

The Wayback Machine - <https://web.archive.org/web/20040606014049/http://www.xml.com:80/pub/a/2004/05/26/qa...>



XML.COM

WEBSERVICES.XML

O'REILLY NETWORK

OREILLY.COM

[Resources](#) | [Buyer's Guide](#) | [Newsletter](#) | [Safari Bookshelf](#)

Download Stylus Studio - The World's Best XML Editor

TOPICS

[Business](#)[Databases](#)[Graphics](#)[Metadata](#)[Mobile](#)[Programming](#)[Schemas](#)[Style](#)[Web](#)[Web Services](#)

From XML to SMIL

by [John E. Simpson](#)
May 26, 2004

[Print](#)
[Email article link](#)
[Discuss](#)

Sponsored By:

Q: How do I create a SMIL text region using XML input?

I am new to XSLT and SMIL, but I am attempting to create a SMIL presentation based on an XML document, using XSLT. Can anyone tell me how to take the text contained in my XML document, and output it to a region of my SMIL presentation? As yet I have only been able to work out how to output text using a separate .txt file, like this:

```
<text src="../media/audio/intro.txt"
id="intro" region="txt"/>
```

This works fine, but now I need to take some text from my XML file.

A: If I understand the question correctly, the answer is pretty simple. But before getting to it, some SMIL background for those who might be *really* new to it.

SMIL (Synchronized Multimedia Integration Language) is now at version 2.0, a full-fledged [Recommendation](#) of the W3C. (SMIL's version 1.0 was one of the first officially endorsed XML vocabularies out of the gate following the release of the [XML 1.0 Recommendation](#) in 1998.) But what is it?

SMIL defines an XML vocabulary for describing the content and presentation of not just text, but also images, audio, and all the other beasts in the multimedia zoo. A SMIL document defines the layout of the presentation -- where visual objects are placed physically in a window -- as well as its timing -- how long to display a particular window or portion of it.

ESSENTIALS

[Annotated XML](#)[What is XML?](#)[What is XSLT?](#)[What is XSL-FO?](#)[What is XLink?](#)[What is XML Schema?](#)[What is XQuery?](#)[What is RDF?](#)[What is RSS?](#)[What are Topic Maps?](#)[What are Web Services?](#)[What are XForms?](#)[XSLT Recipe of the Day](#)[Manage Your Account](#)[Forgot Your Password?](#)

FIND

[Search](#)[Article Archive](#)



► COLUMNS

[<taglines/>](#)
[Dive into XML](#)
[Hacking the Library](#)
[Jon Udell](#)
[Perl and XML](#)
[Practical XQuery](#)
[Python and XML](#)
[Rich Salz](#)
[Sacré SVG](#)
[Standards Lowdown](#)
[Transforming XML](#)
[XML Q&A](#)
[XML-Deviant](#)

► GUIDES

[XML Resources](#)
[Buyer's Guide](#)
[Events Calendar](#)
[Standards List](#)
[Submissions List](#)

► TOOLBOX

[Syntax Checker](#)

Developer Shed

- Open Source
- ASP Help
- Developer Tutorials
- Computer Hardware
- Search Engine Optimization
- Scripts

ATOM FEED

RSS 1.0

Traveling to a Tech Show?

[New York City Hotels](#)
[Canada Hotels](#)
[Chicago Hotels](#)
[Hotel Discounts](#)
[Discount Hotels](#)
[California Hotels](#)
[Hotel Rooms](#)

XML.com
supported by:

[Debt Consolidation Loans](#)
[Home Refinance](#)

Here's an example, taken from the SMIL Recommendation:

```

<smil
xmlns="http://www.w3.org/2001/SMIL20/">
  <head>
    <layout>
      <root-layout width="320" height="480" />
      <region id="a" top="5" />
    </layout>
  </head>
  <body>
    <text region="a" src="text.html" dur="10s" />
  </body>
</smil>

```

Translated, this establishes a simple presentation, the characteristics of which include the following:

- The physical dimensions of the presentation window are 320 pixels by 480 pixels.
- The portion of the presentation, the region, named "a" begins 5 pixels down from the top of the presentation window.
- Region "a" contains simple text, taken from a file named text.html, and its contents are to be displayed in the presentation window for ten seconds.

Of particular interest to the questioner is the `text` element. Like all the so-called SMIL media object elements (the others include, for example, `video`, `audio`, and `img`), `text` is always an empty element: its "content" is located not in the SMIL document itself, but merely pointed to by way of the `src` attribute. This is where the questioner is getting hung up.

A SMIL document author who wants to include the text content of an XML document has two choices:

1. Transform the document's contents (probably using XSLT) into a simple text document, which can then be targeted using the `src` attribute in the expected manner (as in the example above from the SMIL Rec).
2. Use a special form of the `src` attribute value to include literal text.

Each approach has its pros and cons. I'll discuss them in turn, and use the following as the XML document whose content is to be included in the SMIL presentation:

```

<mydoc>
  <verbiage>Now is the winter of our
discontent</verbiage>
</mydoc>

```

Option 1: Use XSLT to transform the XML document to text

This option requires a separate step, run once (whenever the XML document's content changes). That step is an XSLT transformation to extract the XML document's content and save it to a simple text file; this text file then becomes the target of the `text` element's `src` attribute.

The key to a transformation such as this is to use in the stylesheet an `xsl:output` element, with a `method` attribute whose value is "text". For this extremely simple sample document, a stylesheet such as the following will suffice:

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="text"/>
  <xsl:template match="verbiage">
    <xsl:value-of select="."/>
  </xsl:template>
</xsl:stylesheet>
```

Directing the output of the transformation to a file enables you to use a SMIL `text` element just as the questioner proposes:

```
<text src="intro.txt" id="intro"
  region="txt"/>
```

The advantage of this option is that it's relatively simple. The disadvantage is that, despite the simplicity, it's also somewhat inelegant. This is especially true if you've got a lot of text elements in your SMIL presentation, each with different content, even if the content for them all is derived from the same XML source document. In this case, you need to run a separate transformation -- resulting in a separate text file -- for each text element. You may end up having to juggle way too many files for your comfort: an `intro.txt`, yes, but also (say) a `slide01.txt` through `slidenn.txt`.

Option 2: Use the special data: form of the src attribute value

You won't, to my knowledge, find this option described anywhere in the SMIL 2.0 Recommendation. That's because it relies on an otherwise seldom-used form of a URI; this form, the "data" scheme, is documented in the Internet Engineering Task Force (IETF) [RFC2397](#) -- dating (like SMIL itself, come to think of it) all the way back to the summer of 1998.

The data scheme, unlike other URI forms, doesn't open with a protocol signifier (expressed in an absolute URI or implied in a relative one) such as `http://`, `ftp://`, or `gopher://`. Furthermore, it doesn't point to external content. Rather, it embeds the content directly in the URI itself, as inline data. Here's an example of such a URI as it might appear in the questioner's SMIL presentation:

```
<text
  src="data:text/plain,Introduction"
  id="intro" region="txt" />
```

A data-schemed URI thus consists of the word `data` followed by a colon, an Internet media type (here, `text/plain`), a comma, and the inline content itself. If the Internet media type is `text/plain`, it may be omitted, in which case the colon is followed directly by the comma:

```
<text
  src="data:,Introduction"
  id="intro"
  region="txt"
/>
```

Simple enough so far. The catch, of course, is that the questioner's content still resides entirely in an external file. Ultimately what this suggests is that you shouldn't handcraft your SMIL presentation at all; instead, use XSLT to crank it out, with your external document as the source tree. A portion of such a stylesheet, given our sample XML document, might look like this:

```
<xsl:template match="verbiage">
  <text src="data:,{.}"
  id="intro" region="txt"
  />
</xsl:template>
```

Note that the `src` attribute's value contains an attribute value template (AVT) delimited by curly braces. This AVT instructs the XSLT processor to replace the curly-braced expression with the string-value of the context node -- i.e., in this case, the `verbiage` element. Also note that because the result of the transformation is to be SMIL, which is a form of XML, the stylesheet should not include the `<xsl:output method="text"/>` code used in the previous example.

When passed through an XSLT processor such as Saxon, this transformation produces the expected result:

```
<text src="data:,Now is the winter of our
discontent" id="intro"
region="txt"
/>
```

Expected, yes. Unfortunately, we're not out of the woods yet: the "data" URI scheme, odd-looking though it is, must still conform to the rules of other URI schemes. Among those rules are that certain special characters must be escaped, including spaces.

You probably already know that the proper URI escape sequence for a space is `%20`. Thus, what we need to end up with is a text element which looks like this:

```
<text
src="data: ,Now%20is%20the%20winter%20of%20our%20discontent"

id="intro" region="txt"
/>
```

Depending on the [SMIL processor](#) you intend to use, and the server (if any) involved, you may also be able to substitute a plus sign for each space:

```
<text
src="data: ,Now+is+the+winter+of+our+discontent"

id="intro" region="txt"
/>
```

Regardless of the replacement characters, this might seem an insurmountable difficulty -- requiring, as an XSLT-phobic imagination might have it, some kind of exotic substring-and-test operation against each character in the inline text. If you're lucky and can use the plus signs, calm down, recalling the availability of XSLT's `transform()` function. (I leave to you, fortunate ones, the working out of the actual XSLT code at this point.) If you're not so lucky, you may indeed need additional help; in this case, look into using one of the implementations of the [EXSLT `str:encode-uri\(\)` function](#) or the [XQuery `escape-uri\(\)` function](#).

(By the way, spaces aren't the only special characters in URIs which require escaping. Others may be found in Section 2.4 of IETF [RFC2396](#), "Uniform Resource Identifiers (URI): Generic Syntax.")

So which option should you use?

Ultimately, you'll have to answer that question for yourself. It depends on your requirements, your skill, and comfort levels with XSLT and SMIL, and perhaps your personal taste. I would probably choose Option 2. I like the idea of machine-generating SMIL documents rather than hand-coding them and working backwards to shoehorn the content into the hand-coded framework. For the same reason, I'd resist hand-coding SVG and XSL-FO documents, but those are subjects for later columns, not this one.



Share your experience in our forum.

(* You must be a [member](#) of XML.com to use this feature.)

[Comment on this Article](#)

[Contact Us](#) | [Our Mission](#) | [Privacy Policy](#) | [Advertise With Us](#) | [Site Help](#) | [Submissions Guidelines](#)

Copyright © 2004 O'Reilly Media, Inc.