DO MORE .

TYPEFI®

TECHNICAL DOCUMENTATION:

Typefi CXML Guide

Automation for print, online and mobile

release: 3.0

Australia . USA . UK . Netherlands . typefi.com

Typefi Systems Pty Ltd
Suite 1 / 61–63 Primary School Ct
Maroochydore QLD 4558
+61 7 3102 5444
www.typefi.com

Release: 3.0

Contents

Typefi CXML Guide

Introduction

CXML is an abbreviation for Content XML. It is the data format used by Typefi Publish to encode content.

In many Typefi Publish deployments CXML is completely hidden from users. Content is marked up in the Typefi Writer and then automatically converted to CXML before being sent to the Typefi Engine.

Sometimes a custom conversion to CXML from a third party format is performed. This guide is aimed at developers who are creating these conversions.

This guide covers CXML 3.0. The schema can be found at:

https://www.typefi.com/TPS/content/3_0/ContentXML.xsd

Hello, World

Here is a simple CXML document that will run through the Typefi Engine.

```
<content xmlns="http://www.typefi.com/ContentXML">
 <section type="Chapter">
  <p>Hello, World.</p>
 </section>
</content>
```

Although it will typically look more like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<tps:content
 xmlns:tps="http://www.typefi.com/ContentXML"
 xmlns:xsl="http://www.w3.org/2001/XMLSchema-instance"
 schemaVersion="3.0"
 xsl:schemaLocation="http://www.typefi.com/ContentXML
 http://www.typefi.com/TPS/content/3_0/ContentXML.xsd"
 whiteSpaceMode="strict">
 <tps:section type="Chapter" id="49937a1b-bbe9-4836-8651-670a164af8ee">
  <tps:p type="Body">Hello, World.</tps:p>
 </tps:section>
</tps:content>
```

For clarity all our examples with be in the simpler form above.

Notes:

- The document encoding must be `UTF-8`. This is the only supported encoding for CXML.
- For a job to run through the Typefi Engine the root tag must be `<content>` (even though `<section>` is valid CXML).
- The XML namespace MUST be set to `http://www.typefi.com/ContentXML` otherwise the Engine will ignore the tags.
- All content is broken into `<section>`s. No content can appear outside of a `<section>`.
- Each `<section>` requires a `type` attribute which must correspond to a section name created in the Typefi Designer.
- Under `<section>` we see one of the most common tags in CXML, the paragraph: `<p>`.

Styling

CXML text can be styled at three different levels.

Paragraph Styles

Paragraph Styles are introduced through the type attribute on the `<p>` tag.

They must correspond to paragraph styles created in Typefi Designer.

Character Styles

Character Styles are introduced through the type attribute on the `<c>` tag.

They must correspond to character styles created in Typefi Designer.

Soft Styles

Soft Styles are introduced through the type attribute on the `<style>` tag.

The 8 supported soft style types are `bold`, `italic`, `underline`, `strikethrough`, `superscript`, `subscript`, `allCaps`, and `smallCaps`.

```
<content>
 <section type="Chapter">
  <p type="First">Some <c type="Emphasis">character styled</c> text.</p>
  <p type="Body">Some <style type="bold">bold</style> text.</p>
  <p type="Body">Some <style type="italic">italic</style> text.</p>
  <p type="Body">Some <style type="bold">
   <style type="italic">bold italic</style>
   </style> text.</p>
  <p type="Body">Some <c type="Emphasis">character styled and
   <style type="bold><style type="italic">bold italic</style></style>
   </c> text.</p>
 </section>
</content>
```

White Space

White space processing was significantly enhanced in CXML 2.0.

There is an attribute on `<content>` and `<section>` tags called `whiteSpaceMode`. Like `strictSpace` it is optional (to maintain backwards compatibility). If both are present `whiteSpaceMode` overrides `strictSpace`.

The `strictSpace` attribute is now deprecated. Still supported, but deprecated.

Like strictSpace, whiteSpaceMode applies to all content within the relevant content or section, and a single document can mix up white space modes on a section by section basis.

The 3 valid values for whiteSpaceMode are classic, strict and preserve:

Classic White Space

This is the behaviour of early versions of Typefi Publish. It is available for backward compatibility only. It is not recommended for use.

- All occurrences of #x9 (tab), #xA (line feed) and #xD (carriage return) are replaced with #x20 (space).
- Contiguous sequences of spaces are collapsed to a single space.
- Leading spaces are trimmed from the beginning of <p> elements.

It can be activated in 3 ways:

- It's the default setting when no strictSpace or whiteSpaceMode attribute exists on the <content> or <section> tag, or
- Add the attribute strictSpace="false" on the <content> or <section> tag, or
- Add the attribute whiteSpaceMode="classic" on the <content> or <section> tag.

Notes:

- If you pretty-print your CXML it may alter the white space in the final document.
- If you wish to preserve sequences of white space they must be escaped with the tags <s> (space), <t> (tab) or <l> (newline/soft return).
- Each of these tags has the optional attribute c which indicates a count of how many times to repeat.

Example:

```
<content whiteSpaceMode="classic">
 <section type="Chapter">
  <p>Five spaces:<s/><s/><s/><s/><s/></p>
  <p>Five spaces:<s c="5"/></p>
  <p>Five tabs:<t/><t/><t/><t/><t/></p>
  <p>Five tabs:<t c="5"/></p>
  <p>Five newlines:<l/><l/><l/><l/><l/></p>
  <p>Five newlines:<l c="5"/></p>
 </section>
</content>
```

Strict White Space

This mode was added to make CXML completely immune from all pretty-printing.

- All occurrences of #x9 (tab), #xA (line feed) and #xD (carriage return) are replaced with #x20 (space).
- Contiguous sequences of spaces are collapsed to a single space.
- Leading and trailing spaces are removed from all text nodes.

It can be activated in 2 ways:

- Add the attribute strictSpace="true" on the <content> or <section> tag, or
- Add the attribute whiteSpaceMode="strict" on the <content> or <section> tag.

Example:

```
<content whiteSpaceMode="strict">
 <section type="Chapter">
  <p>Some<s/><style type="italic">italic</style><s/>text.</p>
 </section>
</content>
```

Note the two `<s/>` tags are required for spaces to appear in the final document.

Preserve White Space

This was added to give CXML absolute, literal control of white space.

All white space is considered significant and will be passed through to the final document. Do not use this mode if your CXML could be pretty-printed somewhere along the line.

It can be activated in only 1 way:

- Add the attribute `whiteSpaceMode="preserve"` on the `<content>` or `<section>` tag.

Example:

```
<content whiteSpaceMode="preserve">
 <section type="Chapter">
  <p>Some <style type="italic">italic</style> text.</p>
 </section>
</content>
```

Fields

Both Project Fields and Section Fields can be set in CXML.

Project Fields

Project Fields are set with the `<fieldSet>` tag directly under the `<content>` tag.

Project Fields can also be set on the engine job description. When a particular Project Field is specified in both places the engine job description value overrides the CXML value.

Section Fields

Section Fields are set with the `<fieldSet>` tag directly under the `<section>` tag.

```
<content>
 <fieldSet name="Title" value="A Tale of Two Cities"/>
 <fieldSet name="Author" value="Charles Dickens"/>
 <section type="Chapter">
  <fieldSet name="ChapterTitle" value="The Period"/>
  <p>It was the best of times, it was the worst of times,
```

Images

Images are placed in CXML with the `<image>` tag.

They can exist both inside and outside paragraphs.

When the ref attribute is missing or the image cannot be located the comment attribute is displayed in its place.

```
<content>
```

```
 <section type="Chapter">
  <p>The official flag of the Soviet Union consisted of a plain red
   flag, with a hammer <image ref="hammer.png" comment="Hammer"/>
   crossed with a sickle <image ref="sickle.png" comment="Sickle"/>
   and a red star <image ref="redstar.png" comment="Red Star"/> in
   the upper hoist.</p>
  <image ref="sovietflag.png" comment="Soviet Flag"/>
  <p>With the disintegration of the USSR on 25 December 1991, the
   flag ceased to be a national flag.</p>
 </section>
</content>
```

An image inside a paragraph is placed as an inline image.

An image outside a paragraph is placed into an *Element Image Frame*. If there are no (or not enough) *Element Image Frames* then the image is ignored and a warning is logged.

Elements

Elements are used to represent a chunk of content that is outside the main story flow.

Elements (fixed, inline and floating) are all placed in CXML with the `<context>` tag.

The mandatory `type` attribute should refer to an element name in the Typefi Designer.

The mandatory `id` attribute is a unique ID.

The optional `variant` attribute should refer to a variant name in the Typefi Designer. Variants are alternate renderings of a element.

Element fields are specified with the `<fieldSet>` tag directly under the `<context>` tag.

They can exist both inside and outside paragraphs.

They can be nested to any number of levels.

```
<content>
 <section type="Chapter">
  <p>The proto-Baltic forefathers of the Latvian people have lived on
  the eastern coast of the Baltic Sea since the third millennium BC.</p>
  <context type="Map" id="1">
   <image ref="Baltic_Tribes_c_1200.png"/>
   <p>Baltic Tribes, about 1200 CE.</p>
  </context>
  <p>In the 13th century, the Livonian Confederation developed under
  the Germanic authorities consisting of Latvia and Estonia.</p>
  <context type="Map" id="2">
   <image ref="Confederation_of_Livonia_1260.png"/>
   <p>The Livonian Confederation in 1260.</p>
  </context>
  <p>The 1490s were a time of great changes for the inhabitants of
  Latvia, notable for the reformation and the collapse of the
  Livonian nation.</p>
  <context type="Sidebar" id="3">
   <p>After the Livonian War (1558-1583) today's Latvian territory
   came under Polish-Lithuanian rule.</p>
  </context>
  <p>The Lutheran faith was accepted in Kurzeme, Zemgale and Vidzeme,
   but the Roman Catholic faith maintained its dominance in Latgale -
   it remains so to this day.</p>
 </section>
</content>
```

Conditions

Conditions are used to filter content.

Conditions are placed in CXML with the `<condition>` tag and the condition attribute on the `<section>` tag.

The content inside the `<condition>` tag will only appear when the condition specified by the type attibute is included in the job description.

They can exist both inside and outside paragraphs.

```
<content>
 <section type="Questions">
  <condition type="US"><p>Analyze this:</p></condition>
  <condition type="UK"><p>Analyse this:</p></condition>
  <p>1. What is your
   <condition type="US">favorite color</condition>
   <condition type="UK">favourite colour</condition>?</p>
 </section>
 <section type="Answers" condition="TeacherEdition">
  <p>1. Blue, no Yellow...</p>
 </section>
</content>
```

Tables

Recent Changes

The `<table> type` attribute is now optional. This allows unstyled tables to appear in the content. Previous versions of Typefi Publish forced unstyled tables to adopt the awkward name "Non-Typefi Table". Unstyled tables will use the [Basic Table] style by default (unknown types will be based on [Basic Table]).

`<table>` has an optional width attribute, which can be either a fixed, percentage or proportional width.

`<table>` tags can now be placed inside paragraphs. A table wrapped within a CXML paragraph style will override the inherited (external) paragraph style of a master table. When a percentage or proportional width is specified, the effective table width is reduced by the sum of half the left and right table border stroke weight and any left/right text frame insets and any left/right paragraph indents.

The `<table>` attributes `keepProportionalWidth`, `keepVerticalCellAlignment` and `keepHorizontalCellAlignment` are all deprecated.

The `<colspec>` colwidth attribute accepts fixed and proportional measurements, in addition to legacy support for percentage measurements. The Oasis Exchange table specification states:

Either proportional measure of the form number*, e.g., "5*" for 5 times the proportion, or "*" (which is equivalent to "1*"); fixed measure, e.g., 2pt for 2 point, 3pi for 3 pica. (Mixed measure, e.g., 2*+3pt, while allowed in the full CALS table model, is not supported in this Exchange model.) Coefficients are positive integers or fixed point numbers; for fixed point numbers, a leading (possibly 0) integer part is required, and implementations should support at least 2 decimal places. A value of "" [the null string] is treated as a proportional measure of "1*".

The fixed unit values are case insensitive. The standard list of allowed unit values is "pt" (points), "cm" (centimeters), "mm" (millimeters), "pi" (picas), and "in" (inches). The default fixed unit should be interpreted as "pt" if neither a proportion nor a fixed unit is specified.

These `colwidth` values are all valid:

```
<table>
 <tgroup cols="8">
  <colspec colname="1" colwidth="25"/>
  <colspec colname="2" colwidth="40pt"/>
  <colspec colname="3" colwidth="6.35cm"/>
  <colspec colname="4" colwidth="31.75mm"/>
  <colspec colname="5" colwidth="3.5pi"/>
  <colspec colname="6" colwidth="1in"/>
  <colspec colname="7" colwidth="1*"/>
  <colspec colname="8" colwidth="2*"/>
```

Overview

The CXML table schema is loosely based on the OASIS format.

This schema can be found at: https://www.oasis-open.org/specs/a503.htm.

The top level tag is `<table>`. The mandatory `type` attribute should refer to an table name in the Typefi Designer.

Inside a table is a collection of `<tgroup>` tags. These allow different numbers of columns within the one table. Each `<tgroup>` has the mandatory attribute `cols` which specify the number of columns in that group.

Directly below `<tgroup>` can be `<colspec>`, `<thead>`, `<tbody>` and `<tfoot>` tags. Only `<tbody>` is mandatory. The `<thead>`, `<tbody>` and `<tfoot>` tags can all contain any number of `<row>` tags.

The `<row>` tag contains `<entry>` tags for each table cell.

```
<content>
 <section type="Chapter">
  <table type="Population">
   <tgroup cols="3">
    <thead>
     <row>
      <entry><p>Rank</p></entry>
      <entry><p>Country</p></entry>
      <entry><p>Population</p></entry>
     </row>
    </thead>
    <tbody>
     <row>
      <entry><p>1</p></entry>
      <entry><p>China</p></entry>
      <entry><p>1,316,200,000</p></entry>
     </row>
     <row>
      <entry><p>2</p></entry>
      <entry><p>India</p></entry>
      <entry><p>1,123,980,000</p></entry>
     </row>
     <row>
      <entry><p>3</p></entry>
      <entry><p>USA</p></entry>
      <entry><p>301,215,000</p></entry>
     </row>
```

```
    </tbody>
   </tgroup>
  </table>
 </section>
</content>
```

Column Spans

Column spans are accomplished via the namest and nameend attributes on the `<entry>` tag. The value of these attributes corresponds to a value in the colname attribute of a `<colspec>` tag.

Row Spans

Row spans are accomplished via the morerows attribute on the `<entry>` tag. The value of these attributes corresponds to a value in the colname attribute of a `<colspec>` tag.

Cell Merging

Date	Consumer price index					Private consumption chain price index	Other consumer price measures	
	All groups	Excluding volatile items	Market prices excluding volatile items				Based on seasonally adjusted quarterly price changes	
			Goods	Services	Total		Weighted median	Trimmed mean
2003/04								
Dec	2.4	2.4	1.6	2.2	1.8	1.0	2.8	2.5

CXML:

```
<table type="CPI">
 <tgroup cols="9">
  <colspec colname="1"/>
  <colspec colname="2"/>
  <colspec colname="3"/>
  <colspec colname="4"/>
  <colspec colname="5"/>
  <colspec colname="6"/>
  <colspec colname="7"/>
  <colspec colname="8"/>
  <colspec colname="9"/>
  <thead>
   <row>
    <entry namest="1" morerows="2" align="right" valign="middle">
     <p>Date</p>
    </entry>
    <entry namest="2" nameend="6" align="center" valign="middle">
     <p>Consumer price index</p>
    </entry>
    <entry namest="7" morerows="2" align="center" valign="middle">
     <p>Private consumption chain price index</p>
    </entry>
   </row>
   <row>
    <entry namest="2" morerows="1" align="center" valign="middle">
     <p>All groups</p>
    </entry>
    <entry namest="3" morerows="1" align="center" valign="middle">
     <p>Excluding volatile items</p>
    </entry>
    <entry namest="4" nameend="6" align="center" valign="middle">
```

```
     <p>Market prices excluding volatile items</p>
    </entry>
   </row>
   <row>
    <entry namest="4" align="center" valign="middle">
     <p>Goods</p>
    </entry>
    <entry namest="5" align="center" valign="middle">
     <p>Services</p>
    </entry>
    <entry namest="6" align="center" valign="middle">
     <p>Total</p>
    </entry>
    <entry namest="8" align="center" valign="middle">
     <p>Weighted median</p>
    </entry>
    <entry namest="9" align="center" valign="middle">
     <p>Trimmed mean</p>
    </entry>
   </row>
  </thead>
  <tbody>
   <row>
    <entry namest="1" align="right" valign="middle">
     <p>2003/04</p>
    </entry>
    <entry namest="2" nameend="9" align="center" valign="middle"/>
   </row>
   <row>
    <entry namest="1" align="right" valign="middle"><p>Dec</p></entry>
    <entry namest="2" align="center" valign="middle"><p>2.4</p></entry>
    <entry namest="3" align="center" valign="middle"><p>2.4</p></entry>
    <entry namest="4" align="center" valign="middle"><p>1.6</p></entry>
    <entry namest="5" align="center" valign="middle"><p>2.2</p></entry>
    <entry namest="6" align="center" valign="middle"><p>1.8</p></entry>
    <entry namest="7" align="center" valign="middle"><p>1.0</p></entry>
    <entry namest="8" align="center" valign="middle"><p>2.8</p></entry>
    <entry namest="9" align="center" valign="middle"><p>2.5</p></entry>
   </row>
  </tbody>
 </tgroup>
</table>
```

Table Cell Styles

Cell styles are supported via the `type` attribute on the `<entry>`.

```
<entry type="Green">
 <p>Cell content</p>
</entry>
```

Hyperlinks

Hyperlink targets are specified via the ref attribute on the `<link>` tag.

The displayed link content is enclosed by the `<link>` tag and can be arbitrarily complex.

```
<content>
 <section type="Chapter">
  <p><link ref="http://maps.google.com/">Google Maps</link></p>
 </section>
</content>
```

Document links

Document links behave just like hyperlinks that link to a location in the same document.

```
<content>
 <section type="Chapter">
  <p id="1">Paragraph one.</p>
  <p><anchor id="1"/>Anchor one.</p>
  <p>Document link to
   <doclink refType="paragraph" refId="1">paragraph one</doclink>.
  </p>
  <p>Document link to
   <doclink refType="anchor" refId="1">anchor one</doclink>.
  </p>
 </section>
</content>
```

Cross-References

Cross-references have a completely new representation in CXML 2.0. This is because the Typefi Engine has switched to the native cross-reference implementation in InDesign. It was just not possible to retain support for the existing `<xref>` tag.

And so the `<xref>` tag is deprecated. Jobs will still run but `<xref>` tags will be ignored. Instead, a warning will be written to the log file.

A newer and much simpler tag has been introduced `<ref>`. It has three required attributes:

- The `refType`. Either `anchor` or `paragraph`.
- The `refId`. The unique id of the anchor or paragraph.
- The `format`. Corresponds to the name of the cross-reference format in InDesign.

Anchor example:

```
<p><anchor id="1"/>Anchor one.</p>
<p>Document link to anchor 1 on page <ref refType="anchor" refId="1" format="Page
Number"/>.</p>
```

Paragraph example:

```
<p id="1">Paragraph one.</p>
<p>Document link to paragraph 1 on page <ref refType="paragraph" refId="1"
format="Page Number"/>.</p>
```

Footnotes

Footnotes are embedded anywhere in the content via the `<footnote>` tag.

Note that you must create a new `<p>` tag inside the `<footnote>` tag itself.

```
<p>Relatively few of his designs were constructed or were even feasible during
his lifetime,<footnote><p>Modern scientific approaches to metallurgy and
engineering were only in their infancy during the Renaissance.</p></footnote>
but some of his smaller inventions, such as an automated bobbin winder and a
machine for testing the tensile strength of wire, entered the world of
manufacturing unheralded.<footnote><p>A number of Leonardo's most practical
inventions are displayed as working models at the Museum of
Vinci.</p></footnote></p>
```

Note that there was a change to footnote processing in Typefi 6. Leading whitespace in footnote paragraphs (inserted by Word) is now stripped.

So we used to have this:

```
<footnote><p type="footnote text"><s/>Footnote one.</p></footnote>
```

Now we have this:

```
<footnote><p type="footnote text">Footnote one.</p></footnote>
```

Indexing

Index terms were introduced in CXML 2.0.

They are loosely based on DocBook's `<indexterm>` tag.

A simple CXML index reference:

```
<indexterm>
 <primary>New York</primary>
</indexterm>
```

A two level CXML index reference:

```
<indexterm>
 <primary>New York</primary>
 <secondary>accommodation</secondary>
</indexterm>
```

InDesign supports up to four index levels. So do we:

```
<indexterm>
 <primary>Level one</primary>
 <secondary>Level two</secondary>
 <tertiary>Level three</tertiary>
 <quaternary>Level four</quaternary>
</indexterm>
```

You can specify a character style to apply to the index page number:

```
<indexterm pageNumberStyle="Index Bold">
 <primary>New York</primary>
</indexterm>
```

See references are handled through their own tag:

```
<indexterm>
 <primary>Potato</primary>
 <see>Tuber</see>
</indexterm>
```

And so with *See also* references:

```
<indexterm>
 <primary>Apple</primary>
 <seealso>Fruit</seealso>
</indexterm>
```

You can apply alternative sort orders to any level:

```
<indexterm>
 <primary sortas="Prince">The Prince</primary>
</indexterm>
```

Normal old CXML character styles are supported within index levels:

```
<indexterm>
 <primary>Doctrine of <c type="Index Italic">stare decisis</c></primary>
</indexterm>
```

An index reference to a range of text can be created in two ways.

(1) The Microsoft Word approach is how the CXML comes out of the Typefi Writer. An indexterm begins the range, a bookmark (anchor) ends it:

```
<indexterm class="bookmark" id="Arizona">
 <primary>Arizona</primary>
</indexterm>
.
.
.
<anchor name="Arizona"/>
```

(2) The DocBook approach can be used in workflows where the CXML is not coming from the Typefi Writer/Converter:

```
<indexterm class="startofrange" id="Arizona"/>
 <primary>Arizona</primary>
</indexterm>
.
.
.
<indexterm class="endofrange" startref="Arizona"/>
```

Special characters

The <char> tag was introduced in CXML 2.0. It enables the insertion of some special characters. The valid values are:

```
<char type="currentPageNumber"/>
<char type="nextPageNumber"/>
<char type="previousPageNumber"/>
<char type="sectionMarker"/>
<char type="rightIndentTab"/>
<char type="indentToHere"/>
<char type="endNestedStyleHere"/>
```

Breaks

The <break> tag was introduced in CXML 2.0. It is used to control the type of paragraph break between two paragraphs. The valid values are:

```
<break type="column"/>
<break type="frame"/>
<break type="evenPage"/>
<break type="oddPage"/>
<break type="page"/>
```

Note that this tag is only valid *between* paragraphs. It affects the break of the previous paragraph. e.g.

```
<p>Paragraph one.</p>
<break type="column"/>
<p>Paragraph two.</p>
```

Lists

Lists are mostly controlled within InDesign using the standard paragraph style properties.

Most of the CXML list markup is simply ignored by the Typefi Engine.

Except for two attributes, both on the `<ol>` tag:

`<ol start="5">` forces the list numbering to restart at 5.

`<ol style="a">` forces the list numbering to lowercase alpha. The 5 valid values are:

- `1` for Arabic
- `A` for uppercase alpha
- `a` for lowercase alpha
- `I` for uppercase roman
- `i` for lowercase roman

Video

The `<video>` tag was introduced in CXML 3.0.

Simple video:

```
<video ref="dancing-baby.mp4" />
```

Using all the available attributes:

```
<video ref="dancing-baby.mp4"
 comment="Dancing Baby"
 autoplay="true"
 controls="true"
 loop="true"
 id="1234" />
```

With no poster image:

```
<video ref="dancing-baby.mp4">
 <poster type="none" />
</video>
```

With the default (standard, generic) poster image:

```
<video ref="dancing-baby.mp4">
 <poster type="default" />
</video>
```

With a custom poster image:

```
<video ref="dancing-baby.mp4">
 <poster type="image" ref="dancing-baby.png" />
</video>
```

With alternate video files:

```
<video ref="dancing-baby.mp4">
 <altRef ref="dancing-baby.avi" />
 <altRef ref="dancing-baby.ogg" />
</video>
```

One possible use of the (freeform text) altRef.target attribute:

```
<video ref="dancing-baby.mp4">
 <altRef ref="dancing-baby.avi" target="pdf" />
 <altRef ref="dancing-baby.ogg" target="epub" />
</video>
```

Everything:

```
<video ref="dancing-baby.mp4"
 comment="Dancing Baby"
 autoplay="true"
 controls="true"
 loop="true"
 id="1234">
 <poster type="image" ref="dancing-baby.png" />
 <altRef ref="dancing-baby.avi" target="pdf" />
 <altRef ref="dancing-baby.ogg" target="epub" />
</video>
```

Audio

The <audio> tag was introduced in CXML 3.0.

Simple audio:

```
<audio ref="never-gonna-give-you-up.mp3" />
```

Using all the available attributes:

```
<audio ref="never-gonna-give-you-up.mp3"
 comment="Never Gonna Give You Up"
 autoplay="true"
 controls="true"
 loop="true"
 id="1234" />
```

With no poster image:

```
<audio ref="never-gonna-give-you-up.mp3">
 <poster type="none" />
</audio>
```

With the default (standard, generic) poster image:

```
<audio ref="never-gonna-give-you-up.mp3">
 <poster type="default" />
</audio>
```

With a custom poster image:

```
<audio ref="never-gonna-give-you-up.mp3">
 <poster type="image" ref="never-gonna-give-you-up.png" />
</audio>
```

With alternate audio files:

```
<audio ref="never-gonna-give-you-up.mp3">
 <altRef ref="never-gonna-give-you-up.m4a" />
 <altRef ref="never-gonna-give-you-up.ogg" />
</audio>
```

One possible use of the (freeform text) altRef.target attribute:

```
<audio ref="never-gonna-give-you-up.mp3">
 <altRef ref="never-gonna-give-you-up.m4a" target="pdf" />
 <altRef ref="never-gonna-give-you-up.ogg" target="epub" />
</audio>
```

Everything:

```
<audio ref="never-gonna-give-you-up.mp3"
 comment="Never Gonna Give You Up"
 autoplay="true"
 controls="true"
 loop="true"
 id="1234">
 <poster type="image" ref="never-gonna-give-you-up.png" />
 <altRef ref="never-gonna-give-you-up.m4a" target="pdf" />
 <altRef ref="never-gonna-give-you-up.ogg" target="epub" />
</audio>
```