

XSpec for XProc, and XProc for XSLT

Amanda Galtman

August 2026

Key Points

If you write XProc 3 steps (pipelines), you can:

1. Test steps using XSpec
2. Call steps as functions, in XPath

Why?

1. Test steps using XSpec
 - Big/Important/Enduring code? Test it!
2. Call steps as functions, in XPath
 - XPath lives in XSLT, XQuery, etc.

The Connection

1. Call steps as functions, in XPath^{*}



2. Test steps using XSpec

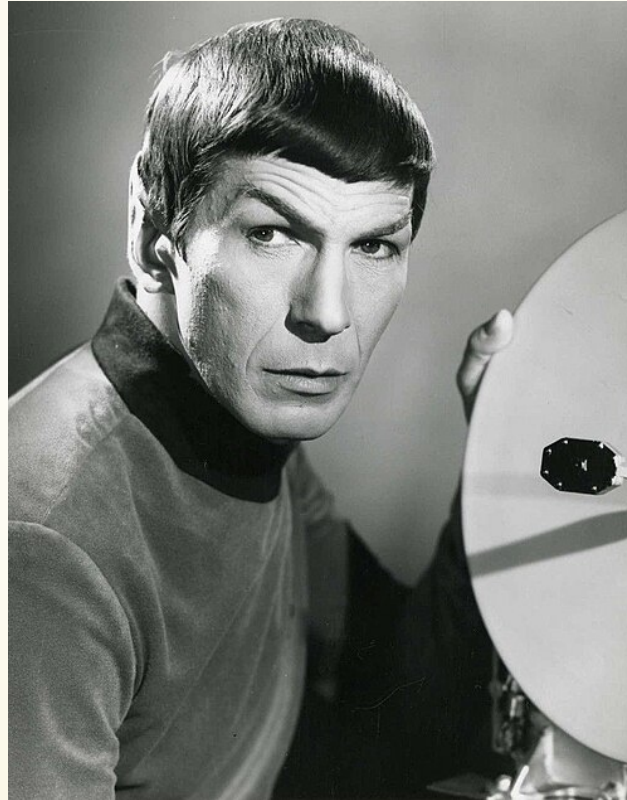
** Implemented by Norm Tovey-Walsh and Achim Berndzen*

Agenda

1. Test steps using XSpec
 - Vocabulary, semantics of a test
2. Call steps as functions, in XPath
 - Example in XSpec implementation

Non-Agenda

Apologizing in advance if I say **XSpoc** by mistake...



Public domain image, https://commons.wikimedia.org/wiki/File:Leonard_Nimoy_as_Spock_1967.jpg

TEST STEPS USING XSPEC

What Is XSpec?

- Open-source software for testing
 - XSLT
 - XQuery
 - Schematron
 - XProc ([XSpec v4.0, 2026](#))
- Created by Jeni Tennison, 2009

Parts of XSpec

- XML vocabulary to express
 - How to call the code you're testing, including inputs
 - What you expect to happen
- Running a test suite automates
 - Calling
 - Verifying: Do results meet expectations?
 - Reporting pass/fail outcomes

Sample Report

TEST REPORT

XProc: C:/balisage2026/count-words.xpl

Generated by XSpec v4.0.3

XSpec: C:/balisage2026/verification-example.xspec

Tested: 30 June 2026 at 12:10

Contents

	passed: 4	pending: 0	failed: 0	total: 4
✓ With one document having 3 words	2	0	0	2
✓ With two documents, requesting total count	2	0	0	2

With one document having 3 words

passed: 2 / pending: 0 / failed: 0 / total: 2

With one document having 3 words	passed: 2 / pending: 0 / failed: 0 / total: 2
✓ there is exactly one output document	Success
✓ an XML c:result document with content 3	Success

With two documents, requesting total count

passed: 2 / pending: 0 / failed: 0 / total: 2

With two documents, requesting total count	passed: 2 / pending: 0 / failed: 0 / total: 2
✓ there is exactly one output document	Success
✓ for the number of words in both documents	Success

XSpec XML: Structure

```
1 <x:description xproc="count-words.xpl"
2   xmlns:x="http://www.jenitennison.com/xslt/xspec"
3   xmlns:c="http://www.w3.org/ns/xproc-step"
4   xmlns:my="http://example.com/syntax"
5   xmlns:p="http://www.w3.org/ns/xproc">
6
7   <x:scenario label="With one document having 3 words">
8     <x:call/>
9     <x:expect label="there is exactly one output document"/>
10    <x:expect label="an XML c:result document with content 3"/>
11  </x:scenario>
12
13  <x:scenario label="With two documents, requesting total count">
14    <x:call/>
15    <x:expect label="there is exactly one output document"/>
16    <x:expect label="for the number of words in both documents"/>
17  </x:scenario>
18
19 </x:description>
```

XSpec XML: Call a Step

```
1 <x:scenario label="With one document having 3 words">
2   <x:call step="my:count-words">
3     <x:input port="source" as="document-node(element(doc))">
4       <doc>One two three</doc>
5     </x:input>
6   </x:call>
7   <x:expect label="there is exactly one output document"/>
8   <x:expect label="an XML c:result document with content 3"/>
9 </x:scenario>
```

XSpec XML: Call a Step (2)

```
1 <x:scenario label="With two documents, requesting total count">
2   <x:call step="my:count-words">
3     <x:input port="source">
4       <p:document href="three-words.xml"/>
5       <p:document href="two-words.xml"/>
6     </x:input>
7     <x:option name="total" select="true()"/>
8   </x:call>
9   <x:expect label="there is exactly one output document"/>
10  <x:expect label="for the number of words in both documents"/>
11 </x:scenario>
```

x:expect as You Expect

- Boolean condition
- Actual result equals expected result
- Verify portion of the result
- Custom filtering/normalization
- `$x:result`*

* *Fixed name, unrelated to output port name*

x:expect XProc-only Features

- `port="result"`
- `$x:document-properties`
- Wrapping in document nodes

XSpec XML: Verify Things

```
1 <x:scenario label="With one document having 3 words">
2   <x:call step="my:count-words">
3     <x:input port="source" as="document-node(element(doc))">
4       <doc>One two three</doc>
5     </x:input>
6   </x:call>
7   <x:expect label="there is exactly one output document"
8     port="result" test="count($x:result)" select="1"/>
9   <x:expect label="an XML c:result document with content 3"
10    port="result">
11     <c:result>3</c:result>
12   </x:expect>
13 </x:scenario>
```

XSpec XML: Verify Things (2)

```
1 <x:scenario label="With two documents, requesting total count">
2   <x:call step="my:count-words">
3     <x:input port="source">
4       <p:document href="three-words.xml"/>
5       <p:document href="two-words.xml"/>
6     </x:input>
7     <x:option name="total" select="true()"/>
8   </x:call>
9   <x:expect label="there is exactly one output document"
10     port="result" test="count($x:result)" select="1"/>
11   <x:expect label="for the number of words in both documents"
12     port="result" select="5"/>
13 </x:scenario>
```

Blending XSpec and XProc

- XSpec for XProc mostly feels like XSpec
 - Main structural elements
 - Ways to provide content
 - HTML report
 - Pending
 - Inheritance, code reuse
- But is also tailored for XProc
 - `x:input`, `x:option` elements
 - `p:document` within XSpec
 - Access to document properties
 - Port-specific verifications

Limitations

- With XSpec v4.0, requires [XML Calabash 3](#)
 - [MorganaXProc-III_{ee}](#) later in 2026
 - (Why not MorganaXProc-III_{se}? `p:run`)
- Only custom, public steps with `@type`
- A few other limitations; see the paper

CALL STEPS AS FUNCTIONS, IN XPATH

Recall: The Connection

1. Call steps as functions, in XPath



2. Test steps using XSpec

Approach

- Use XProc processor's "step function" feature
- Leverage maturity of "XSpec for XSLT"

**Goal: Adapt "XSpec for XSLT" to
create "XSpec for XProc"**

Step Functions

- *Nonstandard* feature
 - XML Calabash 3
 - MorganaXProc-III (June 2026)
- XProc processor creates a function that
 - Behaves like your custom step
 - Is callable from XPath expressions

XProc processor is doing the work.

(Configuration Details)

What makes your step eligible to be called as a function? It depends...

- Calling steps as functions (XML Calabash)
- XProc steps as XPath functions (in XSLT or XQuery) (MorganaXProc-III)

Step Function Syntax

```
1 <xsl:variable name="my-port1-docs".../>
2 <xsl:variable name="my-port2-docs".../>
3 <xsl:variable name="my-options" as="map(xs:QName,item()*)".../>
4 <xsl:sequence select="
5     my:custom-xproc-step(
6         $my-port1-docs,
7         $my-port2-docs,
8         $my-options (: last arg :)
9     )
10     ?result
11 "/>
```

Hold that thought...

**Goal: Adapt "XSpec for XSLT"
to create "XSpec for XProc"**

Inside "XSpec for XSLT"

- XSpec Compiler: XSpec file → Test Runner
- Test Runner is XSLT
 - Sets up arguments
 - Calls your XSLT function
 - Gets result
 - Decides pass/fail for each `x:expect`

Inside "XSpec for XProc"

- XSpec Compiler: XSpec file → Test Runner
- Test Runner is *still* XSLT!
 - Handles inputs/options
 - Calls your XProc step
 - Gets result
 - Decides pass/fail for each x:expect

Calls Your Step — How?

- As a step function? ~~my:step-to-test(...)?~~
- Some test cases need more XProc behaviors than just calling your step.
 - Set/get XProc document properties
 - `p:catch`, not `xs1:catch`
 - Default documents for input port

So, where *is* the step function?

Part 1: Test-Case Step

- Test runner includes "test-case step" declaration
- p:declare-step within xsl:variable

```
1 <xsl:variable name="impl:test-case-step-based-on-x-call" as="element()">
2   <p:declare-step version="3.1" xmlns:p="http://www.w3.org/ns/xproc">
3     <p:import href="file:/C:/balisage2026/count-words.xpl"/>
4     <p:output port="map-of-outputs"/>
5     <p:option name="map-of-inputs"/>
6     <p:option name="map-of-options"/>
7     <my:count-words xmlns:my="..." name="test-target">
8       <p:with-input port="source" select="$map-of-inputs?source">
9         <p:inline/>
10       </p:with-input>
11     </my:count-words>
12     [data processing...]
13   </p:declare-step>
14 </xsl:variable>
```

Part 2: From XSLT to XProc

- Test runner calls "step runner" step
- Passes test-case step as input document
- Here's the step function in action!

```
1 <xsl:sequence select="impl:step-runner(  
2     $impl:test-case-step-based-on-x-call,  
3     $impl:options-for-step-function  
4     )('map-of-outputs')"/>
```

Part 3: Generic to Specific

Step runner runs test-case step (`p:run`), which

- Invokes your XProc step (as a step)
- Packages docs/properties to return to XSLT

**Key: Test-case step has space to
*work within XProc***

Part 4: XProc Back to XSLT

- Defines `$x:result`
- Defines `$x:document-properties`
- Performs verifications

Step Function Lesson 1

Document properties can differ in XDM vs. XProc

- Property info can be lost at XProc/XPath border (e.g., text/csv seen as plain text)
- Can't send document back to XProc to inspect properties if they're already lost

Test-case step takes care of properties itself.

Step Function Lesson 2

The first answer to "What do I need XProc to do?" might not be the step function to call.

- Calling the XProc step being tested
 - But wait, there's more...
- `p:system-property` function
 - A true function — can't call from XSLT
 - So make a step!

Running Tests...

Testing with SAXON HE 12.10 and XML Calabash 3.0.51

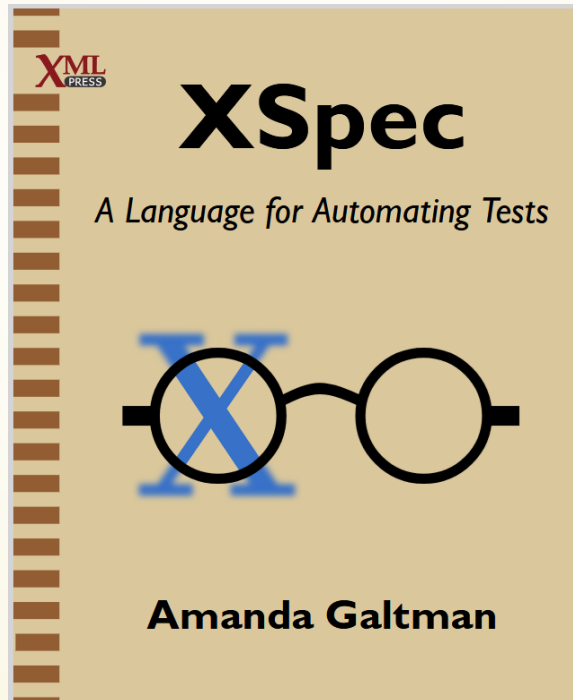
CONCLUSION

- To XProc 3 users: Welcome to XSpec!
- Step functions were the catalyst
- Step functions bring XProc to XSLT, making XProc more versatile

Thanks to Norm Tovey-Walsh, Adrian Bird, Achim Berndzen, Wendell Piez, Christophe Marchand, Sheila Thomson

FYI

My book on XSpec is coming soon (expected in Fall 2026), from XML Press! Here's a draft cover.



<https://xmlpress.net/>