# Appendix III (informative): ITU-T G.8052 Information Model

All the text in red is comments and won’t appear in the output. Do not remove the drop EOL. <drop/>

2 variables are needed: <drop/>

* **model\_name**: only change the default if you rename the papyrus model file <drop/>
* **model\_version**: align if needed to the current version <drop/>

Only change text in brown below. <drop/>

By default, the generated documentation will be stored in the doc folder in the project directory to store the generated documentation. <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/>

<drop/>

<config>

<param key=**’model\_name’** value==’ G.8052\_v2.07-model.uml' />

<param key=**’model\_version’** value==’ v2.07' />

<**output path**=’${project\_loc}/doc/G.8052-${model\_version}-DD.docx' />

</config>

<context model='${project\_loc}/${model\_name}' element=’{0}’ importedBundles='gmf;papyrus' searchMetamodels='true'/>

This informative annex contains the result of the generation of the NFV Information model using gendoc.

Add specific text related to your contribution here that you would like to see in the output document <drop/>

<gendoc><drop/>

[for (p:Package | self.ownedElement->filter(Package)->sortedBy(name))]<drop/>

## [p.name/]

[p.displayInfo(3)/]

[for (p2 :Package| p.ownedElement->filter(Package)->sortedBy(name))]<drop/>

### [p2.name/]

[p2.displayInfo(4)/]

[for (p3 :Package| p2.ownedElement->filter(Package)->sortedBy(name))] <drop/>

#### [p3.name/]

[p3.displayInfo(5)/]

[for (p4 :Package| p3.ownedElement->filter(Package)->sortedBy(name))] <drop/>

##### [p4.name/]

[p4.displayInfo(6)/]

[for (p5 :Package| p4.ownedElement->filter(Package)->sortedBy(name))] <drop/>

###### [p5.name/]

[p5.displayInfo(7)/]

[for (p6 :Package| p5.ownedElement->filter(Package)->sortedBy(name))] <drop/>

[p6.name/]

[p6.displayInfo(8)/]

[for (p7 :Package| p6.ownedElement->filter(Package)->sortedBy(name))] <drop/>

[p7.name/]

[p7.displayInfo(9)/]

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

</gendoc><drop/>

# Display text with headers procedure <drop/>

Display text using the header of the level provided <drop/>

<fragment name=’displayText’ importedBundles=’commons;gmf;papyrus’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’text1’ type=’String’/><drop/>

<arg name=’text2’ type=’String’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[if (level=2)]<drop/>

## [text1/] [text2/]

[else] [if (level =3)]<drop/>

### [text1/] [text2/]

[else] [if (level =4)]<drop/>

#### [text1/] [text2/]

[else] [if (level =5)]<drop/>

##### [text1/] [text2/]

[else] [if (level =6)]<drop/>

###### [text1/] [text2/]

[else] [if (level =7)]<drop/>

[text1/] [text2/]

[else] [if (level =8)]<drop/>

[text1/] [text2/]

[else] [if (level =9)]<drop/>

[text1/] [text2/]

[else]<drop/>

**[text1/] [text2/]**

[/if] [/if] [/if] [/if] [/if] [/if] [/if] [/if]<drop/>

</fragment><drop/>

# Display single element attributes procedure <drop/>

Display the attributes of a single element (class, notification, datatype). <drop/>

Element is the element for which attributes will be displayed. <drop/>

Base is the original class. If B inherits from A, to display all attributes of B, <drop/>

the fragment will be called twice: first element=base=B, element=A, base=B <drop/>

<fragment name=’displaySingleElementAttributes’ importedBundles=’commons;gmf;papyrus’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’base’ type=’uml::Element’/><drop/>

**[for (p:uml::Property|element.eContents()->filter(uml::Property))]<drop/>**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [p.name/] | [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] | [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/]  [/for] | [if not(element.getId()=base.getId())]<drop/>  Inherited from [element.getText()/]  [/if]<drop/>  [for (st:Stereotype | p.getAppliedStereotypes()->sortedBy(name))]<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('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).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]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<drop/>  [/for]<drop/> |

**[/for]<drop/>**

</fragment><drop/>

# Display comments procedure <drop/>

Display all the comments of the element <drop/>

<fragment name=’displayComments’ importedBundles=’commons;gmf;papyrus’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

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

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

[/for]<drop/>

</fragment><drop/>

# Display stereotypes procedure <drop/>

Display a list of stereotypes <drop/>

Only for simple stereotypes, not for classes and notifications <drop/>

<fragment name=’displayStereotypes’ importedBundles=’commons;gmf;papyrus’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

[if (element.getAppliedStereotypes()->notEmpty())]<drop/>

**Applied Stereotypes:**

[for (st:Stereotype | element.getAppliedStereotypes()->sortedBy(name))]<drop/>

* [st.name/]

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display all attributes procedure <drop/>

Display all the attribute information in a class, a notification or a datatype<drop/>

It includes attributes from parents. Only 2 levels of parent supported <drop/>

<fragment name=’displayAllAttributes’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displaySingleElementAttributes'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

[if element.eContents()->filter(uml::Property)->notEmpty()]<drop/>

Table Attributes for [element.getText()/]

<table> <drop/>

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

[element.displaySingleElementAttributes(element) /]<drop/>

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

[for (parent:uml::Classifier | element.oclAsType(uml::Classifier).general->asSequence())]<drop/>

[if (not(parent.oclIsUndefined()))]<drop/>

[parent.displaySingleElementAttributes(element) /]<drop/>

[/if]<drop/>

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

[for (parent2:uml::Classifier | parent.oclAsType(uml::Classifier).general->asSequence())]<drop/>

[if (not(parent2.oclIsUndefined()))]<drop/>

[parent2.displaySingleElementAttributes(element) /]<drop/>

[/if]<drop/>

[/for] <drop/>

[/if]<drop/>

[/for] <drop/>

[/if]<drop/>

</table><drop/>

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

Table Attributes for [element.getText()/]

<table> <drop/>

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

[for (parent:uml::Classifier | element.oclAsType(uml::Classifier).general->asSequence())]<drop/>

[if (not(parent.oclIsUndefined()))]<drop/>

[parent.displaySingleElementAttributes(element) /]<drop/>

[/if]<drop/>

[if (parent.oclAsType(uml::Classifier).general->notEmpty())]

[for (parent2:uml::Classifier | parent.oclAsType(uml::Classifier).general->asSequence())]<drop/>

[if (not(parent2.oclIsUndefined()))]<drop/>

[parent2.displaySingleElementAttributes(element) /]<drop/>

[/if]<drop/>

[/for] <drop/>

[/if]<drop/>

[/for] <drop/>

[else]<drop/>

[/if]<drop/>

</table><drop/>

[/if]<drop/>

</fragment><drop/>

# Display package procedure <drop/>

Display all the information of a package: QName, comments, stereotypes <drop/>

<fragment name=’displayPackage’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayStereotypes'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

**Qualified Name:** [element.oclAsType(Package).qualifiedName/]

[element.displayComments()/]<drop/>

[element.displayStereotypes()/]<drop/>

</fragment><drop/>

# Display classes procedure <drop/>

Display all the information in a class: comments, stereotypes and attributes <drop/>

<fragment name=’displayClasses’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayAllAttributes;displayText'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[if element.ownedElement->filter(Class)->notEmpty()]<drop/>

[element.displayText(‘Classes’, ‘’,level)/]<drop/>

[for (cl:uml::Class | element.ownedElement->filter(Class) ->sortedBy(name))]<drop/>

[element.displayText(cl.name, ‘class’, level+1)/]<drop/>

**Qualified Name:** [cl.qualifiedName/]

[cl.displayComments()/]<drop/>

[if (cl.isAbstract)]<drop/>

**Abstract class**

[/if]<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:uml::Stereotype | cl.getAppliedStereotypes()->sortedBy(name))]<drop/>

* [st.name/]

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

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

[/if] <drop/>

[/for]<drop/>

[/for]<drop/>

[cl.displayAllAttributes()/]

[/for] <drop/>

[/if]<drop/>

</fragment><drop/>

# Display notifications procedure <drop/>

Display all the information in a notification: comments, stereotypes and attributes <drop/>

<fragment name=’displayNotifications’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayAllAttributes;displayText'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[if element.ownedElement->filter(Signal)->notEmpty()]<drop/>

[element.displayText(‘Notifications’, ‘’,level)/]<drop/>

[for (cl:uml::Signal | element.ownedElement->filter(Signal) ->sortedBy(name))]<drop/>

[element.displayText(cl.name, ‘notification’, level+1)/]<drop/>

**Qualified Name:** [cl.qualifiedName/]

[cl.displayComments()/]<drop/>

**Applied Stereotypes:**

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

* [st.name/]

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

[if (not oa.name.contains('base'))]<drop/>

* + [oa.name.clean()/]: [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('support’)] [cl.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/]

[else]

[if oa.name.contains('triggerConditionList')]<drop/>

[for (s : String | Sequence{cl.getValue(st, oa.name)}->flatten().oclAsType(String))]<drop/>

* [s.clean()/]

[/for]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

[/for]<drop/>

[cl.displayAllAttributes()/]

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display datatypes procedure <drop/>

Display all the information in datatypes: comments, stereotypes and attributes <drop/>

Valid for primitive, enumeration and standard datatypes <drop/>

<fragment name=’displayDatatypes’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayAllAttributes;displayText;displayStereotypes'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[if element.ownedElement->filter(DataType)->notEmpty()]<drop/>

[element.displayText(‘Datatypes’, ‘’,level)/]<drop/>

[for (dt:uml::DataType | element.ownedElement->filter(DataType) ->sortedBy(name))]<drop/>

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

[element.displayText(dt.name, ‘datatype’, level+1)/]<drop/>

**Qualified Name:** [dt.qualifiedName/]

[dt.displayComments()/]<drop/>

[dt.displayStereotypes()/]<drop/>

[dt.displayAllAttributes()/]

[else][/if] <drop/>

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

[element.displayText(dt.name, ‘enumeration’, level+1)/]<drop/>

**Qualified Name:** [dt.qualifiedName/]

[dt.displayComments()/]<drop/>

[dt.displayStereotypes()/]<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]

[else][/if] <drop/>

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

[element.displayText(dt.name, ‘primitive type’, level+1)/]<drop/>

**Qualified Name:** [dt.qualifiedName/]

[dt.displayComments()/]<drop/>

[dt.displayStereotypes()/]<drop/>

[else][/if]<drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display associations procedure <drop/>

Display all the information in an association: comments, stereotypes and attributes <drop/>

<fragment name=’displayAssociations’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayText;displayStereotypes’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[if (element.ownedElement->filter(Association)->notEmpty())]<drop/>

[element.displayText(‘Associations’, ‘’,level)/]<drop/>

[for (as:uml::Association | element.ownedElement->filter(Association) ->sortedBy(name))]<drop/>

[element.displayText(as.name, ‘association’, level+1)/]<drop/>

**Qualified Name:** [as.qualifiedName/]

[as.displayComments()/]<drop/>

[as.displayStereotypes()/]<drop/>

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

Table Member ends for [as.getText()/]

<table> <drop/>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Attribute Name | Aggreg. | Navig. | Mult. | Type | Description | Applied Stereotypes |

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

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

**[/for]<drop/>**

</table><drop/>

[else][/if]<drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display diagrams procedure <drop/>

Display all the diagrams in a package<drop/>

<fragment name=’displayDiagrams’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayText’><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[if element.getPapyrusDiagrams()->notEmpty()]<drop/>

[element.displayText(‘Diagrams’, ‘’,level)/]<drop/>

[for (d:Diagram|element.getPapyrusDiagrams())]<drop/>

Figure Diagram [d.name.clean()/]

<drop/>

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

</image><drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>

# Display info procedure <drop/>

Display all the information in a package: comments, classes, notifications, datatypes and diagrams <drop/>

<fragment name=’displayInfo’ importedBundles=’commons;gmf;papyrus’ importedFragments='displayComments;displayClasses;displayNotifications;displayDatatypes;displayAssociations;displayDiagrams;displayPackage'><drop/>

<arg name=’element’ type=’uml::Element’/><drop/>

<arg name=’level’ type=’Integer’/><drop/>

[element.displayPackage()/]<drop/>

[element.displayDiagrams(level)/]<drop/>

[element.displayClasses(level)/]<drop/>

[element.displayNotifications(level)/]<drop/>

[element.displayDatatypes(level)/]<drop/>

[element.displayAssociations(level)/]<drop/>

</fragment><drop/>