<config>  
<output path=’C:\Users\ndavis\git\OnfInfoModelOutput\ModelDescriptions\TR-512.14\_OnfCoreIm-Location.docx' />  
</config>

<context model=’C:\Users\ndavis\git\ONFInfoModel\OnfModel\Location.uml' element=’{0}’ importedBundles='gmf;papyrus' searchMetamodels='true'/>

<gendoc><drop/>

Change path substrings above from “{path for output files}\” to your local path for the output files and “{path for CoreModel}\” to your local path for the Core Model. <drop/>

DELETE: Prior to publishing this –gd.docx (including for review), change path substrings above from “C:\Users\ndavis\git\OnfInfoModelOutput\” to “{path for output files}\” and from “C:\Users\ndavis\git\ONFInfoModel\OnfModel\” to “{path for CoreModel}\” <drop/>



Core Information Model (CoreModel)

TR-512.14

Location

Version 1.6

January 2024

ONF Document Type: Technical Recommendation

ONF Document Name: Core Information Model version 1.6

**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)

©2024 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.

**Important note**

This Technical Recommendations has been approved by the Project TST, but has not been approved by the ONF board.  This Technical Recommendation is an update to a previously released TR specification, but it has been approved under the ONF publishing guidelines for ‘Informational’ publications that allow Project technical steering teams (TSTs) to authorize publication of Informational documents.  The designation of ‘-info’ at the end of the document ID also reflects that the project team (not the ONF board) approved this TR.

Finalizing this document once generated… delete this text prior to publication:

* Replace “{{..}}” with square brackets (which trip up Gendoc)
* Select text in document from beginning of table of contents (first line) to end of document
  + Click menu item “Update Field” (on this large block of text)
    - if “Update Table…” dialogue appears select “Update entire table”
  + Repeat “update fields” 2 more times (on the same large block of text)
    - if “Update Table…” dialogue appears select “Update entire table”
* Remove reviewer comment

Note that the table of contents and figures need to be updated several times as the table length changes the page numbering and the cross references will need to be re-updated.

Table of Contents

[Disclaimer 2](#_Toc456706149)

[Open Networking Foundation 2](#_Toc456706150)

[Document History 4](#_Toc456706151)

[1 Introduction 4](#_Toc456706152)

[2 Model Overview 5](#_Toc456706153)

[2.1 Model Focuses 5](#_Toc456706154)

[2.1.1 Core Network Model Overview 5](#_Toc456706155)

[2.1.2 Core Foundation Model 6](#_Toc456706156)

[2.1.3 Layered Topology 8](#_Toc456706157)

[2.2 Other documents 10](#_Toc456706158)

[2.3 Supporting Guidelines 10](#_Toc456706159)

[2.4 Key reference material 10](#_Toc456706160)

[3 Summary of main changes between version 1.1 and 1.2 11](#_Toc456706161)

[4 References 11](#_Toc456706162)

[5 Definitions 12](#_Toc456706163)

[5.1 Terms defined elsewhere 12](#_Toc456706164)

[5.2 Terms defined in this TR 12](#_Toc456706165)

[6 Abbreviations and acronyms 13](#_Toc456706166)

[7 Conventions 15](#_Toc456706167)

[7.1 Lifecycle Stereotypes 15](#_Toc456706168)

[7.2 Diagram Keys 15](#_Toc456706169)

[8 Future CoreModel areas 17](#_Toc456706170)

[9 Terminology Translation table 18](#_Toc456706171)

[10 Back matter 19](#_Toc456706172)

[10.1 Editors 19](#_Toc456706173)

[10.2 Contributors 19](#_Toc456706174)

List of Figures

[Figure 1-1 Methodology of IM and DS Development 6](#_Toc430780029)

Document History

| **Version** | **Date** | **Description of Change** |
| --- | --- | --- |
| 1.0 | September 2021 | Initial Version |
| 1.6 | January 2024 | Updated release and dates. |

# Introduction to the document suite

This document is an addendum to the TR-512 ONF Core Information Model and forms part of the description of the ONF-CIM. For general overview material and references to the other parts refer to [TR-512.1](../TR-512.1_OnfCoreIm-Overview.pdf).

## References

For a full list of references see [TR-512.1](../TR-512.1_OnfCoreIm-Overview.pdf).

## Definitions

For a full list of definition see [TR-512.1](../TR-512.1_OnfCoreIm-Overview.pdf).

## Conventions

See [TR-512.1](../TR-512.1_OnfCoreIm-Overview.pdf) for an explanation of:

* UML conventions
* Lifecycle Stereotypes
* Diagram symbol set

## Viewing UML diagrams

Some of the UML diagrams are very dense. To view them either zoom (sometimes to 400%) or open the associated image file (and zoom appropriately) or open the corresponding UML diagram via Papyrus (for each figure with a UML diagram the UML model diagram name is provided under the figure or within the figure).

## Understanding the figures

Figures showing fragments of the model using standard UML symbols as well as figures illustrating application of the model are provided throughout this document. Many of the application-oriented figures also provide UML class diagrams for the corresponding model fragments (see [TR-512.1](../TR-512.1_OnfCoreIm-Overview.pdf) for diagram symbol sets). All UML diagrams depict a subset of the relationships between the classes, such as inheritance (i.e. specialization), association relationships (such as aggregation and composition), and conditional features or capabilities. Some UML diagrams also show further details of the individual classes, such as their attributes and the data types used by the attributes.

# Introduction to Location

The focus of this document is on concepts that relate to ‘where’ something is.

This topic is a common one, but unlike party, there are no good open models that can be leveraged.

One approach that could be used in a model is just to have location (site) name and address attributes spread throughout the model as required. The problem is that name is often not a good identifier, address formats can be quite complex and different spellings can make it hard to compare them, and then updating the data will become very complex.

The best approach is to factor out all location related information into a separate set of classes and to reference them from the rest of the model as required, and this is the approach taken in this document.

In the past, most location information was textual based, but now a lot more information systems also support map-based locations, and our model needs to support this too.

Another complication is that locating things ‘internally’ to a building is often done differently from ‘outside’ locations, so this model will cover both these cases.

This model covers the most common alternatives for external location:

* (named) site
* (postal) Address
* (Geospatial) position (such as GPS coordinates)

For internal locations, the model will cover:

* Local address (such as a bin location in a warehouse)
* (local) position in a cartesian reference system (often relative to an internal reference point, and more accurate than GPS)

Note also that this document doesn’t propose a full, robust enterprise grade location model as it is targeted at supporting a network management environment only.

Also, the lack of a single standard for postal and internal address formats precludes a simple solution, so the model focusses on providing a framework that can be augmented as required.

The model only provides simple internal and geographic address classes and is designed to be easily extended using the augments / decorates pattern, for example by adding further geographic geometries or different geographic address formats.

A data dictionary that sets out the details of all classes, data types and attributes is also provided ([TR-512.DD](TR-512.DD_OnfCoreIm-DataDictionary.pdf)).

# Location model detail

[for(p:Package|Package.allInstances())]<drop/>

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘Location’, ’Location Model’)/]

[/for]<drop/>

## Location Model

….

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

[if (cl.qualifiedName.contains(‘Location’))]<drop/>

### [cl.name/]

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[cl.insertClass(cl.name,’Location’)/]

Inserts the attributes of the class <drop/>   
[cl.insertAttributeTableBrief ()/]

[/if]<drop/>

[/for]<drop/>

## Further detail

To link the Location model into the CIM core, it makes sense to decouple the network function classes from the Location model. The network functions deliberately don’t have an abstract parent (for modularity reasons) which means that the best approach is to link the inventory model to the Location model via ConstraintDomain. This decouples the two modules (reducing the number of associations) and allows inventory items to be grouped into a ConstraintDomain before relating them to a Location, reducing the number of association *instances* to be managed and also giving more sensible semantics.

Linking local and geographic locations is a common need, and roles don't really play a factor, so a special association “LocalLocationWithinGeographicLocation” is provided for that case, simplifying the number of instances required.

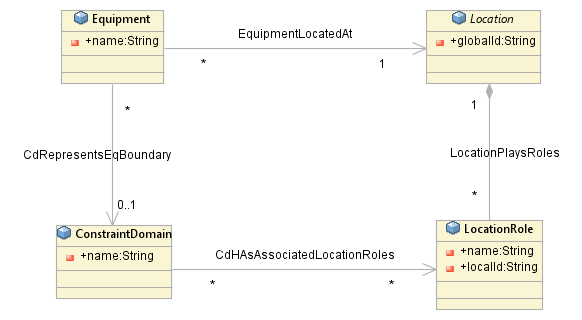


Figure 3‑2 Equipment Location Association

Locating Equipment is a special case that deserves an optimized solution and for that case an association EquipmentLocatedAt is provided.

# Location model examples

## Site Contact

A site contact is the person who would be contacted to gain access to a Site.

We could create a special PartyRole of “site-contact” or we could just allow all employees to be site contacts.

The example below shows the latter option.

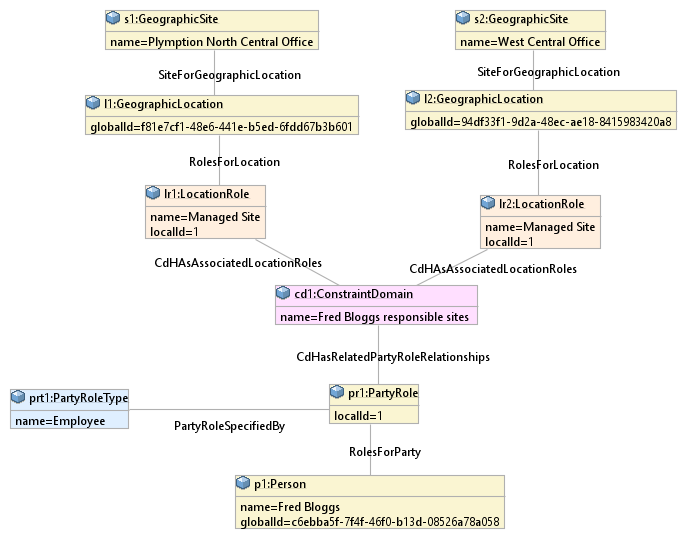


Figure 4‑1 Simple Site Contact Example

If many people play roles in relation to the group of sites, then the ConstraintDomain should be made more generic, say to “Adelaide North Central Offices” and then specific site related roles can be used. This is better than creating and maintaining many equivalent groupings.

Below is shown another option.

Here, we have defined primary and secondary site contact Party roles.

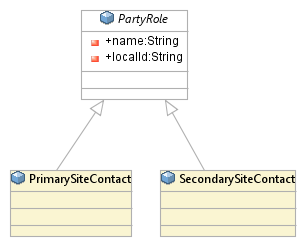


Figure 4‑2 Site Contact Party Roles

So now we can show that Irma is the primary contact and Fred is the secondary contact for the “Adelaide North Central Offices” group of Locations.

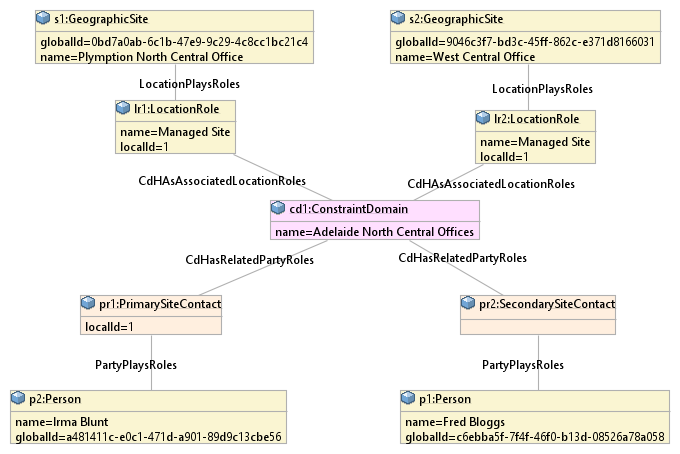


Figure 4‑3 Primary and Secondary Site Contact Example

## Global and Local Location Options

The global and local locations have been deliberately decoupled, so that they can be 'mixed and matched' as required.

This allows for various options, some of which are shown below.

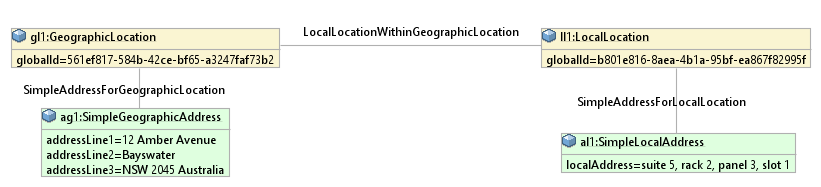


Figure 4‑4 Global and Local Address

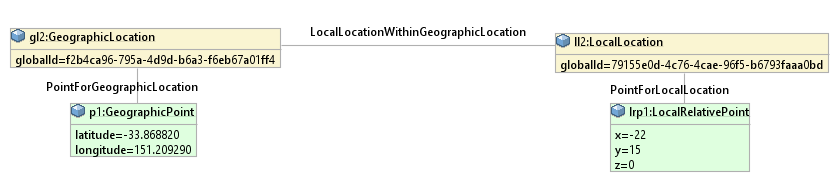


Figure 4‑5 Global and Local points

In the example below, we have a GeographicSite that is related to an address and a point on a map.

Two LocalPositions are defined in the Site, one is a local reference point and the other is a local point measured relative to that local reference point.

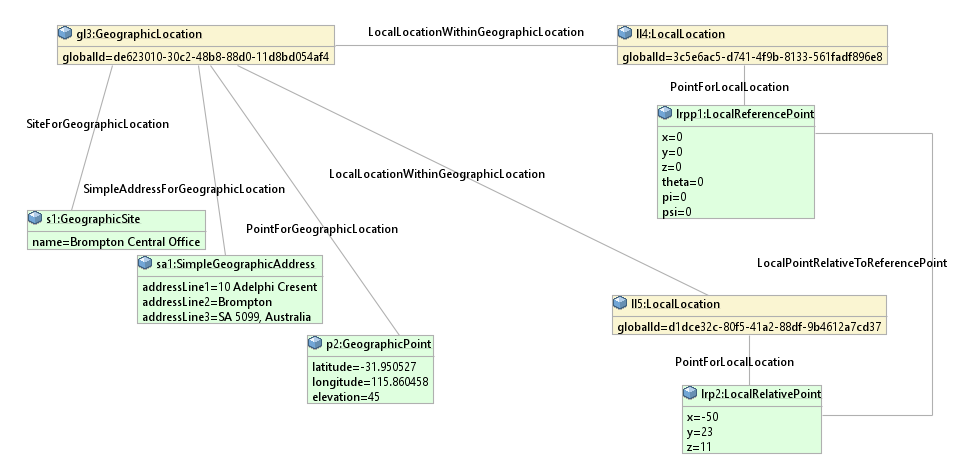


Figure 4‑6 Site with local locations

## Device Location

For Equipment, a special association is provided so that its representation can be optimized (without needing to use role instances).

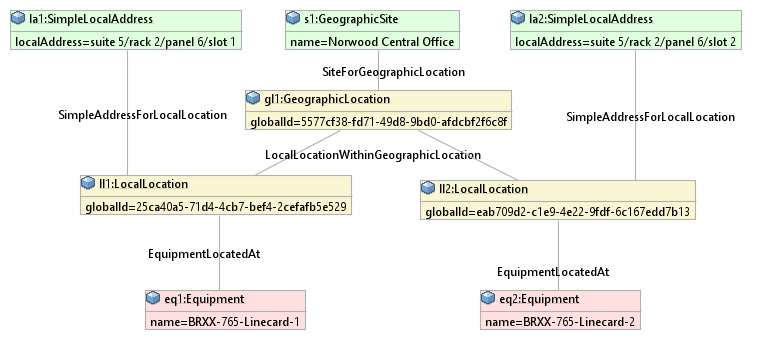


Figure 4‑7 Device Location Example

Note that for SimpleLocalAddress, we are just using a single string with some sort of structure and delimiters.

If required, a more complex local address, such as one that had fields for each level in the local address could be added as another decorator of LocalLocation. The issue is that the naming is likely to be company specific and may not follow simple rules, so it can't really be added to this document.

## Wifi Heat Map

A common task is to be able to show a WiFi ‘heat map”.

In the diagram below, assume that the grid is a 1 meter spacing and that the local reference point is at the bottom left of the diagram and that the Y axis is aligned with north and the Z axis point straight up.

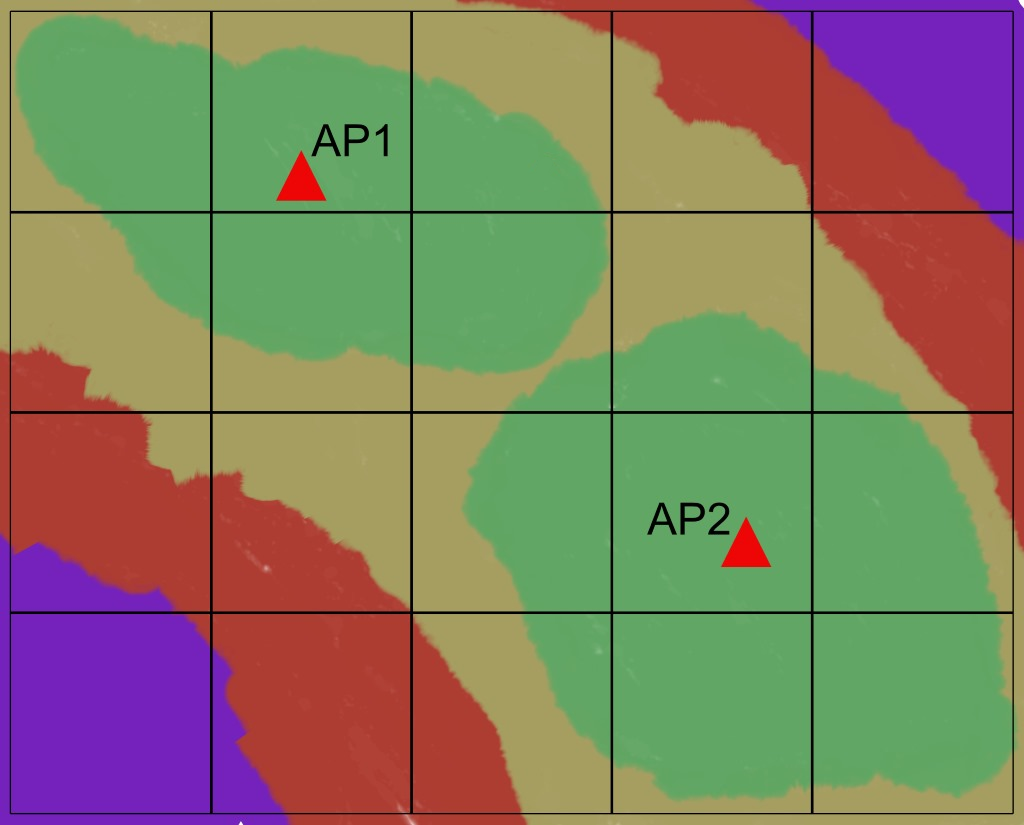


Figure 4‑8 Heat Map Example

For this example, assume that the reference point is at floor level on the fifth floor of the building, and the access points are mounted on the ceiling (which is 4.0m from the floor).

Note that in this example, a structured local address of the form suite/rack/panel/slot is not appropriate, a relative position is the best way of representing where the APs are.

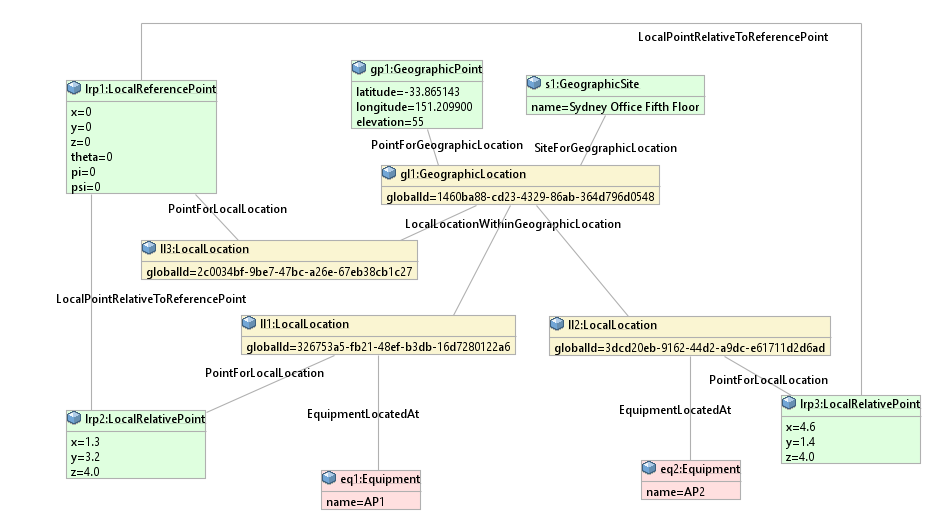


Figure 4‑9 Heat Map Instance Diagram

## Complex Address

As well as the current simple address, the model could be extended by adding a more detailed address class as a subclass of GeographicAddress.

For example, in MEF standard 57.1, they define a Fielded address with attributes

* Fielded Address Identifier
* Street Number
* Street Number Suffix
* Street Number Last
* Street Number Suffix Last
* Street Name
* Street Type
* Street Suffix
* Locality
* City
* Postal Code
* Postal Code Extension
* State Or Province
* Country
* Sub Unit List
* Level Type
* Level Number
* Building Name
* Private Street Number
* Private Street Name

As well as a FormattedAddress (which is similar to our SimpleGeographicAddress) with attributes

* Formatted Address Identifier
* Locality
* City
* Postal Code
* Postal Code Extension
* State Or Province
* Country
* Address Line 1
* Address Line 2

Another definition of a complex address can be found in IETF RFCs

* RFC 5774
* RFC 5139
* RFC 5491
* RFC 7459
* RFC4776

**End of Document**

</gendoc><drop/>

To take latest template: <drop/>

* delete text from “Template version…” to end of file <drop/>
* insert a line in “Normal” style<drop/>
* insert text (Insert 🡪 Object 🡪 Text from File… (alt njf)) from: <drop/>

TR-512.GT\_OnfCoreIm-CommonGendocTemplate-Fragments.docx <drop/>

Template version 0.0.11 1 June 2018 <drop/>

# Fragment: Insert class <drop/>

<fragment name=’insertClass’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’cl’ type=’uml::Class’/><drop/>  
<arg name=’className’ type=’String’/><drop/>  
<arg name=’packageName’ type=’String’/><drop/>  
[if (not cl.qualifiedName.contains(packageName))]<drop/>  
[else] <drop/>  
[if(cl.name.contains(className))]<drop/>

Qualified Name: [cl.qualifiedName/]

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

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

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

[/for]<drop/>

[else]To be provided

[/if]<drop/>

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

This class is abstract.

[/if]<drop/>

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

Inherits properties from:

[for (gen:Class | cl.oclAsType(uml::Class).general)]<drop/>

* [gen.name/]

[/for]<drop/>

[/if]<drop/>

[for (st:Stereotype | cl.getAppliedStereotypes())]<drop/>  
[if(not st.name.contains(‘OpenModelClass’))]<drop/>

This class is [st.name/].

[else] <drop/>  
[/if]<drop/>  
[/for]<drop/>  
[else] <drop/>  
[/if]  
[/if]  
</fragment><drop/>

# Fragment: Insert standard diagram <drop/>

<fragment name=’insertStandardDiagram’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’p’ type=’uml::Package’/><drop/>  
<arg name=’diagramName’ type=’String’/><drop/>  
<arg name=’diagramTitle’ type=’String’/><drop/>

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

[if d.name.contains(diagramName)]

<drop/>

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

CoreModel diagram: [d.name/]

Figure 12-1 [diagramTitle/]

[else]<drop/>

[/if]<drop/>

[/for]<drop/>  
</fragment><drop/>

# Fragment: Insert small diagram <drop/>

<fragment name=’insertSmallDiagram’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’p’ type=’uml::Package’/><drop/>  
<arg name=’diagramName’ type=’String’/><drop/>  
<arg name=’diagramTitle’ type=’String’/><drop/>

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

[if d.name.contains(diagramName)]

<drop/>

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

CoreModel diagram: [d.name/]

Figure 13-1 [diagramTitle/]

[else]<drop/>

[/if]<drop/>

[/for]<drop/>  
</fragment><drop/>

# Fragment: Insert attribute row brief not Obsolete<drop/>

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

Does not work unless we have Mature stereotype… <drop/>  
<arg name=’p’ type=’uml::Property’/><drop/>

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

[if(not st.name.contains(‘OpenModelAttribute’))]

[if(not st.name.contains(‘Obsolete’))]

| [p.name/] | [for (st:Stereotype | p.getAppliedStereotypes())]<drop/>  [if(not st.name.contains(‘OpenModelAttribute’))] [st.name/]  [/if]<drop/>  [/for]<drop/>  Do NOT remove the previous line as word throws an error if the cell is empty <drop/> | [if p.ownedComment->notEmpty()]<drop/>  [for (c:Comment | p.ownedComment)] <drop/>  [cleanAndFormat(c.\_body.clean())/]  [/for]  [else] [if (p.name.contains (‘\_’))]See referenced class  [else]To be provided  [/if]<drop/>  [/if]<drop/>  Do NOT remove the previous line as word throws an error if the cell is empty <drop/> |
| --- | --- | --- |

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>  
</fragment><drop/>

# Fragment: Insert attribute row brief <drop/>

<fragment name=’insertAttributeRowBrief’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’p’ type=’uml::Property’/><drop/>

| [p.name/] | [for (st:Stereotype | p.getAppliedStereotypes())]<drop/>  [if(not st.name.contains(‘OpenModelAttribute’))] [st.name/]  [/if]<drop/>  [/for]<drop/>  Do NOT remove the previous line as word throws an error if the cell is empty <drop/> | [if p.ownedComment->notEmpty()]<drop/>  [for (c:Comment | p.ownedComment)] <drop/>  [cleanAndFormat(c.\_body.clean())/]  [/for]  [else] [if (p.name.contains (‘\_’))]See referenced class  [else]To be provided  [/if]<drop/>  [/if]<drop/>  Do NOT remove the previous line as word throws an error if the cell is empty <drop/> |
| --- | --- | --- |

</fragment><drop/>

# Fragment: Start attribute table brief <drop/>

<fragment name=’insertAttributeTableHeader’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’cl’ type=’uml::Class’/><drop/>

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Lifecycle Stereotype (empty = Mature)** | **Description** |

</fragment><drop/>

# Fragment: Insert Attribute table brief <drop/>

<fragment name=’insertAttributeTableBrief’ importedBundles=’commons;gmf;papyrus’ importedFragments='insertAttributeTableHeader;insertAttributeRowBrief’><drop/>  
<arg name=’cl’ type=’uml::Class’/><drop/>  
[if cl.ownedAttribute->notEmpty()]<drop/>

Table 2: Attributes for [cl.name/]

<table><drop/>

[cl.insertAttributeTableHeader ()/]

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

[if (not p.name.contains(‘\_’))]<drop/>

[p.insertAttributeRowBrief ()/]

[/if]<drop/>

[/for]<drop/>

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

[if (p.name.contains(‘\_’))]<drop/>

[p.insertAttributeRowBrief ()/]

[/if]<drop/>

[/for]<drop/>

</table><drop/>

[/if]<drop/>

</fragment><drop/>

# Fragment: Insert Ten Specified Attribute table brief <drop/>

<fragment name=’insertTenSpecifiedAttributeTableBrief’ importedBundles=’commons;gmf;papyrus’ importedFragments='insertAttributeTableHeader;insertAttributeRowBrief’><drop/>  
<arg name=’cl’ type=’uml::Class’/><drop/>

<arg name=’p1’ type=‘String’/><drop/>

<arg name=’p2’ type=‘String’/><drop/>  
<arg name=’p3’ type=‘String’/><drop/>  
<arg name=’p4’ type=‘String’/><drop/>  
<arg name=’p5’ type=‘String’/><drop/>  
<arg name=’p6’ type=‘String’/><drop/>  
<arg name=’p7’ type=‘String’/><drop/>  
<arg name=’p8’ type=‘String’/><drop/>  
<arg name=’p9’ type=‘String’/><drop/>  
<arg name=’p10’ type=‘String’/><drop/>  
[if cl.ownedAttribute->notEmpty()]<drop/>

Table 3: Attributes for [cl.name/]

<table><drop/>

[cl.insertAttributeTableHeader ()/]

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

[if (p.name.contains(p1) or p.name.contains(p2) or p.name.contains(p3) or p.name.contains(p4) or p.name.contains(p5) or p.name.contains(p6) or p.name.contains(p7) or p.name.contains(p8) or p.name.contains(p9) or p.name.contains(p10))]<drop/>

[if (not p.name.contains(‘\_’))]<drop/>

[p.insertAttributeRowBrief ()/]

[/if]<drop/>

[/if]<drop/>

[if (p.name.contains(p1) or p.name.contains(p2) or p.name.contains(p3) or p.name.contains(p4) or p.name.contains(p5) or p.name.contains(p6) or p.name.contains(p7) or p.name.contains(p8) or p.name.contains(p9) or p.name.contains(p10))]<drop/>

[if (p.name.contains(‘\_’))]<drop/>

[p.insertAttributeRowBrief ()/]

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

</table><drop/>

[/if]<drop/>

</fragment><drop/>

# Fragment: Insert DataType <drop/>

<fragment name=’insertDataType’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’dt’ type=’uml::DataType’/><drop/>  
<arg name=’dataTypeName’ type=’String’/><drop/>  
<arg name=’packageName’ type=’String’/><drop/>  
[if (dt.qualifiedName.contains(packageName))]<drop/>  
[if(dt.name.contains(dataTypeName))]<drop/>

Qualified Name: [dt.qualifiedName/]

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

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

[/for]<drop/>  
[if (dt.oclAsType(uml::DataType).general ->notEmpty())]<drop/>

Inherits properties from:

[for (tp:DataType | dt.oclAsType(uml::DataType).general)]<drop/>

* [tp.name/]

[/for]<drop/>

[for (gen:Class | dt.oclAsType(uml::DataType).general)]<drop/>

* [gen.name/]

[/for]<drop/>

[/if]<drop/>

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

[/for]<drop/>  
[else] <drop/>  
[/if]  
[/if]  
</fragment><drop/>

# Fragment: Start Data Type attribute table brief <drop/>

<fragment name=’insertDataTypeAttributeTableHeader’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’dt’ type=’uml::DataType’/><drop/>

|  |  |  |
| --- | --- | --- |
| **Attribute Name** | **Lifecycle Stereotype (empty = Mature)** | **Description** |

</fragment><drop/>

# Fragment: Insert Data Type Attribute table brief <drop/>

<fragment name=’insertDataTypeAttributeTableBrief’ importedBundles=’commons;gmf;papyrus’ importedFragments='insertDataTypeAttributeTableHeader;insertAttributeRowBrief’><drop/>  
<arg name=’dt’ type=’uml::DataType’/><drop/>  
[if dt.ownedAttribute->notEmpty()]<drop/>

Table 4: Attributes for [dt.name/]

<table><drop/>

[dt.insertDataTypeAttributeTableHeader ()/]

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

[p.insertAttributeRowBrief ()/]

[/for]<drop/>

</table><drop/>

[/if]<drop/>

</fragment><drop/>

# Fragment: Insert enums <drop/>

<fragment name=’insertEnums’ importedBundles=’commons;gmf;papyrus’><drop/>  
<arg name=’dt’ type=’uml::DataType’/><drop/>

#### [dt.name/]

Qualified Name: [dt.qualifiedName/]

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

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

[/for]<drop/>

Applied stereotypes:

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

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

* [st.name/]

[/for]<drop/>

[else] No stereotypes applied

[/if]<drop/>

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

Inherits literals from:

[for (tp:DataType | dt.oclAsType(uml::DataType).general)]<drop/>

* [tp.name/]

[/for]

[/if]<drop/>

[if (dt.oclAsType(Enumeration).ownedLiteral->notEmpty())]<drop/>

Contains Enumeration Literals:

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

* [e.name/]:
  + [for (co:Comment | e.ownedComment)]<drop/>
  + <dropEmpty>[cleanAndFormat(co.\_body.clean())/]
  + </dropEmpty>[/for]<drop/>
  + [if dt.getAppliedStereotypes()->notEmpty()] <drop/>
  + Applied stereotypes:
    - [for (st:Stereotype | e.getAppliedStereotypes())]<drop/>
    - [st.name/]
    - [/for]<drop/>
  + [/if]<drop/>

[/for]<drop/>

[/if]<drop/>

</fragment><drop/>