**[RICHARD SEROTER'S ARCHITECTURE MUSINGS](https://seroter.com/" \o "Richard Seroter's Architecture Musings)**

BLOG FEATURING CODE, THOUGHTS, AND EXPERIENCES WITH SOFTWARE AND SERVICES

[**Skip to content**](https://seroter.com/2008/10/07/splitting-delimited-values-in-biztalk-maps/#content)

* [**ABOUT**](https://seroter.com/about/)

* [**MY PLURALSIGHT COURSES**](https://seroter.com/my-pluralsight-courses/)

* [**CONTACT ME**](https://seroter.com/contact-me/)

Splitting Delimited Values in BizTalk Maps

* BY [RICHARD SEROTER](https://seroter.com/author/rseroter/)

* POSTED ON[OCTOBER 7, 2008](https://seroter.com/2008/10/07/splitting-delimited-values-in-biztalk-maps/)

Today, one of our BizTalk developers asked me how to take a delimited string stored in a single node, and extract all those values into separate destination nodes.  I put together a quick XSLT operation that makes this magic happen.

So let’s say I have a source XML structure like this:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

I need to get this pipe-delimited value into an unbounded destination node.  Specifically, the above XML should be reshaped into the format here:

Texto

Descripción generada automáticamente

Notice that each pipe-delimited value is in its own “value” node.  Now I guess I could chained together 62 functoids to make this happen, but it seemed easier to write a bit of XSLT that took advantage of recursion to split the delimited string and emit the desired nodes.

My map has a scripting functoid that accepts the three values from the source (included the pipe-delimited “values” field) and maps to a parent destination record.

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Because I want explicit input variables  to my functoid (vs. traversing the source tree just to get the individual nodes I need), I’m using the “Call Templates” action of the **Scripting** functoid.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

My XSLT script is as follows:

<!-- This template accepts three inputs and creates the destination

"Property" node. Inside the template, it calls another template which

builds up the potentially repeating "Value" child node -->

**<xsl:template name="WritePropertyNodeTemplate">**

<xsl:param name="name" />

<xsl:param name="type" />

<xsl:param name="value" />

<!-- create property node -->

<Property>

<!-- create single instance children nodes -->

<Name><xsl:value-of select="$name" /></Name>

<Type><xsl:value-of select="$type" /></Type>

<!-- call splitter template which accepts the "|" separated string -->

<xsl:call-template name="StringSplit">

<xsl:with-param name="val" select="$value" />

</xsl:call-template>

</Property>

**</xsl:template>**

<!-- This template accepts a string and pulls out the value before the

designated delimiter -->

**<xsl:template name="StringSplit">**

<xsl:param name="val" />

<!-- do a check to see if the input string (still) has a "|" in it -->

<xsl:choose>

<xsl:when test="contains($val, '|')">

<!-- pull out the value of the string before the "|" delimiter -->

<Value><xsl:value-of select="substring-before($val, '|')" /></Value>

<!-- recursively call this template and pass in

value AFTER the "|" delimiter -->

<xsl:call-template name="StringSplit">

<xsl:with-param name="val" select="substring-after($val, '|')" />

</xsl:call-template>

</xsl:when>

<xsl:otherwise>

<!-- if there is no more delimiter values, print out

the whole string -->

<Value><xsl:value-of select="$val" /></Value>

</xsl:otherwise>

</xsl:choose>

**</xsl:template>**