-

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.

<config services=’TagFileBuffer’>  
<output path=’C:\Users\ks0567\gendoc\OnapVnfModel.docx' />

</config>

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=’C:\Users\ks0567\git\onap-modeling\ONAP Information Model\Vnf.notation' element=’{0}’ importedBundles='gmf;papyrus' />

<gendoc><drop/>

This document was generated on 17 April 2018 by “ONAP Model GenDoc Template” version 1.0

[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=’C:\Users\ks0567\git\onap-modeling\ONAP Information Model\Vnf**.**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.ownedAttribute->notEmpty()]<drop/>

<table><drop/>

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

[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('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]   [/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/>

[/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/>

[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/>

## Primitives

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

## Interfaces

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

### [it.name/]

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

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

[/for]<drop/>

Applied stereotypes:

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

* [st.name/]

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

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

[/if] <drop/>

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

## Operations

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

### [op.name/]

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

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

[/for]<drop/>

Applied stereotypes:

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

* [st.name/]

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

* [if (not oa.name.contains('base'))][oa.name/]:[if (not op.getValue(st, oa.name).oclIsUndefined())][if oa.name.contains('isOperation')][op.getValue(st, oa.name).oclAsType(Boolean)/][else][if oa.name.contains('condition')][op.getValue(st, oa.name).oclAsType(String)/][else][op.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/][/if][/if]

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

[/for]<drop/>

[if op.ownedParameter->notEmpty()]<drop/>

<table><drop/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter Name** | **Type** | **Direction** | **Multi.** | **Description** |

[for (p:Parameter|op.ownedParameter)]<drop/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [p.name/] | [p.type.name/] | [p.direction/] | [if(p.lower=p.upper)]1[else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/]  [/for] |

[/for]<drop/>

</table><drop/>

[else][/if]<drop/>

[/for]<drop/>

</gendoc><drop/>