Typefi Publish 6 - CXML Guide

February 2013

Typéfi®
typefi.com

This document was created with Typefi Publish 6.

6.0.1

Contents

CXML 2.0

What is CXML?

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.

The current version of CXML (at the time of writing) is 2.0. The schema can be found at ***http://www.typefi.com/TPS/content/2_0/ContentXML.xsd***.

CXML 2.0

Typefi Publish 6 incorporates a major revision to CXML. For the first time in the history of Typefi Publish we have broken backwards compatibility with previous versions of CXML. This is why the version number has jumped from 1.6 to 2.0.

We have done our best to minimize the CXML changes required to process legacy content and, in many documents, no CXML changes will need to be made.

We have documented the process for migrating from CXML 1.6 to 2.0 in a section at the end of this guide.

Hello, World

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

```
<content>
 <section type="Chapter" id="1">
  <p>Hello, World.</p>
 </section>
</content>
```

Although, it will usually look a bit 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="2.0"
 xsl:schemaLocation="http://www.typefi.com/ContentXML
 http://www.typefi.com/TPS/content/2_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:

- For a job to run through the Typefi Engine the root tag must be `<content>` (even though `<section>` is valid CXML).
- 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.
- Each `<section>` requires a `id` attribute which must be unique across sections.
- Under `<section>` we see one of the most common tags in CXML, the paragraph: `<p>`.

Styling CXML

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

```
<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>
```

White space processing

White space processing has been further enhanced in CXML 2.0.

We introduce a new 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 three 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 backwards 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.

```
<content whiteSpaceMode="classic">
 <section type="Chapter" id="1">
  <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" id="1">
  <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 one way:

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

Here is a ***preserve white space*** example:

```
<content whiteSpaceMode="preserve">
 <section type="Chapter" id="1">
  <p>Some <style type="italic">italic</style> text.</p>
 </section>
</content>
```

Fields in CXML

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.

Section Counter Fields do not require a value attribute:

```
<content>
 <fieldSet name=”Title” value=”A Tale of Two Cities”/>
 <fieldSet name=”Author” value=”Charles Dickens”/>
 <section type=”Chapter” id=”1”>
  <fieldSet name=”ChapterTitle” value=”The Period”/>
  <fieldSet name=”ChapterNumber”/>
  <p>It was the best of times, it was the worst of times, it was the age of
wisdom, it was the age of foolishness,
  .
  .
  .
```

Images in CXML

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

The location of the image is specified via the `ref` attribute.

When the `ref` attribute is missing or the image cannot be located a `comment` attribute (if present) will be displayed in its place.

There are no CXML 2.0 schema changes related to images. But there has been a significant change in how images are processed.

`<image>`s that are inline to a story MUST be wrapped in a paragraph.

In earlier version of Typefi Publish images outside of a `<p>` are wrapped in their own paragraph.

In Typefi Publish 6 images outside of a `<p>` are only eligible to be placed in Element Image Frames. If there are no (or not enough) Element Image frames then the image is ignored and a warning is logged.

```
<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>
```

Elements in CXML

Elements are used to represent 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 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.

```
<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>
```

Conditions in CXML

Conditions are a mechanism to selectively use content depending on the output source.

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" id="1">
  <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" id="2" condition="TeacherEdition">
  <p>1. Blue, no Yellow...</p>
 </section>
</content>
```

Tables in CXML

Changes in CXML 2.0

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>` now 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 now 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.

```
<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: ***http://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.

```
<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>
```

Column spans

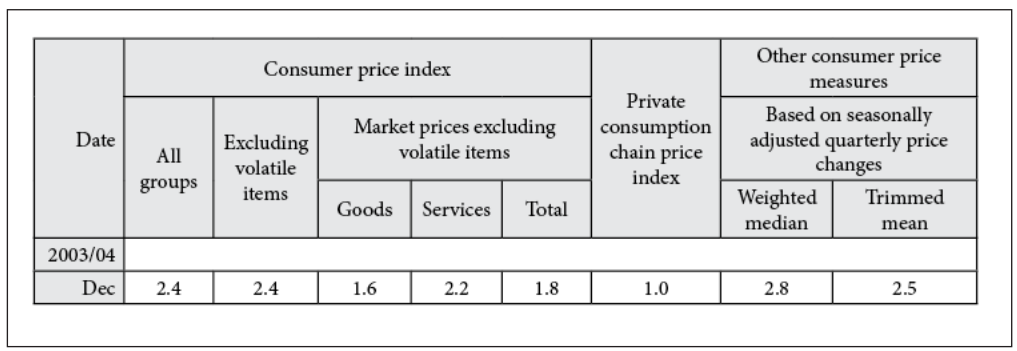
Column spans are accomplished via the `namest` and `nameend` attributes on the `<entry>` tag. The value of these attributes corresponds to a value in the `colname` attribute of a `<colspec>` tag.

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.

Complicated table example

InDesign



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

Desired Output

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

CXML 2.0 adds support for cell styles via the `type` attribute on the `<entry>`.

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

Hyperlinks in CXML

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.

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

Document links in CXML

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

They are new in CXML 2.0.

```
<p><anchor id="1"/>Anchor one.</p>
<p>See <doclink refType="anchor" refId="1">anchor one</doclink>.</p>
```

```
<p id="1">Paragraph one.</p>
<p>See <doclink refType="paragraph" refId="1">paragraph one</doclink>.</p>
```

Cross-references in CXML

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.

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

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

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

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

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

Footnotes in CXML

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 has been a change to footnote processing in Typefi Publish 6. Leading whitespace in footnote paragraphs (inserted by Word) is now stripped by the Converter.

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>
```

Index terms in CXML

Index terms are a new feature in CXML 2.0.

The additions to CXML are based on DocBook's `<indexterm>` tag.

Here is a simple CXML index reference:

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

And here is a two level CXML index reference:

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

InDesign supports up to four index levels. So does Typefi Publish:

```
<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 are ***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 in CXML

The `<char>` tag is new 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 in CXML

The `<break>` tag is new 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:

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

Lists in CXML

Lists in Typefi Publish 6 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

Migrating to CXML 2.0

Here is a checklist to run through when migrating content to CXML 2.0:

- Ensure the document encoding is set to `UTF-8`. This is the only supported encoding in Typefi Publish 6.
- Ensure the `schemaVersion="2.0"` on the `<content>` tag.
- Ensure the `schemaLocation` is set to `http://www.typefi.com/TPS/content/2_0/ContentXML.xsd`.
- The `id` attribute is now required on `<section>` and `<context>`. If you were not previously setting the `id` attribute on these tags then you will have to add this to your CXML.
- `<image>`s intended for Element Image frames MUST NOT be inside a `<p>`.
- `<image>`s intended for inline display MUST be inside a `<p>`.
- You MAY have to remove leading space from `<footnote>` tags. The Typefi Engine 6 no longer expects footnotes to begin with leading spaces.

- `<xref>` tags must be mapped to `<doclink>` and `<ref>` equivalents. **This is potentially the biggest change for legacy CXML content**. The actual mapping will depend on your InDesign template. Here are some example mappings:

A cross-reference to the page number of an anchor:

```
OLD SYNTAX:

<p><anchor id="1"/>Anchor one.</p>
<p>See page <xref refType="anchor" refId="1" partType="pageNumber"/>.</p>

NEW:

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

A cross-reference to the value of a project field:

```
OLD SYNTAX:

<xref refType="project" partType="field" partName="BookTitle"/>

NEW:

No mapping (due to InDesign cross-ref limitations)
```

A cross-reference to the value of a section field:

```
OLD SYNTAX:

<xref refType="section" partType="field" partName="ChapterName"/>

NEW:

No mapping (due to InDesign cross-ref limitations)
```

A cross-reference to the content of a particular paragraph:

```
OLD SYNTAX:

<xref refType="paragraph" refId="1"/>

NEW:

<ref refType="paragraph" refId="1" format="Full paragraph"/>
```

A cross-reference to the page number of a particular paragraph:

```
OLD SYNTAX:

<xref refType="paragraph" refId="1" partType="pageNumber"/>

NEW:

<ref refType="paragraph" refId="1" format="Page Number"/>
```

A cross-reference to the content of an anchor:

```
OLD SYNTAX:

<xref refType="anchor" refId="2"/>

NEW:

No mapping (due to InDesign cross-ref limitations)
```