<config>  
<output path=’${project\_loc}\Gendoc\BackupAndRestore\_1.0.0\_tsi.d.t+gendoc.1.docx' />

</config><drop/>

<context model=’${project\_loc}\BackUp.uml' element=’{0}’ element=’{0}’ importedBundles='gmf;papyrus' /><drop/>

<gendoc><drop/>

You need to modify the path name for the output file and for the input model above (in brown). <drop/>

Note that none of the red text in this document will appear in the output <drop/>

Note that commands that do not result in text being printed are highlighted in purple. Text and commands that cause print are in black <drop/>

</gendoc><drop/>

<context model=’${project\_loc}\BackUp.uml' element=’{0}’ importedBundles='gmf;papyrus' />

<gendoc><drop/>

You need to put the appropriate path name for the output file and for the input model above (in brown). Take care not to insert spaces. <drop/>

This section provides the data dictionary. <drop/>

# Classes

[for (cl:Class | Class.allInstances()->sortedBy(name))]<drop/>

## [cl.name/]

[for (co:Comment | cl.ownedComment)]<drop/>

<dropEmpty>[co.\_body.clean()/]</dropEmpty>

[/for]<drop/>

Applied stereotypes:

[for (st:Stereotype | cl.getAppliedStereotypes())]<drop/>

* [st.name/]

[for (oa:Property|st.ownedAttribute)]<drop/>

* [if (not oa.name.contains('base'))][oa.name/]: [if (not cl.getValue(st, oa.name).oclIsUndefined())][if oa.name.contains('condition')][cl.getValue(st, oa.name).oclAsType(String)/] [else][cl.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/][/if][else]<drop/>[/if]

[/if] <drop/>

[/for]<drop/>

[/for]<drop/>

[if cl.ownedAttribute->notEmpty()]<drop/>

Use one of the following two forms to lead into the table depending upon whether you want table numbering or not. Note that you will need to post-process the output document to get the right table numbers by selecting all text in the document (select all) and the updating the fields (just bring up the menu over one and select update field <drop/>

Attributes for [cl.name/]

Table 1: Attributes for [cl.name/]

<drop/>

<table><drop/>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Multiplicity** | **Access** | **Stereotypes** | **Description** |

[for (p:Property|cl.ownedAttribute)]<drop/>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [p.name/][if(not p.qualifiedName.contains(cl.name))], Inherited[/if] | [p.type.name/]  [if p.defaultValue->notEmpty()][p.defaultValue.stringValue()/][else]./.[/if] | [if(p.lower=p.upper)]1[else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [if(not(p.isReadOnly))]RW[else]R[/if] | [for (st:Stereotype | p.getAppliedStereotypes())]<drop/>  [st.name/]  [for(oa:Property|st.ownedAttribute)]<drop/>  • [if oa.name.contains('part')]partOfObjectKey: [p.getValue(st, oa.name).oclAsType(Integer)/]  [else]<drop/>  • [if oa.name.contains('attribute')]AVC: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]  [else]<drop/>  • [if oa.name.contains('Invariant')]isInvariant: [p.getValue(st, oa.name).oclAsType(Boolean)/]  [else]<drop/>  • [if oa.name.contains('value')]valueRange: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String)/][else]no range constraint[/if]  [else]<drop/>  • [if oa.name.contains('Length')]bitLength: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]  [else]<drop/>  • [if oa.name.contains('unit')]unit: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String)/][else]no unit defined[/if]  [else]<drop/>  • [if oa.name.contains('support')]support: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]  [else]<drop/>  • [if oa.name.contains('condition')][if (not p.getValue(st, oa.name).oclIsUndefined())]condition:[p.getValue(st, oa.name).oclAsType(String)/][else] <drop/> [/if]  [else]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<drop/>  [/for]<drop/> | [if p.ownedComment->notEmpty()]<drop/>  [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/]  [/for]<drop/>  [else][if (p.name.contains (‘\_’))] See referenced class  [else]  [/if]<drop/>  [/if]<drop/> |

[/for]<drop/>

</table><drop/>

[else][/if]<drop/>

[/for]<drop/>

# Data Types

[for (dt:DataType | DataType.allInstances()->sortedBy(name))]<drop/>

[if dt.oclIsTypeOf(DataType)]<drop/>

## [dt.name/]

[for (co:Comment | dt.ownedComment)]<drop/>

<dropEmpty>[co.\_body.clean()/]</dropEmpty>

[/for]<drop/>

Applied Stereotypes:

[for (st:Stereotype | dt.getAppliedStereotypes())]<drop/>

* [st.name/]

[/for]<drop/>

[if dt.ownedAttribute->notEmpty()]<drop/>

Use one of the following two forms to lead into the table depending upon whether you want table numbering or not. Note that you will need to post-process the output document to get the right table numbers by selecting all text in the document (select all) and the updating the fields (just bring up the menu over one and select update field <drop/>

Attributes for [dt.name/]

Table 1: Attributes for [dt.name/]

<drop/>

<table><drop/>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Multiplicity** | **Access** | **Stereotypes** | **Description** |

[for (p:Property|dt.ownedAttribute)]<drop/>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [p.name/] | [p.type.name/]  [if p.defaultValue->notEmpty()][p.defaultValue.stringValue()/][else]./.[/if] | [if(p.lower=p.upper)]1[else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [if(not(p.isReadOnly))]RW[else]R[/if] | [for (st:Stereotype | p.getAppliedStereotypes())]<drop/>  [st.name/]  [for(oa:Property|st.ownedAttribute)]<drop/>   * [if oa.name.contains('part')]partOfObjectKey: [p.getValue(st, oa.name).oclAsType(Integer)/]   [else]<drop/>   * [if oa.name.contains('attribute')]AVC: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('Invariant')]isInvariant: [p.getValue(st, oa.name).oclAsType(Boolean)/]   [else]<drop/>   * [if oa.name.contains('value')]valueRange: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String)/][else]no range constraint[/if]   [else]<drop/>   * [if oa.name.contains('Length')]bitLength: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('unit')]unit: [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(String)/][else]no unit defined[/if]   [else]<drop/>   * [if oa.name.contains('support')]support: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('condition')][if (not p.getValue(st, oa.name).oclIsUndefined())]condition:[p.getValue(st, oa.name).oclAsType(String)/][else] <drop/> [/if]   [else]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<drop/>  [/for]<drop/> | [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/]  [/for]<drop/> |

[/for]<drop/>

</table><drop/>

[else][/if]<drop/>

[else][/if]<drop/>

[/for]<drop/>

# Enumeration Types

[for (dt:DataType | DataType.allInstances()->sortedBy(name))]<drop/>

[if dt.oclIsTypeOf(Enumeration)]<drop/>

## [dt.name/]

[for (co:Comment | dt.ownedComment)]<drop/>

<dropEmpty>[co.\_body.clean()/]</dropEmpty>

[/for]<drop/>

Contains Enumeration Literals:

[for (e:EnumerationLiteral|dt.oclAsType(Enumeration).ownedLiteral)]<drop/>

* [e.name/]:
  + [for (co:Comment | e.ownedComment)]<drop/>
  + <dropEmpty>[co.\_body.clean()/]
  + </dropEmpty>[/for]<drop/>

[/for]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

# Primitive Types

[for (dt:DataType | DataType.allInstances()->sortedBy(name))]<drop/>

[if dt.oclIsTypeOf(PrimitiveType)]<drop/>

## [dt.name/]

[for (co:Comment | dt.ownedComment)]<drop/>

<dropEmpty>[co.\_body.clean()/]</dropEmpty>

[/for]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

</gendoc><drop/>