<config>  
<output path=’C:\Users\ndavis\git\OnfInfoModelOutput\ModelDescriptions\TR-512.DD\_OnfCoreIm-DataDictionary.docx' />  
</config>

<context model=’C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreModel.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.DD

Data Dictionary

Version 1.5

September 2021

ONF Document Type: Technical Recommendation

ONF Document Name: Core Information Model version 1.5

**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)

©2021 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](#_Toc434403079)

[Open Networking Foundation 2](#_Toc434403080)

[Document History 5](#_Toc434403081)

[1 Introduction 5](#_Toc434403082)

[2 References 6](#_Toc434403083)

[3 Definitions 8](#_Toc434403084)

[3.1 Terms defined elsewhere 8](#_Toc434403085)

[3.2 Terms defined in this TR 8](#_Toc434403086)

[4 Abbreviations and acronyms 8](#_Toc434403087)

[5 Conventions 10](#_Toc434403088)

[5.1 UML modeling conventions 10](#_Toc434403089)

[5.2 Lifecycle Stereotypes 10](#_Toc434403090)

[5.3 Diagram Keys 11](#_Toc434403091)

[6 Overview of the CoreModel Fragment 13](#_Toc434403092)

[6.1 Overview of the CoreNetworkModule of the CoreModel 15](#_Toc434403093)

[6.1.1 LogicalTerminationPoint (LTP) and LayerProtocol (LP) 17](#_Toc434403094)

[6.1.2 ForwardingDomain (FD) 18](#_Toc434403095)

[6.1.3 ForwardingConstruct (FC), EndPoint (EP), FcRoute and FcSwitch 18](#_Toc434403096)

[6.1.4 Link and LinkEnd 18](#_Toc434403097)

[6.1.5 NetworkElement, NetworkControlDomain and SdnController 19](#_Toc434403098)

[6.2 CoreFoundationModule 19](#_Toc434403099)

[6.2.1 Naming and identifiers 19](#_Toc434403100)

[6.2.2 States 22](#_Toc434403101)

[6.2.2.1 Relationship between states in Provider context 22](#_Toc434403102)

[6.2.2.2 Relationship between states in the client and provider context 23](#_Toc434403103)

[6.3 Termination Fragment 24](#_Toc434403104)

[6.4 Forwarding Fragment 27](#_Toc434403105)

[6.4.1 Basic Forwarding 27](#_Toc434403106)

[6.4.2 Forwarding Construct Specification and other details of Forwarding 28](#_Toc434403107)

[6.5 Topology Fragment 32](#_Toc434403108)

[6.5.1 Basic Topology 35](#_Toc434403109)

[6.5.2 Topology and views 40](#_Toc434403110)

[6.5.3 View boundaries and intermediates 46](#_Toc434403111)

[6.5.4 More on views and names/identifiers 47](#_Toc434403112)

[6.5.5 Off-network reference and the clients view 50](#_Toc434403113)

[6.5.6 Physical Port reference 51](#_Toc434403114)

[6.5.7 Detailed properties of Topology 53](#_Toc434403115)

[6.6 Directionality 55](#_Toc434403116)

[7 Future CoreModel areas 60](#_Toc434403117)

[8 UML model files 60](#_Toc434403118)

[8.1 Papyrus File 60](#_Toc434403119)

[9 Data Dictionary 62](#_Toc434403120)

[9.1 Core Network Module data dictionary 62](#_Toc434403121)

[9.1.1 Classes 62](#_Toc434403122)

[9.1.1.1 {{cl.name/}} 62](#_Toc434403123)

[9.1.2 Data Types 65](#_Toc434403124)

[9.1.2.1 {{dt.name/}} 65](#_Toc434403125)

[9.1.3 Enumeration Types 67](#_Toc434403126)

[9.1.3.1 {{dt.name/}} 67](#_Toc434403127)

[9.1.4 Primitive Types 68](#_Toc434403128)

[9.1.4.1 {{dt.name/}} 68](#_Toc434403129)

[9.2 Core Foundation Module data dictionary 68](#_Toc434403130)

[9.2.1 Classes 68](#_Toc434403131)

[9.2.1.1 {{cl.name/}} 68](#_Toc434403132)

[9.2.2 Data Types 71](#_Toc434403133)

[9.2.2.1 {{dt.name/}} 71](#_Toc434403134)

[9.2.3 Enumeration Types 73](#_Toc434403135)

[9.2.3.1 {{dt.name/}} 73](#_Toc434403136)

[9.2.4 Primitive Types 74](#_Toc434403137)

[9.2.4.1 {{dt.name/}} 74](#_Toc434403138)

[9.3 Core Enhancements data dictionary 74](#_Toc434403139)

[9.3.1 Classes 74](#_Toc434403140)

[9.3.1.1 {{cl.name/}} 74](#_Toc434403141)

[9.3.2 Data Types 77](#_Toc434403142)

[9.3.2.1 {{dt.name/}} 77](#_Toc434403143)

[9.3.3 Enumeration Types 79](#_Toc434403144)

[9.3.3.1 {{dt.name/}} 79](#_Toc434403145)

[9.3.4 Primitive Types 80](#_Toc434403146)

[9.3.4.1 {{dt.name/}} 80](#_Toc434403147)

[9.4 Core Enhancement Fragments data dictionary 80](#_Toc434403148)

[9.4.1 Classes 80](#_Toc434403149)

[9.4.1.1 {{cl.name/}} 80](#_Toc434403150)

[9.4.2 Data Types 83](#_Toc434403151)

[9.4.2.1 {{dt.name/}} 83](#_Toc434403152)

[9.4.3 Enumeration Types 85](#_Toc434403153)

[9.4.3.1 {{dt.name/}} 85](#_Toc434403154)

[9.4.4 Primitive Types 86](#_Toc434403155)

[9.4.4.1 {{dt.name/}} 86](#_Toc434403156)

[10 Additional figures related to potential extensions 87](#_Toc434403157)

[10.1 State extensions 87](#_Toc434403158)

[10.2 LTP Specification 87](#_Toc434403159)

[10.3 Model structure rules 88](#_Toc434403160)

[11 Addendum Translation table 91](#_Toc434403161)

[12 Back matter 93](#_Toc434403162)

[12.1 Editors 93](#_Toc434403163)

[12.2 Contributors 93](#_Toc434403164)

List of Figures

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

Document History

| **Version** | **Date** | **Description of Change** |
| --- | --- | --- |
| 1.0 | March 30, 2015 | Initial version of the base document of the “Core Information Model” fragment of the ONF Common Information Model (ONF-CIM). |
| 1.1 | November 24, 2015 | Version 1.1 |
| 1.2 | September 20, 2016 | Version 1.2 {{Note Version 1.1 was a single document whereas 1.2 is broken into a number of separate parts}} |
| 1.3 | September 2017 | Document name changed. Was TR-512.8 in Version 1.2. {{Published via wiki only}} |
| 1.3.1 | January 2018 | Addition of text related to approval status. |
| 1.4 | November 2018 | Aligned with 1.4 model content. |
| 1.5 | September 2021 | Enhancements to model structure |

# 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](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\TR-512.1_OnfCoreIm-Overview.pdf).

## References

For a full list of references see [TR-512.1](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\TR-512.1_OnfCoreIm-Overview.pdf).

## Definitions

For a full list of definition see [TR-512.1](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\TR-512.1_OnfCoreIm-Overview.pdf).

## Conventions

See [TR-512.1](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\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%), 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](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\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.

# Data Dictionary

The data dictionary provides details of all classes, attributes and types in the model. The data dictionary is divided up into sections based upon the division of the CoreModel and maturity of work.

* Section 2.1 Core Network Model: includes Forwarding, Termination, Topology and Resilience (see [TR-512.2](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.2_OnfCoreIm-ForwardingAndTermination.pdf), [TR-512.4](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.4_OnfCoreIm-Topology.pdf) and [TR-512.5](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.5_OnfCoreIm-Resilience.pdf))
* Section 2.2 Core Foundation Model: includes naming, identification and states (see [TR-512.3](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.3_OnfCoreIm-Foundation.pdf))
* Section 2.3 Core Physical Model: includes including Equipment and Connector (see [TR-512.6](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.6_OnfCoreIm-Physical.pdf))
* Section 2.4 Core Specification Model: covers specification (see [TR-512.7](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.7_OnfCoreIm-Specification.pdf))
* Section 2.5 General Processing Model: covers the generalized representation of processing capability (see [TR-512.11](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.11_OnfCoreIm-ProcessingConstruct.pdf))
* Section 2.6 General Control Model: covers the generalized representation of control functionality (see [TR-512.8](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.8_OnfCoreIm-Control.pdf))
* Section 2.7 Core Operations Model: covers the generalized, outcome oriented, operations pattern (see [TR-512.10](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.10_OnfCoreIm-OperationPatterns.pdf))
* Section 2.8 Core Software Model data dictionary: covers the software model (see [TR-512.12](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.12_OnfCoreIm-Software.pdf))
* Section 2.9 Model Patterns data dictionary (see [TR-512.A.2](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.A.2_OnfCoreIm-Appendix-ModelStructurePatternsAndArchitecture.pdf))

## Core Network Model data dictionary

This section provides the model details for Forwarding, Termination, Topology and Protection.

### Classes

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

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

[if (not cl.qualifiedName.contains(’SdnController’))]<drop/>

[if (not cl.qualifiedName.contains(’NetworkControlDomain’))]<drop/>

[if (not cl.qualifiedName.contains(’NetworkElement’))]<drop/>

[cl.insertClassesDd()/]

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### Data Types

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

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

[if (dt.qualifiedName.contains(’CoreNetworkModel’))]<drop/>

[dt.insertDataTypesDd()/]

[/if]<drop/>

[else][/if]<drop/>

[/for]<drop/>

### Enumeration Types

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

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

[if (dt.qualifiedName.contains(‘CoreNetworkModel’))]<drop/>

[dt.insertEnumsDd()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

### Primitive Types

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

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

[if (dt.qualifiedName.contains(‘CoreNetworkModel’))]<drop/>

[dt.insertPrimitiveTypesDd()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

## Core Foundation Model data dictionary

This section provides the model details for the foundation.

### Classes

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

[if (not cl.qualifiedName.contains(’CoreFoundationModel’))]<drop/>

[else] <drop/>

[cl.insertClassesDd()/]

[/if]<drop/>

[/for]<drop/>

### Data Types

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

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

[if (not dt.qualifiedName.contains(’CoreFoundationModel’))]<drop/>

[else] <drop/>

[dt.insertDataTypesDd()/]

[/if]<drop/>

[else][/if]<drop/>

[/for]<drop/>

### Enumeration Types

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

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

[if (not dt.qualifiedName.contains(‘CoreFoundationModel’))]<drop/>

[else] <drop/>  
[dt.insertEnumsDd()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

### Primitive Types

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

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

[if (not dt.qualifiedName.contains(‘CoreFoundationModel’))]<drop/>

[else] <drop/>

[dt.insertPrimitiveTypesDd()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

## Core Physical Model data dictionary

This section provides the details for the model of physical things including equipment and connectors.

### Classes

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

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

[cl.insertClassesDd()/]

[/if]<drop/>

[/for]<drop/>

### Data Types

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

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

[if (dt.qualifiedName.contains(’CorePhysicalModel’))]<drop/>

[dt.insertDataTypesDd()/]

[/if]<drop/>

[else][/if]<drop/>

[/for]<drop/>

### Enumeration Types

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

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

[if (dt.qualifiedName.contains(‘CorePhysicalModel’))]<drop/>

[dt.insertEnumsDd()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

### Primitive Types

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

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

[if (dt.qualifiedName.contains(‘CorePhysicalModel’))]<drop/>

[dt.insertPrimitiveTypesDd()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

## Core Specification Model data dictionary

This section provides the details for the model of Specification.

### Classes

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

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

[cl.insertClassesDd()/]

[/if]<drop/>

[/for]<drop/>

### Data Types

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

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

[if (dt.qualifiedName.contains(’CoreSpecificationModel’))]<drop/>

[dt.insertDataTypesDd()/]

[/if]<drop/>

[else][/if]<drop/>

[/for]<drop/>

### Enumeration Types

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

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

[if (dt.qualifiedName.contains(’CoreSpecificationModel’))]<drop/>

[dt.insertEnumsDd()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

### Primitive Types

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

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

[if (dt.qualifiedName.contains(’CoreSpecificationModel’))]<drop/>

[dt.insertPrimitiveTypesDd()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

## General Processing Model data dictionary

This section provides the model details for generalized model of processing.

### Classes

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

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

[cl.insertClassesDd()/]

[/if]<drop/>

[/for]<drop/>

### Data Types

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

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

[if (dt.qualifiedName.contains(’ProcessingConstructModel’))]<drop/>

[dt.insertDataTypesDd()/]

[/if]<drop/>

[else][/if]<drop/>

[/for]<drop/>

### Enumeration Types

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

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

[if (dt.qualifiedName.contains(‘ProcessingConstructModel’))]<drop/>

[dt.insertEnumsDd()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

### Primitive Types

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

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

[if (dt.qualifiedName.contains(‘ProcessingConstructModel’))]<drop/>

[dt.insertPrimitiveTypesDd()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

## General Control Model data dictionary

This section provides the model details for Control model (other than specific control classes used in modeling of resilience).

### Classes

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

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

[if (not cl.qualifiedName.contains(’Explanatory’))]<drop/>

[cl.insertClassesDd()/]

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### Data Types

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

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

[if (dt.qualifiedName.contains(’GeneralControllerModel’))]<drop/>

[dt.insertDataTypesDd()/]

[/if]<drop/>

[else][/if]<drop/>

[/for]<drop/>

### Enumeration Types

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

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

[if (dt.qualifiedName.contains(‘GeneralControllerModel’))]<drop/>

[dt.insertEnumsDd()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

### Primitive Types

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

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

[if (dt.qualifiedName.contains(‘GeneralControllerModel’))]<drop/>

[dt.insertPrimitiveTypesDd()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

## Core Interactions Model data dictionary

This section provides the model details for Interactions model supporting the generalized operations pattern, notification patterns, etc..

### Classes

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

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

[if (not cl.qualifiedName.contains(’Example’))]<drop/>

[cl.insertClassesDd()/]

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### Data Types

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

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

[if (dt.qualifiedName.contains(’CoreOperationsModel’))]<drop/>

[if (not dt.qualifiedName.contains(’Example’))]<drop/>

[dt.insertDataTypesDd()/]

[/if]<drop/>

[/if]<drop/>

[else][/if]<drop/>

[/for]<drop/>

### Enumeration Types

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

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

[if (dt.qualifiedName.contains(‘CoreOperationsModel’))]<drop/>

[if (not dt.qualifiedName.contains(’Example’))]<drop/>

[dt.insertEnumsDd()/]

[/if]<drop/>

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

### Primitive Types

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

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

[if (dt.qualifiedName.contains(‘CoreOperationsModel’))]<drop/>

[dt.insertPrimitiveTypesDd()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

## Core Software Model data dictionary

This section provides the model details for Software model supporting modeling of file systems, running software, software containers and virtual machines.

### Classes

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

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

[if (not cl.qualifiedName.contains(’Example’))]<drop/>

[cl.insertClassesDd()/]

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### Data Types

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

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

[if (dt.qualifiedName.contains(’CoreSoftwareModel’))]<drop/>

[if (not dt.qualifiedName.contains(’Example’))]<drop/>

[dt.insertDataTypesDd()/]

[/if]<drop/>

[/if]<drop/>

[else][/if]<drop/>

[/for]<drop/>

### Enumeration Types

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

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

[if (dt.qualifiedName.contains(‘CoreSoftwareModel’))]<drop/>

[if (not dt.qualifiedName.contains(’Example’))]<drop/>

[dt.insertEnumsDd()/]

[/if]<drop/>

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

### Primitive Types

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

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

[if (dt.qualifiedName.contains(‘CoreSoftwareModel’))]<drop/>

[dt.insertPrimitiveTypesDd()/]

[/if]<drop/>

[else] [/if]<drop/>

[/for]<drop/>

## Core Location Model data dictionary

This section provides the model details for the Location model supporting modeling of where something is including various forms of location identification.

</gendoc><drop/>

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

<gendoc><drop/>

### Classes

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

[cl.insertClassesDd()/]

[/for]<drop/>

### Data Types

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

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

[dt.insertDataTypesDd()/]

[else][/if]<drop/>

[/for]<drop/>

### Enumeration Types

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

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

[dt.insertEnumsDd()/]

[else] [/if]<drop/>

[/for]<drop/>

### Primitive Types

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

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

[dt.insertPrimitiveTypesDd()/]

[else] [/if]<drop/>

[/for]<drop/>

## Core Party Model data dictionary

This section provides the model details for the Party model supporting modeling of an individual/organization from the perspective of its roles.

</gendoc><drop/>

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

<gendoc><drop/>

### Classes

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

[cl.insertClassesDd()/]

[/for]<drop/>

### Data Types

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

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

[dt.insertDataTypesDd()/]

[else][/if]<drop/>

[/for]<drop/>

### Enumeration Types

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

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

[dt.insertEnumsDd()/]

[else] [/if]<drop/>

[/for]<drop/>

### Primitive Types

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

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

[dt.insertPrimitiveTypesDd()/]

[else] [/if]<drop/>

[/for]<drop/>

## Model Patterns data dictionary

This section provides a view of the Classes related to the patterns that underpin the model. These classes are experimental and are provided for rough guidance only.

</gendoc><drop/>

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

<gendoc><drop/>

The focus of this section is the description of the classes for the Component-System pattern.

### Classes for Management/Control modeling

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

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

[cl.insertClassesDd()/]

[/if]<drop/>

[/for]<drop/>

**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-FragmentsDD.docx <drop/>
* Validate the DD pages are in Landscape, if not then correct to landscape. <drop/>

Template version 0.0.10 17 September 2017 <drop/>

# Fragment: Insert classes (DD only) <drop/>

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

#### [cl.name/]

Qualified Name: [cl.qualifiedName/]

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

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

[/for]<drop/>

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

This class is abstract.

[/if]<drop/>

Applied stereotypes:

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

[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/>

[else] No stereotypes applied

[/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/>

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

<drop/>

Table 1: Attributes for [cl.name/]

<table><drop/>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Multiplicity** | **Access** | **Stereotypes** | **Description** |

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

| [p.name/]  [if(not p.qualifiedName.contains(cl.name))]Inherited[/if] | [if p.type.name->notEmpty()]<drop/>  [p.type.name/]  [else]<drop/>  To be defined  [/if]<drop/>  Do NOT remove the previous line as word throws an error if the cell is empty (note that in this case the else is not working).<drop/> | [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('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)/][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)/][else] <drop/> [/if]   [else]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<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/> |
| --- | --- | --- | --- | --- | --- |

[/for]<drop/>

</table><drop/>

[else][/if]<drop/>

</fragment><drop/>

# Fragment: Insert data types (DD only) <drop/>

<fragment name=’insertDataTypesDd’ 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 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/>

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

<drop/>

Table 1: Attributes for [dt.name/]

<table><drop/>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **Type** | **Multiplicity** | **Access** | **Stereotypes** | **Description** |

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

| [p.name/] | [if p.type.name->notEmpty()]<drop/>  [p.type.name/]  [else]<drop/>  To be defined  [/if]<drop/>  Do NOT remove the previous line as word throws an error if the cell is empty (note that in this case the else is not working).<drop/> | [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('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)/][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)/][else] <drop/> [/if]   [else]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/if]<drop/>  [/for]<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/> |
| --- | --- | --- | --- | --- | --- |

[/for]<drop/>

</table><drop/>

[else][/if]<drop/>

</fragment><drop/>

# Fragment: Insert enums (DD only) <drop/>

<fragment name=’insertEnumsDd’ 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/>

# Fragment: Insert primitive types (DD only) <drop/>

<fragment name=’insertPrimitiveTypesDd’ 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/>

</fragment><drop/>