net.sf.saxon.expr.XPathContext context) Unknown
  saxon-he-10.6.dll!net.sf.saxon.expr.instruct.Block.processLeavingTail(net.sf.saxon.event.Outputter output, net.sf.saxon.expr.XPathContext context) Unknown
  saxon-he-10.6.dll!net.sf.saxon.expr.instruct.TemplateRule.applyLeavingTail(net.sf.saxon.event.Outputter output, net.sf.saxon.expr.XPathContext context) Unknown
  saxon-he-10.6.dll!net.sf.saxon.trans.Mode.applyTemplates(net.sf.saxon.expr.instruct.ParameterSet parameters, net.sf.saxon.expr.instruct.ParameterSet tunnelParameters, net.sf.saxon.om.NodeInfo separator, net.sf.saxon.event.Outputter output, net.sf.saxon.expr.XPathContextMajor context, net.sf.saxon.s9api.Location locationId) Unknown
  saxon-he-10.6.dll!net.sf.saxon.trans.XsltController.applyTemplates(net.sf.saxon.om.Sequence source, net.sf.saxon.event.Receiver out) Unknown
  saxon-he-10.6.dll!net.sf.saxon.s9api.AbstractXsltTransformer.applyTemplatesToSource(javax.xml.transform.Source value, net.sf.saxon.event.Receiver value) Unknown
  saxon-he-10.6.dll!net.sf.saxon.s9api.XsltTransformer.transform() Unknown
  saxon-he-api-10.6.dll!Saxon.Api.XsltTransformer.Run(Saxon.Api.XmlDestination destination) Unknow
```

```
n
```

...and is followed up by this when ignoring exceptions that are caught internally:

```
java.lang.RuntimeException
  HResult=0x80131500
  Message=Internal error evaluating template rule  in module file:///D:/_dev/SaxonMLP2NullPointer/bin/Debug/SaxonMLP2NullPointer.exe
  Source=saxon-he-10.6
  StackTrace:
   at net.sf.saxon.expr.instruct.TemplateRule.applyLeavingTail(Outputter output, XPathContext context)
   at net.sf.saxon.trans.Mode.applyTemplates(ParameterSet parameters, ParameterSet tunnelParameters, NodeInfo separator, Outputter output, XPathContextMajor context, Location locationId)
   at net.sf.saxon.trans.XsltController.applyTemplates(Sequence source, Receiver out)
   at net.sf.saxon.s9api.AbstractXsltTransformer.applyTemplatesToSource(Source , Receiver )
   at net.sf.saxon.s9api.XsltTransformer.transform()
   at Saxon.Api.XsltTransformer.Run(XmlDestination destination)
   at SaxonMLP2NullPointer.Program.Main(String[] args) in D:\_dev\SaxonMLP2NullPointer\Program.cs:line 39

Inner Exception 1:
NullPointerException: 
```

It appears that in those cases, SourceLocator sl inside MessagListenerProxy2.message is net.sf.saxon.expr.parser.Loc@1029822 { columnNumber: -1, lineNumber: -1, systemId: null } and fails when attempting to set location.BaseUri = new Uri(sl.getSystemId()); due to sl.getSystemId() returning null.

I managed to create a stripped down example that still causes the issue, and it appears to be related to having both a custom MessageListener2 on the XsltTransformer, as well as writing a <xsl:message> that contains either <xsl:value-of/> or <xsl:sequence/> to produce the message itself from a non-xs:string source (that is: no select attribute, any expression that is not of type xs:string or uses any function to produce a xs:string; including a call to concat()).

Here's the (minimalistic) source:

```
internal class Program
{
    private static void Main(string[] args)
    {
        var xslt = XDocument.Parse(@"
        <stylesheet xmlns='http://www.w3.org/1999/XSL/Transform' version='3.0'>
            <template match='/'>
                <!-- this message causes an error when MessageListener2 is used -->
                <message><value-of select='local-name()'/></message>
                <result xmlns=''/>
            </template>
        </stylesheet>");
        var input = xslt; // does not matter for this test.
        var output = new XDocument();

        var processor = new Processor();
        using (var xsltReader = xslt.CreateReader())
        using (var inputReader = input.CreateReader())
        using (var outputWriter = output.CreateWriter())
        {
            var compiler = processor.NewXsltCompiler();
            compiler.BaseUri = new Uri(typeof(Program).Assembly.Location);
            var xsltExecutable = compiler.Compile(xsltReader);

            var transformer = xsltExecutable.Load();

            var builder = processor.NewDocumentBuilder();
            transformer.InitialContextNode = builder.Build(inputReader);
            // comment this out to prevent the issue
            transformer.MessageListener2 = new TestMessageListener();

            var destination = new TextWriterDestination(outputWriter) { CloseAfterUse = true };
            transformer.Run(destination);
        }
    }
    private sealed class TestMessageListener : IMessageListener2
    {
        public void Message(XdmNode content, QName
errorCode, bool terminate, IXmlLocation location) { }
    }
}
```

The code will work when:

1. transformer.MessageListener2 is not set by the code (or transformer.MessageListener if the errorCode parameter is not neede by the listener)
2. The contents of <message> is plain text (such as <message>test</message>)
3. The contents of <message> is a string variable (such as <variable name="message" select="'test'"/><message><value-of select="$message"/></message>)

The code will **not** work when:

1. The contents of <message> is produced by a function call (such as <message><value-of select="concat('Message here: ', $message)"/></message>)
2. The contents of <message> is not of type xs:string (such as <variable name="message" select="42"/><message><value-of select="$message"/></message>)
3. The contents of <message> use the concatenation operator (such as <message><value-of select="Message here: " || $message"/></message> - which *might* be the same as the first case though)

For obvious reasons, we can't simply *not* add the MessageListener2 because we want to capture <xsl:message> including the errorCode. And the same thing happens if we revert back to MessageListener,  except that it happens in MessageListenerProxy instead of MessageListenerProxy2 in that case.

This is using the currently most recent NuGet package with version 10.6.0 on .NET Framework 4.8

## History

**#1 - 2021-09-22 13:55 - Emanuel Wlaschitz**

Oops, my bad, I started typing the title, then got side-tracked and forgot about it. Should probably something like "NullPointerException in MessageListenerProxy2 for non-string xsl:message content" or so.

**#2 - 2021-09-22 13:56 - Michael Kay**

*- Subject changed from Null to System.ArgumentNullException from message listener (unknown location URI?)*

Thanks for reporting it, we'll take a look.

Missing location URIs are a constant source of trouble.

**#3 - 2021-11-02 23:59 - Michael Kay**

Sorry for the delay in responding.

I think there are two separate issues here.

The first is that the C# proxy code crashes if the location has a null system ID - it should be prepared to accept that, as there are always some cases where the system ID is unknown (e.g. when the stylesheet is loaded from an anonymous Stream).

The second is that the Java code is sometimes supplying a location with a null system ID when it should be able to supply an actual system ID.

**#4 - 2021-11-03 00:59 - Michael Kay**

Looking first at cases where the system ID is needlessly absent from the location passed to the message listener:

I have reproduced this for the first case of <xsl:message><xsl:value-of select="local-name()"/></xsl:message> where the context item is a document node (which means that local-name() returns a zero-length string, which means that <xsl:value-of> constructs a zero-length text node, which means that the content of the message is a document node with no children). The issue here is simply that the message is fed through to the message listener as a sequence of Receiver events, and this sequence consists solely of a startDocument followed by endDocument, and neither of these events carries location information. This case (of an empty message) is presumably untypical, so it's probably not worth focusing on it.

If we take the case where the message content is produced using a concat() call, the issue is a little different. The concat() function is evaluated in push mode (so the string result does not need to be built in contiguous memory) and this loses the location information. The sequence of calls for generating the message is fairly convoluted but I think I can probably fix this for most paths.

**#5 - 2021-11-03 07:55 - Emanuel Wlaschitz**

Huh, wait, does that mean the event is supposed to have the location inside the *produced result document* rather than the *XSLT source*?

Previously, we were logging this information alongside the other parameters so we can figure out where in the XSLT things happened.

Also, now that I write this, I think i accidentally omitted LoadOptions.SetLineInfo from the XDocument.Parse call in the minimalistic sample. Without that, XDocument would not keep track of (and populate) the IXmlLineInfo. On 9.9, the above code passes IXmlLocation.LineNumber = -1 without the LoadOptions.SetLineInfo and IXmlLocation.LineNumber = 4 (which includes the line break immediately following XDocument.Parse, when removed it returns 3) with LoadOptions.SetLineInfo - so it clearly refers to the XSLT source (and not the result).

A bit of background on the concat() thing: We often create messages that interweave a message level (such as "Warning" or "Error", which is later handled by the host application), static text (such as "Hey, we found " and " which seems unexpected here."), variables (such as $elementName) and QNames to be used as error-code (such as "ELEM-UNEXPECTED"). This is dumped into an <xsl:message> to produce the end result "[WARN] Hey, we found &lt;foo&gt; which seems unexpected here.", which is later logged as "Warning: [ELEM-UNEXPECTED] Hey, we found &lt;foo&gt; which seems unexpected here. (at bar.xsl:42)" after going thru some parsing inside our custom IMessageListener2. And, as already mentioned, the same code works just fine on 9.9 without any changes.

**#6 - 2021-11-03 08:39 - Emanuel Wlaschitz**

Scratch that remark, I somewhat failed to parse the statement about "where the context item is a document node" correctly.

That one's on me, though; just like the missing LoadOptions.SetLineInfo I adjusted the sample before submitting it here, and apparently missed the fact that it can never return anything useful there.

Just to make sure, the statement I have in the example is

```
<message><variable name='message' select=""42""/><value-of select='$message || $message'/></message>
```

Conversely, since I just downgraded the sample to 9.9; the code with <value-of select='local-name()'/> *also* causes the NullPointerException there. But as you mentioned, this is a flawed sample.

For completeness sake, this is the source again:

```
internal class Program
{
    private static void Main(string[] args)
    {
        var xslt = XDocument.Parse(@"
    <stylesheet xmlns='http://www.w3.org/1999/XSL/Transform' version='3.0'>
        <template match='/'>
            <message><variable name='message' select=""42""
/><value-of select='$message || $message'/></message>
            <result xmlns=''/>
        </template>
    </stylesheet>", LoadOptions.SetLineInfo);
        var input = xslt; // does not matter for this test.
        var output = new XDocument();

        var processor = new Processor();
        using (var xsltReader = xslt.CreateReader())
        using (var inputReader = input.CreateReader())
        using (var outputWriter = output.CreateWriter())
        {
            var compiler = processor.NewXsltCompiler();
            compiler.BaseUri = new Uri(typeof(Program).Assembly.Location);
            var xsltExecutable = compiler.Compile(xsltReader);

            var transformer = xsltExecutable.Load();

            var builder = processor.NewDocumentBuilder();
            transformer.InitialContextNode = builder.Build(inputReader);
            // comment this out to prevent the issue
            transformer.MessageListener2 = new TestMessageListener();

            var destination = new TextWriterDestination(outputWriter) { CloseAfterUse = true };
            transformer.Run(destination);
        }
    }
    private sealed class TestMessageListener : IMessageListener2
    {
        public void Message(XdmNode content, QName errorCode, bool terminate, IXmlLocation location) { }
    }
}
```

**#7 - 2021-11-03 09:32 - Michael Kay**

The location is supposed to be the location in the stylesheet where the message is constructed. Unfortunately the code to achieve that is very convoluted (in 10.x) because the location parameter was grafted onto an existing interface and we were trying to retain compatibility. (It's all been redesigned in 11.x).

There are a few functions like concat(), string-join(), unparsed-text(), and things that compile to calls on these functions, where the function is evaluated in "push mode", appending output piecemeal to the result document rather than constructing the function result as an in-memory string (rather like a C# method that uses "yield"). Some of your examples where location information isn't retained turn out to be calls on these push-mode functions.

**#8 - 2021-11-03 10:21 - Michael Kay**

*- Category set to Internals*

*- Status changed from New to Resolved*

*- Assignee set to Michael Kay*

*- Fix Committed on Branch 10 added*

No changes needed on the 11 branch, other than adding a couple of unit tests.

On the 10 branch:

(a) in the C# code of MessageListenerProxy2, check that a system ID is available before converting it to a System.Uri

(b) in the Java code of MessageListener2Proxy (a completely unrelated class, despite the name), the location is now picked up from the processingInstruction() call that also supplies the error code. This call happens even if there is no message content, so it's the most reliable way of ensuring that the location finds its way down the pipeline.