#### xmlColumnPr (XML Column Properties)

An element defining the XML column properties for a column. This is only used for [tables](tables.docx) created from XML mappings.

[Example: Here is a simple example showing a [table](table.docx) column that has an xmlColumPr.

<[tableColumn](tableColumn.docx) id="1" uniqueName="SomeElement" name="SomeElement">
 <xmlColumnPr mapId="1" xpath="/xml/foo/element" xmlDataType="string"/>
</[tableColumn](tableColumn.docx)>

end example]

|  |
| --- |
| Parent Elements |
| [tableColumn](tableColumn.docx) (§) |

|  |  |
| --- | --- |
| Child Elements | Subclause |
| [extLst](extLst.docx) (Future Feature Data Storage Area) | § |

|  |  |
| --- | --- |
| Attributes | Description |
| denormalized (Denormalized) | A Boolean that indicates whether the contents of the column have been filled down due to flattening. True if it has been filled down (denormilized), false otherwise. This should be used when an XML mapping parent value has many children, and both the parent and child fields are mapped to their own column in the table.[Example:<Order ID="3"> <Item>Milk</Item> <Item>Bread</Item> <Item>Cheese</Item></Order>The resulting [table](table.docx) in the spreadsheet application would have two columns, the first with the [item](item.docx) ID, filled down for each [item](item.docx) in the [table](table.docx) as follows:Item ID Item 3 Milk 3 Bread 3 Cheeseend example]The possible values for this attribute are defined by the XML [Schema](Schema.docx) boolean datatype. |
| mapId (XML [Map](Map.docx) Id) | An integer representing the ID of the XML [map](map.docx) this [table](table.docx) [field](field.docx) is associated with. The XML [map](map.docx) will be defined in the xml [maps](maps.docx) part, and the [Map](Map.docx) element should have the corresponding id. The possible values for this attribute are defined by the XML [Schema](Schema.docx) unsignedInt datatype. |
| xmlDataType (XML Data Type) | An enumeration indicating which XML data type is used by this column.The possible values for this attribute are defined by the [ST\_XmlDataType](ST_XmlDataType.docx) simple type (§). |
| xpath (XPath) | A string representing the XML path to the element this column is associated with.The spreadsheet application should support XPath limited to the following * The XPath is an absolute path to a simple-content element or attribute

[Example:"/ns1:root/ns1:row/ns1:column1" is supported if 'column1' is a child-most node, but not "/ns1:root/ns1:row" for the same document since 'row' is not a child. end example]* The XPath does not express axes, but uses the default child axes

[Example: "/ns1:root/ns1:row" is supported but not "/ns1:root/child::ns1:rowend example] * An optional [filter](filter.docx) can be expressed at the end of the xpath

[Example:"/ns1:root/ns1:row/ns1:column1[@foo='abc']" is supported but not "/ns1:root/ns1:row[@foo='abc']/ns1:column1" end example]* The [filter](filter.docx) can only contain a single expression comparing a named attribute to a specific value
* Filters are only supported on XPaths that resolve to a simple-content element (not attributes)
* The named attribute must be defined as an attribute of the simple-content element
* The attribute name must be preceded by the short-hand (@) symbol representing the axes 'attribute'

[Example:"/ns1:root/ns1:row/ns1:column1[@foo='abc']" is supported not "/ns1:root/ns1:row/ns1:column1[attribute::foo='abc']"end example]* An arbitrary amount of white-space can be embedded between [filter](filter.docx) tokens

[Example: "/ns1:root/ns1:row/ns1:column1[ @ foo='abc']" is validend example]The possible values for this attribute are defined by the [ST\_Xstring](ST_Xstring.docx) simple type (§). |

The following XML [Schema](Schema.docx) fragment defines the contents of this element:

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

 <sequence>

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

 </sequence>

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

 <attribute [name](name.docx)="xpath" type="[ST\_Xstring](ST_Xstring.docx)" use="required"/>

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

 <attribute [name](name.docx)="xmlDataType" type="[ST\_XmlDataType](ST_XmlDataType.docx)" use="required"/>

</complexType>