

TAPI UML Model

Version 2.5

ONF Document Type: Technical Recommendation

**Disclaimer**

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

Any marks and brands contained herein are the property of their respective owners.

Open Networking Foundation  
1000 El Camino Real, Suite 100, Menlo Park, CA 94025  
[www.opennetworking.org](http://www.opennetworking.org)

©2023 Open Networking Foundation. All rights reserved.

Open Networking Foundation, the ONF symbol, and OpenFlow are registered trademarks of the Open Networking Foundation, in the United States and/or in other countries. All other brands, products, or service names are or may be trademarks or service marks of, and are used to identify, products or services of their respective owners.

Table of Contents

[Disclaimer 2](#_Toc72086477)

[Document History 7](#_Toc72086478)

[1 Common Model 8](#_Toc72086479)

[1.1 Diagrams 8](#_Toc72086480)

[1.2 Classes 9](#_Toc72086481)

[1.2.1 [cl.name/] 9](#_Toc72086482)

[1.3 Data Types 9](#_Toc72086483)

[1.3.1 [dt.name/] 9](#_Toc72086484)

[1.4 Enumerations 10](#_Toc72086485)

[1.4.1 [dt.name/] 10](#_Toc72086486)

[1.5 Primitives 11](#_Toc72086487)

[1.5.1 [dt.name/] 11](#_Toc72086488)

List of Figures

[Figure 1 – *[d.name/]* 6](#_Toc72086489)

List of Tables

[Table 1 Attributes for Class [cl.name/] 11](#_Toc122427728)

Document History

| **Version** | **Date** | **Description of Change** |
| --- | --- | --- |
| 2.3 | May 27, 2021 | Model Dump  *Gendoc generates documentation from Eclipse Modeling Framework (EMF) models using document templates in formats such as OpenOffice Writer (.odt), Microsoft Word (.docx), Microsoft Excel (.xlsx) and Microsoft Powerpoint (.pptx).* |
| 2.4.0 | December 2022 | See high level diff document in Github |
| 2.4.1 | March 2023 | See high level diff document in Github |
| 2.5 | October 2023 | See high level diff document in Github |

# Path Computation Model

<config services=’TagFileBuffer’><drop/>

<config>

<param key='model\_path' value='C:\Users\amazzini\ONF-TAPI-2.x\TAPI\UML\'/>

<param key='model\_name' value='TapiPathComputation'/>

</config>

<output path=’${model\_path}\GenDocTemplates\output\${model\_name}.docx' /><drop/>  
</config> <drop/>

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

<bookmarks>

<alias source=’ as\_name\_bookmark’ target=’ A[as.getId()/]’ />

<alias source=’ cl\_name\_bookmark’ target=’ B[cl.getId()/]’ />

<alias source=’ dt\_name\_bookmark’ target=’ C[dt.getId()/]’ />

<alias source=’ en\_name\_bookmark’ target=’ D[dt.getId()/]’ />

</bookmarks>

<gendoc><drop/>

[for (p:Package|Package.allInstances())][for (c:Comment| p.ownedComment)]

[if (not p.name.contains('Diagrams'))][**p.name/**]: [c.\_body/][else]<drop/>[/if][/for] [/for]

</gendoc><drop/>

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

<gendoc><drop/>

## Diagrams

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

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

</image><drop/>

**Figure 1 – Diagram *[d.name/]***

[/for]<drop/>

</gendoc><drop/>

<context model=’${model\_path}${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/>

Description:

* <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.allAttributes()->notEmpty()]<drop/>

<table><drop/>

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

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

| [p.name/]  [if p.qualifiedName.contains(cl.name)][else]Inherited: *[p.qualifiedName/]*[/if][if p.association.oclIsUndefined()][else]*Navigable association end of:* [*[p.association.name/*]](#A[p.association.getId()/])[/if] | [if p.type.qualifiedName.contains(‘Path’)][if p.type.oclIsTypeOf(DataType)][[p.type.name/]](#C[p.type.getId()/])[else][if p.type.oclIsTypeOf(Enumeration)][[p.type.name/]](#D[p.type.getId()/])[else][if p.type.oclIsTypeOf(Class)][[p.type.name/]](#B[p.type.getId()/])[/if][/if][/if][else][p.type.qualifiedName/][/if]  [if (not(p.default.oclIsUndefined()))]<drop/>  Default value: *[p.default/]*  [/if]<drop/> | [if(p.lower=p.upper)] [p.lower/] [else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [if(p.isReadOnly)]R[else]RW[/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('protobuf')][if p.getValue(st, oa.name).oclAsType(Integer)=0][else]   * Protobuf Index: [p.getValue(st, oa.name).oclAsType(Integer)/][/if]   [else]<drop/>   * [if oa.name.contains('isInvariant')]isInvariant: [p.getValue(st, oa.name).oclAsType(Boolean)/]   [else]<drop/>   * [if oa.name.contains('partOfObjectKey')]isKey:[if (p.getValue(st, oa.name).oclAsType(Integer))>0] yes – part: [p.getValue(st, oa.name).oclAsType(Integer)/] [else] No [/if]   [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/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<drop/>  [/for]<drop/> |
| --- | --- | --- | --- | --- |
| **Description:**  [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/][/for] | | | |

[/for]<drop/>

**Table 1 – Attributes for class *[cl.name/]***

</table><drop/>

[else][/if]<drop/>

[/for]<drop/>

## Signals

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

### [si.name/]

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

Description:

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

[/for]<drop/>

Applied stereotypes:

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

* [st.name/]

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

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

[/if] <drop/>

[/for]<drop/>

[/for]<drop/>

[if si.allAttributes()->notEmpty()]<drop/>

<table><drop/>

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

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

| [p.name/]  [if p.qualifiedName.contains(si.name)][else]Inherited: *[p.qualifiedName/]*[/if] | [if p.type.qualifiedName.contains(‘Path’)][if p.type.oclIsTypeOf(DataType)][[p.type.name/]](#C[p.type.getId()/])[else][if p.type.oclIsTypeOf(Enumeration)][[p.type.name/]](#D[p.type.getId()/])[else][if p.type.oclIsTypeOf(Class)][[p.type.name/]](#B[p.type.getId()/])[/if][/if][/if][else][p.type.qualifiedName/][/if]  [if (not(p.default.oclIsUndefined()))]<drop/>  Default value: *[p.default/]*  [/if]<drop/> | [if(p.lower=p.upper)] [p.lower/] [else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] | [if(p.isReadOnly)]R[else]RW[/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('protobuf')][if p.getValue(st, oa.name).oclAsType(Integer)=0][else]   * Protobuf Index: [p.getValue(st, oa.name).oclAsType(Integer)/][/if]   [else]<drop/>   * [if oa.name.contains('isInvariant')]isInvariant: [p.getValue(st, oa.name).oclAsType(Boolean)/]   [else]<drop/>   * [if oa.name.contains('partOfObjectKey')]isKey:[if (p.getValue(st, oa.name).oclAsType(Integer))>0] yes – part: [p.getValue(st, oa.name).oclAsType(Integer)/] [else] No [/if]   [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/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<drop/>  [/for]<drop/> |
| --- | --- | --- | --- | --- |
| **Description:**  [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/][/for] | | | |

[/for]<drop/>

</table><drop/>

[else][/if]<drop/>

[/for]<drop/>

## Associations

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

### [as.name/]

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

Description:

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

[/for]<drop/>

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

Applied stereotype:

* <dropEmpty>[st.name/]</dropEmpty>

[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/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Association end role name** | **Aggreg. type** | **Navigable** | **Target Class** | **Mult.** |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [p.name/] | [p.aggregation/] | [if (p.isNavigable())]Yes[else]No[/if] | [if (not (p.type.name.oclIsUndefined()))][if p.type.qualifiedName.contains(‘Path’)][[p.type.name/]](#B[p.type.getId()/])[else][p.type.qualifiedName/][/if][/if] | [if(p.lower=p.upper)]1[else][p.lower/]..[if(p.upper=-1)]\*[else][p.upper/][/if][/if] |

[/for]<drop/>

**Table 1 – Member ends for association *[as.name/]***

</table><drop/>

[else][/if]<drop/>

[/for]<drop/>

## Abstractions

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

### [ab.name/]

[if ab.getAppliedStereotypes()->notEmpty()]

<table><drop/>

|  |  |  |
| --- | --- | --- |
| **Augmenting Class** | **Augmented Class** | **Comment** |
| [[ab.client.name/]](#B[ab.client.getId()/]) | [ab.supplier.qualifiedName/] | [for (co:Comment | ab.ownedComment)]<drop/>  <dropEmpty>[co.\_body.clean()/]</dropEmpty>  [/for]<drop/> |
| [for (st:Stereotype | ab.getAppliedStereotypes())]<drop/>  [for (oa:Property|st.ownedAttribute)]<drop/>  [if oa.name.contains('target')][oa.name/]: “[ab.getValue(st, oa.name).oclAsType(OclAny)/]” [else]<drop/>[/if]  [/for]<drop/>  [/for]<drop/> | | |

**Table 1 – Member ends for class abstraction *[ab.name/]***

</table><drop/>

[else]

<table><drop/>

|  |  |
| --- | --- |
| **Augmenting Enumeration** | **Augmented Enumeration** |
| [ab.client.name/]  [for (e:EnumerationLiteral|ab.client.oclAsType(Enumeration).ownedLiteral->sortedBy(name))]<drop/>   * [e.name/]   [/for]<drop/> | [ab.supplier.name/]  [for (e:EnumerationLiteral|ab.supplier.oclAsType(Enumeration).ownedLiteral->sortedBy(name))]<drop/>   * [e.name/]   [/for]<drop/> |
| **Comment**  [for (co:Comment | ab.ownedComment)]<drop/>  <dropEmpty>[co.\_body.clean()/]</dropEmpty>  [/for]<drop/> | |

**Table 1 – Member ends for enum abstraction *[ab.name/]***

</table>[/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/>

Description:

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

[/for]<drop/>

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

Applied stereotype:

* <dropEmpty>[st.name/]</dropEmpty>

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

[/if] <drop/>

[/for]<drop/>

[/for]<drop/>

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

<table><drop/>

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

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

| [p.name/]  [if p.qualifiedName.contains(dt.name)][else]Inherited: *[p.qualifiedName/]*[/if] | [if p.type.qualifiedName.contains(‘Path’)][if p.type.oclIsTypeOf(DataType)][[p.type.name/]](#C[p.type.getId()/])[else][if p.type.oclIsTypeOf(Enumeration)][[p.type.name/]](#D[p.type.getId()/])[else][if p.type.oclIsTypeOf(Class)][[p.type.name/]](#B[p.type.getId()/])[/if][/if][/if][else][p.type.qualifiedName/][/if]  [if (not(p.default.oclIsUndefined()))]<drop/>  Default value: *[p.default/]*  [/if]<drop/> | [if(p.lower=p.upper)] [p.lower/] [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('protobuf')][if p.getValue(st, oa.name).oclAsType(Integer)=0][else]   * Protobuf Index: [p.getValue(st, oa.name).oclAsType(Integer)/][/if]   [else]<drop/>   * [if oa.name.contains('isInvariant')]isInvariant: [p.getValue(st, oa.name).oclAsType(Boolean)/]   [else]<drop/>   * [if oa.name.contains('partOfObjectKey')]isKey:[if (p.getValue(st, oa.name).oclAsType(Integer))>0] yes – part: [p.getValue(st, oa.name).oclAsType(Integer)/] [else] No [/if]   [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/>  [/if]<drop/>  [/if]<drop/>  [/for]<drop/>  [/for]<drop/> |
| --- | --- | --- | --- | --- |
| **Description:**  [for (c:Comment | p.ownedComment)] <drop/>  [c.\_body.clean()/][/for] | | | |

[/for]<drop/>

**Table 1 – Attributes for data type *[dt.name/]***

</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/>

Description:

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

[/for]<drop/>

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

Applied stereotype:

* <dropEmpty>[st.name/]</dropEmpty>

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

[/if] <drop/>

[/for]<drop/>

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

* + Applied stereotype:
    - <dropEmpty>[st.name/]</dropEmpty>

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

* + - [if (not oa.name.contains('base'))][oa.name/]: [if (not e.getValue(st, oa.name).oclIsUndefined())][if oa.name.contains('condition')][e.getValue(st, oa.name).oclAsType(String)/] [else][if oa.name.contains('reference')][e.getValue(st, oa.name).oclAsType(String)/] [else][if oa.name.contains('protobuf')][if e.getValue(st, oa.name).oclAsType(Integer)<>0][e.getValue(st, oa.name).oclAsType(Integer)/] [/if][else][e.getValue(st, oa.name).oclAsType(EnumerationLiteral).name/][/if][/if][/if][else]<drop/>[/if]

[/if] <drop/>

[/for]<drop/>

[/for]<drop/>

[/for]<drop/>

[for (e1:EnumerationLiteral|dt.oclAsType(Enumeration).generalization.general.oclAsType(Enumeration).ownedLiteral)]<drop/>

* (Inherited from *[dt.oclAsType(Enumeration).generalization.general.oclAsType(Enumeration).name/]*) [e1.name/]

[/for]<drop/>

[for (e2:EnumerationLiteral|dt.oclAsType(Enumeration).generalization.general.oclAsType(Enumeration).generalization.general.oclAsType(Enumeration).ownedLiteral)]<drop/>

* (Inherited from *[dt.oclAsType(Enumeration).generalization.general.oclAsType(Enumeration).generalization.general.oclAsType(Enumeration).name/]*) [e2.name/]

[/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/>

Description:

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

[/for]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

</gendoc><drop/>