<config>  
<output path=’C:\Users\ndavis\git\OnfInfoModelOutput\ModelDescriptions\TR-512.15\_OnfCoreIm-Compute.docx' />  
</config>

<context model=’C:\Users\ndavis\git\ONFInfoModel\OnfModel\Compute.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-512.15

Compute

Version 1.6

January 2024

ONF Document Type: Technical Recommendation

ONF Document Name: Core Information Model version 1.6

**Disclaimer**

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

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

Open Networking Foundation

1000 El Camino Real, Suite 100, Menlo Park, CA 94025

[www.opennetworking.org](http://www.opennetworking.org)

©2024 Open Networking Foundation. All rights reserved.

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

**Important note**

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

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

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

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

Table of Contents

[Disclaimer 2](#_Toc16002872)

[Open Networking Foundation 2](#_Toc16002873)

[Document History 4](#_Toc16002874)

[1 Introduction 5](#_Toc16002875)

[1.1 References 5](#_Toc16002876)

[1.2 Definitions 5](#_Toc16002877)

[1.3 Conventions 5](#_Toc16002878)

[1.4 Viewing UML diagrams 5](#_Toc16002879)

[1.5 Understanding the figures 5](#_Toc16002880)

[2 Introduction to Storage 6](#_Toc16002881)

[2.1 Introduction 6](#_Toc16002882)

[2.2 Challenges 6](#_Toc16002883)

[2.3 Storage Options 7](#_Toc16002884)

[2.4 Storage or Memory ? 7](#_Toc16002885)

[2.5 Existing Standard Storage Models 8](#_Toc16002886)

[2.6 Partitioning and Aggregation 9](#_Toc16002887)

[2.7 Storage Pooling 10](#_Toc16002888)

List of Figures

[Figure 1 – Storage Options 7](#_Toc16087097)

[Figure 2 - StorageExtent as a ‘piece of tape’ 10](#_Toc16087098)

[Figure 3 – StorageExtent concatenation and striping 10](#_Toc16087099)

[Figure 4 - Storage Model Class Diagram 14](#_Toc16087100)

[Figure 5 - Example - Single Disk 15](#_Toc16087101)

Document History

| **Version** | **Date** | **Description of Change** |
| --- | --- | --- |
| 1.0 | January 2024 | Initial Version |

# Introduction

This document is an addendum to the TR-512\_v1.5 ONF Core Information Model and forms part of the description of the ONF-CIM. For general overview material and references to the other parts refer to [TR-512.1](../TR-512.1_OnfCoreIm-Overview.pdf).

## References

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

## Definitions

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

## Conventions

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

* UML conventions
* Lifecycle Stereotypes
* Diagram symbol set

## Viewing UML diagrams

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

## Understanding the figures

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

# Introduction to Compute

This document describes a general high-level model for compute functionality including processing and storage. The model is considered sufficient to represent the capabilities of the compute functions that may be present in a network device or in a controller of network devices.

For storage the document covers management of Block, File and Object storage, both directly attached and over a network. It includes standalone hosts with local storage, Redundant Array of Inexpensive Discs (RAID), Small Computer System Interface (SCSI) as well as network-based storage, including enterprise and cloud storage.

Note that this model excludes physical devices such as Central Processing Unit (CPU) chips and memory chips. All physical device considerations are covered by the existing Equipment model (see [TR-512.6](TR-512.6_OnfCoreIm-Physical.pdf)).

A data dictionary that sets out the details of all classes, data types and attributes is also provided ([TR-512.DD](file:///C:\Users\ndavis\git\ONFInfoModel\OnfModel\CoreGendoc\ModelDescriptions\TR-512.DD_v1.3_OnfCoreIm-DataDictionary.pdf)).

## Background

The compute model covers representation of the functions of CPU, memory and storage.

This model is designed to represent compute architectures in a technology independent manner and is focused on the management and control of the compute functions.

### CPU

The original CPU consisted of many physical units, as technology evolved it became possible to implement a CPU in single chip. Now a single chip may contain many CPU cores, and each core may run more than one thread. So, a definition of a ‘logical CPU’ function is required.

Some chips have a mix of architectures and/or capabilities (asymmetric), others simply have several replications of the same architecture (symmetric).

For example, an asymmetric CPU may have 4 + 4 cores ( 4 \* 1.8 GHz Type-A + 4 \* 1.4 GHz Type-B).

The CPU hardware may be housed in an Field Replaceable Unit (FRU) or a non-FRU. This is covered in the Equipment model (see [TR-512.6](TR-512.6_OnfCoreIm-Physical.pdf)).

### Memory

Memory chip(s), Single In-line Memory Module (SIMM) and Dual In-line Memory Modules (DIMM) may be an FRU or non-FRU. This is covered in the Equipment model (see [TR-512.6](TR-512.6_OnfCoreIm-Physical.pdf)). The model in this document describes the memory functionality (capabilities and capacity).

### Storage

#### Challenges

The challenges in producing an abstract, standard storage model include :

* The large number of variations in storage options
* the lack of standard terminology.

For example, the definition of a LUN is problematic :

* “A LUN, is a number used to identify a logical unit, which is a device addressed by the SCSI protocol or Storage Area Network protocols which encapsulate SCSI, such as Fibre Channel or iSCSI[”  
  https://en.wikipedia.org/wiki/Logical\_unit\_number](https://en.wikipedia.org/wiki/Logical_unit_number)
* “a logical unit *number* (LUN) is a slice or portion of a configured set of disks that is presentable to a host and mounted as a volume within the OS.”  
  <https://www.computerweekly.com/answer/What-is-a-LUN-and-why-do-we-need-storage-LUNs>

#### Storage Options

Storage can be provided in many forms, some of the options commonly used today are shown below.

With each of the options, there could be more than one protocol used, and the diagram shows some of these in pink.

A screen shot of a computer

Description automatically generated

Figure 1 – Storage Options

#### Data “at rest”

Both storage and memory allow data to be ‘at rest’, ready for later retrieval. Memory can perform as volatile storage “RAM drives” or “RAM disks” and storage devices can be used as “virtual memory” where volatile memory is paged in and out of disk to increase the apparent amount of main memory.

The storage model is used for both storage and memory. It covers both the case where the access is to files, blocks or objects using a storage protocol and the case where access is to locations via the memory protocols.

## Storage Extent

In the model set out in this document, StorageExtent is defined as the key unit of storage capacity that the rest of the model is built around. The extent is a block or segment of storage (contiguous bytes).

The model covers ranges of extents.

## Partitioning and Aggregation

The model supports both partitioning and aggregation of StorageExtent ranges.

The simple example below, where StorageExtent range is represented as a ‘piece of tape’, should help clarify the concepts.

A green square with black background

Description automatically generated

Figure 2 - StorageExtent as a ‘piece of tape’

The tape can be aggregated in two ways, by end-to-end concatenation and by striping.

A colorful rectangular shapes on a black background

Description automatically generated

Figure 3 – StorageExtent concatenation and striping

The tape can be cut into sub extents, i.e., can be partitioned (the opposite of concatenation).

A green and black rectangles

Description automatically generated

Figure 3 – StorageExtent partitioning

Note that the operations are ‘closed’ that is both the inputs and the results of the operations are StorageExtent ranges, allowing the operations to be performed recursively.

## Storage Pooling

Originally, due to the limitations of the hardware, only local storage was available. As technology evolved, the support of shared storage (provided over a network) became feasible. Therefore, the model needs to support pooling of physical storage that can then be allocated logically to various consumers.

To do that, the model defines a ComputePool with ComputePoolEntries.

Note that the decision was made to have a single compute pool rather than separate Storage, CPU and memory pools[[1]](#footnote-1), because :

* CPU and memory are usually tightly coupled, and the pool can then allocate these consistently
* *Sometimes* storage is tightly coupled with CPU and memory and the pool can then allocate these consistently

There are two types of entries :

* Pool inputs for SSD, PhysicalDisk, VM VirtualDisk and LogicalEntry
* A pool output as an extent allocation (volume)

Note that the extent allocation from a pool can be a LogicalEntry to another pool allowing for allocation chaining.

Note also that the pools aren’t hierarchical (deliberately no ComputePool contained in self-join)

* The association StorageExtentPoolEntryIsLogical allows an output from one pool to become the input of another pool
* This needs to form a directed acyclic graph (no loops)

Note that there is no association linking the pool inputs and outputs. The ordering of the inputs allows the input to output extent mapping to be determined.

It is assumed that there will be many simple pools rather than few large complex pools with complex mappings.

# Compute model and context

## ComputeConstruct positioning

The following figures set out the core of the compute model.

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

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

[/for]<drop/>

### ComputeConstruct

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

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

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

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

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

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

## ComputePool

This part of the model allows for basic allocation of compute resources.

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

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

[/for]<drop/>

### ComputePool

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

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

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

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

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

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

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

[else]

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

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

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### ComputePoolInput

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

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

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

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

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

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### ComputePoolOutput

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

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

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

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

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

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### ComputePoolSegment

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

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

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

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

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

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### ComputePoolTransferFunction

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

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

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

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

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

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### RoleInPool

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

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

[if (dt.name.contains(‘RoleInPool’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertEnums()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

## Compute model data types

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

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘Compute-DataTypes’, ’Compute Data Tyoes’)/]

[/for]<drop/>

### ProcessingProperties

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

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

[if (dt.name.contains(‘ProcessingProperties’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertDataType(dt.name,’Compute’)/] <drop/>

Inserts the attributes of the data type <drop/>   
[dt.insertDataTypeAttributeTableBrief ()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### StorageProperties

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

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

[if (dt.name.contains(‘StorageProperties’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertDataType(dt.name,’Compute’)/] <drop/>

Inserts the attributes of the data type <drop/>   
[dt.insertDataTypeAttributeTableBrief ()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### StorageStrategy

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

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

[if (dt.name.contains(‘StorageStrategy’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertDataType(dt.name,’Compute’)/] <drop/>

Inserts the attributes of the data type <drop/>   
[dt.insertDataTypeAttributeTableBrief ()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### ErrorCorrectionStrategy

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

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

[if (dt.name.contains(‘ErrorCorrectionStrategy’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertDataType(dt.name,’Compute’)/] <drop/>

Inserts the attributes of the data type <drop/>   
[dt.insertDataTypeAttributeTableBrief ()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### ExtentRange

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

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

[if (dt.name.contains(‘ExtentRange’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertDataType(dt.name,’Compute’)/] <drop/>

Inserts the attributes of the data type <drop/>   
[dt.insertDataTypeAttributeTableBrief ()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### Media

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

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

[if (dt.name.contains(‘Media’))]<drop/>

[if (dt.name.contains(‘MediaType’))]<drop/>

[else]

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertDataType(dt.name,’Compute’)/] <drop/>

Inserts the attributes of the data type <drop/>   
[dt.insertDataTypeAttributeTableBrief ()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### Lifetime

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

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

[if (dt.name.contains(‘Lifetime’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertDataType(dt.name,’Compute’)/] <drop/>

Inserts the attributes of the data type <drop/>   
[dt.insertDataTypeAttributeTableBrief ()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### SpeedCharacteristic

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

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

[if (dt.name.contains(‘SpeedCharacteristic’))]<drop/>

[if (dt.name.contains(‘SpeedCharacteristicName’))]<drop/>

[else] <drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertDataType(dt.name,’Compute’)/]<drop/>

Inserts the attributes of the data type <drop/>   
[dt.insertDataTypeAttributeTableBrief ()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### Status

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

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

[if (dt.name.contains(‘Status’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertDataType(dt.name,’Compute’)/]<drop/>

Inserts the attributes of the data type <drop/>   
[dt.insertDataTypeAttributeTableBrief ()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### AccessStrategy

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

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

[if (dt.name.contains(‘AccessStrategy’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertEnums()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### ReadWriteStrategy

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

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

[if (dt.name.contains(‘ReadWriteStrategy’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertEnums()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### ApplicationRole

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

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

[if (dt.name.contains(‘ApplicationRole’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertEnums()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### SpeedProfile

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

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

[if (dt.name.contains(‘SpeedProfile’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertEnums()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### SpeedCharacteristicName

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

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

[if (dt.name.contains(‘SpeedCharacteristicName’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertEnums()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### InstructionSet

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

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

[if (dt.name.contains(‘InstructionSet’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertEnums()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### ProcessorArchitecture

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

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

[if (dt.name.contains(‘ProcessorArchitecture’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertEnums()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### Availability

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

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

[if (dt.name.contains(‘Availability’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertEnums()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### MediaType

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

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

[if (dt.name.contains(‘MediaType’))]<drop/>

Inserts the details of the class in first quotes from the package in second quotes <drop/>  
[dt.insertEnums()/]<drop/>

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

## Relationship to File System and Software

The following diagram shows the relationships between this compute model and other existing models, such as the software model described in [TR-512.6](TR-512.12_OnfCoreIm-Software.pdf).

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

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

[/for]<drop/>

# A simple compute example

This simple example shows how the concepts in the model fit together. The figure below shows an assembly of hardware.



Figure 5 - Simple Compute hardware example

The instance representation below shows how a slice of the capability of the hardware can be represented using the compute model. Not all of the hardware is required in the slice depicted.

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

Inserts the diagram identified in first quotes with the title identified in second quotes <drop/>  
[p.insertStandardDiagram(‘Compute-SimpleCompute’, ’Compute example instance diagram’)/]

[/for]<drop/>

# Model considerations

## Pooling

The decision was made to have a single compute pool rather than separate Storage, CPU and memory pools[[2]](#footnote-2), because :

* CPU and memory are usually tightly coupled (vie the CPU and memory buses) and the pool can then allocate these consistently
* *Sometimes* storage is tightly coupled with CPU and memory and the pool can then allocate these consistently (local attached un-sharable storage)

The pools aren’t hierarchical, i.e., there is not a self-join association on ComputePool (deliberately), because:

* The association StorageExtentPoolEntryIsLogical allows an output from one pool to become the input of another pool
* The structure needs to form a directed acyclic graph (no loops)

Note that there is no association linking the pool inputs and outputs. The ComputePoolTransferFunction will determine the order and association between inputs and outputs. In addition, the ordering of the inputs allows the input-to-output extent mapping to be determined in simple cases.

It is assumed that there will be a large number of simple pools rather than few large complex pools with complex mappings.

## Partitioning and Aggregation

The partitioning and aggregation of CPU and memory is subtly different from that of Storage.

With Storage, in theory each extent can be considered separately. Some extents could come from a disk A, some from disk B. With a sensible pool allocation there is no need to worry about segmenting the pool.

With CPU and storage, we have 2 issues :

1. The pool needs to remember and enforce segmentation (resource chunks)
2. The pool needs to pair CPU and Memory segments

For example, if there are 2 blade servers Blade-A and Blade-B.

* A software process cannot be allocated a CPU from Blade-A and memory from Blade-B.
* A software thread cannot be allocated half its CPU requirements from Blade-A and half from Blade-B.
* A software thread cannot be allocated half its memory requirements from Blade-A and half from Blade-B.

A multi-threaded application may be able to run across multiple CPU/memory pairs, but it would have to know the segmentation.

This model will use the following :

* ComputePoolSegment is defined to allow the definition of ‘segments’
* Inputs to the pool can be optionally assigned to a segment
* Each pool input/output may optionally be related to one segment
* If an input entry relates to a segment, then no output can be assigned that ‘crosses’ the segment (i.e., each output can only relate to 1 segment)
* Multiple inputs can only be combined if both do not relate to a segment or if both relate to the same segment
* If an input entry is related to a segment, then it should be propagated to any outputs that relate to the same segment
* If a CPU output relates to a segment, then it can only be used with memory that relates to the same segment (and visa-versa)
* Each segment should have its own internal number range in the pool

## Items for Further Investigation

### ComputePoolTransferFunction

This is currently a placeholder. The transfer function model will be developed in a future release. The model should take advantage of the work on the modeling of Task and ViewMappingFunction both of which have elements of transfer function. The work will also need to leverage the spec model patterns.

The transfer function in this model is intended to be basic. A more sophisticated form could be developed.

### Application of specification model

The spec model approach should be applied to all aspects of this model.

Some of the properties of the compute and storage could be moved to spec model occurrences.

### Physical model considerations

The physical model should be extended to cover media type which should then be removed from this document.

Rotation properties should also be considered in the context of the physical model and removed from this document. However, application of rotation properties may not be straightforward as rotation is physical behavior and the current physical model focusses (intentionally) on physical inventory. The rotation consideration has similar challenges to temperature and power (both of which overload the physical model).

Similarly, the concept of removable media belongs to the physical model. This could be covered by the equipment in holder structure.

### Component-System Pattern

The component-system pattern has been mentioned in this document, but it has not been fully expanded in the model described in this document.

The ComputeConstruct should align more strongly with the component-system pattern and the “..emergent..” associations should be worked further (as there is mapping complexity hidden in these associations). It could be argued that the ViewMappingFunction is necessarily present in the relationship between a system and the apparent emergent component. This should be explored further.

### Other areas

What units should be defined for memory sizes, for CPU clock speed etc.

Note that:

* Kubernetes works in units of CPU, where “One CPU, in Kubernetes, is equivalent to a *Hyperthread* on a bare-metal Intel processor with Hyperthreading”
* A CPU hardware thread is also called a vCPU (virtual CPU)

Zone size needs to be defined (block, sector, byte…).

The implications of address and data bus limitations needs to be explored.

**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\_v1.3\_OnfCoreIm-CommonGendocTemplate-Fragments.docx <drop/>

Template version 0.0.10 17 September 2017 <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/]

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

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

[/for]<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 7-1 [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 8-1 [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 1: 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 1: 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] <drop/>  
[/if] <drop/>  
</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 1: 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/>

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. Note that this document only considers storage entries. [↑](#footnote-ref-1)
2. Note that this document only considers CPU and memory entries. [↑](#footnote-ref-2)