<config>  
<output path=’C:\Users\ndavis\git\OnfInfoModelOutput\TR-512.1\_OnfCoreIm-Overview.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-512.1

Overview

Version 1.6

November 2023

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](#_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 | 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 | Version 1.3 {{Published via wiki only}} |
| 1.3.1 | January 2018 | Addition of text related to approval status. |
| 1.4 | November 2018 | Modifications to accommodate 1.4 enhancements.  Addition of a simplified introductory section of the key object classes of the Core model. |
| 1.5 | September 2021 | Enhancements to model structure |

# Introduction to the document suite

## General introduction to the model

This ONF Technical Recommendation (TR) focuses on the Core Information Model (CoreModel) of the ONF Common Information Model (ONF-CIM). An information model describes the things in a domain in terms of objects, their properties (represented as attributes), and their relationships.

The ONF-CIM is expressed in a formal language called Unified Modeling Language (UML). UML defines a number of basic model elements, called UML artifacts. In order to assure consistent modeling, only a subset of these artifacts was used in the development of the ONF-CIM according to guidelines for creating an information model expressed in UML (documented in {{ONF TR-514}}).

The ONF-CIM is formed from a number of pieces and is focused on the CoreModel. At its heart, the CoreModel provides a representation of network forwarding resources[[1]](#footnote-1) from a management-control perspective. The CoreModel is independent of:

* Specific forwarding technology, i.e. the CoreModel is forwarding technology neutral.
* Specific management-control interface protocol, i.e. the CoreModel is management-control interface protocol neutral (as described in {{ONF TR-513}}).

The ONF-CIM supports forwarding technology specific properties via application of the specification model (see 2.1.6 Specification Model (TR-512.7) on page 9) enabling reuse of existing technology specific standards definitions (e.g., from {{ITU-T G.875}}), pruned and refactored as appropriate (see {{ONF TR-513}}). The technology specific content, acquired in a runtime solution via "filled in" cases of specification, augment the CoreModel to provide a forwarding technology specific representation.

From an interfacing perspective, considering the SDN architecture {{ONF TR-521}} as an example, a controller may expose a view of the network in terms of ONF-CIM entities to client SDN controllers or applications to meet the needs of that client. The interface may expose the information in a client specific form where that form can be deterministically mapped to the ONF-CIM[[2]](#footnote-2). Tooling is used to generate an interface specific form from the UML[[3]](#footnote-3) model[[4]](#footnote-4).

## Introduction to this document

This document acts as a guide to the set of documents that describe the CoreModel of the ONF-CIM, providing:

* An introduction to the CoreModel of the ONF-CIM in the form of a brief overview of the model with links to the other documents in the set (see section 2.1 Model Document Structure on page 8 and section 2.2 Supporting documents on page 15 including a reference to the data dictionary (see section 2.2.2 Data Dictionary (TR-512.DD) on page 16)).
* A brief explanation of how to introduce attributes and structure related to a specific network technology (see section 2.1.6 Specification Model (TR-512.7) on page 11).
* A terminology translation table (see section 2.2.3 Terminology mapping (TR-512.TM) on page 16)
* An explanation of supporting guidelines with references to the guideline documents (see section 2.3 Supporting Guidelines on page 17).
* A summary of the main changes from the previous versions (see section 3 Summary of changes on page 24).
* A list of references used in the document set (see section 4 References 28).
* A list of definitions used in the document set (see section 5 Definitions on page 31)
* A list of abbreviations used in the document set (see section 5.3 Abbreviations and acronyms on page 32).
* Some conventions used in the document set including key stereotypes and keys to the diagram symbol sets (see section 6 Conventions on page 38).
* A summary of future work (see section 7 Future CoreModel work areas on page 40)

In addition, a number of appendix documents, that provide examples and further explanatory details, are included with the deliverables. These are summarized in [TR-512.A.1](ModelDescriptions/TR-512.A.1_OnfCoreIm-AppendixOverview.pdf).

Separate work activities are taking the CoreModel and deriving interface models (see {{ONF-TAPI}}).

## Tool versions

In addition to the documentation referenced above and throughout this document, the TR-512 delivery package includes the CoreModel in Papyrus UML (see [OnfModel folder](OnfModel)). The ongoing intention is to publish using the environment versions as stated in the guidelines {{ONF TR-515}}. The precise versions are stated below. The OpenModelProfile is the latest available from GitHub at the time of publication:

**Table 1: Tooling and Profile Versions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **{{ONF TR-515}} Version** | **GitHub Version** | **Version used for {{ONF TR-512}}** | **Comments** |
| Eclipse | 4.13.0  (2019-09) |  | 4.13.0.v20190916-1323 |  |
| Papyrus | 4.5.0 |  | 4.5.0.201909110925 |  |
| Gendoc | 0.7.2 |  | 0.7.2.201909241003 |  |
| OpenModelProfile |  | 0.2.14 | 0.2.14 |  |

In addition, an Experimental profile has been used for some of the Experimental model. This profile will be integrated into the formal profile structure in a later release.

## Viewing UML diagrams

Some of the UML diagrams in figures 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).

The UML diagram convention is provided in {{ONF TR-514}}. There are some key aspects of the diagrams that need to be emphasized.

* Association end attribute (the name of which always starts with “\_”) highlighted in the diagrams by the navigable end of the association (arrow head) is an attribute of the class at the non-navigable end of the association. It is the convention not to show the attribute in the class in the diagrams. The attributes for non-navigable ends (owned by the association) are not shown in the diagram (so in the figure there is no attribute name by the black diamond).
* On some occasions, other properties of the association end attribute are also shown.

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

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertSmallDiagram(‘Overview-LtpAndLp’, ’Illustrating navigable association end attributes’)/]

[/for]<drop/>

In the diagram above, the text at the arrow head end \_lp… is an attribute of the Logical Termination Point.

This attribute is shown in the fragment of abbreviated data dictionary below for LogicalTerminationPoint.

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

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

[if (cl.name.contains(‘LogicalTerminationPoint’))]<drop/>

Inserts the attributes of the class <drop/>   
[cl.insertTenSpecifiedAttributeTableBrief (‘\_lp’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, ‘10’)/]

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

This sort of table is used in each of the documents on a section of the model and only provides summary information. For full information the reader should refer to the data dictionary (see section 2.2.2 Data Dictionary (TR-512.DD) on page 9) or the model itself (see [OnfModel folder](OnfModel)).

# Model Overview

This section provides an overview of the ONF Core Information Model (CoreModel) and of the structure of the model description documentation. Each document described has a hyperlink that will take you to the document in your system[[5]](#footnote-5). The documents referenced in this section are all in the “ModelDescriptions” folder and are covered by two subsections:

* The documents referred to in Section 2.1 describe the core model artifacts progressing through the model from the basics of forwarding and termination through to a description of the augmentation mechanism of the specification model.
* The documents referred to in Section 2.2 provide additional supporting material including the Data Dictionary.

The remaining subsections provide:

* Related guidelines for model generation and usage (section 2.3)
* Key references (section 2.4)
* A brief overview of the Papyrus files (section 2.5)
* A simplified introduction of the key object classes across multiple model parts (section 2.7).

## Model Document Structure

The CoreModel of the ONF-CIM consists of model artifacts that are intended for use by multiple applications and/or forwarding technologies. For navigability, the CoreModel is further sub-structured into Core Network Model (CNM), Core Foundation Model, Core Physical Model, and the Core Specification Model.

The Core Network Model (CNM) consists of artifacts that model the essential network aspects that are neutral to the forwarding technology of the network. The CNM currently encompasses Forwarding, Termination, Timing, Topology, and Resilience aspects (subsets of the CNM).

This section provides a list of all associated documents that describe the model. Each of the following sub-sections provides some brief highlights from the associated document and a link to that associated document.

The model documentation is broken down into a number of key parts which relate to but do not exactly match the model breakdown:

* CoreNetworkModel:
  + Forwarding and Termination (see section 2.1.1)
  + Topology (see section 2.1.3)
  + Resilience (see section 2.1.4)
  + Timing (see section 2.1.1)
* CoreFoundationModel (see section 2.1.2)
* CorePhysicalModel (see section 2.1.5)
* CoreSpecificationModel (see section 2.1.6)
* GeneralControllerModel (see section 2.1.7)
* CoreOperationsModel (see section 2.1.9)
* ProcessingConstructModel (see section 2.1.10)
* Software (see section 2.1.11)
* Party (see section 2.1.12)
* Location (see section 2.1.13

### Core Network Model – Forwarding and Termination Model ([TR-512.2](ModelDescriptions/TR-512.2_OnfCoreIm-ForwardingAndTermination.pdf))

The Forwarding and Termination document provides a high-level overview of the Termination and Forwarding aspects of the CoreNetworkModel. This model is essentially a canonical model of networking from a management-control perspective. The figure below is a skeleton class diagram illustrating the interrelationships between key object classes defined in the CoreNetworkModel of the CoreModel. The classes are colored to help recognize key groupings in the model. The colors are chosen to match the key entity colors in Figure 6-1 Network diagram symbol set (with the Link in the alternative color for clarity). This color scheme for class diagrams is used in some of the figures in the associated documents.

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

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘Forwarding-LtpInterLayerSkeletonOverview’, ’Skeleton Class Diagram of key object classes’)/]

[/for]<drop/>

Examples of applying the Core Network Model for modeling timing and synchronization are provided in [TR-512.A.8](ModelDescriptions/TR-512.A.8_OnfCoreIm-Appendix-TimingAndSynchronizationExamples.pdf).

### Core Foundation Model

The Foundation document provides a detailed view of all aspects of the CoreModel that are relevant to all other parts of the ONF-CIM. Currently this model includes coverage of naming and identifiers as well as states.

#### Naming and Identifiers ([TR-512.3](ModelDescriptions/TR-512.3_OnfCoreIm-Foundation.pdf))

Rationalizing the approach to naming, identification and addressing of entities described in the ONF-CIM.

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

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘Foundation-CommonPackagesNoNotes’, ’Overview of Naming and Identifier of Objects’)/]

[/for]<drop/>

#### States ([TR-512.17](ModelDescriptions/TR-512.17_OnfCoreIm-FoundationState.pdf))

Basic states applicable to a majority of entities in the ONF-CIM.

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

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘State-FullModel’, ’States for all Objects’)/]

[/for]<drop/>

### Core Network Model – Topology Model ([TR-512.4](ModelDescriptions/TR-512.4_OnfCoreIm-Topology.pdf))

The topology document provides a detailed view of the topology model covering both the basic topology pattern with detailed attributes as well as multi-layered topology and topology views.

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

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘Topology-HighLevelOverviewOfStructure-LargeText’, ’Key classes that form the network topology’)/]

[/for]<drop/>



Figure 2-1 ForwardingDomain recursion with Link[[6]](#footnote-6)

### Core Network Model – Resilience Model ([TR-512.5](ModelDescriptions/TR-512.5_OnfCoreIm-Resilience.pdf))

The Resilience document provides a view of the model for resilience (including protection and restoration) and encompasses:

* The basic resilience model structure
* The key attributes relevant to resilience
* The application of the resilience model to various cases

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

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘Resilience-Pattern’, ’Basic resilience pattern’)/]

[/for]<drop/>

### Core Physical Model ([TR-512.6](ModelDescriptions/TR-512.6_OnfCoreIm-Physical.pdf))

The Physical model document provides a view of the model for physical entities (including equipment, holders and connectors). The document:

* Introduces the Physical model structure
* Describes the key classes of the Physical model
* Explains the attributes of the Physical model
* Describes the relationship between the connector and the LTP
* Shows how the model deals with the relationship between physical and functional views including resilience
* Explains how the Specification model describes equipment schemes (e.g. rules, etc.)
* Highlights work in progress to further advance the Equipment model

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

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘Equipment-Pattern’, ’Basic equipment pattern’)/]

[/for]<drop/>

### Specification Model ([TR-512.7](ModelDescriptions/TR-512.7_OnfCoreIm-Specification.pdf))

There are several related needs that have given rise to the Specification model:

* Provide machine-interpretable form of specific localized behavior:
  + Representing rules related to restrictions of specific cases of use of the model
  + Representing capabilities of specific cases of use
* Enable the introduction of run time schema where the essential structure of the model is known up front (at compile time), but the details are not
* Reduce the clutter in a representation where a set of details take the same values for all instances that related to a specific case
* Allow leverage of existing standards definitions (e.g., technology/application specific) in a machine-interpretable language

The combination of the above resulted in a separation in the model of definitions of structure and content such that an instance of a class from one model fragment could have an association instance to another model fragment to enable the provision of a fragment of definition of the class and of subordinates.

The aim of all specification definitions is that they be rigorous definitions of specific cases of usage and enable machine interpretation where traditional interface designs would only allow human interpretation.

The following dedicated spec structures have been considered:

* FC spec: Main focus is to provide a representation of the effective internal structure of a ForwardingConstruct (FC)
* LTP and LP spec: Main focus is to provide a representation of Layer Protocol (LP) specific parameters for the Logical Termination Point (LTP)
* FD and Link spec: Main focus is on capacity and forwarding enablement restrictions
* Equipment spec: Main focus is to provide a representation of equipping constraints
* Scheme spec: Main focus is to provide a mechanism to describe any pattern (arrangement) of entities from the model for some specific purpose (e.g. to describe the structure of a {{ITU-T G.8032}} protection scheme

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

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘Spec-LtpCapabilitySpecWithLtp’, ’Class Diagram of the Spec Model of LTP and LP’)/]

[/for]<drop/>

In addition, there is work on a generalized spec pattern with the main focus to provide a common representation of the mechanism for relating a class to its spec, accounting for implementation needs.

### Control Model ([TR-512.8](ModelDescriptions/TR-512.8_OnfCoreIm-Control.pdf))

The ONF Architecture {{ONF TR-521}} talks of a recursion of control aligning well with the more general concept of the Management-Control Continuum from {{TMF IG1118}}. The control model in {{ONF TR-512 V1.2}} showed a traditional hierarchy rather than a generalized recursion.

Over many years it has become apparent that the traditional representation of the Network Element and of the Managed Element was not correct. It is clear that from one perspective the Network Element is simply a lower-level member of the Management-Control Continuum. It is also apparent that all other aspects of the NE are covered by other parts of the model.

It was concluded that the NE should be remodeled as simply a control capability and that that capability should be generalized so that it could handle all aspects of the Management Control Continuum.

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

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

[/for]<drop/>

As explained in [TR-512.8](ModelDescriptions/TR-512.8_OnfCoreIm-Control.pdf) the classes SdnController, NetworkControlDomain and NetworkElement[[7]](#footnote-7) have been reassessed and deprecated and new classes have been developed to replace them. It has been recognized that a uniform recursive model of control can be developed that provides a consistent treatment of what were previously seen as completely different things.

### OAM Model (TR-512.9)

This document is not part of this release, it will be provided in a later release. The document will provide a view of the multi-technology OAM model.

### Operations Pattern Model ([TR-512.10](ModelDescriptions/TR-512.10_OnfCoreIm-OperationPatterns.pdf))

The work has been carried out with the assumption that the future is cloud oriented such that the controllers are an interconnected system of cloud-based components. It is assumed that in a cloud environment the operations will be "outcome-oriented" interaction[[8]](#footnote-8) where the focus is on stating the constraints that form a boundary that defines the desired target. In outcome-oriented interaction the operations/methods/activities/tasks used to achieve the desired outcome are firmly in the domain of the provider. The client simply provides information about the desired outcome in the context of what has been agreed as possible. Hence the essential need of any interaction is the provision of information about the desired outcome in terms of constraints and potentially in the context of some expected initial system state. Whilst the content of any message may differ per interaction the structure will be consistent[[9]](#footnote-9).

* The Operations Pattern Model is intended to provide a dynamic sophisticated structure that has “foldaway” parts
* The aim is to provide one structure:
  + For all outcome-oriented constraint-based forms including intent
  + Supports traditional Verb driven forms
    - with constrained valued
    - with absolute values
  + Enables operations that:
    - Act on multiple separate independent things
    - Have sequence and interdependency between parts and with other separate interactions
    - Are long lived or short lived (where the life may depend upon the case and may not be knowable before the request
* The aim is that the model will be used to generate schema where there is a continuum of compatible schema from the most basic simple CRUD (Create/Read/Update/Delete) forms to the most sophisticated forms such that the CRUD form can be seen as a tiny subset of the sophisticated form

The following figure shows the model of the request.

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

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘Operation-Structure’, ’The structure of an operation (request)’)/]

[/for]<drop/>

### Processing Construct Model ([TR-512.11](ModelDescriptions/TR-512.11_OnfCoreIm-ProcessingConstruct.pdf))

The ProcessingConstruct (PC) represents generalized functionality. The PC is used in conjunction with the ConstraintDomain (CD) that groups PCs and constrains their usage. In addition to being general applicable to represent functionality that is not being modeled in detail the PC and CD form the fundamental pattern that allows an important transition in the representation of a ‘device’[[10]](#footnote-10).

In the ONF CIM there are already separate classes for special types of functions:

* ForwardingConstruct to represent forwarding functionality
* LogicalTerminationPoint to represent termination, and
* ForwardingDomain to represent forwarding scope constraints.

ProcessingConstruct is in addition to these concepts and is to be used where the major function of interest is related to processing rather than forwarding of information.

While there are a number of grey areas between processing and forwarding, there are a few ‘pure’ ProcessingConstructs:

* Memory
* CPU
* Storage

Another use for ProcessingConstruct is for representing control plane processes such as packet routing processes. Packet routers commonly run many routing protocols and may also run many instances of each routing protocol. Each routing process instance peers independently and using ProcessingConstruct we can show the actual control plane topologies.

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

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘ProcessingConstruct-Core’, ’Processing Construct and Constraint Domain core model’)/]

[/for]<drop/>

### Software Model ([TR-512.12](ModelDescriptions/TR-512.12_OnfCoreIm-Software.pdf))

The software model provides a representation of software aspects of network devices and compute hosts, and can be split into two broad areas:

1. Software inventory (similar to hardware inventory)
2. Software functionality (equivalent to ‘running hardware’, which isn’t explicitly modelled in the ONF CIM)

The figure below shows the support for a Virtual Machine.

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

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘Software-VirtualMachine’, ’Skeleton Class Diagram of key object classes’)/]

[/for]<drop/>

The Software model strengthens linkage between various other parts of the model, so that the model can:

* Show how running software provides functionality (similar to running hardware)
* Support management of memory, CPU and storage capacity (related back to its usage by running software)
* Show how the combination of hardware and software together produces functionality
* Consistently represent software running directly on hardware CPU and memory as well as the VMM/VM and container cases

In combination with other parts of the software model further supports the representation of functions emerging from a equipment.

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

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

[/for]<drop/>

Note that there are a number of scenarios that the software model should cover, including:

* Hypervisor/VMM as running software hosts VMs
* VM as running software has guest operating system
* operating system as running software enables running applications
* VM Image is installed element
* (Linux) container as running software enables running applications
* Software agent as running software
* Container engine as running software hosts containers
* Container as running software enables running applications
* Container image is installed element

Various examples are provided in an appendix (see[TR-512.A.1](ModelDescriptions/TR-512.A.1_OnfCoreIm-AppendixOverview.pdf) for an explanation of the Appendix).

TEST TEST

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

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

[/for]<drop/>

### Party Model ([TR-512.13](ModelDescriptions/TR-512.12_OnfCoreIm-Party.pdf))

</gendoc><drop/>

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

<gendoc><drop/>

The Party model focusses on concepts relating to “who”. Recording information about a person will be useful, but in business, many interactions are done via legal (company) entities.

The model links the company organization to the people it employs and who interact on its behalf. The common modelling term used for both person and organization is party.

It is intended that the model be augmented with specific party structure and details.

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

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

[/for]<drop/>

### Location Model ([TR-512.14](ModelDescriptions/TR-512.12_OnfCoreIm-Location.pdf))

</gendoc><drop/>

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

<gendoc><drop/>

The Location model focusses on concepts relating to ‘where’ something is. All location related information is modelled to be referenced from the rest of the model as required.

The model supports both textual based and map-based locations. It also supports locations ‘internal’ to a building as well as ‘outside’ locations.

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)

It is intended that the model be augmented with specific location structure and details.

[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/>

### Storage TR-512.15

This document is not part of this release, it will be provided in a later release. The document will provide a view of a generalized storage model.

### CPU and Memory TR-512.16

This document is not part of this release, it will be provided in a later release. The document will provide a view of a generalized compute model.

### Core Foundation Model – States ([TR-512.17](ModelDescriptions/TR-512.17_OnfCoreIm-FoundationState.pdf))

</gendoc><drop/>

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

<gendoc><drop/>

See section 2.1.2.2 States (TR-512.17) on page 10.

## Supporting documents

There are further supporting documents described below:

* The “A” series of explanatory documents (see [TR-512.A.1](ModelDescriptions/TR-512.A.1_OnfCoreIm-AppendixOverview.pdf))
* The “double letter” series of supporting documents (described below)
  + The documents of this series were issued in V1.2 as part of the numeric series (e.g., TR-512.DD was TR-512.8 in V1.2)

### Appendix Overview ([TR-512.A.1](ModelDescriptions/TR-512.A.1_OnfCoreIm-AppendixOverview.pdf))

There is a set of supporting appendix documents (the “A” series) that provide further examples and explanation of the model. These documents are briefly summarized in TR-512.A.1.

### Data Dictionary ([TR-512.DD](ModelDescriptions/TR-512.DD_OnfCoreIm-DataDictionary.pdf))

The data dictionary provides details of the classes, attributes and data types (i.e., syntax) that are used in the model. The individual “model focuses” documents provide details on key classes and attributes but do not provide all details to avoid clutter and replication.

An extract from the data dictionary is shown below.

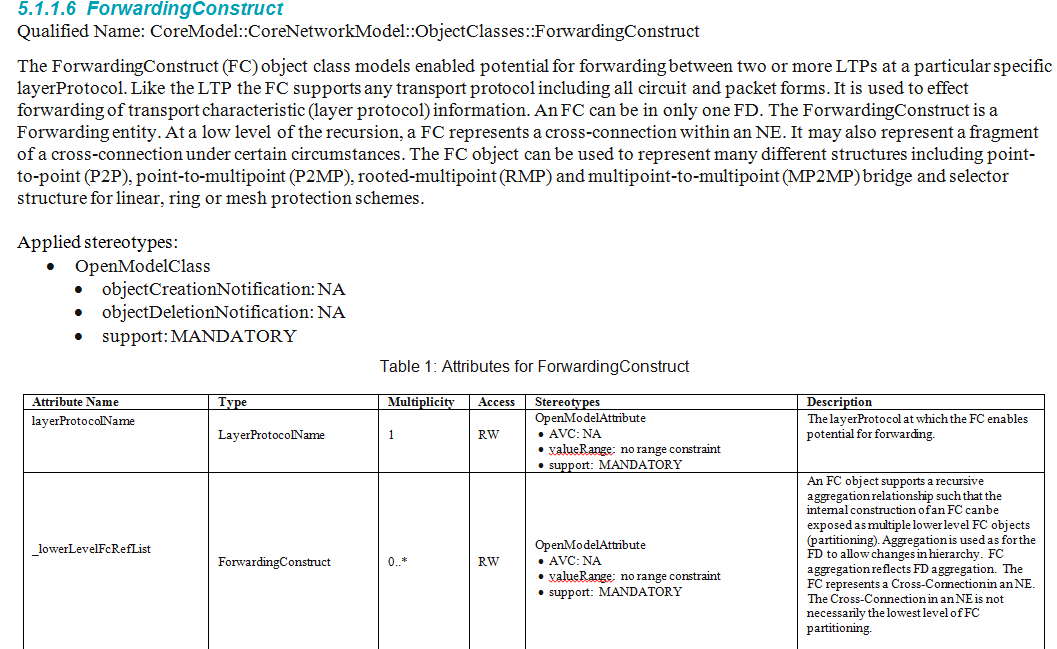


Figure 2-2 Extract from data dictionary (V1.2)

### Terminology mapping ([TR-512.TM](ModelDescriptions/TR-512.TM_OnfCoreIm-TerminologyMapping.pdf))

The terminology mapping document contains a table that provides overview translations from classes in the ONF-CIM to classes (and concepts) in other models. It will be helpful for someone who is familiar with one of the other industry standard terminology sets when working through the ONF-CIM.

### Core Model Future Enhancements ([TR-512.FE](ModelDescriptions/TR-512.FE_OnfCoreIm-FutureEnhancements.pdf))

This document provides fragments of ongoing work and lists all work areas known to require further development. The data dictionary document does NOT include entities from this document. All the work mentioned in this document is experimental.

### Gendoc fragment definitions ([TR-512.GT](ModelDescriptions/TR-512.GT_OnfCoreIm-CommonGendocTemplate.pdf))

This document provides a base document from which all other documents are derived. The document provides some examples of usage.

Note that Gendoc is the tool used to extract model element details and diagrams from the .uml and .notation files and to insert those into the TR-512 documentation.

Note all Gendoc templates are provided in the Gendoc folder.

## Supporting Guidelines

Several guideline documents have been constructed to maintain consistency in the models generated by ONF. These guidelines have also been shared with organizations outside ONF and are now developed in a collaborative mode across multiple bodies in an activity called IISOMI under OIMT in ONF {{ONF-IISOMI}}.

* **{{ONF TR-513}}:** This document specifies the principles and guidelines for the development and use of the ONF-CIM, including guidelines for deriving purpose-specific information model views (through pruning and refactoring selected subsets of artifacts from the ONF-CIM), and mapping to data schemas for protocol-specific control interfaces.
* **{{ONF TR-514}}:** The ONF-CIM is expressed in a formal language called UML (Unified Modeling Language). UML has a number of basic model elements, called UML artifacts. In order to assure consistent modeling, only a subset of the UML artifacts is used in the development of the ONF-CIM. The selected subset of UML artifacts is documented.
* **{{ONF TR-515}}**: This document specifiesthe guidelines for using the Papyrus tool used in the development of the ONF-CIM. It also describes how the ONF CIM modeling teams can cooperate in the GitHub environment for separate and coordinated development of the ONF-CIM fragments.
* {{ONF TR-531}}: This document defines the guidelines for mapping protocol-neutral UML information models to YANG data schemas. The UML information model to be mapped has to be defined based on the UML Modeling Guidelines defined in {{ONF TR-514}}.
* {{ONF TR-543}}: This document defines the guidelines for mapping protocol-neutral UML information model to OpenAPI (also a.k.a Swagger API), which is a RESTful API with JSON data schema. The UML information model to be mapped has to be defined based on the UML Modeling Guidelines defined in {{ONF TR-514}}.
* {{ONF TR-544}}: This document defines the guidelines for mapping protocol-neutral UML information model to the ProtoBuf schema language. The UML information model to be mapped has to be defined based on the UML Modeling Guidelines defined in {{ONF TR-514}}.

## Key reference material

In the development of the CoreModel, information model work from other SDOs has been used as input, including {{TMF TR215}}, {{TMF TR225}}, {{TMF SID 5LR}}, {{ITU-T G.7711}}, {{ITU-T G.875}}, {{ITU-T G.8052}}, and {{ITU-T G.8152}}. The CoreModel is being shared with other bodies via various mechanisms including publication of a view of the model as an IETF draft {{draft-lam}}.

## Papyrus File

This section provides the link to the information model file and the companion Open Model Profile file specified using the “Papyrus” modeling tool.

Link to the Core Model files: [OnfModel folder](OnfModel).

The file structure is as follows:

* + .project,
  + CoreModel.di,
  + CoreModel.notation
  + CoreModel.uml
  + OpenModel\_Profile.profile.di
  + OpenModel\_Profile.profile.notation
  + OpenModel\_Profile.profile.uml
  + Location.di
  + Location.notation
  + Location.uml
  + Party.di
  + Party.notation
  + Party.uml
  + Experimental.profile.di[[11]](#footnote-11)
  + Experimental.profile.notation
  + Experimental.profile.uml

In order to view and further extend or modify the information model, install the open source Eclipse software and the Papyrus tool. The installation guide for Eclipse and Papyrus can be found in {{ONF TR-515}}.

## Boundary of the work

As noted, the ONF Core IM does not cover interface definition. As a consequence, certain stereotype values are not relevant and hence are left at default including objectCreationNotification, objectDeletionNotification and passByReference. A majority of the attributes are read/write as in most cases a view can be conceived that will allow the attribute to be written.

## Key Model Classes

This section aims to take a different approach to introducing the CIM Core model classes. Note that this section is non-normative, that is it is a simplified introductory section only and not the actual model definition.

One of the key principles that the model is based on is that the physical[[12]](#footnote-12) inventory needs to be clearly segregated from the (logical) functional model. It is important that there is a clean boundary as any mixing will cause problems that then ‘distort the rest of the model’.

The physical inventory model has two key classes, Equipment and Holder. Equipment is the physical unit and the holders are the ‘places’ that can hold other physical units. This allows us to show the physical interrelationships but no functional ones, so for example concepts like a switch stack need to be done in the logical model.



Figure 2‑3 – Physical Inventory

On the logical side of the model, it is useful to categorize the functions in a network in terms of the type of functions that they provide. Two key function types are processing and transport of information.

For both of these logical function types it is useful to represent their association[[13]](#footnote-13) using a common pattern. Each of the logical functions can be thought of as a component. If the component is ‘symmetric’ then the association is between the components. If the component is asymmetric and the asymmetry is relevant to the association, then the component needs to have ports defined and the association is between the ports.

The figure below shows the key model entities and the functions that they perform.



Figure 2‑12 Key Class Functions

The first set of logical functions that we will discuss is those related to information transport.

The key transport classes are ForwardingDomain, ForwardingConstruct and Link.

ForwardingDomain and Link are both constraint boundaries that allow ForwardingConstruct creation between the LTPs that their ports are bound to. Links are also constrained in that their ports need to be bound to ForwardingDomain port LTP.

ForwardingConstruct represents enabled forwarding capability.

LogicalTerminationPoint and LayerProtocol are specialized processing functions used to terminate the forwarding capability and process the information.



Figure 2‑13 – Key Transport classes

For the processing functions we also have ProcessingConstruct and ControlConstruct.

Both ProcessingConstruct and ControlConstruct perform processing functions, but while a ProcessingConstruct just processes its own information, a ControlConstruct processes information to control other functions (such as ProcessingConstructs, ForwardingConstructs etc.). It is this additional controlling responsibility that means that it makes sense to have a separate model entity for ControlConstruct.

The last class that we will discuss is ConstraintDomain. ConstraintDomain provides general grouping / scoping and a place to attach constraints (to be added in a later release of the model).

The figure below shows the key model classes, and we can see the common component-port pattern repeated.

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

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

[/for]<drop/>

Note that LogicalTerminationPoint plays a special role in the model. It is like a ‘binding post’ that is used to decouple the associations between each of the other classes. Without it we would need N \* (N-1) /2 associations to show the options (possibly 55 associations instead of the 11 in the model above). The table below summarizes the association options in the model.



Figure 2‑4 – Key Model Class Association Options

Now we can see how the model fits together. The loose coupling between the physical inventory and the logical functions allows for complex physical – logical mappings as shown in the diagram below.



Figure 2‑5 - Distributed Device – Split Chassis

ConstraintDomain (CD in the diagrams) is used to scope physical, logical and control boundaries.



Figure 2‑6- Distributed Device – Split Chassis

Rather than trying to ‘decompose’ a network element black-box concept, the model ‘turns the network element inside-out’ and focusses on the network functions and their relationships and allows these functions to be grouped as required using ConstraintDomains. The association through LTP allows for the functional port-port bindings to be related to any underlying transport objects.



Figure 2‑7 – Example network function connectivity

The figure above shows a typical device which is running many protocols, and hence will have a number of network function instances. Each network function instance may peer with different remote devices to form different network function topologies.

# Summary of changes

## Summary of main changes between version 1.1 and 1.2

Changes to the model and/or related documentation:

* General
  + Change to doc structure
  + Change to gendoc
* Forwarding and Termination
  + Minor corrections made to multiplicities
  + Improvements made to documentation
* Foundation
  + An address structure has been added as «Experimental»
* Topology
  + Change of name of TopologicalEntity to ForwardingEntity
  + Incorporation of FC under ForwardingEntity and consideration of FC as closely related to topology
  + Capacity
* Resilience
  + Protection, restoration and recovery attributes added
  + Structure enhanced
  + Association to LTP from protection added in preparation for G.8032 modeling
  + Various examples of usage developed to both prove and document the resilience model
  + Corrections made to some multiplicities
  + Lifecycle stereotypes adjusted to reflect the advancing maturity of the model
* Equipment
  + New model added as «Experimental»
  + Focus of the model is on the pattern but all experimental work has been published as equal
* Specification
  + Addition of FD/Link spec details in terms of FD/Link capability statements
  + Refinement of FC spec to accommodate the Link (removal of Fc from class names in the spec and generalization to Forwarding recognizing the Link as a Forwarding entity).
  + Addition of sketches of the generalized spec model
  + Enhancements to details on LTP/LP spec and discussion on migration

## Summary of main changes between version 1.2 and 1.3

* General
  + Change to doc structure:
    - To focus numeric series on describing the model
    - Adding an appendix (“A”) series that provides further explanation
    - Adding a “double letter” series to capture ongoing model support material
  + Clean up of model structure
* Forwarding and Termination
  + Photonic/Media model including examples in an Appendix
  + Deprecating of NE, SdnController and NetworkControlDomain (replaced by the Control model and ProcessingConstruct/ConstraintDomain (see below)
  + LtpHasServerLtp is changed from composition to aggregation
  + Addition of Clock
* Foundation
  + Enhancements to the state model
* Topology
  + FdPort added to model as an optional
  + Aggregation of FD by FD via HigherLeeFdEncompassesLowerLevelFDs allows for many FDs to aggregate the same subordinate FD and this is reflected in the FcHasLowerLevelFc association multiplicity
  + Added explanation of use of model to support:
    - Serial-Compound Link
    - Inverse Multiplexing
    - Transitional Link
    - Multi-Port Links
    - State depedency
  + Added a clarification of the definition of Link
  + Explained the relationship between topology and the new control model
  + Enhanced the explanation of the approach of using the FC to represent the Call
  + Added explanation of the Resource-Service Continuum
* Resilience
  + G.8032 model with examples of use in an Appendix
  + Use of scheme spec concept
  + Timing protection
* Equipment
  + Upgrade of some classes and attributes from experimental to preliminary or mature
  + Integration with PC model providing an enhanced support for “Equipment protection” (via a model of function resilience)
* Specification
  + Improved introductory material
  + Refinements to the model to improve the decoupling of specification from the specified class
  + Addition of scheme spec concept
* Addition of the Control model
  + In conjunction with Processing Construct and Constraint Domain (see below) this provides:
    - A full consistent model of control
    - A replacement for the NE
* Addition of Operations Pattern model
  + This provides a generalized model of operations for an outcome-oriented interaction
* Addition of Processing Construct (PC) and Constraint Domain (CD)
  + This provides a generalized representation of functionality beyond Forwarding and Termination
* Addition of supporting information on patterns underpinning the model

## Summary of main changes between version 1.3 and 1.3.1

* General
  + An important note related to the approval status of TR-512 was added to each of the documents.
* Addition of the Circuit switched examples

Note that there were no changes in the UML model in 1.3.1.

## Summary of main changes between version 1.3.1 and 1.4

* Addition of Simplified Introduction section to this document (see section 2.7 Key Model Classes on page 17).
* Additional explanation to account for monitoring and overhead of media network ([TR-512.2](ModelDescriptions/TR-512.2_OnfCoreIm-ForwardingAndTermination.pdf)). There has also been:
  + Addition of the PHOTONIC\_MEDIA LayerProtocol string literal
  + Extension of layerProtocol to include a qualifier (relevant to both OTN and photonic-media)
* Addition of explanatory figure to show topology in a deep non-intrusive monitoring context ([TR-512.4](ModelDescriptions/TR-512.4_OnfCoreIm-Topology.pdf))
* Addition of description to the attributes and promotion to some model artifacts of the Physical model in [TR-512.6](ModelDescriptions/TR-512.6_OnfCoreIm-Physical.pdf))
* Enhancement to Forwarding Spec and Termination Spec to cater for the photonic model ([TR-512.7](ModelDescriptions/TR-512.7_OnfCoreIm-Specification.pdf))
  + The model now allows a recursion of specification of FCs inside LP and LPs inside FCs[[14]](#footnote-14)
* Enhancements to the model of Control to improve consistency with other parts of the model and addition of representation of the operations on the ControlConstruct ([TR-512.8](ModelDescriptions/TR-512.8_OnfCoreIm-Control.pdf))
  + The new work builds a bridge between the model of Control and the Operations model in [TR-512.10](ModelDescriptions/TR-512.10_OnfCoreIm-InteractionPatterns.pdf)
* Addition of the Software model (new [TR-512.12](ModelDescriptions/TR-512.12_OnfCoreIm-Software.pdf)) and examples (new [TR-512.A.13](ModelDescriptions/TR-512.A.13_OnfCoreIm-Appendix-SoftwareExamples.pdf))
* Refinement and further development to the Component-System pattern ([TR-512.A.2](ModelDescriptions/TR-512.A.2_OnfCoreIm-Appendix-ModelStructurePatternsAndArchitecture.pdf))
* Addition of new sections to [TR-512.A.4](ModelDescriptions/TR-512.A.4_OnfCoreIm-Appendix-AnalogueAndMediaExamples-L0.pdf) (new section 4 on L0 monitoring and overhead, section 5 on Relationship between L0 functional and physical, section 6 on photonic cases)
* Addition of the Packet switched examples (new [TR-512.A.6](ModelDescriptions/TR-512.A.6_OnfCoreIm-Appendix-PacketSwitchedExamples-L2-L3.pdf))
* Addition of Control and Interaction examples (new [TR-512.A.7](ModelDescriptions/TR-512.A.7_OnfCoreIm-Appendix-ControlAndInteractionExamples.pdf))

## Summary of main changes between version 1.4 and 1.5

* Addition of models of Party ([TR-512.13](ModelDescriptions/TR-512.12_OnfCoreIm-Party.pdf)) and Location ([TR-512.14](ModelDescriptions/TR-512.12_OnfCoreIm-Location.pdf))
* Split of Foundation across two documents (Naming and Identifiers ([TR-512.3](ModelDescriptions/TR-512.3_OnfCoreIm-Foundation.pdf))and State ([TR-512.17](ModelDescriptions/TR-512.17_OnfCoreIm-FoundationState.pdf)))
* Enhancements to state model ([TR-512.17](ModelDescriptions/TR-512.17_OnfCoreIm-FoundationState.pdf))
* Addition of model sketch of streaming and notification functions to the Control model ([TR-512.8](ModelDescriptions/TR-512.8_OnfCoreIm-Control.pdf))
* Update to modeling tools (see section 1.3 Tool versions on page 6)
* Enhancements to model structure in preparation for partitioning the model into separate papyrus projects.
* Removal of various broken diagrams

# References

{{BPMN}} Business Process Model and Notation <https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation>

{{draft-lam}} IETF draft-lam-teas-usage-info-model-net-topology-04, Usage of IM for network topology to support TE Topology YANG Module Development

{{Fernandez}} A Pattern for a Virtual Machine Environment  
Madiha H. Syed and Eduardo B. Fernandez  
Dept. of Computer and Elect. Eng. and Computer Science  
Florida Atlantic University, Boca Raton, FL 33431, USA

{{GNMI-SPEC}} <https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md> (version 0.10.0 from master branch current on 20230712)

{{IEEE 1588}} 1588 -2008 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems

{{IETF RFC4122}} IETF RFC 4122 (July 2005) A Universally Unique Identifier (UUID) URN Namespace

{{ISO/IEC 19505}} ISO/IEC 19505:2012 Information technology -- Object Management Group Unified Modeling Language (OMG UML)

{{ITU-T G.694.1}} Recommendation ITU-T G.694.1 (02/12), *Spectral grids for WDM applications: DWDM frequency grid*

{{ITU-T G.709}} Recommendation ITU-T G.709/Y.1331 (06/16), *Interfaces for the optical transport network*

{{ITU-T G.780}} Recommendation ITU-T G.780/Y.1351 (07/10), *Terms and definitions for synchronous digital hierarchy (SDH) networks*

{{ITU-T G.798}} Recommendation ITU-T G.798 (12/17), *Characteristics of optical transport network hierarchy equipment functional blocks*

{{ITU-T G.800}} Recommendation ITU-T G.800 (04/2016), *Unified functional architecture of transport networks*

{{ITU-T G.805}} Recommendation ITU-T G.805 (03/2000), *Generic functional architecture of transport networks*

{{ITU-T G.808.1}} Recommendation ITU-T G.808.1 (05/2014), *Generic protection switching – Linear trail and subnetwork protection*

{{ITU-T G.812}} Recommendation ITU-T G.812 (06/04), *Timing requirements of slave clocks suitable for use as node clocks in synchronization networks*

{{ITU-T G.813}} Recommendation ITU-T G.813 (03/03), *Timing characteristics of SDH equipment slave clocks (SEC)*

{{ITU-T G.852.1}} Recommendation ITU-T G.852.1 (11/1996), *Enterprise viewpoint for simple subnetwork connection management*

{{ITU-T G.852.2}} Recommendation ITU-T G.852.1 (11/1996), *Enterprise viewpoint description of transport network resource model*

{{ITU-T G.872}} Recommendation ITU-T G.872 (01/17), Architecture of optical transport networks

{{ITU-T G.874}} Recommendation ITU-T G.874 (08/2017), *Management aspects of optical transport network elements*

{{ITU-T G.875}} Recommendation ITU-T G.875 (06/2020), *Optical transport network: Protocol-neutral management information model for the network element view*

{{ITU-T G.7702}} Recommendation ITU-T G.7702 (03/18), Architecture for SDN control of transport networks

{{ITU-T G.7711}} Recommendation ITU-T G.7711/Y.1702 (03/2018), *Generic Protocol-Neutral Information Model for Transport Resources*

{{ITU-T G.7712}} Recommendation ITU-T G.7712/Y.1703 (09/10), *Architecture and specification of data communication network*

{{ITU-T G.8001}} Recommendation ITU-T G.8001/Y.1354 (04/16), *Terms and definitions for Ethernet frames over transport*

{{ITU-T G.8013}} Recommendation ITU-T G.8013/Y.1731 (08/15), *Operations, administration and maintenance (OAM) functions and mechanisms for Ethernet-based networks*

{{ITU-T G.8032}} Recommendation ITU-T G.8032/Y.1344 (08/15), *Ethernet Ring Protection Switching*

{{ITU-T G.8052}} Recommendation ITU-T G.8052/Y.1346 (11/2016), *Protocol-neutral management information model for the Ethernet Transport capable network element*

{{ITU-T G.8081}} Recommendation ITU-T G.8081 (02/2012), *Terms and definitions for automatically switched optical networks*

{{ITU-T G.8152}} Recommendation ITU-T G.8152/Y.1375 (12/2016), *Protocol-neutral management information model for the MPLS-TP network element*

{{ITU-T M.3100}} Recommendation ITU-T M.3100 (04/2005), *Generic network information model*

{{ITU-T M.3400}} Recommendation ITU-T M.3400 (02/2000), *TMN management functions*

{{ITU-T Q.1741.9}} Recommendation ITU-T Q.1741.9 (06/15), IMT-2000 references to Release 11 of GSM evolved UMTS core network

{{ITU-T X.731}} Recommendation ITU-T X.731 (01/1992), *Information technology - Open Systems Interconnection - Systems management: State management function*

{{KAFKA-COMP}} <https://kafka.apache.org/documentation/#compaction>

{{OASIS TOSCA}} <https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca>

{{ONF}} <https://www.opennetworking.org/>

{{ONF-IISOMI}} Project EAGLE: ONF Open Model, Profiles and Tools (<https://github.com/OpenNetworkingFoundation/EAGLE-Open-Model-Profile-and-Tools> and https://wiki.opennetworking.org/display/OIMT/IISOMI)

{{ONF OpenFlow}} <https://www.opennetworking.org/technical-communities/areas/specification/open-datapath/>

{{ONF-TAPI}} ONF Transport API (activity under the ONF OTCC project). (<https://github.com/OpenNetworkingFoundation/TAPI> and https://wiki.opennetworking.org/display/OTCC/TAPI)

{{ONF-TMF-MEF}} MEF ONF TMF Collaboration Agreement (see <https://login.opennetworking.org/bin/c5i?mid=38&rid=61&cid=3&k1=1567&tid=1483824677>)

{{ONF TR-502}} TR-502 (V1.0): SDN architecture, June 2014  
<https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf>

{{ONF TR-512}} This document series (V1.3.1, V1.2)  
(<https://www.opennetworking.org/software-defined-standards/models-apis/> for the project deliverables (including the documents), <https://www.opennetworking.org/wp-content/uploads/2014/10/TR-512_CIM_(CoreModel)_1.2.zip> for V1.2 and navigate the project deliverables for V1.3.1 (as the link was not available during the final editing))

{{ONF TR-513}} TR-513 (V1.2): ONF Common Information Model Overview   
(<https://www.opennetworking.org/software-defined-standards/models-apis/> for the project deliverables (including the document) and <https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/TR-513_CIM_Overview_1.2.pdf> for the document)

{{ONF TR-514}} TR-514 (V1.3): ONF UML Model Guidelines  
(<https://www.opennetworking.org/software-defined-standards/models-apis/> for the project deliverables (including the document) and [https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2018/08/TR-514\_UML\_Modeling\_Guidelines\_v1.3-1-1.pdf](https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2018/08/TR-514_UML_Modeling_Guidelines_v1.3-1-1.pdff) for the document)

{{ONF TR-515}} TR-515 (V1.3): ONF Papyrus Guidelines  
(<https://www.opennetworking.org/software-defined-standards/models-apis/> for the project deliverables (including the document) and https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2018/08/TR-515\_Papyrus\_Guidelines\_v1.3-1-1.pdf for the document)

{{ONF TR-521}} TR-521 (V1.1): ONF SDN Architecture 1.1, February 2016 (<https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/TR-521_SDN_Architecture_issue_1.1.pdf>)

{{ONF TR-523}} TR-523: Intent NBI – Definition and Principles, October 2016  
(<https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2014/10/TR-523_Intent_Definition_Principles.pdf>)

{{ONF TR-548}} TR-548 v2.0: TAPI v2.4.0 Reference Implementation Agreement for Streaming (<https://opennetworking.org/wp-content/uploads/2022/12/TR-548-TAPI_ReferenceImplementationAgreement-Streaming_v2.0.pdf>)

{{OpenConfig}} <https://github.com/openconfig/public>

{{TMF 612}} TM Forum MTOSI (4.0), Multi-Technology OS Interface <https://www.tmforum.org/resources/suite/mtosi-solution-suite-release-2-1/>

{{TMF IG1118}} TM Forum IG1118 OSS/BSS Futures – Architecture R15.5.1 (liaised to ONF) <https://www.tmforum.org/resources/exploratory-report/ig1118-ossbss-futures-architecture-r15-5-1/>

{{TMF MTNM}} Multi-Technology Network Management (MTNM) <https://www.tmforum.org/mtnm/>

{{TMF TR215}} TMF TR215 (V0.5.3) Logical Resource Network Model Advancements and Insights (liaised to ONF) <https://www.tmforum.org/resources/standard/tr215-logical-resource-network-model-advancement-and-insightstr215-logicalresourcenetworkmodeladvancementandinsights_version0-5-3/>

{{TMF TR225}} TM Forum TR225 (R15.0.0), Logical Resource: Network Function Model (liaised to ONF) <https://www.tmforum.org/resources/exploratory-report/tr225-logical-resource-network-function-model-r15-0-0-2/>

{{TMF SID 5LR}} TM Forum GB922 (R15.0.0) Information Framework (SID) Addendum 5LR (liaised to ONF) – latest version at <https://www.tmforum.org/resources/reference/gb922-information-framework-model-differences-r18-0-0/>

{{UML-YANG GUIDE}} TR-531 (V1.1) UML- YANG Mapping Guidelines  
https://3vf60mmveq1g8vzn48q2o71a-wpengine.netdna-ssl.com/wp-content/uploads/2018/08/TR-531\_UML-YANG\_Mapping\_Gdls\_v1.1-1-1.pdf

{{UML-YANG TOOL}} UML- YANG Mapping Tooling Navigate via <https://github.com/OpenNetworkingFoundation/EAGLE-Open-Model-Profile-and-Tools>

# Definitions

## Terms defined elsewhere

This document uses terms defined elsewhere. These terms are highlighted in section 5.3 Abbreviations and acronyms below by referring to the definition source document.

## Terms defined in this TR

The primary purpose of TR-512 is to define terms and hence terms are defined throughout the document suite and model. Key terms are highlighted in section 5.3 Abbreviations and acronyms below. Where a term is defined elsewhere there is a reference to the document where the term is defined.

## Abbreviations and acronyms

This TR uses the following abbreviations and acronyms:

Ø Phase

(A) Assembly

(e) electrical

(G) Group

(O) Overhead

(p) photonic

Analogue Continuously variable quantity (not relevantly quantized, not digitized). Used to convey information.

AP Access Point {{ITU-T G.805}}

API Application Programmer’s Interface

BBF BroadBand Forum (see <https://www.broadband-forum.org/>)

BC Boundary Clock

BMCA Best Master Clock Algorithm

C&SC Configuration and Switch Controller (model entity)

CASC C&SC

CC ControlConstruct (see [TR-512.8](ModelDescriptions/TR-512.8_OnfCoreIm-Control.pdf))

CD ConstraintDomain (see [TR-512.11](ModelDescriptions/TR-512.11_OnfCoreIm-ProcessingConstruct.pdf))

cir circulator

CNM Customer Network Management

CP Connection Point {{ITU-T G.805}}

CRUD Create Read Update Delete

CTP Connection Termination Point. Note that definitions differ between TM Forum {{TMF 612}} and {{ITU-T M.3100}}. Both usages apply here when referring to legacy cases and the abbreviation is qualified in all cases of use.

DSP Digital Signal Processor

DSRA Digital Services Reference Architecture (see TMF)

E-O Electrical - Optical

ECC Embedded Communications Channel {{ITU-T G.874}}

EDFA Erbium Doped Fiber Amplifier

EMS Element Management System {{definition reference ITU-T M.3400 - TMN}}[[15]](#footnote-15)

ERP Ethernet Ring Protection

ERPS Ethernet Ring Protection Switching

ETH Ethernet MAC Layer {{definition reference ITU-T G.8001}}

eTOM enhanced Telecommunications Operations Map (see TMF)

ETY Ethernet Physical Layer {{definition ITU-T G.8001}}

FC ForwardingConstruct (defined in the ONF-CIM - see [TR-512.2](ModelDescriptions/TR-512.2_OnfCoreIm-ForwardingAndTermination.pdf)).

* Note that at this point the definition is subtly different to that in {{TMF TR225}}. The aim is to align the terms usage

FD ForwardingDomain (defined in the ONF-CIM - see [TR-512.2](ModelDescriptions/TR-512.2_OnfCoreIm-ForwardingAndTermination.pdf))

FDFr FlowDomainFragment {{TMF 612}}

FRE ForwardingRelationshipEncapsulation {{TMF TR215}}

FRU Field Replaceable Unit

FTP FloatingTerminationPoint {{TMF 612}}

g Glass (used on an FC representing a fibre)

GitHub See www.github.com

GUID Globally Unique IDentifier (see www.wikipedia .org/Globally\_unique\_identifier)

H2M Human to Machine

IISOMI Informal Inter-SDO Open Model Initiative (see {{ONF-IISOMI}})

IM Information Model (see section 1 Introduction above)

IMP Inverse MultiPlexing {{ITU-T G.805}}

ISO International Organization for Standardization (see www.iso.org)

ITC Information Transfer Channel

ITU International Telecommunications Union (see www.itu.int)

ITU-T Telecommunications Standardization Sector of ITU-T (see <http://www.itu.int/en/ITU-T/Pages/default.aspx>)

JSON JavaScript Object Notation ([www.json.org/](http://www.json.org/))

La Laser

LP LayerProtocol (defined in the ONF-CIM – see [TR-512.2](ModelDescriptions/TR-512.2_OnfCoreIm-ForwardingAndTermination.pdf)). Note that there are two related terms:

* layer-protocol: used to refer to the information transfer protocol (or Characteristic Information of the signal)
* layerProtcolName: used to refer to the attribute in the LP class that carries the value that identifies the characteristic layer-protocol of the LP
* LayerProtocolName: used to refer to the data type that holds the formal name of the layer-protocol

LTP LogicalTerminationPoint (defined in the ONF-CIM - see [TR-512.2](ModelDescriptions/TR-512.2_OnfCoreIm-ForwardingAndTermination.pdf))

M2M Machine to Machine

MA Management Agent

MAC Media Access Control

MC Media Channel

MCA Media Channel Assembly

MCC Management Control Continuum (see {{TMF IG1118}}

MCG Media Channel Group

ME Managed Element OR Media Element (clarified per usage)

Media Substances (singular Medium) through which signal carrier is conveyed. In this case a vacuum is also considered as a medium.

MEF MEF Forum (see <https://mef.net/>)

MEG Maintenance Entity Group

MEP MEG End Point

MFDFr MatrixFlowDomainFragment

MIP MEG Intermediate Point

MLSN MultiLayerSubNetwork {{TMF 612}}

MP2MP Multi-Point to Multi-Point

MPLS-TP Multi-Protocol Label Switching Transport Profile {{definition reference RFC6378}}

MSS Multiple Strand Span

NCD NetworkControlDomain

NDC Network Domain Channel

NE NetworkElement

NFV Network Function Virtualization

NMC Network Media Channel

NMCA Network Media Channel Assembly

NMCG Network Media Channel Group

NMS Network Management System

NVP Name-Value Pair

OAM Operations Administration and Maintenance

OC Ordinary Clock OR Overall Controller

OCh Optical Channel

ODU Optical Data Unit

OIF Optical Interworking Forum (see http://www.oiforum.com/)

OIMT Open Information Model & Tooling (an ONF project)

OME Optical Maintenance Entity

OMS Optical Multiplex Section

OMS-O Optical Multiplex Section-Overhead

ONF Open Networking Foundation

ONF-CIM ONF Common Information Model

OPS Optical Protection Switch

OS Operations System (same as OSS) OR Optical Section

OSC Optical Supervisory Channel

OSME Optical Signal Maintenance Entity

OSNR Optical Signal to Noise Ratio

OSS Operation Support System

OTCC Open Transport Configuration and Control (an ONF project)

OTN Optical Transport Network

OTS Optical Transmission Section

OTS-O Optical Transmission Section-Overhead

OTSi Optical Tributary Signal

OTSiG Optical Tributary Signal Group

OTSiG-O Optical Tributary Signal Group-Overhead

OTU Optical Transport Unit

OTU-CN Optical Transport Unit beyond 100G (B100G)

P&R Pruning and Refactoring (Prune and Refactor)

P2MP Point to Multi-Point

P2P Point to Point

PC ProcessingConstruct (see [TR-512.11](ModelDescriptions/TR-512.11_OnfCoreIm-ProcessingConstruct.pdf))

Pd Photodiode

Pe Pump of electrons (constant power stream)

Pm Phase modulation

PoC Proof of Concept

PON Passive Optical Network

Pp Pump of photons (constant power stream)

PRC Primary Reference Clock

PTP Physical Termination Point {{TMF 612}}

PTP Precision Time Protocol {{IEEE 1588}}

RMP Rooted Multi-Point

ROADM Reconfigurable Optical Add-Drop Multiplexor

SD FEC Soft Decision Forward Error Correction

SDN Software Defined Networking {{ONF}}

SDO Standards Development Organization

Se Signal encoded on an electron stream

Si Signal

SID Shared Information and Date model (see TMF)

SiG Signal Group

SNC SubNetworkConnection {{TMF 612}}

SNP SubNetworkPoint {{ ITU-T G.8081}}

Sp Signal encoded on a photon stream

SSM Synchronization Status Message

TAPI Transport API

TBD To Be Defined

TC Transparent Clock

TCP Termination Connection Point {{ITU-T G.805}}

TDM Time Division Multiplex

TL1 Transaction Language 1 (<https://en.wikipedia.org/wiki/Transaction_Language_1>)

TMF TeleManagement Forum (see [www.tmforum.org](http://www.tmforum.org))

TOSCA {{OASIS TOSCA}}

TP Termination Point {{ITU-T M.3100}}

TPE TerminationPointEncapsulation {{TMF TR215}}

TR Technical Recommendation {{ONF}} Technical Report {{TM Forum}}

TRI Transport Resource Identifier {{ ITU-T G.8081}}

TTP Trail Termination Point {{ITU-T M.3100}}

UML Unified Modelling Language (see [www.omg.org](http://www.omg.org))

UUID Universally Unique IDentifier (see https://en.wikipedia.org/wiki/Universally\_unique\_identifier)

VCAT Virtual Concatenation

VM Virtual Machine

VMM Virtual Machine Manager

VNE Virtual Network Element

VNF Virtual Network Function

XC CrossConnection

YANG <https://en.wikipedia.org/wiki/YANG>

# Conventions

## Lifecycle Stereotypes

Lifecycle stereotypes (see {{ONF TR-514}}) are applied to entities in the model to indicate their degree of maturity[[16]](#footnote-16). These are made visible in many of the figures in this document.

The following stereotypes appear in TR-512:

* «Experimental»: Indicates that the entity is at a very early stage of development and will almost certainly change. The entity is NOT mature enough to be used in implementation[[17]](#footnote-17).
* «Preliminary» : Indicates that the entity is at a relatively early stage of development and is likely to change but is mature enough to be used in implementation.

If no stereotype is shown (or the entity is marked «Mature») the entity is mature. Other Lifecycle Stereotypes are defined in {{ONF TR-514}}.

## Key to diagram symbol set

This document set includes a number of UML diagrams. The UML symbol set is suitably explained in {{ONF TR-514}}. Many of the UML diagrams in this document have small font (due to density of information conveyed). It will be necessary for the reader to zoom in and pan across the figure to see the detail[[18]](#footnote-18).

This document set also contains a number of non-UML diagrams, which use the symbols highlighted below in pictorial representations of network examples. The symbol set is an advanced partial hybrid of symbols used by other bodies (see {{TMF TR215}} and {{ITU-T G.805}}).



Figure 6-1 Network diagram symbol set[[19]](#footnote-19)

In addition, in the diagrams related to media the following symbols and labels are used.



Figure 6-2 Additional media diagram symbol set

# Future CoreModel work areas

Future work areas are covered in [TR-512.FE](ModelDescriptions/TR-512.FE_OnfCoreIm-FutureEnhancements.pdf).

# Terminology Translation table

The translations provided in this release are early draft (see [TR-512.TM](ModelDescriptions/TR-512.TM_OnfCoreIm-TerminologyMapping.pdf)). There may be errors in the table and the table is not complete. It should be used for guidance only.

# Documentation structure

TR-512 is delivered in a .zip file. The file has key documents at the top level and several folders that contain other documents, UML figures and the model.

The .zip file is structured as below:

* ReadMe.txt
* TR-512.1\_OnfCoreIm-Overview.pdf (inc. Links to ModelDescription documents and OnfModel folder)
* ModelDescriptions folder (each document includes navigable links to other relevant documents)
  + TR-512.2\_OnfCoreIm-ForwardingAndTermination.pdf
  + TR-512.3\_OnfCoreIm-Foundation.pdf
  + TR-512.4\_OnfCoreIm-Topology.pdf
  + TR-512.5\_OnfCoreIm-Resilience.pdf
  + TR-512.6\_OnfCoreIm-Physical.pdf
  + TR-512.7\_OnfCoreIm-Specification.pdf
  + TR-512.8\_OnfCoreIm-Control.pdf
  + TR-512.10\_OnfCoreIm-OperationPatterns.pdf
  + TR-512.11\_OnfCoreIm-ProcessingConstruct.pdf
  + TR-512.12\_OnfCoreIm-Software.pdf
  + TR-512.13\_OnfCoreIm-Party.pdf
  + TR-512.14\_OnfCoreIm-Location.pdf
  + TR-512.17\_OnfCoreIm-FoundationState.pdf
  + TR-512.A.1\_OnfCoreIm-AppendixOverview.pdf
  + TR-512.A.2\_OnfCoreIm-Appendix-ModelStructurePatternsAndArchitecture.pdf
  + TR-512.A.3\_OnfCoreIm-Appendix-ModelRationale.pdf
  + TR-512.A.4\_OnfCoreIm-Appendix-AnalogueAndMediaExamples-L0.pdf
  + TR-512.A.5\_OnfCoreIm-Appendix-CircuitSwitchedExamples-L1-L2-gd.pdf
  + TR-512.A.6\_OnfCoreIm-Appendix-PacketSwitchedExamples-L2-L3-gd.pdf
  + TR-512.A.7\_OnfCoreIm-Appendix-ControlAndInteractionExamples-gd.pdf
  + TR-512.A.8\_OnfCoreIm-Appendix-TimingAndSynchronizationExamples.pdf
  + TR-512.A.9\_OnfCoreIm-Appendix-ProcessingConstructExamples.pdf
  + TR-512.A.10\_OnfCoreIm-Appendix-SpecificationExamples.pdf
  + TR-512.A.11\_OnfCoreIm-Appendix-ResilienceExamples.pdf
  + TR-512.A.13\_OnfCoreIm-Appendix-SoftwareExamples.pdf
  + TR-512.DD\_OnfCoreIm-DataDictionary.pdf
  + TR-512.FE\_OnfCoreIm-FutureEnhancements.pdf
  + TR-512.GT\_OnfCoreIm-CommonGendocTemplate.pdf
  + TR-512.TM\_OnfCoreIm-TerminologyMapping.pdf
* UmlFigures folder (has a subfolder for each document that includes one or more UML figures. These figures have been included to aid viewing as some are very detailed (as the .pdf figures do not scale sufficiently)).
  + TR-512.1
  + TR-512.2
  + TR-512.3
  + TR-512.4
  + TR-512.5
  + TR-512.6
  + TR-512.7
  + TR-512.8
  + TR-512.10
  + TR-512.11
  + TR-512.12
  + TR-512.13
  + TR-512.14
  + TR-512.17
  + TR-512.A.2
  + TR-512.FE
* OnfModel folder
  + .project,
  + CoreModel.di,
  + CoreModel.notation
  + CoreModel.uml
  + Location.di
  + Location.notation
  + Location.uml
  + Party.di
  + Party.notation
  + Party.uml
  + CoreCommonDataTypes.di
  + CoreCommonDataTypes.notation
  + CoreCommonDataTypes.uml
  + CommonDataTypes folder
  + UmlProfiles folder

The Gendoc Templates used to generate the TR-512 documents and other documentation aids are provided via another .zip file. They have been delivered to ensure inter-release continuity and will be used as a basis for the construction of the documentation for the next release of model.

# Individuals engaged

## Editors

Nigel DAVIS Ciena

Kam LAM FiberHome

## Contributors

Martin SKORUPSKI AT&T

Hui DING CATR, China

Yun Bin XU CATR, China

Xing ZHAO CATR, China

Weiqiang CHENG China Mobile

Rod LU China Mobile (Rod was previously at ZTE)

Ruiquan JING China Telecom

Nigel DAVIS Ciena

Lyndon ONG Ciena

Stephen SHEW Ciena

Christopher HARTLEY Cisco

Jonathan SADLER Infinera (Jonathan was previously at Coriant)

Bernd ZEUNER Deutsche Telekom

Erez SEGEV ECI

Dave HOOD Ericsson

Meral SHIRAZIPOUR Ericsson

Scott MANSFIELD Ericsson

Xiang YUN FiberHome

LiPing CHEN FiberHome (LiPing was previously at ZTE)

Kam LAM FiberHome (Kam was previously at Nokia)

Yuji TOCHIO Fujitsu

Italo BUSI Huawei

Maarten VISSERS Huawei

Stephane ST-LAURENT Infinera

Raymond CHEN Juniper

Leo NEDERLOF Merkator

Akira SAKURAI NEC

Karthik SETHURAMAN NEC  
Sibylle SCHALLER NEC

Andre MAZZINI Nokia

Dieter BELLER Nokia

Sergio BELOTTI Nokia

Eve VARMA Nokia

Shahar STEIFF PCCW Global

Germano GASPARINI SM-Optics

Adriana Veronica ERRIGO Telefonica

Thorsten HEINZE Telefonica

Malcolm BETTS ZTE

Xiaobing NIU ZTE

Qi Lei WANG ZTE

Xiong QUAN ZTE

Jin ZHOU ZTE

**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/>

1. It is focused on representation of the functions/resources that have the primary purpose of supporting information forwarding (transfer and transform functions), that form a network that realizes virtual adjacency, for the purpose of control of those functions/resources. Those resources are referred to as network forwarding resources. The information model is not intended to cover functional resources that have a primary purpose of supporting storage or compute solutions. [↑](#footnote-ref-1)
2. The Transport API (TAPI) provides an interface-oriented representation (in UML) derived from the CoreModel using the “Pruning & Refactoring” process {{ONF TR-513}} supported by tooling. [↑](#footnote-ref-2)
3. UML is not an interface protocol language. [↑](#footnote-ref-3)
4. For example, TAPI {{ONF-TAPI}} uses IISOMI tooling {{ONF-IISOMI}} to generate interface specific form (Yang, JSON etc). [↑](#footnote-ref-4)
5. The link will only work if you have unzipped the whole package as one. [↑](#footnote-ref-5)
6. The numbering of the FDs on the figure implies strict and fixed hierarchy. It should be noted that the association is aggregation and hence the hierarchy can change, and an FD may move from being encompassed by one FD to being encompassed by another. Consider the numbering as simply a view of the current structure. [↑](#footnote-ref-6)
7. The Network Element scope of the direct interface from a SDN controller to a Network Element in the infrastructure layer is similar to the EMS-to-NE management interface defined in the information models {{ITU-T G.875}} (OTN), {{ITU-T G.8052}} (Ethernet), and{{ITU-T G.8152}} (MPLS-TP). [↑](#footnote-ref-7)
8. Intent is an outcome-oriented form of interaction. [↑](#footnote-ref-8)
9. Again, human language is a good analogy. The grammar remains constant, simple and repeating but the vocabulary is broad and changes/grows often rapidly. [↑](#footnote-ref-9)
10. Here we will use the term ‘device’ in a loose and undefined manner to aid in the discussion. The term is not defined because it is not important for our discussion, the generally understood concept is sufficient. [↑](#footnote-ref-10)
11. The Experimental profile provides some stereotypes related to experimental rules (e.g. in the Physical model). The relevant stereotypes in this profile will be moved to a formal profile in the next release. [↑](#footnote-ref-11)
12. A Physical thing is a thing that can be measured with a ruler. [↑](#footnote-ref-12)
13. A binding together in some way such that the components are adjacent and such that information flows between them. [↑](#footnote-ref-13)
14. This will probably lead to a generalized pattern for LTP, FC etc. and scheme/system spec. This will be covered in the next release. [↑](#footnote-ref-14)
15. This term is not intended for use other than in reference to legacy systems. [↑](#footnote-ref-15)
16. The whole model including all degrees of work in progress has been published to allow the user maximum opportunity to set a most consistent direction with the work at hand. It is considered important to expose work in progress especially where this may have an impact on a choice of implementation. There may be some experimental structure that contains some very stable parts, without that structure those parts might be quite uninterpretable. A user who decides to take a low risk approach can ignore preliminary and experimental parts. A user who is more inclined to take a risk or who is looking for inspiration for their work can take the experimental and preliminary parts, understanding the risk involved. [↑](#footnote-ref-16)
17. The implementer can clearly choose to use the item at risk (expecting change and accounting for this in deployments etc.) [↑](#footnote-ref-17)
18. The aim is to improve the figure readability in future releases. [↑](#footnote-ref-18)
19. It should be noted that in this version and future versions the terms ForwardingDomain (FD) and ForwardingConstruct (FC) are used in place of SubNetwork (SN) and SubNetworkConnection (SNC) (used in the earlier versions of the ONF-CIM). [↑](#footnote-ref-19)