#### Body Formatting

Being the highest level of formatting available within a shape, the body properties allow for the manipulation of the text area as a whole. This means that all paragraphs and runs of text for the shape in question would be encompassed within here and, therefore, follow the text body style defined here.

[Example: Consider a shape that has three paragraphs within it, each with a different sized text. If this shape is resized to be smaller, then the text will no longer fit the same way within the shape. Thus, we see that to maintain visual quality the size must be changed.

Rather than try to change each of the paragraphs to a smaller font size to fit in the shape, just use a body-level format property such as the normAutofit. This will apply to all text within the shape and scale the text by a certain percentage in [order](order.docx) to fit within the shape.

<p:txBody>
 <a:bodyPr>
 <a:normAutofit fontScale="20000" lnSpcReduction="20000"/>

 </a:bodyPr>

 …

 <a:p>
 …
 </a:p>

 <a:p>
 …
 </a:p>

 <a:p>
 …
 </a:p>
</p:txBody>

[end](end.docx) example]

##### bodyPr (Body Properties)

This element defines the body properties for the text body within a shape.

[Example: Consider a shape with a text body that has some formatting properties associated with it. For the formatting of text body properties, the bodyPr element should be used as follows:

<p:sp>
 …
 <p:txBody>
 <a:bodyPr>
 (text body properties)
 </a:bodyPr>
 …
 </p:txBody>
</p:sp>

[end](end.docx) example]

|  |
| --- |
| 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 |
| [extLst](extLst.docx) (Extension List) | § |
| [flatTx](flatTx.docx) (No text in [3D](3D.docx) scene) | § |
| noAutofit (No AutoFit) | § |
| normAutofit (Normal AutoFit) | § |
| [prstTxWarp](prstTxWarp.docx) (Preset Text Warp) | § |
| [scene3d](scene3d.docx) (3D Scene Properties) | § |
| [sp3d](sp3d.docx) (Apply [3D](3D.docx) shape properties) | § |
| spAutoFit (Shape AutoFit) | § |

|  |  |
| --- | --- |
| Attributes | Description |
| [anchor](anchor.docx) (Anchor) | Specifies the anchoring position of the [txBody](txBody.docx) within the shape. If this attribute is omitted, then a value of [t](t.docx), or top is implied.[Example: Consider the following DrawingML:<p:txBody> <a:bodyPr anchor="ctr" … /> …</p:txBody>Here the text is vertically aligned in the center of the shape within which it is contained. [end](end.docx) example]The possible values for this attribute are defined by the [ST\_TextAnchoringType](ST_TextAnchoringType.docx) simple type (§). |
| anchorCtr (Anchor Center) | Specifies the centering of the text box. The way it works fundamentally is to determine the smallest possible "bounds box" for the text and then to center that "bounds box" accordingly. Note that this is different than paragraph alignment, which aligns the text within the "bounds box" for the text. This flag is compatible with all of the different types of anchoring. If this attribute is omitted, then a value of 0, or [off](off.docx) is implied.[Example: The text within this shape has been both vertically centered with the anchor attribute and horizontally centered with the anchorCtr attribute.<p:txBody> <a:bodyPr anchor="ctr" anchorCtr="1" … /> …</p:txBody>[end](end.docx) example]The possible values for this attribute are defined by the XML Schema boolean datatype. |
| bIns (Bottom Inset) | Specifies the bottom inset of the bounding rectangle. Insets are used just as internal margins for text boxes within shapes. If this attribute is omitted, a value of 45720 or 0.05 inches is implied.[Example: Consider the following DrawingML:<p:txBody> <a:bodyPr lIns="91440" tIns="91440"  rIns="91440" bIns="91440" … /> …</p:txBody>The text box having the above body properties will have inset margins of 91440 or 0.1 inches on all four sides. [end](end.docx) example]The possible values for this attribute are defined by the [ST\_Coordinate32](ST_Coordinate32.docx) simple type (§). |
| compatLnSpc (Compatible Line Spacing) | Specifies that the line spacing for this text body will be decided in a simplistic manner using the font scene. If this attribute is omitted, a value of 0 or false is implied.The possible values for this attribute are defined by the XML Schema boolean datatype. |
| forceAA (Force Anti-Alias) | Forces the text to be rendered anti-aliased regardless of the font size. Certain fonts may appear grainy around their edges unless they are anti-aliased. Therefore this attribute allows for the specifying of which bodies of text should always be anti-aliased and which ones should not. If this attribute is omitted, then a value of 0, or [off](off.docx) is implied.The possible values for this attribute are defined by the XML Schema boolean datatype. |
| fromWordArt (From WordArt) | Specifies that text within this textbox is converted text from a WordArt object. This is more of a backwards compatibility attribute that is useful to the application from a tracking perspective. WordArt was the former way to apply text effects and therefore this attribute is useful in document conversion scenarios. If this attribute is omitted, then a value of 0, or [off](off.docx) is implied.[Example: Consider the following DrawingML:<p:txBody> <a:bodyPr wrap="none" fromWordArt="1" …  /> …</p:txBody>[end](end.docx) example] Because of the presence of the fromWordArt attribute the text within this shape can be mapped back to the corresponding WordArt during document conversion. [end](end.docx) example]The possible values for this attribute are defined by the XML Schema boolean datatype. |
| horzOverflow (Text Horizontal Overflow) | Determines whether the text can flow out of the bounding box horizontally. This is used to determine what will happen in the event that the text within a shape is too large for the bounding box it is contained within. If this attribute is omitted, then a value of overflow is implied.[Example: Consider the case where we have multiply paragraphs within a shape and the second is greater in length and causes text to flow outside the shape. By applying the clip value of the horzOverflow attribute as a body property this overflowing text will now be cut [off](off.docx) instead of extending beyond the bounds of the shape.<p:txBody> <a:bodyPr horzOverflow="clip" … /> … <a:p> … (Some text) … </a:p> <a:p> … (Some more text) … </a:p></p:txBody>[end](end.docx) example]The possible values for this attribute are defined by the [ST\_TextHorzOverflowType](ST_TextHorzOverflowType.docx) simple type (§). |
| lIns (Left Inset) | Specifies the left inset of the bounding rectangle. Insets are used just as internal margins for text boxes within shapes. If this attribute is omitted, then a value of 91440 or 0.1 inches is implied.[Example: Consider the following DrawingML:<p:txBody> <a:bodyPr lIns="91440" tIns="91440" rIns="91440"  bIns="91440" … /> …</p:txBody>The text box having the above body properties will have inset margins of 91440 or 0.1 inches on all four sides. [end](end.docx) example]The possible values for this attribute are defined by the [ST\_Coordinate32](ST_Coordinate32.docx) simple type (§). |
| numCol (Number of Columns) | Specifies the number of columns of text in the bounding rectangle. When applied to a text run this property takes the width of the bounding box for the text and divides it by the number of columns specified. These columns are then treated as overflow containers in that when the previous column has been filled with text the next column will act as the repository for additional text. When all columns have been filled and text still remains then the overflow properties set for this text body will be used and the text will be reflowed to make room for additional text. If this attribute is omitted, then a value of 1 is implied.[Example: Consider the case where a text area would need to be split [up](up.docx) into four separate columns. Then simply specifying one paragraph with one run of text is enough to describe four columns of text here.<p:txBody> <a:bodyPr numCol="4" … /> <a:p> <a:r> … (Some text) … </a:r> </a:p></p:txBody>[end](end.docx) example]The possible values for this attribute are defined by the [ST\_TextColumnCount](ST_TextColumnCount.docx) simple type (§). |
| rIns (Right Inset) | Specifies the right inset of the bounding rectangle. Insets are used just as internal margins for text boxes within shapes. If this attribute is omitted, then a value of 91440 or 0.1 inches is implied.[Example: Consider the following DrawingML:<p:txBody> <a:bodyPr lIns="91440" tIns="91440"  rIns="91440" bIns="91440" … /> …</p:txBody>The text box having the above body properties will have inset margins of 91440 or 0.1 inches on all four sides. [end](end.docx) example]The possible values for this attribute are defined by the [ST\_Coordinate32](ST_Coordinate32.docx) simple type (§). |
| [rot](rot.docx) (Rotation) | Specifies the rotation that is being applied to the text within the bounding box. If it not specified, the rotation of the accompanying shape is used. If it is specified, then this is applied independently from the shape. That is the shape can have a rotation applied in addition to the text itself having a rotation applied to it. If this attribute is omitted, then a value of 0, is implied.[Example: Consider the case where a shape has a rotation of 5400000, or 90 degrees clockwise applied to it. In addition to this, the text body itself has a rotation of -5400000, or 90 degrees counter-clockwise applied to it. Then the resulting shape would appear to be rotated but the text within it would appear as though it had not been rotated at all. The DrawingML specifying this would look like the following:<p:sp> <p:spPr> <a:xfrm [rot](rot.docx)="5400000"> … </a:xfrm> </p:spPr> … <p:txBody> <a:bodyPr [rot](rot.docx)="-5400000" … /> … (Some text) … </p:txBody></p:sp>[end](end.docx) example]The possible values for this attribute are defined by the [ST\_Angle](ST_Angle.docx) simple type (§). |
| rtlCol (Columns Right-To-Left) | Specifies whether columns are used in a right-to-left or left-to-right order. The usage of this attribute only sets the column [order](order.docx) that is used to determine which column overflow text should go to next. If this attribute is omitted, then a value of 0, or [off](off.docx) is implied in which case text will [start](start.docx) in the leftmost column and flow to the right.[Note: This attribute in no way determines the direction of text but merely the direction in which multiple columns are used. [end](end.docx) note]The possible values for this attribute are defined by the XML Schema boolean datatype. |
| spcCol (Space Between Columns) | Specifies the space between text columns in the text area. This should only apply when there is more than 1 column present. If this attribute is omitted, then a value of 0 is implied.The possible values for this attribute are defined by the [ST\_PositiveCoordinate32](ST_PositiveCoordinate32.docx) simple type (§). |
| spcFirstLastPara (Paragraph Spacing) | Specifies whether the before and after paragraph spacing defined by the user is to be respected. While the spacing between paragraphs is helpful, it is additionally useful to be able to set a flag as to whether this spacing is to be followed at the edges of the text body, in other words the first and last paragraphs in the text body. More precisely since this is a text body level property it should only [effect](effect.docx) the before paragraph spacing of the first paragraph and the after paragraph spacing of the last paragraph for a given text body. If this attribute is omitted, then a value of 0, or false is implied.[Example: Consider the case where spacing has been defined between multiple paragraphs within a text body using the spcBef and spcAft paragraph spacing attributes. For this text body however the user would like to not have this followed for the edge paragraphs and thus we have the following DrawingML.<p:txBody> <a:bodyPr spcFirstLastPara="0" … /> … <a:p> <a:pPr> <a:spcBef> <a:spcPts [val](val.docx)="1800"/> </a:spcBef> <a:spcAft> <a:spcPts [val](val.docx)="600"/> </a:spcAft> </a:pPr> … (Some text) … </a:p> <a:p> <a:pPr> <a:spcBef> <a:spcPts [val](val.docx)="1800"/> </a:spcBef> <a:spcAft> <a:spcPts [val](val.docx)="600"/> </a:spcAft> </a:pPr> … (Some text) … </a:p> …</p:txBody>[end](end.docx) example]The possible values for this attribute are defined by the XML Schema boolean datatype. |
| tIns (Top Inset) | Specifies the top inset of the bounding rectangle. Insets are used just as internal margins for text boxes within shapes. If this attribute is omitted, then a value of 45720 or 0.05 inches is implied.[Example: Consider the following DrawingML:<p:txBody> <a:bodyPr lIns="91440" tIns="91440"  rIns="91440" bIns="91440" … /> …</p:txBody>The text box having the above body properties will have inset margins of 91440 or 0.1 inches on all four sides. [end](end.docx) example]The possible values for this attribute are defined by the [ST\_Coordinate32](ST_Coordinate32.docx) simple type (§). |
| upright (Text Upright) | Specifies whether text should remain upright, regardless of the transform applied to it and the accompanying shape transform. If this attribute is omitted, then a value of 0, or false will be implied.[Example: Consider text that has been rotated within the text body but has the upright flag set.<p:txBody> <a:bodyPr upright="1" [rot](rot.docx)="5400000" …/> … (Some text) …</p:txBody>This text will appear as though no transform has been applied to it. [end](end.docx) example]The possible values for this attribute are defined by the XML Schema boolean datatype. |
| vert (Vertical Text) | Determines if the text within the given text body should be displayed vertically. If this attribute is omitted, then a value of horz, or no vertical text is implied.[Example: Consider the case where the user needs to display text that appears vertical and has a right to left flow with respect to its columns.<p:txBody> <a:bodyPr vert="wordArtVertRtl" … /> … <a:p> … <a:t>This is</a:t> … </a:p> <a:p> … <a:t>some text.</a:t> … </a:p></p:txBody>In the above sample DrawingML there are two paragraphs denoting a separation between the text otherwise which are known as either a line or paragraph break. Because wordArtVertRtl is used here this text will not only be displayed in a stacked manner flowing from top to bottom but also have the first paragraph be displayed to the right of the second. This is because it is both vertical text and right to left. [end](end.docx) example]The possible values for this attribute are defined by the [ST\_TextVerticalType](ST_TextVerticalType.docx) simple type (§). |
| vertOverflow (Text Vertical Overflow) | Determines whether the text can flow out of the bounding box vertically. This is used to determine what will happen in the event that the text within a shape is too large for the bounding box it is contained within. If this attribute is omitted, then a value of overflow is implied.[Example: Consider the case where we have multiply paragraphs within a shape and the second causes text to flow outside the shape. By applying the clip value of the vertOverflow attribute as a body property this overflowing text will now be cut [off](off.docx) instead of extending beyond the bounds of the shape.<p:txBody> <a:bodyPr vertOverflow="clip" … /> … <a:p> … (Some text) … </a:p> <a:p> … (Some longer text) … </a:p></p:txBody>[end](end.docx) example]The possible values for this attribute are defined by the [ST\_TextVertOverflowType](ST_TextVertOverflowType.docx) simple type (§). |
| wrap (Text Wrapping Type) | Specifies the wrapping options to be used for this text body. If this attribute is omitted, then a value of square is implied which will wrap the text using the bounding text box. The possible values for this attribute are defined by the [ST\_TextWrappingType](ST_TextWrappingType.docx) simple type (§). |

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

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

 <sequence>

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

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

 <element name="[scene3d](scene3d.docx)" type="CT\_Scene3D" minOccurs="0" maxOccurs="1"/>

 <group ref="EG\_Text3D" minOccurs="0" maxOccurs="1"/>

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

 </sequence>

 <attribute [name](name.docx)="[rot](rot.docx)" type="[ST\_Angle](ST_Angle.docx)" use="optional"/>

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

 <attribute [name](name.docx)="vertOverflow" type="[ST\_TextVertOverflowType](ST_TextVertOverflowType.docx)" use="optional"/>

 <attribute [name](name.docx)="horzOverflow" type="[ST\_TextHorzOverflowType](ST_TextHorzOverflowType.docx)" use="optional"/>

 <attribute [name](name.docx)="vert" type="[ST\_TextVerticalType](ST_TextVerticalType.docx)" use="optional"/>

 <attribute [name](name.docx)="wrap" type="[ST\_TextWrappingType](ST_TextWrappingType.docx)" use="optional"/>

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

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

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

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

 <attribute [name](name.docx)="numCol" type="[ST\_TextColumnCount](ST_TextColumnCount.docx)" use="optional"/>

 <attribute [name](name.docx)="spcCol" type="[ST\_PositiveCoordinate32](ST_PositiveCoordinate32.docx)" use="optional"/>

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

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

 <attribute [name](name.docx)="[anchor](anchor.docx)" type="[ST\_TextAnchoringType](ST_TextAnchoringType.docx)" use="optional"/>

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

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

 <attribute name="upright" type="xsd:boolean" use="optional" default="false"/>

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

</complexType>

##### noAutofit (No AutoFit)

This element specifies that text within the text body should not be auto-fit to the bounding box. Auto-fitting is when text within a text box is scaled in [order](order.docx) to remain inside the text box. If this element is omitted, then noAutofit or auto-fit [off](off.docx) is implied.

[Example: Consider a text box where the user wishes to have the text extend outside the bounding box. The following DrawingML would describe this.

<p:txBody>
 <a:bodyPr wrap="none" rtlCol="0">
 <a:noAutofit/>
 </a:bodyPr>

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

[end](end.docx) example]

|  |
| --- |
| Parent Elements |
| bodyPr (§) |

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

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

##### normAutofit (Normal AutoFit)

This element specifies that text within the text body should be normally auto-fit to the bounding box. Auto-fitting is when text within a text box is scaled in [order](order.docx) to remain inside the text box. If this element is omitted, then noAutofit or auto-fit [off](off.docx) is implied.

[Example: Consider the situation where a user is building a diagram and needs to have the text for each shape that they are using stay within the bounds of the shape. An easy way this might be done is by using normAutofit. The following DrawingML will illustrate how this might be accomplished.

<p:sp>
 <p:txBody>
 <a:bodyPr rtlCol="0" anchor="ctr">
 <a:normAutofit fontScale="92000" lnSpcReduction="20000"/>
 </a:bodyPr>
 …

 <a:p>
 …
 <a:t>Diagram Object 1</a:t>
 …
 </a:p>
 </p:txBody>
</p:sp>

<p:sp>
 <p:txBody>
 <a:bodyPr rtlCol="0" anchor="ctr">
 <a:normAutofit fontScale="92000" lnSpcReduction="20000"/>
 </a:bodyPr>
 …

 <a:p>
 …
 <a:t>Diagram Object 2</a:t>
 …
 </a:p>
 </p:txBody>
</p:sp>

In the above example there are two shapes that have normAutofit turned on so that when the user types more text within the shape that the text will actually resize to accommodate the new data. For the application to know how and to what degree the text should be resized two attributes are set for the auto-fit resize logic. [end](end.docx) example]

|  |
| --- |
| Parent Elements |
| bodyPr (§) |

|  |  |
| --- | --- |
| Attributes | Description |
| fontScale (Font Scale) | Specifies the percentage of the original font size to which each run in the text body is scaled. In [order](order.docx) to auto-fit text within a bounding box it is sometimes necessary to decrease the font size by a certain percentage. Using this attribute the font within a text box can be scaled based on the value provided. A value of 100000 is treated as 100% while a value of 1000 is consequently 1%. If this attribute is omitted, then a value of 100000 or 100% is implied.The possible values for this attribute are defined by the [ST\_TextFontScalePercent](ST_TextFontScalePercent.docx) simple type (§). |
| lnSpcReduction (Line Space Reduction) | Specifies the percentage amount by which the line spacing of each paragraph in the text body is reduced. The reduction is applied by subtracting it from the original line spacing value. Using this attribute the vertical spacing between the lines of text can be scaled by a percent amount. A value of 100000 is treated as 100% while a value of 1000 is consequently 1%. If this attribute is omitted, then a value of 0 or 0% is implied.[Note: This attribute applies only to paragraphs with percentage line spacing. [end](end.docx) note]The possible values for this attribute are defined by the [ST\_TextSpacingPercent](ST_TextSpacingPercent.docx) simple type (§). |

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

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

 <attribute name="fontScale" type="[ST\_TextFontScalePercent](ST_TextFontScalePercent.docx)" use="optional" default="100000"/>

 <attribute name="lnSpcReduction" type="[ST\_TextSpacingPercent](ST_TextSpacingPercent.docx)" use="optional" default="0"/>

</complexType>

##### spAutoFit (Shape AutoFit)

This element specifies that a shape should be auto-fit to fully contain the text described within it. Auto-fitting is when text within a shape is scaled in [order](order.docx) to contain all the text inside. If this element is omitted, then noAutofit or auto-fit [off](off.docx) is implied.

[Example: Consider the situation where a user is building a diagram and needs to have the text for each shape that they are using stay within the bounds of the shape. An easy way this might be done is by using spAutofit. The following DrawingML will illustrate how this might be accomplished.

<p:sp>
 <p:txBody>
 <a:bodyPr rtlCol="0" anchor="ctr">
 <a:spAutoFit/>
 </a:bodyPr>
 …
 <a:p>
 …
 <a:t>Diagram Object 1</a:t>
 …
 </a:p>
 </p:txBody>
</p:sp>

<p:sp>
 <p:txBody>
 <a:bodyPr rtlCol="0" anchor="ctr">
 <a:spAutoFit/>
 </a:bodyPr>
 …
 <a:p>
 …
 <a:t>Diagram Object 2</a:t>
 …
 </a:p>
 </p:txBody>
</p:sp>

In the above example there are two shapes that have spAutoFit turned on so that when the user types more text within the shape that the shape will actually resize to accommodate the new data. [end](end.docx) example]

|  |
| --- |
| Parent Elements |
| bodyPr (§) |

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

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