This template needs to be imported into a papyrus project (not your model project as it shouldn’t end up in Github). You then "right click" on the selected template and choose "generate documentation from gendoc". The output will be in the file you list in the "output path" below. <drop/>

<config services='TagFileBuffer'>  
<param key='model\_name' value='Pnf'/><drop/>  
<param key='user\_path' value='D:\eclipse-workspace\ONAP\_IM'/><drop/>  
<output path='${user\_path}\GenDoc\_output\${model\_name}Model-${date}.docx' /><drop/>

</config><drop/>

You need to put the appropriate path name for the input model file. Note that this is an associated UML file. Take care not to insert spaces. <drop/>

<context model='${user\_path}\${model\_name}.notation' element='{0}' importedBundles='gmf;papyrus' /><drop/>

<gendoc><drop/>

## Diagrams

[for (d : notation::Diagram |notation::Diagram.allInstances()->sortedBy(name))]<drop/>

### [d.name/]

<image object='[d.getDiagram()/]' maxW='true' keepH='false' keepW=’false’ ><drop/>

</image>

[/for]<drop/>

</gendoc><drop/>

You need to put the appropriate path name for the input model file. Note that this is an associated UML file. Take care not to insert spaces. <drop/>

<context model=’${user\_path}\${model\_name}.uml’ element=’{0}’ importedBundles='gmf;papyrus' />

<gendoc><drop/>

## Classes

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

### [cl.name/]

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

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

[/for]<drop/>

[if (cl.oclAsType(uml::Classifier).general->notEmpty())]<drop/>

**Parent class:** [cl.oclAsType(uml::Classifier).general ->asSequence()->first().name/]

[/if]<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] [if oa.name.contains('reference')][cl.getValue(st, oa.name).oclAsType(String)/] [else] [cl.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/][/if][/if][else]<drop/>[/if]

[/if] <drop/>

[/for]<drop/>

[/for]<drop/>

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

<table><drop/>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Mult.** | **Stereotypes** | **Description** | **Defined in** |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [p.name/] | [p.type.name/] | [if(p.lower=p.upper)]1[else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [for (st:Stereotype | p.getAppliedStereotypes())]<drop/>  [st.name/]  [for(oa:Property|st.ownedAttribute)]<drop/>   * [if oa.name.contains('attribute')]AVC: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('isInvariant')]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).clean()/][else] no range constraint [/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).clean()/][else] <drop/> [/if]   [else]<drop/>   * [if oa.name.contains('passedByRef')] [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(Boolean)/][else] undefined [/if]   [else]<drop/>   * [if oa.name.contains('reference')][if (not p.getValue(st, oa.name).oclIsUndefined())]reference:[p.getValue(st, oa.name).oclAsType(String).clean()/][else] <drop/> [/if]   [else]<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] | [p.qualifiedName/] |

[/for]<drop/>

</table><drop/>

[else][/if]<drop/>

[/for]<drop/>

## Associations

[for (as:Association | self.eAllContents(Association))]<drop/>

### [as.name/]

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

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

[/for]<drop/>

Applied stereotypes:

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

* [st.name/]

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

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

[/if] <drop/>

[/for]<drop/>

[/for]<drop/>

[if as.memberEnd->notEmpty()]<drop/>

<table><drop/>

| **Attribute Name** | **Aggreg.** | **Navig.** | **Type** | **Mult.** | **Description** |
| --- | --- | --- | --- | --- | --- |

[for (p:Property|as.memberEnd)]<drop/>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [p.name/] | [p.aggregation/] | [if (p.isNavigable())]<drop/>  Navig.[else]<drop/>  Not navig. [/if] | [if (not (p.type.name.oclIsUndefined()))]<drop/>  [p.type.name/]  [/if] | [if(p.lower=p.upper)]1[else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [if (p.ownedComment->notEmpty())]<drop/>  [for (c:Comment | p.ownedComment)]<drop/>  [c.\_body.cleanAndFormat()/]  [/for] <drop/>  [/if] |

[/for]<drop/>

</table><drop/>

[else][/if]<drop/>

[/for]<drop/>

## Data Types

[for (dt:DataType | self.eAllContents(DataType)->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 (oa:Property|st.ownedAttribute)]<drop/>

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

[/if] <drop/>

[/for]<drop/>

[/for]<drop/>

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

<table><drop/>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Mult.** | **Access** | **Stereotypes** | **Description** |

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| [p.name/] | [p.type.name/] | [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('attribute')]AVC: [p.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]   [else]<drop/>   * [if oa.name.contains('isInvariant')]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).clean()/][else] no range constraint [/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).clean()/][else] <drop/> [/if]   [else]<drop/>   * [if oa.name.contains('passedByRef')] [if (not p.getValue(st, oa.name).oclIsUndefined())][p.getValue(st, oa.name).oclAsType(Boolean)/][else] undefined [/if]   [else]<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] |

[/for]<drop/>

</table><drop/>

[else][/if]<drop/>

[else][/if]<drop/>

[/for]<drop/>

## Enumerations

[for (dt:DataType | self.eAllContents(DataType)->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/>

</gendoc><drop/>