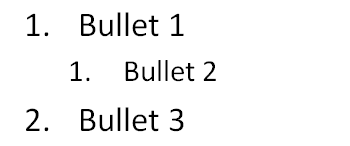
#### Bullets and Numbering

In addition to the above body, paragraph and text run properties there can also be a structure of bullets and numbering that can be defined by utilizing a few of these layers. Since Bullet and Numbering does span multiple formatting levels it will be described on it's own in the following section.

##### buAutoNum (Auto-Numbered Bullet)

This element specifies that automatic numbered bullet points should be applied to a paragraph. These are not just numbers used as bullet points but instead automatically assigned numbers that are based on both buAutoNum attributes and paragraph level.

[Example: Consider the DrawingML content shown below.



<p:txBody>  
 …  
 <a:p>  
 <a:pPr …>  
 <a:buAutoNum type="arabicPeriod"/>  
 </a:pPr>

…  
 <a:t>Bullet 1</a:t>  
 …  
 </a:p>

<a:p>  
 <a:pPr [lvl](lvl.docx)="1"…>  
 <a:buAutoNum type="arabicPeriod"/>  
 </a:pPr>  
 …  
 <a:t>Bullet 2</a:t>  
 …  
 </a:p>

<a:p>  
 <a:pPr …>  
 <a:buAutoNum type="arabicPeriod"/>  
 </a:pPr>  
 …  
 <a:t>Bullet 3</a:t>  
 …  
 </a:p>  
 …  
</p:txBody>

For the above text there are a total of three bullet points. Two of which are at [lvl](lvl.docx)="0" and one at [lvl](lvl.docx)="1". Due to this breakdown of levels, the numbering sequence that should be automatically applied will be 1, 1, 2 as is shown in the picture above. [end](end.docx) example]

|  |
| --- |
| Parent Elements |
| defPPr (§); lvl1pPr (§); lvl2pPr (§); lvl3pPr (§); lvl4pPr (§); lvl5pPr (§); lvl6pPr (§); lvl7pPr (§); lvl8pPr (§); lvl9pPr (§); pPr (§) |

|  |  |
| --- | --- |
| Attributes | Description |
| startAt (Start Numbering At) | Specifies the number that will [start](start.docx) number for a given sequence of automatically numbered bullets. When the numbering is alphabetical, the number should map to the appropriate letter. For instance 1 will map to 'a', 2 to 'b' and so on. If the numbers are larger than 26, then multiple letters should be used. For instance 27 should be represented as 'aa' and similarly 53 should be 'aaa'.  The possible values for this attribute are defined by the [ST\_TextBulletStartAtNum](ST_TextBulletStartAtNum.docx) simple type (§). |
| type (Bullet Autonumbering Type) | Specifies the numbering scheme that is to be used. This allows for the describing of formats other than strictly numbers. For instance a set of bullets may need to be represented by a series of Roman numerals instead of the standard 1,2,3,etc. number set.  The possible values for this attribute are defined by the [ST\_TextAutonumberScheme](ST_TextAutonumberScheme.docx) simple type (§). |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextAutonumberBullet">

<attribute [name](name.docx)="type" type="[ST\_TextAutonumberScheme](ST_TextAutonumberScheme.docx)" use="required"/>

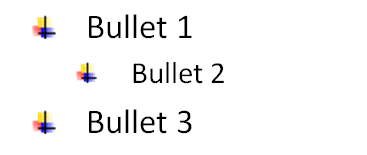
<attribute name="startAt" type="[ST\_TextBulletStartAtNum](ST_TextBulletStartAtNum.docx)" use="optional" default="1"/>

</complexType>

##### buBlip (Picture Bullet)

This element specifies that a picture be applied to a set of bullets. This element allows for any standard picture format graphic to be used instead of the typical bullet characters. This opens [up](up.docx) the possibility for bullets to be anything the generating application would seek to apply.

[Example: Consider the DrawingML shown below.



<p:txBody>  
 …  
 <a:p>  
 <a:pPr …>  
 <a:buBlip>  
 <a:blip r:embed="rId2"/>  
 </a:buBlip>  
 </a:pPr>

…  
 <a:t>Bullet 1</a:t>  
 …  
 </a:p>

<a:p>  
 <a:pPr [lvl](lvl.docx)="1"…>  
 <a:buBlip>  
 <a:blip r:embed="rId2"/>  
 </a:buBlip>  
 </a:pPr>  
 …  
 <a:t>Bullet 2</a:t>  
 …  
 </a:p>

<a:p>  
 <a:pPr …>  
 <a:buBlip>  
 <a:blip r:embed="rId2"/>  
 </a:buBlip>  
 </a:pPr>  
 …  
 <a:t>Bullet 3</a:t>  
 …  
 </a:p>  
 …  
</p:txBody>

For the above text there are a total of three bullet points. Two of which are at [lvl](lvl.docx)="0" and one at [lvl](lvl.docx)="1". Because the same picture is specified for each bullet the levels do not stand out here. The only difference is the indentation as shown in the picture above. [end](end.docx) example]

|  |
| --- |
| Parent Elements |
| defPPr (§); lvl1pPr (§); lvl2pPr (§); lvl3pPr (§); lvl4pPr (§); lvl5pPr (§); lvl6pPr (§); lvl7pPr (§); lvl8pPr (§); lvl9pPr (§); pPr (§) |

|  |  |
| --- | --- |
| Child Elements | Subclause |
| [blip](blip.docx) (Blip) | § |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextBlipBullet">

<sequence>

<element name="[blip](blip.docx)" type="CT\_Blip" minOccurs="1" maxOccurs="1"/>

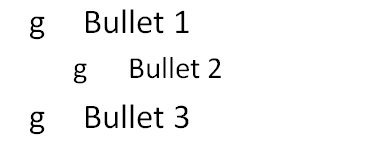
</sequence>

</complexType>

##### buChar (Character Bullet)

This element specifies that a character be applied to a set of bullets. These bullets are allowed to be any character in any font that the system is able to support. If no bullet font is specified along with this element then the paragraph font will be used.

[Example: Consider the DrawingML shown below.



<p:txBody>  
 …  
 <a:p>  
 <a:pPr …>  
 <a:buFont typeface="Calibri"/>  
 <a:buChar char="g"/>  
 </a:pPr>  
 …  
 <a:t>Bullet 1</a:t>  
 …  
 </a:p>

<a:p>  
 <a:pPr [lvl](lvl.docx)="1"…>  
 <a:buFont typeface="Calibri"/>  
 <a:buChar char="g"/>  
 </a:pPr>  
 …  
 <a:t>Bullet 2</a:t>  
 …  
 </a:p>

<a:p>  
 <a:pPr …>  
 <a:buFont typeface="Calibri"/>  
 <a:buChar char="g"/>  
 </a:pPr>  
 …  
 <a:t>Bullet 3</a:t>  
 …  
 </a:p>  
 …  
</p:txBody>

For the above text there are a total of three bullet points. Two of which are at [lvl](lvl.docx)="0" and one at [lvl](lvl.docx)="1". Because the same character is specified for each bullet the levels do not stand out here. The only difference is the indentation as shown in the picture above. [end](end.docx) example]

|  |
| --- |
| Parent Elements |
| defPPr (§); lvl1pPr (§); lvl2pPr (§); lvl3pPr (§); lvl4pPr (§); lvl5pPr (§); lvl6pPr (§); lvl7pPr (§); lvl8pPr (§); lvl9pPr (§); pPr (§) |

|  |  |
| --- | --- |
| Attributes | Description |
| char (Bullet Character) | Specifies the character to be used in place of the standard bullet point. This character may be any character for the specified font that will be supported by the system upon which this document is being viewed.  The possible values for this attribute are defined by the XML Schema string datatype. |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextCharBullet">

<attribute [name](name.docx)="char" type="xsd:string" use="required"/>

</complexType>

##### buClr (Color Specified)

This element specifies the color to be used on bullet characters within a given paragraph. The color is specified using the numerical RGB color format.

[Example: Consider the DrawingML shown below.

<p:txBody>  
 …  
 <a:p>  
 <a:pPr …>  
 <a:buClr>  
 <a:srgbClr [val](val.docx)="FFFF00"/>  
 </a:buClr>  
 </a:pPr>  
 …  
 <a:t>Bullet 1</a:t>  
 …  
 </a:p>  
 …  
</p:txBody>

The color of the above bullet will not follow the text color but instead will have a yellow color specified by [val](val.docx)="FFFF00". This color should only apply to the actual bullet character and not to the text within the bullet. [end](end.docx) example]

|  |
| --- |
| Parent Elements |
| defPPr (§); lvl1pPr (§); lvl2pPr (§); lvl3pPr (§); lvl4pPr (§); lvl5pPr (§); lvl6pPr (§); lvl7pPr (§); lvl8pPr (§); lvl9pPr (§); pPr (§) |

|  |  |
| --- | --- |
| Child Elements | Subclause |
| hslClr (Hue, Saturation, Luminance Color Model) | § |
| prstClr (Preset Color) | § |
| schemeClr (Scheme Color) | § |
| scrgbClr (RGB Color Model - Percentage Variant) | § |
| srgbClr (RGB Color Model - Hex Variant) | § |
| sysClr (System Color) | § |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_Color">

<sequence>

<group ref="EG\_ColorChoice"/>

</sequence>

</complexType>

##### buClrTx (Follow Text)

This element specifies that the color of the bullets for a paragraph should be of the same color as the text run within which each bullet is contained.

[Example: Consider the DrawingML shown below.

<p:txBody>  
 …  
 <a:p>  
 <a:pPr …>  
 <a:buClrTx>  
 </a:pPr>  
 …  
 <a:t>Bullet 1</a:t>  
 …  
 </a:p>  
 …  
</p:txBody>

The color of the above bullet will follow the default text color of the text for the run of text shown above since no specific text color was specified. [end](end.docx) example]

|  |
| --- |
| Parent Elements |
| defPPr (§); lvl1pPr (§); lvl2pPr (§); lvl3pPr (§); lvl4pPr (§); lvl5pPr (§); lvl6pPr (§); lvl7pPr (§); lvl8pPr (§); lvl9pPr (§); pPr (§) |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextBulletColorFollowText"/>

##### buFont (Specified)

This element specifies the font to be used on bullet characters within a given paragraph. The font is specified using the typeface that it is registered as within the generating application.

[Example: Consider the DrawingML shown below.

<p:txBody>  
 …  
 <a:p>  
 <a:pPr …>  
 <a:buFont typeface="Arial"/>  
 <a:buChar char="g"/>  
 </a:pPr>  
 …  
 <a:t>Bullet 1</a:t>  
 …  
 </a:p>  
 …  
</p:txBody>

The font of the above bullet will not follow the text font but instead will have Arial font specified by typeface="Arial". This font should only apply to the actual bullet character and not to the text within the bullet. [end](end.docx) example]

|  |
| --- |
| Parent Elements |
| defPPr (§); lvl1pPr (§); lvl2pPr (§); lvl3pPr (§); lvl4pPr (§); lvl5pPr (§); lvl6pPr (§); lvl7pPr (§); lvl8pPr (§); lvl9pPr (§); pPr (§) |

|  |  |
| --- | --- |
| Attributes | Description |
| charset (Similar Character Set) | Specifies the most similar character set to the one being used. This is useful if the generating application cannot use the current font and must [choose](choose.docx) a similar one.  The possible values for this attribute are defined by the XML Schema byte datatype. |
| panose (Panose Setting) | Specifies the panose standard setting that will be used to determine the closest matching font by any generating application that employs this method.  The possible values for this attribute are defined by the [ST\_Panose](ST_Panose.docx) simple type (§). |
| pitchFamily (Similar Font Family) | Specifies the most similar font family to the one being used. This is useful if the generating application cannot use the current font and must [choose](choose.docx) a similar one.  The possible values for this attribute are defined by the XML Schema byte datatype. |
| typeface (Text Typeface) | Specifies the typeface, or name of the font that is to be used for a bullet. The typeface used here should be selected from the font list of the generating application.  The possible values for this attribute are defined by the [ST\_TextTypeface](ST_TextTypeface.docx) simple type (§). |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextFont">

<attribute [name](name.docx)="typeface" type="[ST\_TextTypeface](ST_TextTypeface.docx)"/>

<attribute [name](name.docx)="panose" type="[ST\_Panose](ST_Panose.docx)" use="optional"/>

<attribute name="pitchFamily" type="xsd:byte" use="optional" default="0"/>

<attribute name="charset" type="xsd:byte" use="optional" default="1"/>

</complexType>

##### buFontTx (Follow text)

This element specifies that the font of the bullets for a paragraph should be of the same font as the text run within which each bullet is contained.

[Example: Consider the DrawingML shown below.

<p:txBody>  
 …  
 <a:p>  
 <a:pPr …>  
 <a:buFontTx>  
 </a:pPr>  
 …  
 <a:t>Bullet 1</a:t>  
 …  
 </a:p>  
 …  
</p:txBody>

The font of the above bullet will follow the default text font of the text for the run of text shown above since no specific text font was specified. [end](end.docx) example]

|  |
| --- |
| Parent Elements |
| defPPr (§); lvl1pPr (§); lvl2pPr (§); lvl3pPr (§); lvl4pPr (§); lvl5pPr (§); lvl6pPr (§); lvl7pPr (§); lvl8pPr (§); lvl9pPr (§); pPr (§) |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextBulletTypefaceFollowText"/>

##### buNone (No Bullet)

This element specifies that the paragraph within which it is applied is to have no bullet formatting applied to it. That is to say that there should be no bulleting found within the paragraph where this element is specified.

[Example: Consider the DrawingML shown below.

<p:txBody>  
 …  
 <a:p>  
 <a:pPr …>  
 <a:buNone/>  
 </a:pPr>  
 …  
 <a:t>Bullet 1</a:t>  
 …  
 </a:p>  
 …  
</p:txBody>

The above paragraph will be formatted with no bullets. [end](end.docx) example]

|  |
| --- |
| Parent Elements |
| defPPr (§); lvl1pPr (§); lvl2pPr (§); lvl3pPr (§); lvl4pPr (§); lvl5pPr (§); lvl6pPr (§); lvl7pPr (§); lvl8pPr (§); lvl9pPr (§); pPr (§) |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextNoBullet"/>

##### buSzPct (Bullet Size Percentage)

This element specifies the size in percentage of the surrounding text to be used on bullet characters within a given paragraph. The size is specified using a percentage where 1000 is equal to 1 percent of the font size and 100000 is equal to 100 percent font of the font size.

[Example: Consider the DrawingML shown below.

<p:txBody>  
 …  
 <a:p>  
 <a:pPr …>  
 <a:buSzPct [val](val.docx)="111000"/>  
 </a:pPr>  
 …  
 <a:t>Bullet 1</a:t>  
 …  
 </a:p>  
 …  
</p:txBody>

The size of the above bullet will follow the text size in that it will always be rendered at 111% the size of the text within the given text run. This is specified by [val](val.docx)="111000", with a restriction on the values not being less than 25% or more than 400%. A value of 100000 is equal to 100%, similarly a value of 1000 is equal to 1%. This percentage size should only apply to the actual bullet character and not to the text within the bullet. [end](end.docx) example]

|  |
| --- |
| Parent Elements |
| defPPr (§); lvl1pPr (§); lvl2pPr (§); lvl3pPr (§); lvl4pPr (§); lvl5pPr (§); lvl6pPr (§); lvl7pPr (§); lvl8pPr (§); lvl9pPr (§); pPr (§) |

|  |  |
| --- | --- |
| Attributes | Description |
| [val](val.docx) (Value) | Specifies the percentage of the text size that this bullet should be. It is specified here in terms of 100% being equal to 100000 and 1% being specified in increments of 1000. This attribute should not be lower than 25%, or 25000 and not be higher than 400%, or 400000.  The possible values for this attribute are defined by the [ST\_TextBulletSizePercent](ST_TextBulletSizePercent.docx) simple type (§). |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextBulletSizePercent">

<attribute [name](name.docx)="[val](val.docx)" type="[ST\_TextBulletSizePercent](ST_TextBulletSizePercent.docx)"/>

</complexType>

##### buSzPts (Bullet Size Points)

This element specifies the size in points to be used on bullet characters within a given paragraph. The size is specified using the points where 100 is equal to 1 point font and 1200 is equal to 12 point font.

[Example: Consider the DrawingML shown below.

<p:txBody>  
 …  
 <a:p>  
 <a:pPr …>  
 <a:buSzPts [val](val.docx)="1400"/>  
 </a:pPr>  
 …  
 <a:t>Bullet 1</a:t>  
 …  
 </a:p>  
 …  
</p:txBody>

The size of the above bullet will not follow the text size of the text within the given text run. The bullets size is specified by [val](val.docx)="1400", which corresponds to a point size of 14. This bullet size should only apply to the actual bullet character and not to the text within the bullet. [end](end.docx) example]

|  |
| --- |
| Parent Elements |
| defPPr (§); lvl1pPr (§); lvl2pPr (§); lvl3pPr (§); lvl4pPr (§); lvl5pPr (§); lvl6pPr (§); lvl7pPr (§); lvl8pPr (§); lvl9pPr (§); pPr (§) |

|  |  |
| --- | --- |
| Attributes | Description |
| [val](val.docx) (Value) | Specifies the size of the bullets in point size. Whole points are specified in increments of 100 starting with 100 being a point size of 1. For instance a font point size of 12 would be 1200 and a font point size of 12.5 would be 1250.  The possible values for this attribute are defined by the [ST\_TextFontSize](ST_TextFontSize.docx) simple type (§). |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextBulletSizePoint">

<attribute [name](name.docx)="[val](val.docx)" type="[ST\_TextFontSize](ST_TextFontSize.docx)"/>

</complexType>

##### buSzTx (Bullet Size Follows Text)

This element specifies that the size of the bullets for a paragraph should be of the same point size as the text run within which each bullet is contained.

[Example: Consider the DrawingML shown below.

<p:txBody>  
 …  
 <a:p>  
 <a:pPr …>  
 <a:buSzTx>  
 </a:pPr>  
 …  
 <a:t>Bullet 1</a:t>  
 …  
 </a:p>  
 …  
</p:txBody>

The size of the above bullet will follow the default text size of the text for the run of text shown above since no specific text size was specified. [end](end.docx) example]

|  |
| --- |
| Parent Elements |
| defPPr (§); lvl1pPr (§); lvl2pPr (§); lvl3pPr (§); lvl4pPr (§); lvl5pPr (§); lvl6pPr (§); lvl7pPr (§); lvl8pPr (§); lvl9pPr (§); pPr (§) |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextBulletSizeFollowText"/>

##### lstStyle (Text List Styles)

This element specifies the list of styles associated with this body of text.

|  |
| --- |
| Parent Elements |
| lnDef (§); [rich](rich.docx) (§); spDef (§); [t](t.docx) (§); [txBody](txBody.docx) (§); [txBody](txBody.docx) (§); [txBody](txBody.docx) (§); [txBody](txBody.docx) (§); txDef (§); [txPr](txPr.docx) (§) |

|  |  |
| --- | --- |
| Child Elements | Subclause |
| defPPr (Default Paragraph Style) | § |
| [extLst](extLst.docx) (Extension List) | § |
| lvl1pPr (List Level 1 Text Style) | § |
| lvl2pPr (List Level 2 Text Style) | § |
| lvl3pPr (List Level 3 Text Style) | § |
| lvl4pPr (List Level 4 Text Style) | § |
| lvl5pPr (List Level 5 Text Style) | § |
| lvl6pPr (List Level 6 Text Style) | § |
| lvl7pPr (List Level 7 Text Style) | § |
| lvl8pPr (List Level 8 Text Style) | § |
| lvl9pPr (List Level 9 Text Style) | § |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextListStyle">

<sequence>

<element name="defPPr" type="CT\_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>

<element name="lvl1pPr" type="CT\_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>

<element name="lvl2pPr" type="CT\_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>

<element name="lvl3pPr" type="CT\_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>

<element name="lvl4pPr" type="CT\_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>

<element name="lvl5pPr" type="CT\_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>

<element name="lvl6pPr" type="CT\_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>

<element name="lvl7pPr" type="CT\_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>

<element name="lvl8pPr" type="CT\_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>

<element name="lvl9pPr" type="CT\_TextParagraphProperties" minOccurs="0" maxOccurs="1"/>

<element name="[extLst](extLst.docx)" type="CT\_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>

</sequence>

</complexType>

##### lvl1pPr (List Level 1 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute [lvl](lvl.docx)="0". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the [order](order.docx) in which this and other level property elements are specified be in [order](order.docx) of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[Example: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl1pPr and thus create a paragraph of text that has no bullets and is right aligned.

<p:txBody>  
 …  
 <a:lstStyle>  
 <a:lvl1pPr algn="r">  
 <a:buNone/>  
 </a:lvl1pPr>  
 </a:lstStyle>  
 <a:p>  
 <a:pPr [lvl](lvl.docx)="0">  
 </a:pPr>  
 …  
 <a:t>Some text</a:t>  
 …  
 </a:p>  
</p:txBody>

[end](end.docx) example]

[Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]

|  |
| --- |
| Parent Elements |
| bodyStyle (§); defaultTextStyle (§); lstStyle (§); notesStyle (§); otherStyle (§); titleStyle (§) |

|  |  |
| --- | --- |
| Child Elements | Subclause |
| buAutoNum (Auto-Numbered Bullet) | § |
| buBlip (Picture Bullet) | § |
| buChar (Character Bullet) | § |
| buClr (Color Specified) | § |
| buClrTx (Follow Text) | § |
| buFont (Specified) | § |
| buFontTx (Follow text) | § |
| buNone (No Bullet) | § |
| buSzPct (Bullet Size Percentage) | § |
| buSzPts (Bullet Size Points) | § |
| buSzTx (Bullet Size Follows Text) | § |
| defRPr (Default Text Run Properties) | § |
| [extLst](extLst.docx) (Extension List) | § |
| lnSpc (Line Spacing) | § |
| spcAft (Space After) | § |
| spcBef (Space Before) | § |
| tabLst (Tab List) | § |

|  |  |
| --- | --- |
| Attributes | Description |
| algn (Alignment) | Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied.    [Example: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextAlignType](ST_TextAlignType.docx) simple type (§). |
| defTabSz (Default Tab Size) | Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.  [Example: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.  <p:txBody>  …  <a:p>  <a:pPr defTabSz="376300" …/>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_Coordinate32](ST_Coordinate32.docx) simple type (§). |
| eaLnBrk (East Asian Line Break) | Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr eaLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>      [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| fontAlgn (Font Alignment) | Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.  [Example: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.    <a:txtBody>  …  <a:pPr fontAlgn="b" …/>  …  <a:r>  <a:rPr …/>  <a:t>H </a:t>  </a:r>  <a:r>  <a:rPr sz="1200" …/>  <a:t>2</a:t>  </a:r>  <a:r>  <a:rPr …/>  <a:t>O</a:t>  </a:r>  … </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextFontAlignType](ST_TextFontAlignType.docx) simple type (§). |
| hangingPunct (Hanging Punctuation) | Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the [end](end.docx) of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied.  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| indent (Indent) | Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.    [Example: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" indent="571500" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Here is some…</a:t>  …  </a:p> </p:txBody>  By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextIndent](ST_TextIndent.docx) simple type (§). |
| latinLnBrk (Latin Line Break) | Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr latinLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| [lvl](lvl.docx) (Level) | Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.  [Example: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again [lvl](lvl.docx)="1" is considered to be level 2.  <p:txBody>  …  <a:p>  <a:pPr [lvl](lvl.docx)="1" …/>  …  <a:t>Sample text</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  [Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]  The possible values for this attribute are defined by the [ST\_TextIndentLevelType](ST_TextIndentLevelType.docx) simple type (§). |
| marL (Left Margin) | Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| marR (Right Margin) | Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| rtl (Right To Left) | Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.  [Example: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.  <p:txBody>  …  <a:p>  <a:pPr rtl="1" …/>  …  <a:t>Sample text…</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextParagraphProperties">

<sequence>

<element name="lnSpc" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcBef" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcAft" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletColor" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletSize" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBullet" minOccurs="0" maxOccurs="1"/>

<element name="tabLst" type="CT\_TextTabStopList" minOccurs="0" maxOccurs="1"/>

<element name="defRPr" type="CT\_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>

<element name="[extLst](extLst.docx)" type="CT\_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>

</sequence>

<attribute [name](name.docx)="marL" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="marR" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="[lvl](lvl.docx)" type="[ST\_TextIndentLevelType](ST_TextIndentLevelType.docx)" use="optional"/>

<attribute [name](name.docx)="indent" type="[ST\_TextIndent](ST_TextIndent.docx)" use="optional"/>

<attribute [name](name.docx)="algn" type="[ST\_TextAlignType](ST_TextAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="defTabSz" type="[ST\_Coordinate32](ST_Coordinate32.docx)" use="optional"/>

<attribute [name](name.docx)="rtl" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="eaLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="fontAlgn" type="[ST\_TextFontAlignType](ST_TextFontAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="latinLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="hangingPunct" type="xsd:boolean" use="optional"/>

</complexType>

##### lvl2pPr (List Level 2 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute [lvl](lvl.docx)="1". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the [order](order.docx) in which this and other level property elements are specified be in [order](order.docx) of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[Example: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl2pPr and thus create a paragraph of text that has no bullets and is right aligned.

<p:txBody>  
 …  
 <a:lstStyle>  
 <a:lvl2pPr algn="r">  
 <a:buNone/>  
 </a:lvl2pPr>  
 </a:lstStyle>

<a:p>  
 <a:pPr [lvl](lvl.docx)="1">  
 </a:pPr>  
 …  
 <a:t>Some text</a:t>  
 …  
 </a:p>  
</p:txBody>

[end](end.docx) example]

[Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]

|  |
| --- |
| Parent Elements |
| bodyStyle (§); defaultTextStyle (§); lstStyle (§); notesStyle (§); otherStyle (§); titleStyle (§) |

|  |  |
| --- | --- |
| Child Elements | Subclause |
| buAutoNum (Auto-Numbered Bullet) | § |
| buBlip (Picture Bullet) | § |
| buChar (Character Bullet) | § |
| buClr (Color Specified) | § |
| buClrTx (Follow Text) | § |
| buFont (Specified) | § |
| buFontTx (Follow text) | § |
| buNone (No Bullet) | § |
| buSzPct (Bullet Size Percentage) | § |
| buSzPts (Bullet Size Points) | § |
| buSzTx (Bullet Size Follows Text) | § |
| defRPr (Default Text Run Properties) | § |
| [extLst](extLst.docx) (Extension List) | § |
| lnSpc (Line Spacing) | § |
| spcAft (Space After) | § |
| spcBef (Space Before) | § |
| tabLst (Tab List) | § |

|  |  |
| --- | --- |
| Attributes | Description |
| algn (Alignment) | Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied.    [Example: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextAlignType](ST_TextAlignType.docx) simple type (§). |
| defTabSz (Default Tab Size) | Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.  [Example: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.  <p:txBody>  …  <a:p>  <a:pPr defTabSz="376300" …/>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_Coordinate32](ST_Coordinate32.docx) simple type (§). |
| eaLnBrk (East Asian Line Break) | Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr eaLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>      [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| fontAlgn (Font Alignment) | Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.  [Example: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.    <a:txtBody>  …  <a:pPr fontAlgn="b" …/>  …  <a:r>  <a:rPr …/>  <a:t>H </a:t>  </a:r>  <a:r>  <a:rPr sz="1200" …/>  <a:t>2</a:t>  </a:r>  <a:r>  <a:rPr …/>  <a:t>O</a:t>  </a:r>  … </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextFontAlignType](ST_TextFontAlignType.docx) simple type (§). |
| hangingPunct (Hanging Punctuation) | Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the [end](end.docx) of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied.  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| indent (Indent) | Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.    [Example: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" indent="571500" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Here is some…</a:t>  …  </a:p> </p:txBody>  By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextIndent](ST_TextIndent.docx) simple type (§). |
| latinLnBrk (Latin Line Break) | Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr latinLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| [lvl](lvl.docx) (Level) | Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.  [Example: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again [lvl](lvl.docx)="1" is considered to be level 2.  <p:txBody>  …  <a:p>  <a:pPr [lvl](lvl.docx)="1" …/>  …  <a:t>Sample text</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  [Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]  The possible values for this attribute are defined by the [ST\_TextIndentLevelType](ST_TextIndentLevelType.docx) simple type (§). |
| marL (Left Margin) | Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| marR (Right Margin) | Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| rtl (Right To Left) | Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.  [Example: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.  <p:txBody>  …  <a:p>  <a:pPr rtl="1" …/>  …  <a:t>Sample text…</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextParagraphProperties">

<sequence>

<element name="lnSpc" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcBef" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcAft" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletColor" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletSize" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBullet" minOccurs="0" maxOccurs="1"/>

<element name="tabLst" type="CT\_TextTabStopList" minOccurs="0" maxOccurs="1"/>

<element name="defRPr" type="CT\_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>

<element name="[extLst](extLst.docx)" type="CT\_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>

</sequence>

<attribute [name](name.docx)="marL" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="marR" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="[lvl](lvl.docx)" type="[ST\_TextIndentLevelType](ST_TextIndentLevelType.docx)" use="optional"/>

<attribute [name](name.docx)="indent" type="[ST\_TextIndent](ST_TextIndent.docx)" use="optional"/>

<attribute [name](name.docx)="algn" type="[ST\_TextAlignType](ST_TextAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="defTabSz" type="[ST\_Coordinate32](ST_Coordinate32.docx)" use="optional"/>

<attribute [name](name.docx)="rtl" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="eaLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="fontAlgn" type="[ST\_TextFontAlignType](ST_TextFontAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="latinLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="hangingPunct" type="xsd:boolean" use="optional"/>

</complexType>

##### lvl3pPr (List Level 3 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute [lvl](lvl.docx)="2". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the [order](order.docx) in which this and other level property elements are specified be in [order](order.docx) of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[Example: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl3pPr and thus create a paragraph of text that has no bullets and is right aligned.

<p:txBody>  
 …  
 <a:lstStyle>  
 <a:lvl3pPr algn="r">  
 <a:buNone/>  
 </a:lvl3pPr>  
 </a:lstStyle>

<a:p>  
 <a:pPr [lvl](lvl.docx)="2">  
 </a:pPr>  
 …  
 <a:t>Some text</a:t>  
 …  
 </a:p>  
</p:txBody>

[end](end.docx) example]

[Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]

|  |
| --- |
| Parent Elements |
| bodyStyle (§); defaultTextStyle (§); lstStyle (§); notesStyle (§); otherStyle (§); titleStyle (§) |

|  |  |
| --- | --- |
| Child Elements | Subclause |
| buAutoNum (Auto-Numbered Bullet) | § |
| buBlip (Picture Bullet) | § |
| buChar (Character Bullet) | § |
| buClr (Color Specified) | § |
| buClrTx (Follow Text) | § |
| buFont (Specified) | § |
| buFontTx (Follow text) | § |
| buNone (No Bullet) | § |
| buSzPct (Bullet Size Percentage) | § |
| buSzPts (Bullet Size Points) | § |
| buSzTx (Bullet Size Follows Text) | § |
| defRPr (Default Text Run Properties) | § |
| [extLst](extLst.docx) (Extension List) | § |
| lnSpc (Line Spacing) | § |
| spcAft (Space After) | § |
| spcBef (Space Before) | § |
| tabLst (Tab List) | § |

|  |  |
| --- | --- |
| Attributes | Description |
| algn (Alignment) | Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied.    [Example: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextAlignType](ST_TextAlignType.docx) simple type (§). |
| defTabSz (Default Tab Size) | Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.  [Example: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.  <p:txBody>  …  <a:p>  <a:pPr defTabSz="376300" …/>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_Coordinate32](ST_Coordinate32.docx) simple type (§). |
| eaLnBrk (East Asian Line Break) | Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr eaLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>      [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| fontAlgn (Font Alignment) | Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.  [Example: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.    <a:txtBody>  …  <a:pPr fontAlgn="b" …/>  …  <a:r>  <a:rPr …/>  <a:t>H </a:t>  </a:r>  <a:r>  <a:rPr sz="1200" …/>  <a:t>2</a:t>  </a:r>  <a:r>  <a:rPr …/>  <a:t>O</a:t>  </a:r>  … </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextFontAlignType](ST_TextFontAlignType.docx) simple type (§). |
| hangingPunct (Hanging Punctuation) | Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the [end](end.docx) of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied.  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| indent (Indent) | Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.    [Example: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" indent="571500" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Here is some…</a:t>  …  </a:p> </p:txBody>  By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextIndent](ST_TextIndent.docx) simple type (§). |
| latinLnBrk (Latin Line Break) | Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr latinLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| [lvl](lvl.docx) (Level) | Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.  [Example: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again [lvl](lvl.docx)="1" is considered to be level 2.  <p:txBody>  …  <a:p>  <a:pPr [lvl](lvl.docx)="1" …/>  …  <a:t>Sample text</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  [Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]  The possible values for this attribute are defined by the [ST\_TextIndentLevelType](ST_TextIndentLevelType.docx) simple type (§). |
| marL (Left Margin) | Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| marR (Right Margin) | Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| rtl (Right To Left) | Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.  [Example: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.  <p:txBody>  …  <a:p>  <a:pPr rtl="1" …/>  …  <a:t>Sample text…</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextParagraphProperties">

<sequence>

<element name="lnSpc" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcBef" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcAft" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletColor" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletSize" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBullet" minOccurs="0" maxOccurs="1"/>

<element name="tabLst" type="CT\_TextTabStopList" minOccurs="0" maxOccurs="1"/>

<element name="defRPr" type="CT\_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>

<element name="[extLst](extLst.docx)" type="CT\_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>

</sequence>

<attribute [name](name.docx)="marL" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="marR" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="[lvl](lvl.docx)" type="[ST\_TextIndentLevelType](ST_TextIndentLevelType.docx)" use="optional"/>

<attribute [name](name.docx)="indent" type="[ST\_TextIndent](ST_TextIndent.docx)" use="optional"/>

<attribute [name](name.docx)="algn" type="[ST\_TextAlignType](ST_TextAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="defTabSz" type="[ST\_Coordinate32](ST_Coordinate32.docx)" use="optional"/>

<attribute [name](name.docx)="rtl" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="eaLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="fontAlgn" type="[ST\_TextFontAlignType](ST_TextFontAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="latinLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="hangingPunct" type="xsd:boolean" use="optional"/>

</complexType>

##### lvl4pPr (List Level 4 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute [lvl](lvl.docx)="3". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the [order](order.docx) in which this and other level property elements are specified be in [order](order.docx) of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[Example: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl4pPr and thus create a paragraph of text that has no bullets and is right aligned.

<p:txBody>  
 …  
 <a:lstStyle>  
 <a:lvl4pPr algn="r">  
 <a:buNone/>  
 </a:lvl4pPr>  
 </a:lstStyle>

<a:p>  
 <a:pPr [lvl](lvl.docx)="3">  
 </a:pPr>  
 …  
 <a:t>Some text</a:t>  
 …  
 </a:p>  
</p:txBody>

[end](end.docx) example]

[Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]

|  |
| --- |
| Parent Elements |
| bodyStyle (§); defaultTextStyle (§); lstStyle (§); notesStyle (§); otherStyle (§); titleStyle (§) |

|  |  |
| --- | --- |
| Child Elements | Subclause |
| buAutoNum (Auto-Numbered Bullet) | § |
| buBlip (Picture Bullet) | § |
| buChar (Character Bullet) | § |
| buClr (Color Specified) | § |
| buClrTx (Follow Text) | § |
| buFont (Specified) | § |
| buFontTx (Follow text) | § |
| buNone (No Bullet) | § |
| buSzPct (Bullet Size Percentage) | § |
| buSzPts (Bullet Size Points) | § |
| buSzTx (Bullet Size Follows Text) | § |
| defRPr (Default Text Run Properties) | § |
| [extLst](extLst.docx) (Extension List) | § |
| lnSpc (Line Spacing) | § |
| spcAft (Space After) | § |
| spcBef (Space Before) | § |
| tabLst (Tab List) | § |

|  |  |
| --- | --- |
| Attributes | Description |
| algn (Alignment) | Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied.    [Example: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextAlignType](ST_TextAlignType.docx) simple type (§). |
| defTabSz (Default Tab Size) | Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.  [Example: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.  <p:txBody>  …  <a:p>  <a:pPr defTabSz="376300" …/>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_Coordinate32](ST_Coordinate32.docx) simple type (§). |
| eaLnBrk (East Asian Line Break) | Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr eaLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>      [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| fontAlgn (Font Alignment) | Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.  [Example: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.    <a:txtBody>  …  <a:pPr fontAlgn="b" …/>  …  <a:r>  <a:rPr …/>  <a:t>H </a:t>  </a:r>  <a:r>  <a:rPr sz="1200" …/>  <a:t>2</a:t>  </a:r>  <a:r>  <a:rPr …/>  <a:t>O</a:t>  </a:r>  … </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextFontAlignType](ST_TextFontAlignType.docx) simple type (§). |
| hangingPunct (Hanging Punctuation) | Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the [end](end.docx) of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied.  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| indent (Indent) | Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.    [Example: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" indent="571500" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Here is some…</a:t>  …  </a:p> </p:txBody>  By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextIndent](ST_TextIndent.docx) simple type (§). |
| latinLnBrk (Latin Line Break) | Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr latinLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| [lvl](lvl.docx) (Level) | Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.  [Example: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again [lvl](lvl.docx)="1" is considered to be level 2.  <p:txBody>  …  <a:p>  <a:pPr [lvl](lvl.docx)="1" …/>  …  <a:t>Sample text</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  [Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]  The possible values for this attribute are defined by the [ST\_TextIndentLevelType](ST_TextIndentLevelType.docx) simple type (§). |
| marL (Left Margin) | Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| marR (Right Margin) | Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| rtl (Right To Left) | Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.  [Example: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.  <p:txBody>  …  <a:p>  <a:pPr rtl="1" …/>  …  <a:t>Sample text…</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextParagraphProperties">

<sequence>

<element name="lnSpc" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcBef" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcAft" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletColor" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletSize" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBullet" minOccurs="0" maxOccurs="1"/>

<element name="tabLst" type="CT\_TextTabStopList" minOccurs="0" maxOccurs="1"/>

<element name="defRPr" type="CT\_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>

<element name="[extLst](extLst.docx)" type="CT\_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>

</sequence>

<attribute [name](name.docx)="marL" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="marR" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="[lvl](lvl.docx)" type="[ST\_TextIndentLevelType](ST_TextIndentLevelType.docx)" use="optional"/>

<attribute [name](name.docx)="indent" type="[ST\_TextIndent](ST_TextIndent.docx)" use="optional"/>

<attribute [name](name.docx)="algn" type="[ST\_TextAlignType](ST_TextAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="defTabSz" type="[ST\_Coordinate32](ST_Coordinate32.docx)" use="optional"/>

<attribute [name](name.docx)="rtl" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="eaLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="fontAlgn" type="[ST\_TextFontAlignType](ST_TextFontAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="latinLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="hangingPunct" type="xsd:boolean" use="optional"/>

</complexType>

##### lvl5pPr (List Level 5 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute [lvl](lvl.docx)="4". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the [order](order.docx) in which this and other level property elements are specified be in [order](order.docx) of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[Example: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl5pPr and thus create a paragraph of text that has no bullets and is right aligned.

<p:txBody>  
 …  
 <a:lstStyle>  
 <a:lvl5pPr algn="r">  
 <a:buNone/>  
 </a:lvl5pPr>  
 </a:lstStyle>

<a:p>  
 <a:pPr [lvl](lvl.docx)="4">  
 </a:pPr>  
 …  
 <a:t>Some text</a:t>  
 …  
 </a:p>  
</p:txBody>

[end](end.docx) example]

[Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]

|  |
| --- |
| Parent Elements |
| bodyStyle (§); defaultTextStyle (§); lstStyle (§); notesStyle (§); otherStyle (§); titleStyle (§) |

|  |  |
| --- | --- |
| Child Elements | Subclause |
| buAutoNum (Auto-Numbered Bullet) | § |
| buBlip (Picture Bullet) | § |
| buChar (Character Bullet) | § |
| buClr (Color Specified) | § |
| buClrTx (Follow Text) | § |
| buFont (Specified) | § |
| buFontTx (Follow text) | § |
| buNone (No Bullet) | § |
| buSzPct (Bullet Size Percentage) | § |
| buSzPts (Bullet Size Points) | § |
| buSzTx (Bullet Size Follows Text) | § |
| defRPr (Default Text Run Properties) | § |
| [extLst](extLst.docx) (Extension List) | § |
| lnSpc (Line Spacing) | § |
| spcAft (Space After) | § |
| spcBef (Space Before) | § |
| tabLst (Tab List) | § |

|  |  |
| --- | --- |
| Attributes | Description |
| algn (Alignment) | Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied.    [Example: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextAlignType](ST_TextAlignType.docx) simple type (§). |
| defTabSz (Default Tab Size) | Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.  [Example: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.  <p:txBody>  …  <a:p>  <a:pPr defTabSz="376300" …/>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_Coordinate32](ST_Coordinate32.docx) simple type (§). |
| eaLnBrk (East Asian Line Break) | Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr eaLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>      [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| fontAlgn (Font Alignment) | Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.  [Example: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.    <a:txtBody>  …  <a:pPr fontAlgn="b" …/>  …  <a:r>  <a:rPr …/>  <a:t>H </a:t>  </a:r>  <a:r>  <a:rPr sz="1200" …/>  <a:t>2</a:t>  </a:r>  <a:r>  <a:rPr …/>  <a:t>O</a:t>  </a:r>  … </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextFontAlignType](ST_TextFontAlignType.docx) simple type (§). |
| hangingPunct (Hanging Punctuation) | Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the [end](end.docx) of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied.  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| indent (Indent) | Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.    [Example: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" indent="571500" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Here is some…</a:t>  …  </a:p> </p:txBody>  By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextIndent](ST_TextIndent.docx) simple type (§). |
| latinLnBrk (Latin Line Break) | Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr latinLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| [lvl](lvl.docx) (Level) | Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.  [Example: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again [lvl](lvl.docx)="1" is considered to be level 2.  <p:txBody>  …  <a:p>  <a:pPr [lvl](lvl.docx)="1" …/>  …  <a:t>Sample text</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  [Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]  The possible values for this attribute are defined by the [ST\_TextIndentLevelType](ST_TextIndentLevelType.docx) simple type (§). |
| marL (Left Margin) | Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| marR (Right Margin) | Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| rtl (Right To Left) | Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.  [Example: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.  <p:txBody>  …  <a:p>  <a:pPr rtl="1" …/>  …  <a:t>Sample text…</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextParagraphProperties">

<sequence>

<element name="lnSpc" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcBef" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcAft" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletColor" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletSize" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBullet" minOccurs="0" maxOccurs="1"/>

<element name="tabLst" type="CT\_TextTabStopList" minOccurs="0" maxOccurs="1"/>

<element name="defRPr" type="CT\_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>

<element name="[extLst](extLst.docx)" type="CT\_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>

</sequence>

<attribute [name](name.docx)="marL" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="marR" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="[lvl](lvl.docx)" type="[ST\_TextIndentLevelType](ST_TextIndentLevelType.docx)" use="optional"/>

<attribute [name](name.docx)="indent" type="[ST\_TextIndent](ST_TextIndent.docx)" use="optional"/>

<attribute [name](name.docx)="algn" type="[ST\_TextAlignType](ST_TextAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="defTabSz" type="[ST\_Coordinate32](ST_Coordinate32.docx)" use="optional"/>

<attribute [name](name.docx)="rtl" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="eaLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="fontAlgn" type="[ST\_TextFontAlignType](ST_TextFontAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="latinLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="hangingPunct" type="xsd:boolean" use="optional"/>

</complexType>

##### lvl6pPr (List Level 6 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute [lvl](lvl.docx)="5". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the [order](order.docx) in which this and other level property elements are specified be in [order](order.docx) of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[Example: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl6pPr and thus create a paragraph of text that has no bullets and is right aligned.

<p:txBody>  
 …  
 <a:lstStyle>  
 <a:lvl6pPr algn="r">  
 <a:buNone/>  
 </a:lvl6pPr>  
 </a:lstStyle>

<a:p>  
 <a:pPr [lvl](lvl.docx)="5">  
 </a:pPr>  
 …  
 <a:t>Some text</a:t>  
 …  
 </a:p>  
</p:txBody>

[end](end.docx) example]

[Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]

|  |
| --- |
| Parent Elements |
| bodyStyle (§); defaultTextStyle (§); lstStyle (§); notesStyle (§); otherStyle (§); titleStyle (§) |

|  |  |
| --- | --- |
| Child Elements | Subclause |
| buAutoNum (Auto-Numbered Bullet) | § |
| buBlip (Picture Bullet) | § |
| buChar (Character Bullet) | § |
| buClr (Color Specified) | § |
| buClrTx (Follow Text) | § |
| buFont (Specified) | § |
| buFontTx (Follow text) | § |
| buNone (No Bullet) | § |
| buSzPct (Bullet Size Percentage) | § |
| buSzPts (Bullet Size Points) | § |
| buSzTx (Bullet Size Follows Text) | § |
| defRPr (Default Text Run Properties) | § |
| [extLst](extLst.docx) (Extension List) | § |
| lnSpc (Line Spacing) | § |
| spcAft (Space After) | § |
| spcBef (Space Before) | § |
| tabLst (Tab List) | § |

|  |  |
| --- | --- |
| Attributes | Description |
| algn (Alignment) | Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied.    [Example: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextAlignType](ST_TextAlignType.docx) simple type (§). |
| defTabSz (Default Tab Size) | Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.  [Example: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.  <p:txBody>  …  <a:p>  <a:pPr defTabSz="376300" …/>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_Coordinate32](ST_Coordinate32.docx) simple type (§). |
| eaLnBrk (East Asian Line Break) | Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr eaLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>      [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| fontAlgn (Font Alignment) | Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.  [Example: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.    <a:txtBody>  …  <a:pPr fontAlgn="b" …/>  …  <a:r>  <a:rPr …/>  <a:t>H </a:t>  </a:r>  <a:r>  <a:rPr sz="1200" …/>  <a:t>2</a:t>  </a:r>  <a:r>  <a:rPr …/>  <a:t>O</a:t>  </a:r>  … </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextFontAlignType](ST_TextFontAlignType.docx) simple type (§). |
| hangingPunct (Hanging Punctuation) | Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the [end](end.docx) of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied.  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| indent (Indent) | Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.    [Example: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" indent="571500" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Here is some…</a:t>  …  </a:p> </p:txBody>  By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextIndent](ST_TextIndent.docx) simple type (§). |
| latinLnBrk (Latin Line Break) | Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr latinLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| [lvl](lvl.docx) (Level) | Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.  [Example: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again [lvl](lvl.docx)="1" is considered to be level 2.  <p:txBody>  …  <a:p>  <a:pPr [lvl](lvl.docx)="1" …/>  …  <a:t>Sample text</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  [Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]  The possible values for this attribute are defined by the [ST\_TextIndentLevelType](ST_TextIndentLevelType.docx) simple type (§). |
| marL (Left Margin) | Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| marR (Right Margin) | Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| rtl (Right To Left) | Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.  [Example: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.  <p:txBody>  …  <a:p>  <a:pPr rtl="1" …/>  …  <a:t>Sample text…</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextParagraphProperties">

<sequence>

<element name="lnSpc" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcBef" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcAft" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletColor" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletSize" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBullet" minOccurs="0" maxOccurs="1"/>

<element name="tabLst" type="CT\_TextTabStopList" minOccurs="0" maxOccurs="1"/>

<element name="defRPr" type="CT\_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>

<element name="[extLst](extLst.docx)" type="CT\_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>

</sequence>

<attribute [name](name.docx)="marL" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="marR" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="[lvl](lvl.docx)" type="[ST\_TextIndentLevelType](ST_TextIndentLevelType.docx)" use="optional"/>

<attribute [name](name.docx)="indent" type="[ST\_TextIndent](ST_TextIndent.docx)" use="optional"/>

<attribute [name](name.docx)="algn" type="[ST\_TextAlignType](ST_TextAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="defTabSz" type="[ST\_Coordinate32](ST_Coordinate32.docx)" use="optional"/>

<attribute [name](name.docx)="rtl" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="eaLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="fontAlgn" type="[ST\_TextFontAlignType](ST_TextFontAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="latinLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="hangingPunct" type="xsd:boolean" use="optional"/>

</complexType>

##### lvl7pPr (List Level 7 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute [lvl](lvl.docx)="6". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the [order](order.docx) in which this and other level property elements are specified be in [order](order.docx) of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[Example: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl7pPr and thus create a paragraph of text that has no bullets and is right aligned.

<p:txBody>  
 …  
 <a:lstStyle>  
 <a:lvl7pPr algn="r">  
 <a:buNone/>  
 </a:lvl7pPr>  
 </a:lstStyle>

<a:p>  
 <a:pPr [lvl](lvl.docx)="6">  
 </a:pPr>  
 …  
 <a:t>Some text</a:t>  
 …  
 </a:p>  
</p:txBody>

[end](end.docx) example]

[Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]

|  |
| --- |
| Parent Elements |
| bodyStyle (§); defaultTextStyle (§); lstStyle (§); notesStyle (§); otherStyle (§); titleStyle (§) |

|  |  |
| --- | --- |
| Child Elements | Subclause |
| buAutoNum (Auto-Numbered Bullet) | § |
| buBlip (Picture Bullet) | § |
| buChar (Character Bullet) | § |
| buClr (Color Specified) | § |
| buClrTx (Follow Text) | § |
| buFont (Specified) | § |
| buFontTx (Follow text) | § |
| buNone (No Bullet) | § |
| buSzPct (Bullet Size Percentage) | § |
| buSzPts (Bullet Size Points) | § |
| buSzTx (Bullet Size Follows Text) | § |
| defRPr (Default Text Run Properties) | § |
| [extLst](extLst.docx) (Extension List) | § |
| lnSpc (Line Spacing) | § |
| spcAft (Space After) | § |
| spcBef (Space Before) | § |
| tabLst (Tab List) | § |

|  |  |
| --- | --- |
| Attributes | Description |
| algn (Alignment) | Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied.    [Example: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextAlignType](ST_TextAlignType.docx) simple type (§). |
| defTabSz (Default Tab Size) | Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.  [Example: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.  <p:txBody>  …  <a:p>  <a:pPr defTabSz="376300" …/>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_Coordinate32](ST_Coordinate32.docx) simple type (§). |
| eaLnBrk (East Asian Line Break) | Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr eaLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>      [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| fontAlgn (Font Alignment) | Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.  [Example: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.    <a:txtBody>  …  <a:pPr fontAlgn="b" …/>  …  <a:r>  <a:rPr …/>  <a:t>H </a:t>  </a:r>  <a:r>  <a:rPr sz="1200" …/>  <a:t>2</a:t>  </a:r>  <a:r>  <a:rPr …/>  <a:t>O</a:t>  </a:r>  … </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextFontAlignType](ST_TextFontAlignType.docx) simple type (§). |
| hangingPunct (Hanging Punctuation) | Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the [end](end.docx) of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied.  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| indent (Indent) | Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.    [Example: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" indent="571500" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Here is some…</a:t>  …  </a:p> </p:txBody>  By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextIndent](ST_TextIndent.docx) simple type (§). |
| latinLnBrk (Latin Line Break) | Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr latinLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| [lvl](lvl.docx) (Level) | Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.  [Example: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again [lvl](lvl.docx)="1" is considered to be level 2.  <p:txBody>  …  <a:p>  <a:pPr [lvl](lvl.docx)="1" …/>  …  <a:t>Sample text</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  [Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]  The possible values for this attribute are defined by the [ST\_TextIndentLevelType](ST_TextIndentLevelType.docx) simple type (§). |
| marL (Left Margin) | Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| marR (Right Margin) | Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| rtl (Right To Left) | Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.  [Example: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.  <p:txBody>  …  <a:p>  <a:pPr rtl="1" …/>  …  <a:t>Sample text…</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextParagraphProperties">

<sequence>

<element name="lnSpc" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcBef" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcAft" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletColor" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletSize" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBullet" minOccurs="0" maxOccurs="1"/>

<element name="tabLst" type="CT\_TextTabStopList" minOccurs="0" maxOccurs="1"/>

<element name="defRPr" type="CT\_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>

<element name="[extLst](extLst.docx)" type="CT\_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>

</sequence>

<attribute [name](name.docx)="marL" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="marR" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="[lvl](lvl.docx)" type="[ST\_TextIndentLevelType](ST_TextIndentLevelType.docx)" use="optional"/>

<attribute [name](name.docx)="indent" type="[ST\_TextIndent](ST_TextIndent.docx)" use="optional"/>

<attribute [name](name.docx)="algn" type="[ST\_TextAlignType](ST_TextAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="defTabSz" type="[ST\_Coordinate32](ST_Coordinate32.docx)" use="optional"/>

<attribute [name](name.docx)="rtl" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="eaLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="fontAlgn" type="[ST\_TextFontAlignType](ST_TextFontAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="latinLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="hangingPunct" type="xsd:boolean" use="optional"/>

</complexType>

##### lvl8pPr (List Level 8 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute [lvl](lvl.docx)="7". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the [order](order.docx) in which this and other level property elements are specified be in [order](order.docx) of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[Example: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl8pPr and thus create a paragraph of text that has no bullets and is right aligned.

<p:txBody>  
 …  
 <a:lstStyle>  
 <a:lvl8pPr algn="r">  
 <a:buNone/>  
 </a:lvl8pPr>  
 </a:lstStyle>

<a:p>  
 <a:pPr [lvl](lvl.docx)="7">  
 </a:pPr>  
 …

<a:t>Some text</a:t>  
 …  
 </a:p>  
</p:txBody>

[end](end.docx) example]

[Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]

|  |
| --- |
| Parent Elements |
| bodyStyle (§); defaultTextStyle (§); lstStyle (§); notesStyle (§); otherStyle (§); titleStyle (§) |

|  |  |
| --- | --- |
| Child Elements | Subclause |
| buAutoNum (Auto-Numbered Bullet) | § |
| buBlip (Picture Bullet) | § |
| buChar (Character Bullet) | § |
| buClr (Color Specified) | § |
| buClrTx (Follow Text) | § |
| buFont (Specified) | § |
| buFontTx (Follow text) | § |
| buNone (No Bullet) | § |
| buSzPct (Bullet Size Percentage) | § |
| buSzPts (Bullet Size Points) | § |
| buSzTx (Bullet Size Follows Text) | § |
| defRPr (Default Text Run Properties) | § |
| [extLst](extLst.docx) (Extension List) | § |
| lnSpc (Line Spacing) | § |
| spcAft (Space After) | § |
| spcBef (Space Before) | § |
| tabLst (Tab List) | § |

|  |  |
| --- | --- |
| Attributes | Description |
| algn (Alignment) | Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied.    [Example: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextAlignType](ST_TextAlignType.docx) simple type (§). |
| defTabSz (Default Tab Size) | Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.  [Example: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.  <p:txBody>  …  <a:p>  <a:pPr defTabSz="376300" …/>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_Coordinate32](ST_Coordinate32.docx) simple type (§). |
| eaLnBrk (East Asian Line Break) | Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr eaLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>      [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| fontAlgn (Font Alignment) | Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.  [Example: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.    <a:txtBody>  …  <a:pPr fontAlgn="b" …/>  …  <a:r>  <a:rPr …/>  <a:t>H </a:t>  </a:r>  <a:r>  <a:rPr sz="1200" …/>  <a:t>2</a:t>  </a:r>  <a:r>  <a:rPr …/>  <a:t>O</a:t>  </a:r>  … </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextFontAlignType](ST_TextFontAlignType.docx) simple type (§). |
| hangingPunct (Hanging Punctuation) | Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the [end](end.docx) of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied.  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| indent (Indent) | Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.    [Example: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" indent="571500" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Here is some…</a:t>  …  </a:p> </p:txBody>  By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextIndent](ST_TextIndent.docx) simple type (§). |
| latinLnBrk (Latin Line Break) | Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr latinLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| [lvl](lvl.docx) (Level) | Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.  [Example: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again [lvl](lvl.docx)="1" is considered to be level 2.  <p:txBody>  …  <a:p>  <a:pPr [lvl](lvl.docx)="1" …/>  …  <a:t>Sample text</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  [Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]  The possible values for this attribute are defined by the [ST\_TextIndentLevelType](ST_TextIndentLevelType.docx) simple type (§). |
| marL (Left Margin) | Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| marR (Right Margin) | Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| rtl (Right To Left) | Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.  [Example: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.  <p:txBody>  …  <a:p>  <a:pPr rtl="1" …/>  …  <a:t>Sample text…</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextParagraphProperties">

<sequence>

<element name="lnSpc" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcBef" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcAft" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletColor" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletSize" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBullet" minOccurs="0" maxOccurs="1"/>

<element name="tabLst" type="CT\_TextTabStopList" minOccurs="0" maxOccurs="1"/>

<element name="defRPr" type="CT\_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>

<element name="[extLst](extLst.docx)" type="CT\_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>

</sequence>

<attribute [name](name.docx)="marL" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="marR" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="[lvl](lvl.docx)" type="[ST\_TextIndentLevelType](ST_TextIndentLevelType.docx)" use="optional"/>

<attribute [name](name.docx)="indent" type="[ST\_TextIndent](ST_TextIndent.docx)" use="optional"/>

<attribute [name](name.docx)="algn" type="[ST\_TextAlignType](ST_TextAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="defTabSz" type="[ST\_Coordinate32](ST_Coordinate32.docx)" use="optional"/>

<attribute [name](name.docx)="rtl" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="eaLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="fontAlgn" type="[ST\_TextFontAlignType](ST_TextFontAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="latinLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="hangingPunct" type="xsd:boolean" use="optional"/>

</complexType>

##### lvl9pPr (List Level 9 Text Style)

This element specifies all paragraph level text properties for all elements that have the attribute [lvl](lvl.docx)="8". There are a total of 9 level text property elements allowed, levels 0-8. It is recommended that the [order](order.docx) in which this and other level property elements are specified be in [order](order.docx) of increasing level. That is lvl2pPr should come before lvl3pPr. This allows the lower level properties to take precedence over the higher level ones because they are parsed first.

[Example: Consider the following DrawingML code that would specify a paragraph to follow the level style defined in lvl9pPr and thus create a paragraph of text that has no bullets and is right aligned.

<p:txBody>  
 …  
 <a:lstStyle>  
 <a:lvl9pPr algn="r">  
 <a:buNone/>  
 </a:lvl9pPr>  
 </a:lstStyle>

<a:p>  
 <a:pPr [lvl](lvl.docx)="8">  
 </a:pPr>  
 …  
 <a:t>Some text</a:t>  
 …  
 </a:p>  
</p:txBody>

[end](end.docx) example]

[Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]

|  |
| --- |
| Parent Elements |
| bodyStyle (§); defaultTextStyle (§); lstStyle (§); notesStyle (§); otherStyle (§); titleStyle (§) |

|  |  |
| --- | --- |
| Child Elements | Subclause |
| buAutoNum (Auto-Numbered Bullet) | § |
| buBlip (Picture Bullet) | § |
| buChar (Character Bullet) | § |
| buClr (Color Specified) | § |
| buClrTx (Follow Text) | § |
| buFont (Specified) | § |
| buFontTx (Follow text) | § |
| buNone (No Bullet) | § |
| buSzPct (Bullet Size Percentage) | § |
| buSzPts (Bullet Size Points) | § |
| buSzTx (Bullet Size Follows Text) | § |
| defRPr (Default Text Run Properties) | § |
| [extLst](extLst.docx) (Extension List) | § |
| lnSpc (Line Spacing) | § |
| spcAft (Space After) | § |
| spcBef (Space Before) | § |
| tabLst (Tab List) | § |

|  |  |
| --- | --- |
| Attributes | Description |
| algn (Alignment) | Specifies the alignment that is to be applied to the paragraph. Possible values for this include left, right, centered, justified and distributed. If this attribute is omitted, then a value of left is implied.    [Example: Consider the case where the user wishes to have two columns of text that have a justified alignment, much like text within a book. The following DrawingML could describe this.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextAlignType](ST_TextAlignType.docx) simple type (§). |
| defTabSz (Default Tab Size) | Specifies the default size for a tab character within this paragraph. This attribute should be used to describe the spacing of tabs within the paragraph instead of a leading indentation tab. For indentation tabs there are the marL and indent attributes to assist with this.  [Example: Consider the case where a paragraph contains numerous tabs that need to be of a specific size. The following DrawingML would describe this.  <p:txBody>  …  <a:p>  <a:pPr defTabSz="376300" …/>  …  <a:t>Sample Text …</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_Coordinate32](ST_Coordinate32.docx) simple type (§). |
| eaLnBrk (East Asian Line Break) | Specifies whether an East Asian word can be broken in half and wrapped onto the next line without a hyphen being added. To determine whether an East Asian word can be broken the presentation application would use the kinsoku settings here. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable East Asian words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr eaLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>      [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| fontAlgn (Font Alignment) | Determines where vertically on a line of text the actual words are positioned. This deals with vertical placement of the characters with respect to the baselines. For instance having text anchored to the top baseline, anchored to the bottom baseline, centered in between, etc. To understand this attribute and it's use it is helpful to understand what baselines are. A diagram describing these different cases is shown below. If this attribute is omitted, then a value of base is implied.  [Example: Consider the case where the user wishes to represent the chemical compound of a water molecule. For this they will need to make sure the H, the 2, and the O are all in the correct position and are of the correct size. The results below can be achieved through the DrawingML shown below.    <a:txtBody>  …  <a:pPr fontAlgn="b" …/>  …  <a:r>  <a:rPr …/>  <a:t>H </a:t>  </a:r>  <a:r>  <a:rPr sz="1200" …/>  <a:t>2</a:t>  </a:r>  <a:r>  <a:rPr …/>  <a:t>O</a:t>  </a:r>  … </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextFontAlignType](ST_TextFontAlignType.docx) simple type (§). |
| hangingPunct (Hanging Punctuation) | Specifies whether punctuation is to be forcefully laid out on a line of text or put on a different line of text. That is, if there is punctuation at the [end](end.docx) of a run of text that should be carried over to a separate line does it actually get carried over. A true value will allow for hanging punctuation forcing the punctuation to not be carried over and a value of false will allow the punctuation to be carried onto the next text line. If this attribute is omitted, then a value of 0, or false is implied.  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| indent (Indent) | Specifies the indent size that will be applied to the first line of text in the paragraph. An indentation of 0 will be considered to be at the same location as marL attribute. If this attribute is omitted, then a value of -342900 is implied.    [Example: Consider the scenario where the user now wanted to add a paragraph indentation to the first line of text in their two column format book.  <p:txBody>  <a:bodyPr numCol="2" spcCol="914400"…/>  <a:normAutofit/>  </a:bodyPr>  …  <a:p>  <a:pPr marL="0" indent="571500" algn="just">  <a:buNone/>  </a:pPr>  …  <a:t>Here is some…</a:t>  …  </a:p> </p:txBody>  By adding the indent attribute the user has effectively added a first line indent to this paragraph of text. [end](end.docx) example]  The possible values for this attribute are defined by the [ST\_TextIndent](ST_TextIndent.docx) simple type (§). |
| latinLnBrk (Latin Line Break) | Specifies whether a Latin word can be broken in half and wrapped onto the next line without a hyphen being added. This attribute is to be used specifically when there is a word that cannot be broken into multiple pieces without a hyphen. That is it will not be present within the existence of normal breakable Latin words but will when a special case word arises that should not be broken for a line break. If this attribute is omitted, then a value of 1, or true is implied.  [Example: Consider the case where the presentation contains a long word that must not be divided with a line break. Instead it should be placed, in [whole](whole.docx) on a new line so that it may fit. The picture below shows a normal paragraph where a long word has been broken for a line break. The second picture shown below shows that same paragraph with the long word specified to not allow a line break. The resulting DrawingML is as follows.  <p:txBody>  …  <a:p>  <a:pPr latinLnBrk="0" …/>  …  <a:t>Sample text (Long word)</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |
| [lvl](lvl.docx) (Level) | Specifies the particular level text properties that this paragraph will follow. The value for this attribute is numerical and formats the text according to the corresponding level paragraph properties that are listed within the lstStyle element. Since there are nine separate level properties defined, this tag will have an effective range of 0-8 = 9 available values.  [Example: Consider the following DrawingML. This would specify that this paragraph should follow the lvl2pPr formatting style because once again [lvl](lvl.docx)="1" is considered to be level 2.  <p:txBody>  …  <a:p>  <a:pPr [lvl](lvl.docx)="1" …/>  …  <a:t>Sample text</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  [Note: To resolve conflicting paragraph properties the linear hierarchy of paragraph properties should be examined starting first with the pPr element. The [rule](rule.docx) here is that properties that are defined at a level closer to the actual text should take precedence. That is if there is a conflicting property between the pPr and lvl1pPr elements then the pPr property should take precedence because in the property hierarchy it is closer to the actual text being represented. [end](end.docx) note]  The possible values for this attribute are defined by the [ST\_TextIndentLevelType](ST_TextIndentLevelType.docx) simple type (§). |
| marL (Left Margin) | Specifies the left margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marL attributes are additive with respect to the text position. If this attribute is omitted, then a value of 347663 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| marR (Right Margin) | Specifies the right margin of the paragraph. This is specified in addition to the text body inset and applies only to this text paragraph. That is the text body inset and the marR attributes are additive with respect to the text position. If this attribute is omitted, then a value of 0 is implied.  The possible values for this attribute are defined by the [ST\_TextMargin](ST_TextMargin.docx) simple type (§). |
| rtl (Right To Left) | Specifies whether the text is right-to-left or left-to-right in its flow direction. If this attribute is omitted, then a value of 0, or left-to-right is implied.  [Example: Consider the scenario where the user wanted text to flow from right to left. That is within the bounding text box the first word would be right aligned with each additional word being written to the left of the previous while continuing to flow top to bottom. The DrawingML to describe this might be as follows.  <p:txBody>  …  <a:p>  <a:pPr rtl="1" …/>  …  <a:t>Sample text…</a:t>  …  </a:p> </p:txBody>  [end](end.docx) example]  The possible values for this attribute are defined by the XML Schema boolean datatype. |

The following XML Schema fragment defines the contents of this element:

<complexType [name](name.docx)="CT\_TextParagraphProperties">

<sequence>

<element name="lnSpc" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcBef" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<element name="spcAft" type="CT\_TextSpacing" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletColor" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletSize" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBulletTypeface" minOccurs="0" maxOccurs="1"/>

<group ref="EG\_TextBullet" minOccurs="0" maxOccurs="1"/>

<element name="tabLst" type="CT\_TextTabStopList" minOccurs="0" maxOccurs="1"/>

<element name="defRPr" type="CT\_TextCharacterProperties" minOccurs="0" maxOccurs="1"/>

<element name="[extLst](extLst.docx)" type="CT\_OfficeArtExtensionList" minOccurs="0" maxOccurs="1"/>

</sequence>

<attribute [name](name.docx)="marL" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="marR" type="[ST\_TextMargin](ST_TextMargin.docx)" use="optional"/>

<attribute [name](name.docx)="[lvl](lvl.docx)" type="[ST\_TextIndentLevelType](ST_TextIndentLevelType.docx)" use="optional"/>

<attribute [name](name.docx)="indent" type="[ST\_TextIndent](ST_TextIndent.docx)" use="optional"/>

<attribute [name](name.docx)="algn" type="[ST\_TextAlignType](ST_TextAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="defTabSz" type="[ST\_Coordinate32](ST_Coordinate32.docx)" use="optional"/>

<attribute [name](name.docx)="rtl" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="eaLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="fontAlgn" type="[ST\_TextFontAlignType](ST_TextFontAlignType.docx)" use="optional"/>

<attribute [name](name.docx)="latinLnBrk" type="xsd:boolean" use="optional"/>

<attribute [name](name.docx)="hangingPunct" type="xsd:boolean" use="optional"/>

</complexType>