<config>  
<output path=’C:\Users\ndavis\git\OnfInfoModelOutput\ModelDescriptions\TR-512.5\_OnfCoreIm-Resilience.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-512.5

Resilience

(Protection, Restoration and Recovery)

Version 1.4

November 2018

ONF Document Type: Technical Recommendation

ONF Document Name: Core Information Model version 1.4

**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)

©2018 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](#_Toc456952626)

[Open Networking Foundation 2](#_Toc456952627)

[Document History 4](#_Toc456952628)

[1 Introduction 4](#_Toc456952629)

[2 References 4](#_Toc456952630)

[3 Definitions 4](#_Toc456952631)

[4 Conventions 4](#_Toc456952632)

[5 Introduction to the Resilience Model 5](#_Toc456952633)

[5.1 Understanding the figures 5](#_Toc456952634)

[6 Resilience model detail 5](#_Toc456952635)

[6.1 Resilience Pattern 5](#_Toc456952636)

[6.1.1 FcSwitch 6](#_Toc456952637)

[6.1.2 FcRoute 7](#_Toc456952638)

[6.1.3 ConfigurationAndSwitchController 7](#_Toc456952639)

[6.1.4 ControlParameters\_Pac 8](#_Toc456952640)

[6.2 Resilience Attributes 8](#_Toc456952641)

[6.2.1 ForwardingConstruct (FC) 9](#_Toc456952642)

[6.2.2 FcPort 9](#_Toc456952643)

[6.2.3 FcSwitch 10](#_Toc456952644)

[6.2.4 FcRoute 11](#_Toc456952645)

[6.2.5 ConfigurationAndSwitchController 11](#_Toc456952646)

[6.2.6 ControlParameters\_Pac 12](#_Toc456952647)

[7 Resilience Model application 13](#_Toc456952648)

[7.1 Key to diagrams 13](#_Toc456952649)

[7.2 1?1 cases 13](#_Toc456952650)

[7.3 1?1 open protection cases 19](#_Toc456952651)

[7.4 1:N and N:1 cases 22](#_Toc456952652)

[7.5 Signaling information flow 26](#_Toc456952653)

[7.5.1 Closed case 26](#_Toc456952654)

[7.5.2 Open case 27](#_Toc456952655)

[7.5.3 Signaling control 28](#_Toc456952656)

[7.6 Foldaway of complexity 28](#_Toc456952657)

[8 Equipment protection considerations 29](#_Toc456952658)

[9 Resilience conveyed by Spec 29](#_Toc456952659)

[9.1.1 Forwarding Construct Specification and other details of Forwarding 29](#_Toc456952660)

[10 Resilience Work in progress 33](#_Toc456952661)

List of Figures

[Figure 6-2 Multiple open switch case with one client LTP 9](#_Toc460201385)

[Figure 6-2 Sharing FcPorts and switch orientation convention 10](#_Toc460201386)

[Figure 6-2 Instance diagram key 12](#_Toc460201387)

[Figure 6-2 Simple summary example of 1?1 cases (represented via partition) 13](#_Toc460201388)

[Figure 6-2 Showing detail of a single ended view of 1+1 and 1:1 switches 13](#_Toc460201389)

[Figure 6-2 Showing an emergent abstract controller in a 1:1 case 14](#_Toc460201390)

[Figure 6-2 Showing a basic route based representation of the 1?1 protection scheme 15](#_Toc460201391)

[Figure 6-2 Showing a two level route based representation of the 1?1 protection scheme 16](#_Toc460201392)

[Figure 6-2 Showing the preferred route based representation of the 1?1 protection scheme 16](#_Toc460201393)

[Figure 6-2 Route based representation of the 1?1 protection scheme showing C&SCs 17](#_Toc460201394)

[Figure 7-9 Showing detail of a single ended view of 1+1 and 1:1 switches in a route context 18](#_Toc460201395)

[Figure 6-2 Single ended view of 1:1 switches in a route context with peer C&SC coordination 19](#_Toc460201396)

[Figure 7-11 Nodal controller peering in a route context 20](#_Toc460201397)

[Figure 6-2 Simple summary example of open 1?1 cases 20](#_Toc460201398)

[Figure 6-2 Showing an emergent abstract controller in an open 1?1 21](#_Toc460201399)

[Figure 6-2 Simple summary example of open 1?1 cases showing route approach 22](#_Toc460201400)

[Figure 6-2 Single ended view of open 1:1 switches in a route context with peer C&SC coordination 23](#_Toc460201401)

[Figure 6-2 Nodal controller peering in a route context 23](#_Toc460201402)

[Figure 6-2 Simple summary example of 1:N cases (represented via partition) 24](#_Toc460201403)

[Figure 6-2 Showing detail of a single ended view of 1:N line system 25](#_Toc460201404)

[Figure 6-2 Showing an emergent abstract controller in a 1:N case 26](#_Toc460201405)

[Figure 6-2 Showing route based representation of the 1:N protection scheme 27](#_Toc460201406)

[Figure 6-2 Showing Extra Traffic in a route based representation of the 1:N protection scheme 27](#_Toc460201407)

[Figure 6-2 Showing independent two ended view of W1 route detail in a 1:N protection scheme 28](#_Toc460201408)

[Figure 6-2 Showing a unidirectional N:1 scheme fragment 29](#_Toc460201409)

[Figure 6-2 1:1 independent two ended view considering explicit signalling 32](#_Toc460201410)

[Figure 6-13 Sketch of two routes with internal resilience 33](#_Toc460201411)

[Figure 6-12 FcRoute in a complex network 33](#_Toc460201412)

[Figure 6-13 Relationship between FcSwitch approach and ProtectionGroup approach 34](#_Toc460201413)

[Figure 6-2 Speculative view of timing model 35](#_Toc460201414)

[Figure 6-2 Speculative simplified view of timing model 35](#_Toc460201415)

[Figure 6-2 1:1 independent two ended view showing FcPortFedByRoute 36](#_Toc460201416)

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 | No changes. |

# Introduction

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

## References

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

## Definitions

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

## Conventions

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

* UML conventions
* Lifecycle Stereotypes
* Diagram symbol set

## Viewing UML diagrams

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

## Understanding the figures

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

# Introduction to the Resilience Model

The focus of this document is the modeling of resilience in the ONF-CIM.

This document:

* Introduces the resilience model structure
* Describes the key classes of the resilience model
* Explains the attributes of the resilience model
* Shows how the model deals with various resilience schemes
* Explains how the specification model describes resilience schemes (protection etc.)
* Highlights work in progress to further advance the resilience model

The resilience model builds on aspects of the Core Network Model related to Termination and Forwarding described in [TR-512.2](TR-512.2_OnfCoreIm-ForwardingAndTermination.pdf) and related to Topology [TR-512.4](TR-512.4_OnfCoreIm-Topology.pdf). Resilience capability and other specification considerations are covered in [TR-512.7](TR-512.7_OnfCoreIm-Specification.pdf).

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

## Some working definitions

**Resilience:** A mechanism that ensures greater availability of a provided capability than could be achieved by use of a minimal set of dedicated resources. The mechanism uses additional resources to ensure that the provided capability continues to be provided even when one or more resource(s) originally used to provide the capability fails. The degree of failure supported depends upon the scheme. The time taken to recover depends upon the scheme. Several schemes may be used together.

**Protection:** A resilience mechanism where the resources used to achieve resilience against failure are in place and running ready to be selected so as to rapidly recover the service. The resources may be shared by several services such that under certain failure conditions one service may take the resilience resources from another causing the other to fail.

**Restoration:** A resilience mechanism where there are no additional resources over and above those needed to provide the capability in place and running but there is either a plan for resources to be used and/or there is a control capability that can determine which resources can be used to recover a failed service.

# Resilience model detail

## Resilience Pattern

The resilience model unifies a number of apparently different traditional model approaches that are used for various different resilience schemes (see {{ITU-T 808.1}}). The resilience model focus is the FcSwitch which represents the forwarding selector[[1]](#footnote-1) and which enables changes of forwarding to achieve resilience. The model also represents the control element of the resilience control loop that monitors behavior, assesses that behavior identifying necessary configuration changes and applies those configuration changes to make the required adjustments to Forwarding so as to achieve the intended resilience.

Some resilience schemes require combinations of control elements and switches. A particular pattern of combination of control elements and switches along with forwarding of control messages fully describe each scheme. This single uniform approach replaces the various traditional approaches (e.g. in some traditional representations a protection group is used, the protection group is replaced by one or more control elements in the new model).

### Resilience model in the context of other model additions to V1.3

The resilience model is a specialist model that represents the components of resilience at a similar level of abstraction to the LTP and FC. This release of the ONF CIM includes a number of new classes and has further explanation of patterns that related to this model.

As will be observed in the model described below, a key consideration is that of control of the resilience scheme. In V1.3 a generalized Control model has been added (see [TR-512.8](TR-512.8_OnfCoreIm-Control.pdf)). The control elements described in this document, the ConfigurationAndSwitchControl, can be seen as a specialist form of ControlComponent described in the generalized control model. The ControlComponent itself can be seen as a specialist form of ProcessingConstruct (described in [TR-512.11](TR-512.11_OnfCoreIm-ProcessingConstruct.pdf)) and ProcessingConstruct can be seen as a specialist form of Component (described in [TR-512.A.2](TR-512.A.2_OnfCoreIm-Appendix-ModelStructurePatternsAndArchitecture.pdf)). If the entire model was represented as Component this would be particularly opaque.

As a consequence it has been chosen to represent explicit classes to describe the specialist roles. The specialist roles are clearly still generalized forms. An implementer may choose to represent everything as component with layers of spec or use the explicit classes of the Core model. It is recommended to not specialize any further.

## Resilience Model

The figure below shows the key classes involved in modeling resilience and the associations between them.

[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/>

In the diagram:

* The classes shaded green are the classes that are solely present to support resilience
* The associations and classes shown in blue are new in this release.
* The blue and red associations are experimental.

The key classes present in the model that specifically support resilience are described in the following sections. The naming/identification classes, Local\_Pac and Global\_Pac, are discussed in section 6.5 Naming the ConfigurationAndSwitchController (C&SC) on page 7. The FcSpec class is included as it will be used to express the structure of the resilience scheme of the ForwardingConstructs, this is described in detail in [TR-512.7](TR-512.7_OnfCoreIm-Specification.pdf). See also [TR-512.2](TR-512.2_OnfCoreIm-ForwardingAndTermination.pdf) for an explanation of some key classes in the figure.

In this release:

* The C&SC has been extended with ports (reflecting the ComponentSystem pattern detailed in [TR-512.A.2](TR-512.A.2_OnfCoreIm-Appendix-ModelStructurePatternsAndArchitecture.pdf)). The port capability allows for representation of detailed signalling relationships and of asymmetric control (see 3.4.5 Symmetric and asymmetric C&SC on page 13). In basic control cases and for abstract representations the CaSC ports do not need to be expressed and can be omitted (being optional).
* The C&SC can be composed of C&SCs allowing for expression of complex control structures (see 3.3.1 Encapsulation of the ConfigurationAndSwitchController (C&SC) on page 9). This decomposition is also reflected at the C&SC port.
* The C&SC can provide a list of references to controlled FCs (see 3.3.5 C&SC Coordinates FC on page 13) providing clarity as to which FCs are controlled when the C&SC deals with a subset of FCs in the FD and the C&SC is not related to the FC by composition or via another subordinate C&SC. This allows aggregate statements expressed via attributes to be made at the C&SC and/or its ports about control effects on all FCs and/or their ports. As a result of the FC references and other rules the implications of the attributes can be interpreted and the effects on all related FCs can be determined.
* An FcPort can be internal (and hence not associate with an LTP) to allow chaining of switches (see 3.9 Overlaying and chaining switches on page 17).
* An FcPort can be associated with multiple switches to allow both chaining and other more complex arrangements (see 3.9 Overlaying and chaining switches on page 17).
* FcRoute is a GlobalClass.

The model in this release is a superset of that detailed in the previous release and all previous usages should be compatible with this release.

The following sections detail the key classes of the resilience model.

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

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

[if (cl.name.contains(‘FcSwitch’) or cl.name.contains(‘FcRoute’) or cl.name.contains(‘ConfigurationAndSwitchControl’) or cl.name.contains(‘ControlParameters\_Pac’) or cl.name.contains(‘CascPort’))]<drop/>

### [cl.name/]

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

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

## Key to diagrams

The following diagram highlights the symbols used in other diagrams in this document for various classes etc in the resilience model.



Figure 6-2 Instance diagram key

## Further explanation of the model

### Encapsulation of the ConfigurationAndSwitchControl (C&SC)

There are several degrees C&SC independence (see [TR-512.A.11](TR-512.A.11_OnfCoreIm-Appendix-ResilienceExamples.pdf) for examples of each):

* C&SC encapsulated in a FcSwitch
  + Used where a C&SC has a control scope of a single switch and where there is benefit in exposing ControlParameters for a single switch
  + The C&SC may participate in scheme where only per-switch autonomous control is available or may be part of a broader scheme with a hierarchy/mesh of C&SCs
  + The C&SC need have no id as it is identified in the context of the switch and there can only be one C&SC per switch
* C&SC encapsulated in an FC
  + Used where the C&SC has a control scope across several switches in the FC and where there is a need to have consistent parameters across those switches
  + This approach could be used for an FC with a single switch instead of embedding the C&SC in the switch
  + The C&SC may participate in a scheme as part of a hierarchy/mesh of C&SCs
  + The C&SC has a local id in the context of the FC. There may be several C&SCs in the context of an FC
  + The arrangement of C&SCs in the FC is described by the FcSpec (see [TR-512.7](TR-512.7_OnfCoreIm-Specification.pdf))
* C&SC encapsulated in a C&SC
  + Used where a complex control structure needs to be set out as a tightly coupled system of controllers
  + The C&SC has a local id in the context of the encapsulating C&SC
  + The usage if this is described in section 3.3.6 Relation between the ProcessingConstruct and the protection scheme on page 12
* C&SC encapsulated in an LTP
  + Similar to the FC cased but used where there is significant switching capability within the LTP
  + At this release there are no examples of usage for this capability
* C&SC stand-alone
  + Used where the C&SC coordinates switches and other configuration spread across multiple FCs etc
    - In this case it replaces the traditional protection group approach
  + There may be a hierarchy/mesh of C&SCs where a C&SC may govern others and may itself be governed
  + The C&SC may create/delete/adjust FCs as well as activate switches
  + The C&SC is part of the overall Management-Control Continuum (see [TR-512.8](TR-512.8_OnfCoreIm-Control.pdf))
  + The C&SC has a global id
  + The arrangement of C&SCs in a control scheme is described in a ControlSchemeSpec (see [TR-512.7](TR-512.7_OnfCoreIm-Specification.pdf))

This model fragment offers flexibility in the way the FcSwitch gains its ControlParameters and provides an instantiable C&SC that can be positioned with an appropriate scope of control for any particular case.

The control parameters for a number of C&SCs/Switches could be provided by a profile (although this area of model is for further development).

The C&SC can be included in a ConstraintDomain (CD) which may define the scheme that the C&SC is part of or which may simply apply common constraints to a number of C&&SCs. (see [TR-512.8](TR-512.8_OnfCoreIm-Control.pdf))

### An Open FcSwitch

The figure below (see section 3.3 Key to diagrams on page 9 for an explanation of the figure symbol set) shows an example of multiple open switches showing both legal and illegal settings.

The figure assumes a circuit switched technology and shows four cases of an NE[[2]](#footnote-2) with a protected signal flow to one client LTP (green) supported by an LTP (purple) bound to a physical port (on the left of each diagram). The cases highlighted are the two normal states of switches in the upper two diagrams, a transient state in lower left and an illegal state in lower right where the Configuration and Switch Controller (C&SC) has failed.



Figure 6-2 Multiple open switch case with one client LTP

### Sharing FcPorts and switch orientation convention

The diagrams in the figure below (in dotted red ellipses) illustrate usage of a mix of output and input switches (designated by “o” and “i" respectively). The modelling orientation convention is that the switch common is on the sharing FcPort if there is only one sharing FcPort (hence in some cases mixed ingress/egress switches are used). If there are two sharing FcPorts, or no sharing FcPorts the convention is that the input switch (default) is used unless there is specific complexity that can only be resolved with output switches.

See also figures in [TR-512.A.11](TR-512.A.11_OnfCoreIm-Appendix-ResilienceExamples.pdf) for more details on the specific case of use.



Figure 6-2 Sharing FcPorts and switch orientation convention

### Resilience Attributes

The figure below highlights the key attributes of the resilience model.

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

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

[/for]<drop/>

The attributes are described in the following tables (which show all attributes of the classes, not just the attributes related to resilience).

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

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

[if (cl.name.contains(‘FcSwitch’) or cl.name.contains(‘FcRoute’) or cl.name.contains(‘ConfigurationAndSwitchControl’) or cl.name.contains(‘ControlParameters\_Pac’) or cl.name.contains(‘ForwardingConstruct’) or cl.name.contains(‘FcPort’) or cl.name.contains(‘CascPort’))]<drop/>

#### [cl.name/]

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

[/if]<drop/>

[/if]<drop/>

[/for]<drop/>

### Symmetric and asymmetric C&SC

In release 1.2 all C&SC usages were essentially symmetric (see [TR-512.11](TR-512.11_OnfCoreIm-ProcessingConstruct.pdf) for explanation of symmetric and asymmetric) and hence the C&SC did not need ports. In release V1.3 the C&SC may optionally have ports and hence a number of cases that require asymmetric treatment of control are now supported. Like the FC, a C&SC port is associated with up to two LTPs (to allow for directionality differences)[[3]](#footnote-3).

The C&SC port to LTP association is used to represent several distinct flows:

* The flow of C&SC signaling information to/from the LTP where at the LTP it is propagated with the traffic and hence is to/from the adapter in the spec of the LTP
* The flow of control information to be applied to the LTP (e.g. disable traffic flow)
* The flow of monitoring information from the LTP to be used by the C&SC

The purpose of the port with respect to the flows covered is expressed via the ports role. A C&SC port can have a composite role and it may deal with several of the above flows if appropriate and where the same LTP is involved in all aspects described. The C&SC port role is described in the C&SC spec.

For rigid invariant patterned cases the relationship between C&SC port and LTP may be covered fully in the spec allowing a symmetric C&SC instance to be used. For more flexible cases explicit layout of instances of asymmetric C&SC port to LTP associations will be required.

The figure below, which is a sketch of an aspect of the {{ITU-T G.8032}} solution (see also [TR-512.A.11](TR-512.A.11_OnfCoreIm-Appendix-ResilienceExamples.pdf) for more details) shows a C&SC with ports related to an LTP for various purposes. It also includes associations described in the next section.



Figure 6-2 Figure showing C&SC with ports and association to FC

### C&SC Coordinates FC

In some schemes the C&SC may control FC flow indirectly via manipulation of an LTP associated to an FcPort and manipulation of switches in the FC associated with that FcPort. In these complex cases it is not sufficient to embed C&SCs in the FC. The focus of the control action is the combination of FcPort and LTP and it is based on the C&SC asymmetry exposed via the CascPort. The CascPort to LTP association carries some of this information[[4]](#footnote-4). Whilst it is possible to determined indirectly the C&SC association to the FC by examining the C&SC port to LTP associations and identifying the corresponding FCs, to cover these cases more explicitly a direct association between the C&SC and the FC is used. The figure above shows this association in use.

This association indicates which FCs a C&SC coordinates. Where there is a C&SC embedded in the FC (FcCoordinatedByCasc association) that is governed by a superior C&SC, that superior C&SC does not need to reference the coordinated FCs directly as the ControlGovernsControl association provides all necessary information.

### Relating the ProcessingConstruct , C&SC encapsulation and protection schemes

Where there is a complex behavior that does not fit the definition of one of the functional classes such as LTP and FC the ProcessingConstruct (PC) is used. The PC is described in more detail in [TR-512.11](TR-512.11_OnfCoreIm-ProcessingConstruct.pdf). The C&SC is essentially a PC and a ControlComponent (as noted in 3.1.1 Resilience model in the context of other model additions to V1.3 on page 8), but considering importance of network application resilience in SDN, the choice has been made to define a specific class[[5]](#footnote-5) to represent this behaviour instead of using the more general PC class.. The C&SC represents complex behavior of an assembly of parts where the emergent effect is that of Configuration and Switch Control[[6]](#footnote-6). Where a resilience scheme has a specific repetitive structure that is complex it may be beneficial to encapsulate the detail of the various C&SCs etc that enable the scheme in a superior C&SC. The scheme, including encapsulated C&SC and associations, is then describe in a spec structure (see [TR-512.7](TR-512.7_OnfCoreIm-Specification.pdf) for more detail). The complex structure may be summarized as defined in the spec (including key parameters) and may be exposed as a constrained hierarchy.

The figure below shows some alternative encapsulations of C&SC explored for the {{G.8032}} solution[[7]](#footnote-7). The actual {{G.8032}} solution is in [TR-512.A.11](TR-512.A.11_OnfCoreIm-Appendix-ResilienceExamples.pdf).



Figure 6-2 Figure showing basic groupings in CD and in C&SC

The figure above shows a Constraint Domain grouping LTPs, FCs and C&SCs in the “Simple Constraint Domain” diagram and shows the outer C&SC governing the inner C&SCs (via ControlGovernsControl) in the “Simple Control Hierarchy” diagram.

The figure below examines a more sophisticated encapsulation:

* The base model is as above
* The diagram of “The spec pattern (n cases of each item)” which identifies what to encapsulate in the spec, i.e. what would need to be specified for a single C&SC unit
* The diagram of “The spec pattern showing refactoring” highlighting an alternative structure of encapsulated C&SC
* The diagram of “The spec pattern…” shows the key parts and associations in the spec where there can be n of each item in the spec and each item will carry key attribute definitions

An approach that sets out explicit instances for each LTP, FC, C&SC etc as per the “Base model” would be reasonable as would an approach with a single instance of C&SC abiding by the spec shown in the diagram as “The spec pattern showing refactoring”. The spec pattern would need to be related to base model as shown in the “The spec pattern (n cases of each item)” such that if desired a recipient of the information could expand the model.



Figure 6-2 Figure showing a single C&SC encapsulating C&SCs defined by spec

Clearly it is preferable to have a single solution for each protection scheme. However, it is recognized that under particular constrained circumstances a representation of a fully capable realization may be verbose (i.e. expose details that are not of interest for the particular purpose) and hence there may be a desire to compact the representation. On this basis only recommendations are made for representation of particular schemes. These recommendations are in [TR-512.A.11](TR-512.A.11_OnfCoreIm-Appendix-ResilienceExamples.pdf). Clearly, any alternative representation of a scheme should be justified by there being some need to simplify the model due to limitations in capability and any representation of a scheme should abide by the model in terms of entities used and should provide full details in the spec model supporting scheme. This approach offering a variety of interpretable encapsulations is essentially the same as the approach used for the LTP etc (see [TR-512.A.2](TR-512.A.2_OnfCoreIm-Appendix-ModelStructurePatternsAndArchitecture.pdf)).

### Foldaway of complexity – Naming the ConfigurationAndSwitchControl

The ConfigurationAndSwitchControl can be:

* Embedded in an FcSwitch, a local class, essentially as a \_PAC with no need for ids etc.
* Embedded in an FC, a global class, essentially as a local class with need for only relative ids etc.
* Stand alone as a global class with need for a UUID

Where there is one switch controller in a context (e.g. a switch or an FC) and where the controller relates to the context entity by composition it is reasonable to fold the controller into the context entity.

* The context entity gains the controller attributes
* Any reference to the controller becomes a reference to the context entity

Where there are several switch controllers in a context but where those controllers do NOT need to be referenced in any way from outside the context entity it is reasonable to fold the controllers into a data structure within the context entity

* The context entity gains a structure of multiple controller attribute blocks
* The controller “instance” is resolved by position in the structure
* It is NOT POSSIBLE to reference a controller from outside the context entity

Where there are several switch controllers in a context and/or where those controllers need to be referenced from outside the context it is not possible to fold the controllers into the context entities but the entities representing the controllers can have a relative identification (localId) within the scope of the identifier for the context

* References are via an address with contextId and localId as elements

Where the switch controller is not in any stable (long lived) context then it must have a UUID and can be directly referenced via that UUID.

Hence it is necessary to use a mechanism that allows the class to have a variable id strategy. This is achieved using conditional composition rather than inheritance (this approach has only been applied here but may be relevant for other cases in the model).

### FcRoute has FCs and/or Links

There are two methods of describing the forwarding resources used by an FC to achieve forwarding across the network:

* Direct aggregation of FCs via FcHasLowerLevelFcs association where each FC exists in an FD/Link. The aggregation may be:
  + Single layer
  + Multiple layer where some of the FCs represent “Trails”[[8]](#footnote-8)
* Indirect aggregation of FCs and/or Links via the Route. Where the route is described by FC those FCs need not exist in an FD but instead may stand-alone describing some arbitrary fragment of the flow[[9]](#footnote-9).

The direct aggregation approach is the normal approach. The FCs in the Links are omitted and only the FCs in the FDs are provided.



Figure 6-2 Forwarding detail represented via direct aggregation (or partition)

In some cases the direct aggregation is not sufficient and a route mechanism is used.



Figure 6-2 Showing a basic route based representation using FCs

The FcRoute has several potential uses:

* As part of the constraint model related to routing an FC
* As a description of future alternative ways through the network to cover variability in the service need or some other where only one is active at any one time (as depicted above)
* To represent each of a number of alternative ways through the network for a particular FC to provide resilience
* As a description of the current way through the network for a particular FC (current route)

The FcRoute may be fully detailed or quite abstract in terms of constraints.

The key focus in this document is the use of FcRoute for resilience. The actual instantiated active route across the network, i.e. the actual configuration of the real devices, must necessarily be fully detailed (otherwise information could not flow). But the definition of the desired route can be just a set of constraints that the actual route simply needs to satisify. Similarly the alternative routes may be simply constraints.



Figure 6-2 Showing a basic route based representation using abstract FCs

Essentially the route constraints used in conjunction with knowledge of the necessary layout constraints for the type of FC should be sufficient to allow an instance to be created.

The degree of detail available in a route definition depends upon a number of factors including design philosophy and level of risk tolerated. For example, to minimize the risk of the route not being successful when application is requested full detail will be required and the resources will need to be dedicated.

If the route is itself complex including combinations of switched segments etc then FC orientation may be critical and hence will need to minimally include abstract FCs whereas if the route is a trivial point to point structure with no embedded protection and the network technology does not restricted channelization and there is no committed bandwidth etc then the route can suitable be described in terms of just links.



Figure 6-2 Showing a basic route based representation using Links

If there is committed bandwidth and there are restrictions in the FDs transited by the route then the route should be in terms of suitably detailed (partially detailed) FCs[[10]](#footnote-10).



Figure 6-2 Showing a basic route based representation using abstract Link FCs

### FcRoute LifecycleState

As implied in the previous section the FcRoute inherits the Lifecycle state attribute.

#### General considerations

The LifecycleStates of POTENTIAL and INSTALLED allow alternative routes reflect the selection state of a resilience scheme where INSTALLED means selected and POTENTIAL means deselected. When a route is selected it is available for use.

The LifecycleState of an FcRoute is:

* PLANNED: If the resources are not present in the network
* POTENTIAL\_BUSY: If the resources are present in the network but are shared with other FCs and are currently used by those FCs
* POTENTIAL\_AVAILABLE: If the resources are present in the network and are shared with other FCs but are not currently used by any FCs
* INSTALLED: If the resources are present and allocated to this FC (whether shared or not)
* PENDING\_REMOVAL: If the FcRoute is INSTALLED and the intention is to remove the FcRoute

#### Protection

From the perspective of a protection scheme it is usual for all resources for routes to be present in the network.

#### 1+1 Protection

In a 1+1 protection scheme, both of the worker (main) FcRoute and protection (standby) FcRoute have resources active in the network such that the LifecycleState of both will be INSTALLED (even if the route is not selected and there is no continuous traffic path as a result of switch states etc). The switch states are changed to select the route.

#### X:Y Protection

In a X:Y (X<= Y) scheme, although resources are present in the network but are shared and hence not necessarily available to protect a failure in a worker (main). A route may be POTENTIAL\_AVAILABLE if the resources are not currently used by any FCs and POTENTIAL\_BUSY when some or all of the resources necessary for protection are used by one or more other FCs.

When a route is “POTENTIAL\_AVAILBLE” then some other process is required to configure and activate the resources of the route before it can be used by a protection scheme[[11]](#footnote-11). It is possible that even if the state of the FcRoute is POTENTIAL\_BUSY a control process could have the authority to preempt and remove the blocking FcRoute.

#### Restoration schemes

In a restoration scheme there may be a number of alternative routes. At most one of those reoute will be INSTALLED. The other routes will be PLANNED. In some revertive schemes a preferred route (often called the home route) is remembered (pinned, retained) and the resources retained when an alternative is being used (due perhaps to failure of the preferred route). When the preferred route recovers the FC is caused to revert to it. When an alternative is being used the preferred route is POTENTIAL\_AVAILABLE (as it is not shared) and when it is used it is INSTALLED.

If the route is only described in abstract constraints then when it is INSTALLED the actual FCs abiding by the abstract constraints will be created. This actual FC will be added to the actual route and may become part of the route description for later re-instantiation if the policy for that FC indicates this should be the case.

#### Further considerations of state

It is currently not possible to distinguish, using states, between the following cases:

* All resources have been specified but not reserved
* Some resources have been specified but not reserved
* No resources have been specified

### Route Feeds FcPort

In some views it is possible that the detail below the route is not accessible but it is still clear to the viewer that there are multiple alternative routes. In these views it is beneficial to indicate to the viewer that a particular route is being used to feed the output from a particular FcPort.

The FcPortFedByFcRoute association reflected in the \_fcRouteFeedsFcPortEgress of the FcPort identifies which route feeds the FcPort.

The figure below shows a case where there are two routes. From the switch detail it is clear that the upper route is feeding to the right and the lower route is feeding to the left. However, the switch detail of the FC in the route, shown in the figure slightly greyed out, is not visible to the viewer. The \_fcRouteFeedsFcPortEgress attribute in each FcPort of the visible FC references the relevant visible route objects.



Figure 7-11 Understanding the active route in an opaque view

In more complex contexts there may be many routes and/or the upper-most FC may have multiple FcPorts. In these cases it is possible that multiple routes are actively feeding an FcPort such that information is flowing from at least one other FcPort on each of the routes[[12]](#footnote-12). In these cases the \_routeFeedsFcPortEgress in each FcPort simply list all routes that are actively feeding that FcPort.

### Abstraction of resilience viewed through the supported Link

The Link is an abstract view of underlying resources. It exposes the effects of the technology specific aspects and of the protection of the underlying network. The abstract view is in terms of the characteristics of forwarding and is covered in [TR-512.4](TR-512.4_OnfCoreIm-Topology.pdf).

The request for service will be in terms of the essential characteristics of forwarding e.g. timing, integrity etc. The need for a resilience mechanism will be interpreted by considering these characteristics. If the distance between the points of delivery is very short the characteristics may be achieved with unprotected resources, if the delivery points are very distant then the same characteristics may require some form of resilience. The resilience chosen will depend upon the specifics of the characteristics. Certain combinations of characteristics will not be achievable beyond a certain distance.

Because the resilience scheme is chosen to enable particular characteristics to be achieved then the abstraction of the scheme as viewed at the links should be in terms of those characteristics. It is not meaningful to express in the Link the type of scheme used in the underlying network because:

* This violates the intended opaqueness (as the user of the link abstraction would need to understand the meaning of a scheme in a foreign layer or domain)
* It does not provide the user with sufficient information to derive the characteristics of the scheme since these will depend upon many factors including length and these cannot be interpreted without deep knowledge of the underlying network

On this basis the expression of the underlying resilience viewed through the link should simply be in terms of generalized properties of forwarding.

Clearly, if the network is not hidden the user can navigate from the Link to the underlying network structures. The opportunity to navigate to the server FC and from that to the resilience scheme details is fully supported by the model[[13]](#footnote-13).

[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 Resilience-InterLayerSkeletonOverview’, ’Inter-Layer navigation’)/]

[/for]<drop/>

The figure above highlights two key relationships in red. These provide the most direct inter-layer resilience navigation. If an FcRoute is described in terms of Links the underlying FC in the server layer (that represents the Trail as per {{ITU-T G.805}}) can be identified by reverse navigation of the FcSupportsLink association. This will arrive at the supporting FC and all relevant resilience details. It is also possible to navigate via the Linkport to the LTP and then from the LTP to the supporting FC and all resilience details/

### Overlaying and chaining switches

The following figure shows an abstract example of an FC with chained switches using the internal FcPort where some FcPorts are fed by more than one switch. As it notes in the figure the layout is arbitrary, solely for the illustration of the model capability and is not a representation of any particular switching scheme. As usual the FC spec would explain, in terms of switched flows (which represent FcSwitches), the range of switching opportunities and distinguish the port roles.



Figure 6-2 Internal FcPorts and Ports fed by several switches

### Controls from CascPort

The CascPort supports both the sending of signaling through and/or application of control to the associated LTP and/or the gathering of monitoring information from the associated LTP. The controls can be applied directly to the associated LTP and/or indirectly to an appropriately deterministically related LTP peer or server to the associated LTP as described by the scheme spec (see [TR-512.7](TR-512.7_OnfCoreIm-Specification.pdf)). The same applies to the gathering of monitoring information.

Considering {{ITU-T G.8032}} protection as an example the control parameter related to the “isRelatedControlFlowDisabled” property of the port applies also to the indirectly related LTP dealing with the control signal and the “isControlledFcPortDisabled” property of the port applies specifically to the port of the controlled FC as explained by the scheme spec.

In addition the scheme spec will indicate whether the actual state of each individual controlled FC can be determined directly from the FC or whether only the aggregate state is available. Clearly the former may cause performance issues in an implementation if hundreds of FCs are controlled and switched together especially if notifications are sent for changes in every one independently.

### Use of FcSpec to explain unexpected flow through a protection scheme

The FcSpec is used to state rules for and constraints on flows through the FC so as to define the FC internal interconnectivity. The normal usage is to provide an FcSpec per type of FC. Clearly the intended flows in a protection scheme can be stated in terms of an FcSpec.

Under some failure conditions the flows in a protection scheme may not reflect the expected flows. Under these circumstances it is possible to use an FcSpec structure to describe the unexpected flows. Such an FcSpec could be made available as part of the description of the protection scheme if the failure modes are deterministic and the range of different flow patterns were limited.

In the case where the failure patterns are extensive, rare and not readily pre-calculable on occurrence of an unexpected flow state a temporary FcSpec could be constructed to express the current flow case.

The following basic network can be used to illustrate the complex behavior. The network includes four NEs. Each NE is configured as shown for the upper right NE with Interconnect-Protection[[14]](#footnote-14). The external view of the effect of the configuration of FCs in the NEs is Back-to-Back-Protection (depicted as Offered/Desired).



Figure 7-11 Basic network showing back to back protection abstraction of underlying protection

Under single failure conditions the external effect is still Back-to-Back-Protection



Figure 7-11 Single failure in network

When there is a failure of the input to the protection scheme operates as expected by the Dual 1+1 definition.as shown below.



Figure 7-11 Failure at an input to the network

However under certain internal failure scenarios the network is split into two and there is a non-desired flow. Although from the external perspective traffic is being delivered at both ports 3 and 4, external failures will not give the desired Backto-Back-Protection characteristic.



Figure 7-11 Two internal failures

A failure occurring on port 1 will, unexpectedly from the client’s perspective, cause the output at port 3 to fail.



Figure 7-11 Two internal failures with external failure

The figure below shows the potential states of flow of a realization of Back-to-Back-Protection under internal failure modes (some are dependent on their being a more sophisticated underlying network than that shown in the figure above). The states highlighted in the red ellipse are not expected from the simple external presentation of Back-to-Back-Protection. Only one direction of flow is shown to reduce clutter, the diagrams have been simplified to show only the flow (not the specific switches and the port numbers have been generalized (maintaining the orientation as per the the Back-to-back-protection shown in the figures above). This is further explained later in this section.



Figure 7-11 Representation of forwarding under normal and failure conditions

The FcSpec (see [TR-512.7](TR-512.7_OnfCoreIm-Specification.pdf)) can be used to represent the Back-to-Back-Protection scheme as shown in the figure below.



Figure 7-11 Spec for Back-to-BackProtection

The FcSpec can also be used to represent any of the desired and undesired forwarding patterns. An example is shown in the figure below.



Figure 7-11 Spec representation of one of the undesired cases

The figure below shows another one of the failure cases.



Figure 7-11 Spec representation of another of the undesired cases

### Dealing with multiple control domains (this section requires further work)

There are many network cases where an NE participates in several resilience schemes where two or more scheme instances are responsible for the resilience of a single information flow that transit the NE. The model supports two ways of dealing with this:

* Single FC with split control
* Two FCs each with separate dedicated C&SCs where the two FCs are connected by a zero-length-link

Two options… Zero length link or single FC.

# Protection schemes considered

The resilience model is designed to support standard resilience schemes in a consistent fashion. The model has been exercised for a number of schemes (see [TR-512.A.11](TR-512.A.11_OnfCoreIm-Appendix-ResilienceExamples.pdf)).

Resilience schemes considered in detail include:

* Linear protection including
  + 1+1
  + 1:1
  + 1:N
* Mesh protection including
  + N:1
* Ring protection including
  + {{ITU-T G.8032}}

The above schemes are protection schemes. Various restoration schemes are also supported by the model but these have not yet been covered in detail in the examples.

# Protection of other functions of physical things

The Physical model covered by [TR-512.6](TR-512.6_OnfCoreIm-Physical.pdf) which focuses on the modeling of Equipment. The Equipment is considered to be purely physical. The document also provides some modeling of functions that are emergent from a physical assembly when powered. Clearly, all functions including those encapsulated by LTPs and FCs are only realized in by a powered physical assembly.

The functions being supported by the equipment can be protected. This type of protection often goes under the name “Equipment Protection”. This name has not been used as it blurs the intentional constraint that Equipment is purely physical (where a physical thing can be measured with a ruler). Physical things are not protected, the functions that they support are protected; it is functions supported by additional (redundant) physical things that give rise to resilient/protected functions.

The [TR-512.6](TR-512.6_OnfCoreIm-Physical.pdf) document provides a sketch of how functional resilience could be represented. This aspect is for further work in the next release. The intention is to use a switch/controller based pattern to represent functional resilience/protection.

# Work in progress (see also [TR-512.FE](TR-512.FE_OnfCoreIm-FutureEnhancements.pdf))

## Signaling information flow

Some signaling flow considerations have been covered in this release but there is a need to cover the general case of inter-controller signaling. This will be tackled in a later release.

There are two distinct cases to consider:

* Closed: where the signaling/messaging is solely within the visible/controlled network
* Open: where the signaling/messaging emerges from the visible/controlled network

The open case occurs where, for example, there is an admin boundary the cuts a protection scheme and where the administrative entities have agreed to enable their management/control systems to exchange messages/signals to achieve inter-administration automation. This applies to B2B exchanged and E-NNI exchanges[[15]](#footnote-15)

### Closed case

* Current assumption is that a controller that uses signalling is identified in the appropriate spec
  + The model uses ControllerGovernsController in both directions to indicate a peer.
    - Attributes could be added to indicate whether the controller is signalling to a peer or not and that the signalling grouping is determined from the spec and switch orientation
* It is possible to show
  + Signalling flow through the network by associating the C&SC with an LTP as for the {{ITU-T G.8032}} model supported in this release
    - The LTP spec explains the adaptation and hence association with another C&SC can be derived
  + A direct peer association between C&SC with no view of the underlying mechanism
  + A full forwarding model for the signalling information flow
    - This could be in a referenced pattern that is the summarized rigorously in one of the above forms
    - The resilience scheme spec would explain the signalling flow alternatives
* Note that a full forwarding model is necessary when the signalling flow routing is not coincident with the traffic flow routing
  + An attribute could be added to indicate that signalling is co-routed with the traffic being controlled

### Open case

* This case has an open signalling path so there needs to be an expression of the signalling where it will emerge explaining what it is etc. Signalling information is exposed at the edge of the network
  + Again current assumption is that a controller that uses signalling is identified in the appropriate spec
    - Also with the attribute to indicate whether the controller is signalling to a peer or not and that the signalling grouping is determined from the spec and switch orientation
  + The ControllerGovernsController cannot name peer as it is not within the view so an off-net form of foreign pointer is necessary (or there could be a dummy controller with a few parameters (perhaps discoverable, perhaps manually entered) as well as the name)
* Potentially more relevantly in this case we could show
  + Signalling flow through the network by associating the C&SC with an LTP via a new association that indicates that signalling information is sent through the adapter of the LTP
    - The LTP spec would explain the adaptation and hence association with another C&SC could be derived
  + A direct peer association between C&SC with no view of the underlying mechanism
  + A full forwarding model for the signalling information flow
    - This could be in a referenced pattern that is the summarized rigorously in one of the above forms
    - Note that a full forwarding model would appear to make sense when the signalling flow routing is not coincident with the traffic flow routing
      * An attribute could be added to indicate that signalling is co-routed with the traffic being controlled

### Signaling control

* Need to identify parameters related to signalling and control that are independent of switching or only partly dependent on switching
  + Can timers be adjusted?
  + Can signalling be disabled?
  + Can aspects of signalling be disabled?
  + Can control be adjusted?

## Additional considerations for FcRoute

An FcRoute may:

* Be provisioned in the network but turned off
* Have resources reserved but not provisioned in the network
* Have resources that are reserved but shared with one or more other routes (either in the same FC or a different FC)
* Have specified but not reserved resources
* Have partially specified resources
* Have no resources specified and hence no subordinate FC detail

This implies the need to add properties on LifeCycleState (reserved, provisioned etc for the route) and to support a route in terms of constraints

An FcRoute may have encapsulated protection or other complex nesting of resilience schemes. Whilst the model supports this it has not been exercised with any cases. The figure below has a sketch of two alternative routes both of which have internal protection



Figure 6-13 Sketch of two routes with internal resilience

## Representation alternatives – Partition or Route

Consider the figure below of a simple network with relatively sophisticated switching scheme with a single FC spanning from A1 to C2.



Figure 6-12 FcRoute in a complex network

A1-C2 has four routes each of which has one FC

* Blue: A1-A2, B1-B2, C1-C2
* Red: A1-A3, D1-D2, B3-B2, C1-C2
* Green: A1-A3, D1- D3, E1-E2, C3-C2
* Brown: A1-A3, B1- B3, D2-D3, E1-E2, C3-C2

In more complex cases there could be many potential routes for a sophisticated switch configuration where there are only a few well defined switches.

Adding two more nodes and two more switches would double the number of routes. Adding more ends would further multiply the number of routes.

For complex layouts the route approach is not an efficient way of expressing the layout and instead the FC partition should be used.

If there are alternative FC partitions as a result of their being a combination of protection and restoration each FC partition will be considered as a route (where each route is composed of FCs).

## Relationship to the ProtectionGroup approach

The brief figure below sketches the relationship between a Protection Group approach and the FcSwitch. Further work is required to formalize the relationships.



Figure 6-13 Relationship between FcSwitch approach and ProtectionGroup approach

**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 0.0.10 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 6-2 [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 6-2 [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]  
[/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 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/>

#### [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. In this release only a selector that makes an absolute choice is explicitly supported. The switch can select multiple FcPorts at the same time and the assumption is currently that all selected ports are equally weighted. Through this the selector concept extends to cover high rate (e.g. packet rate selection from queues). An extension to this could allow a weighted sharing capability where the weighting may be based upon some algorithm. This would also apply to analogue cases (e.g. for power). Such an extension will be considered for a later release of the model. [↑](#footnote-ref-1)
2. The NetworkElement class has been deprecated in this release (see [TR-512.8](TR-512.8_OnfCoreIm-Control.pdf)). The term NE is used in a general sensein this document. [↑](#footnote-ref-2)
3. An association to the FcPort was also explored but not added in this release. [↑](#footnote-ref-3)
4. The LTP association can mean “controls”, “monitors”, “signals via” or some combination of the three. [↑](#footnote-ref-4)
5. This is true also for the LTP etc. [↑](#footnote-ref-5)
6. This is true for all classes in the model as explained in [TR-512.11](TR-512.11_OnfCoreIm-ProcessingConstruct.pdf) and [TR-512.A.2](TR-512.A.2_OnfCoreIm-Appendix-ModelStructurePatternsAndArchitecture.pdf). [↑](#footnote-ref-6)
7. Overlay of diagram entities represents an explicit relationship between the entities where the inferior entity is shown to the front. [↑](#footnote-ref-7)
8. A Trail is a forwarding relationship between Access Points (as per {{ITU-T G.805}}. In the ONF CIM it is represented by a ForwardingConstruct [↑](#footnote-ref-8)
9. The FC may only exist in the context of the Route and have a lifecycle dependent upon the existence of the route or may exist in several routes. [↑](#footnote-ref-9)
10. It can be argued that the route should always minimally in terms of abstract FCs in either a Link or FD context so that properties such as bandwidth commitment etc can be recorded. This will be explored further in subsequent releases to help normalize route expression. [↑](#footnote-ref-10)
11. This is a typical case for ASON restoration where signaling is used to trigger distributed control components to activate the standby path. [↑](#footnote-ref-11)
12. Clearly this is only applicable to some network technologies. [↑](#footnote-ref-12)
13. In many cases it is not possible/allowed to represent full server details, hence an abstraction would be used. [↑](#footnote-ref-13)
14. This is an illustrative FcSpecName. [↑](#footnote-ref-14)
15. The intention in the long term is to unify these two currently distinct considerations under one single architecture. [↑](#footnote-ref-15)