Core Information Model (CoreModel)

TR-512.7

Specification Model

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

©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.

Table of Contents

 © 2018 Open Networking Foundation

 © 2018 Open Networking Foundation

 © 2018 Open Networking Foundation

List of Figures

 © 2018 Open Networking Foundation

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	Enhancements to spec models for LTP and FC.

 © 2018 Open Networking Foundation

1 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.

1.1 References

For a full list of references see TR-512.1.

1.2 Definitions

For a full list of definition see TR-512.1.

1.3 Conventions

See TR-512.1 for an explanation of:

- UML conventions
- Lifecycle Stereotypes
- Diagram symbol set

1.4 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).

1.5 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 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.

 © 2018 Open Networking Foundation

2 Introduction to the Specification Model

2.1 Introduction to the ONF Specification approach

The focus of this document is the modeling of capability of managed-controlled things from a management-control perspective. The approach is guided strongly towards "outcome-oriented" interaction[1] where the focus is on stating the constraints that form a boundary that defines the desired result. In outcome-oriented interactions the operations/methods/activities/tasks used to achieve the desired outcome are firmly in the domain of the provider. The client simply provides information about the desired outcome in the context of what has been agreed as possible.

What is possible, i.e. the capability of the system, is also stated in terms of constraint oriented information including entities that can exist, values particular properties can take etc. Capability is considered in terms of properties that are expressed as observable or adjustable, the legal value of the properties and the interaction between the properties as well as properties that indicate creation/deletion opportunity.

The modeling of capability necessarily involves the modeling of constraints and rules, as a specific capability is always restricted in some way with respect to the maximum possible capability. The ONF Specification approach focusses on model of constrained capability.

The figure below shows a simple stylized pictorial model where the intention to achieve an outcome is expressed in terms of constraints that do not go beyond the bounds of the expressed capability restrictions.

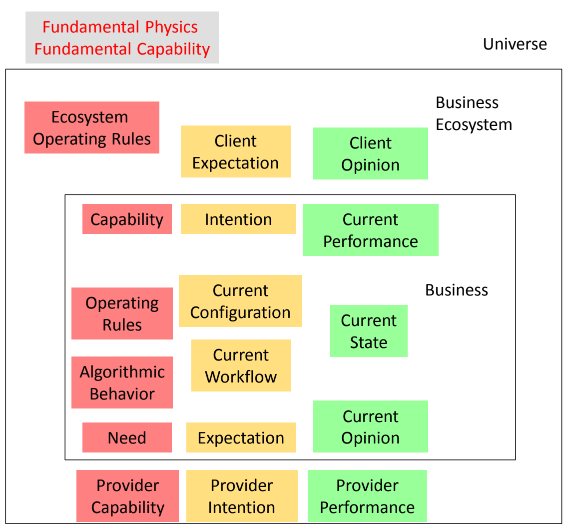


Figure 2-1 Stylized model of capabilities, intention and achievement

Considering the figure above, the restriction in capability offered (compared to ideal) may be due to:

- Provider capability

[1] Intent is an outcome-oriented form of interaction.

 © 2018 Open Networking Foundation

- Provider intention[2]
- Ecosystem operating rules
- Business operating rules
- Business behavioural capability
- Laws of physics
- Etc.

This pattern essentially applies at any level of view.

Regardless of the origin of the constraint, when exposed to the client it can be expressed in terms of Capability to achieve particular outcomes and this will be in terms of the properties of the things involved in the outcome.

Clearly a UML Class model provides a definition of capability in terms of things that can be created and values that can be set. Therefore, the Core Network Model is an expression of capability. However, the full Core Network Model goes way beyond the capability of any real solution. It is therefore necessary for any particular solution to be able to state its specific capabilities. The term used in the CIM for the modeling of capabilities is "Capability Specification" or "ONF Specification" or "Specification" or "Spec" for short.

As will be seen, the Specifications will be stated in the form of UML Class models. In some cases, attributes defined in the Specification will be added to an instance of a Core class, in other cases a Specification will result in addition of classes, refinement of attribute definitions and application of rules.

2.2 Rationale for, and features of, the ONF Specification approach

2.2.1 Formal definition of extension properties

It is often necessary to extend the capability of a function beyond that defined in a standard (i.e. with proprietary functionality). The approach uses a formal machine interpretable definition of all properties including proprietary extensions. This method is used instead of the traditional approach of allowing extension via Name-Value Pairs (NVPs, i.e., "soft properties") that are essentially undefined or are defined only on paper.

2.2.2 Reduction in range of properties defined elsewhere

It is often the case that a real implementation of a capability will not support all features of a standard specification. The approach supports the redefinition of attributes to cover a sub-range of their original definition using formal techniques[3] to relate back to the original definition via a machine interpretable model.

[2] The word intent is often associated with the client, but there is a phase shift considering the definition of the word and the position of the boundary. A party intends to do something but needs support that results in an agreement with another party, the provider. The provider then intends to satisfy the agreement with the client and the client expects that the provider will satisfy the agreement. Hence the agreement is about provider intention.
[3] An approach similar to the "Pruning and Refactoring" approach used to generate TAPI from the Core. Ths is experimental at this stage.

 © 2018 Open Networking Foundation

There is no limit to the degree to which a mandatory attribute can be pruned. It is allowed to prune an attribute below some apparent minimal conformance level where that pruning still allows viable operation is a niche application (where the vendor may make a major saving as a result of the reduced capability and may choose to pass that saving on to the customer).

The approach allows statement relating properties (where the constraints statements can be of arbitrary complexity).

2.2.3 Removing the need to propagate, per instance, properties that are invariant for the type

In some solutions, a standard property may always have the same value for all occurrences of a type/case. Per-type/case invariant properties appear only in the spec not in the instance. A specific property may be invariant for one type/case but may vary for another type/case. As a consequence, depending upon the type/case a specific property will alternatively be defined in the spec:

- As invariant and not appear in the instance[4]
- As variable and appear in the instance
- As variable with specified default and be overridable in the instance potentially only appearing in some expressions when not set to default

2.2.4 Substitution for existing property definitions

It is sometimes the case that there may be restrictions in a property that is seen as fundamental to the object and hence is defined in the core model (e.g. the property name is defined as a string, but some implementations may have a limit to the character set that may be used). The approach allows a definition in a spec to override a definition in the class so as to substitute a new definition of the attribute in place of the original. The attribute definition in the spec will be a "formally" related to the original[5] and must abide by the rules for reduction and extension:

- An attribute that is defined as extensible can be formally extended using the spec (see also 2.2.9 Extension of a specific property on page 12)
    - E.g., an ENUM for Colour could be defined as extensible in the class and have one or more extension literals, e.g., "Cerulean", "Azure", defined.
- An attribute (extensible or not)[6] can be formally reduced using the spec
    - E.g., the Colour ENUM could have the literal "Brown" removed.

2.2.5 Combining properties from elsewhere in an instance

It is often the case that a particular capability abides by standards from several bodies. Multiple specs can be applied such that an instance of a class is an assembly of attributes drawn from multiple places (this could be from different standards bodies). The assembly of properties is interpreted to determine the capability, the label on the class is not particularly relevant. The

[4] For example, all instances of a type of equipment have the same manufacturer. The name of the manufacturer of the type is stated in the type spec only and need not be stated in each instance.
[5] Using a technique similar to Pruning & Refactoring.
[6] It is arguable that an attribute could be defined to not allow reduction.

 © 2018 Open Networking Foundation

method augments the instance of a general class with defined properties. There is no need to rename the class (as for sub-classing) as there are no hidden semantics[7].

2.2.6 Combining properties into one property

In some solutions, in narrow applications, what is a complex set of multi-valued attributes in the standard only requires a few combinations of values such that the full standard seems cumbersome and verbose. The mechanism supports the refactoring of the combination of several properties defined elsewhere (e.g. many attributes compacted into one) using "formal" techniques (P&R) to relate to the original definition via a machine interpretable model.

2.2.7 Deep model-based definition

Often there is a complex multi-stage derivation of one view from another[8]. Properties may be "pulled in" and a derivation from some more detailed model be explained using recursive P&R[9].

This aspect has not yet been exercised for the defined specification structures and is highly experimental at this stage.

2.2.8 Sophisticated traceability

To support complex multi-stage derivation of one view from another, specs can have specs[10] (using recursive P&R). This is useful where two standards bodies describe part of a technology and where there is some overlap such that a property from one body needs to be combined with a property from the other to give the complete attribute.

This aspect has not yet been exercised for the defined specification structures and is highly experimental at this stage.

2.2.9 Extension of a specific property

Often it is necessary to extend a standard property for some application as the standard has not yet covered the full semantic space of the property. Properties may be extended using P&R but only where the property is defined as extensible. The specification structure can convey an extended definition.

For example, a property that provides the protocol name will be extensible and could be extended by a vendor to include a proprietary protocol.

---

[7] Other than those that are always hidden dues to the lack of formal specification of telecommunications protocols and measures etc.

[8] Actually, there is always a complex derivation back to fundamental semantics and true automation (where machines close the control loops) will require derivation from fundamental semantics. For a machine to “understand” a property it needs to have meaning in terms of more fundamental semantics (recursively). A property without meaning is of no value (clearly a messenger may pass on a property that has no meaning to it but that has meaning to the eventual recipient).

[9] As noted there is a complete lack of formal specification of telecommunications protocols and measures such that there are no deep models to trace to recursively. It is hoped that as a side effect of this mechanism more formal machine interpretable definitions of protocols and systems will emerge.

[10] A spec is made of classes in a model and any class, structure of classes can have a spec.

 © 2018 Open Networking Foundation

2.2.10 Expressing constraints and assemblies

When designing a system there are rules that must be followed. Some rules are local policy choices but other are more fundamental (restrictions of component capability, universal restrictions, regulation). The approach supports the providing of constraints for connectability and explains valid assembles of components (physical and functional) in a system. The approach enables the construction of schemes, as descried later in this document, where each scheme represents a pattern of components that occurs often such that that pattern can be identified and/or the assembly of an instance that pattern can be appropriately constrained. Schemes can be intertwined to form complex assemblies.

2.2.11 Run-time application

Not all constraints can be known at control system design time[11]. The approach provides the ability to discover and interpret new capabilities run-time and allows the capabilities and properties of a particular instance to change on-the-fly. Clearly, specs known at design time of a control system could also be captured and delivered as part of the deployed solution.

2.2.12 Long-term vision

The long-term vision is that the development of this approach will support the emergence of generalized controllers that can interpret the meaning from layers of specs that explain the system capability in terms of very fine-grained parts. Ultimately any protocol is defined in terms of a recursion of machine interpretable models such that a new protocol can be discovered, interpreted and controlled without the need for upgrade of the controller.

2.3 The mechanism compared to other mechanisms

The mechanism is in part one of extension via "attribute decoration" as attributes definitions are provided in addition to those of the class definition for a particular case of use of the class. An instance of the case of use of the class will present attributes from the class definition and attributes from the specification definition. The class definition includes a generalized reference to specification to enable extension and so that an instance of the class will have a pointer to each applied specification(s). The user of the extended class will have all definitions necessary to allow full interpretation of the instance. The class makes no reference to any specific specification and hence the approach is different from the "conditional package" (_Pac) approach, where each extension is known upfront during the standard class definition activity and a reference to each potential extension is explicitly stated in an attribute of the class.

The mechanism is distinct from the "GoF decorator pattern"[12] in that operation extensions are not expressed. As noted earlier, the focus is on "outcome oriented" interactions (see TR-512.10) and, as a consequence, relevant operations are related to the expression of outcome and not to the expression of activity to achieve that outcome (as the choice of activity is delegated to the server). The expression of outcome requires a relatively basic set of operations with the focus very much on definition of the desired outcome in the form of constraints expressed in terms of entities and

[11] To enable appropriate solution agility, it is wise to not code specific structure but instead to focus on coding time-invariant patterns that can then be decorate and constrained.
[12] See https://en.wikipedia.org/wiki/Decorator_pattern.

 © 2018 Open Networking Foundation

their properties[13] where those entities are taken from a schema shared between, and understood by, the interacting parties. The outcome-oriented operations are not expressed on the entities of the shared understanding but instead are dealt with by controllers of those entities. The requesting party talks to a controller[14] about entities that it controls.

The mechanism, is similar to but, goes beyond "attribute decoration" in that it allows modulation of existing definition and assumes that invariant definitions will be only present in the specification classes and not in the instance of the specified case.

The approach and mechanism are not directly related to the "specification pattern[15]". The use of the term specification here is distinct from other known usages.

2.4 Introduction to this document

This document considers the general modeling of patterns for the representation of capabilities and constraints and specific patterns for representing capabilities and constraints for all entities represented in the ONF-CIM.

This document:

- Introduces the dedicated forms of capability specification, the primary model structure used to represent the capabilities and constraints that feature in the current model
- Works through the specification mechanism as applied to the following key classes:
    - FC (and Link)
    - LTP/LP
    - FD/Link
- Provides a view of work in progress on a generalized pattern for specification

The specification model relates to all other models

- Core Network Model (Termination and Forwarding) described in TR-512.2
- Foundation model (identifiers and naming) described in TR-512.3
- Topology model described in TR-512.4
- Resilience model TR-512.5
- Physical model TR-512.6
- Control model in TR-512.8
- Operations pattern model in TR-512.10

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

[13] As a consequence, all aspects of the outcome are stated via attributes. Even where there is a fleeting transient state requested, this is expressed in terms of attributes.

[14] The controller is also represented as an entity that itself can be controlled (see TR-512.8).
[15] https://en.wikipedia.org/wiki/Specification_pattern

 © 2018 Open Networking Foundation

3 Purpose and essentials of the specification model

3.1 Background

The Core model classes represent fully flexible capabilities. In real deployments, there are restrictions in capability due to various factors including the need for low cost specific solutions. The essential approach proposed is to associate an instance of a Core model class with a set of constraints that account for the specific case.

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

- Variety: Representing the set of rules related to capabilities and restrictions for each specific case of use/application of the model
- Extension: Enabling the introduction of run time schema where the essential structure of the core model is known up front (at design/compile time) but the usage/application specific details are not known. Subsequently, the detail is added via a run-time reference to a schema that describes attributes and structure that augment the core model at runtime. The attributes may:
    - Add invariant data to the definition of the class
    - Add variable data to the definition of the class that will be represented in the instance
    - Modulate the definition of existing attributes in the class
- Profile: Reducing the clutter in an instance representation where a set of details take the same values for all instances that related to a specific case (reference the case specification). This area is not developed in this release.[16]

The combination of the above resulted in a separation in the model of definitions of structure (with core common content) and variable content such that instance of classes from one model fragment could point to another model fragment to enable the acquisition of that fragment of definition of the class and its subordinates at run time.

This approach is not new and many key aspects were inspired by work in the TMF SID, partly described in [TMF SID 5LR].

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

Whilst the mechanism allows for proprietary extensions, the intention is that primarily standard forms be used to augment the model. The specifications handle:

- Vendor variety (pruning, extension) from standards
- SDO decoupling from the Core and each other's (transport technologies, e.g., ODU, ETH)

[16] The specific structures used in this release are applied such that the structures and values in the spec are unchanging. The specification structure described here is not intended to be used as a common point for configuration change. The common point for configuration change will be developed from the specification mechanism in a later release.

 © 2018 Open Networking Foundation

- Rules that emerge from an assembly of constrained parts
- Constrained roles
- Inter-role relationships

3.2 Cases considered

It was recognized that the specification capability would be required by all classes but that some key per class capabilities were especially important in model. The mechanism was developed using dedicated structures for the following cases (roughly starting in the order below – although clearly much of the work was in parallel):

- Forwarding Spec
    - Main focus to provide a representation of the effective internal structure of an FC accounting for:
        - Asymmetric flow between FcPorts
        - Arrangements of switches and controllers
    - Additionally, the representation of layer protocol specific attributes
    - Covers FC cases such as "Root & Leaf" and "Dual homed resilience"
        - Also, fully applicable to the Link as this reflects the structure of the supporting FC
- LTP and LP spec
    - Main focus to provide a representation of
        - Layer protocol specific parameters
        - Layer protocol adaptation hierarchy rules
        - Termination flexibility rules
    - Provides a mechanism through which to acquire technology specific definition from other specification authorities
        - To also enable proprietary extensions within a technology definition
    - A critical consideration here is that ONF don't own the technology definitions:
        - New technologies need to be introducible with no change to the core
        - Proprietary extensions need to be introducible
        - Approach is to combine pruning & refactoring with the spec model
- FD and Link spec
    - Main focus on capacity and forwarding enablement restrictions
- Equipment spec[17]
    - Main focus to provide a representation of:
        - Equipping constraints
        - Functionality emergent from Equipment configuration
- Generalized spec pattern
    - Main focus to provide a common representation of
        - The mechanism for relating a class to its spec, accounting for implementation needs

[17] Note that the Equipment spec work is currently in TR-512.6. It is likely that the capability definitions for the other aspects of the model will be moved to the documents that define those aspects and this document will eventually cover the generalized specification pattern model and usage of that pattern.

 © 2018 Open Networking Foundation

    - Categories of specification element of the specification via stereotypes[18] that guide tooling and solution code
  - Note that this has been developed by refactoring the class specific specifications forms (mentioned in the bullets above) to result in a single generalized specification form.
    - The intention is that this single generalized form be used in place of the specific forms and that it is shaped by data to represent the specific structure of each of the dedicated specification forms (and any further specification forms necessary)

3.3 Resultant representations and principles

It was recognized that UML itself augmented by stereotypes provides all necessary structure and definition capability to enable representation of the specifications[19]. It was also recognized that the key is in the representation of the relationship of a class that is to be instantiated to a model definition that is not known until run time such that the instance of the instantiated class can point to the model definition (schema) for parts of its content.

The approach uses association stereotype with a member-end attribute stereotype in the class model. This attribute stereotype indicates that the member-end attribute in an instance of a class will reference a case of a schema (class model fragment) instead of, in the normal usage, an instance of a class.

A specification is used to extend the class definition BUT is constrained such that each class in the model has a particular dedicated specification structure.

The specification structure will restrict the statements that can be made and may also restrict the source of the attributes that can be applied to the specification. For example the LP specification has a place where layer protocol specific attributes can be added but it is expected and required that these attributes be derived from a formal layer protocol definition by pruning and refactoring that definition[20].

It is vital that the use of the specifications be constrained so as to enable the extension capability without allowing a model free-for-all that eventually causes the model structure to be lost.

3.3.1 Composition rather than inheritance

There are several UML techniques for model structuring and extension. The work on specification essentially uses a composition approach to extend/augment the model (as opposed to inheritance).

In the normal usage of UML composition, the composed parts are known upfront during the model definition stage. When using the spec approach, the composed parts are controlled by the specification definition[21] and can be added run time. The attributes of the specifications are

[18] Currently highly experimental
[19] It appears that raw UML does not directly support any augmentation via extension schema mechanisms
[20] This will be described in detail later in the document.
[21] Some use is still made of conditional composition but it is expected that the specification approach will be used to drive all conditional content.

 © 2018 Open Networking Foundation

augmented with stereotypes that direct their specific application. Some attributes identified in the specification will appear in the instance and others will not (see section 4.6.3)[22, 23].

Using composition enables the specification structure to reflect and extend the structure of the class model (e.g. the LayerProtocol class gains further sub-structuring from the specification[24]). This approach also allows the specification to incorporate augmenting content by pruning and refactoring other related external models. So for example the specific properties of a particular layer protocol can be acquired from the definitions of the specification authority of that layer protocol (e.g. see [ITU-T G.874.1].

3.3.2 Specification model restructurings during pruning & refactoring

When a specification is being composed by deriving from an external model it is likely to be necessary to refactor that model to reposition attributes to fit the ONF model structure. During that refactoring, significant flattening of the model can take place. Inheritance hierarchies can (and in most cases must) be removed and [0..1] and [1] associations (primarily composition) can be folded (i.e. enable the contents of one class to transferred into the other related class).

The original structure is not lost as it can be re-acquired if beneficial[25] via the pruning and refactoring associations in the mapping model (see [ONF TR-513]).

In addition to flattening, representation transformations are likely to be required while composing a specification. For example:

- The ONF model exclusively uses a "switch and controller" approach for protection whereas some other models use a "protection group" approach. The other model will need to be refactored to the "switch and controller" form.
- The ITU-T models have a different granularity of TP modelling where the LP is modelled as several separate equal parts (TTP, CTP etc.). The ONF specification model is at a finer granularity than the ITU-T model.

When acquiring a property from external definitions via pruning and refactoring it is possible to narrow the property (reduced range etc.) but NOT broaden it[26]. The narrowing must maintain the essential semantics of the property.

3.3.3 Further discussion

The terms specification and template may be confusing (profile, template and specification are used inconsistently across multiple contexts and the definitions overlap with each other). From some perspectives, this is essentially a templating/substituting mechanism. In a way this is

---

[22] It is also intended that the specification mechanism is capable of modulating core attributes and structure. This will also be indicated via stereotypes

[23] The model entities will also provide data and stereotypes etc. to drive the operations content. This will be developed at a later stage

[24] It is expected that this specific sub-structuring will become part of the LP model itself

[25] Much structuring in the models examined was bundling of attributes rather than dependency graph or flow semantic based structuring. This does not allow for enhanced model interpretation over and above that that can be achieved from the attribute alone. Work is progressing slowly on dependency graph (and flow semantic) modelling.

[26] To add something new requires a new property even if the new thing appears to be extending an unextendible standard property.

 © 2018 Open Networking Foundation

"instantiating" a filled in template to form a further class structure. The "instance" of the filled in template defines the substructure of the LP/LTP and provides the definition of attributes.

This is a general principle and the pattern has been there for many years although not necessarily well formulated[27].

The specification method appears necessary for dynamic APIs.

There could be methods like 'verify' to validate that the instance of LTP supports the specification definition.

Specifications will be constructed by the designer/developer of the item being specified and will be available to the controllers via some interface. The hosting entity that provides the specification may be:

- The same as the one providing the interface that uses the specification
- The organization of the designer/developer of the item being specified
- Some central repository/library

3.4 Adoption and migration considerations

This section briefly discusses migration from traditional design time fixed solutions to a fully run time dynamic solution using specifications. The steps discussed are examples; there are many ways of approaching this.

3.4.1 Using as a traditional interface

As noted the specification provides a dynamic run time definition for use in a dynamic API context. However, it is quite possible to use the same APIs in a more traditional static mode.

In a traditional system, the entire schema has been coded prior to compile. The specification reference can be ignored run time and instead the content of the specification can be compiled into the systems on both sides of the interface.

In a slightly more sophisticated solution the specification reference can be used to validate the version compatibility of the interface.

Alternatively, assuming a JSON or XML encoded interface, the additional attributes could be transformed into simple name and value lists much like a traditional semi-dynamic API with soft properties.

3.4.2 Using with basic specs and prior knowledge

The specification, rather than being fully interpreted run time, could be compiled but decoupled, allowing for some dynamic behavior but not for previously unknown specs to be used.

In this case, the specification reference would allow selection of previously compiled schema and associated behavior.

[27] E.g. through “additional info” or “vendor extensions” NVPs added systematically in every class as “last attribute”.

 © 2018 Open Networking Foundation

3.4.3 Using with full specs with discovery

The spec pointer would be used to reference the library to acquire the necessary specification that would then be used to interpret the attributes that were not defined in the compile time schema.

The definition of the attributes will provide information on value ranges, defaults, writeability etc. From this, the application can determine whether to make available on a configuration UI or as part of a flexible profile etc.

Clearly more sophisticated usage will require special code but on arrival of the specification this code can potentially be identified, installed and be available to run on arrival of the first case of use of the specification.

3.5 Enabling innovation whilst removing unnecessary variety

As discussed, the specification mechanism allows the raw model to be augmented at run time. As noted the augmentation may be proprietary. The intention is that the source of the augmentation be identified. The intention is also that the extension is in a place allowed by the specification authority of the area being augmented (e.g. ITU-T for protocol definitions).

There is a challenge here of hitting the right degree of allowed augmentation so that innovation is in no way stifled, whilst model chaos is prevented. This will be a journey and learning experience.

4 Dedicated specification structures

It should be noted that much of this work is experimental.

4.1 Forwarding Specification

4.1.1 Overview of the Forwarding Spec

Prior to embarking on a brief description of the ForwardingSpec and associated classes, it is important to explain the ForwardingSpec in the context of the specification approach in general. In this model, the specification classes provide a mechanism to express the restrictions of a particular case of application of a particularclass or set of classes. For example an FC in general has[2..*] FcPorts while a specific case of FC may have exactly 4. This case may also be such that it has 2 internal switches and such that these switches affect specific flows in the FC. The ForwardingSpec is designed to allow the expression of cases of this sort.

The number of FcPorts alone does not adequately characterize the FC. The figure below shows two FCs both with exactly four FcPorts. The flows through the two FCs are clearly very different. Simply knowing the labels for the roles of the FcPorts does not provide sufficient information on the flows within the FC. To fully describe the FC behavior, it is necessary to represent the flows.

In effect, what we are doing is to model the internals of the FC in an abstract form so as to convey the effect of the internal behavior as perceived from the outside of the FC. Essentially,

 © 2018 Open Networking Foundation

the approach enables the creation of 'prototypical instances' that can then be applied to any FC or Link instances[28]. The figure below shows two distinct 'prototypical instances'.

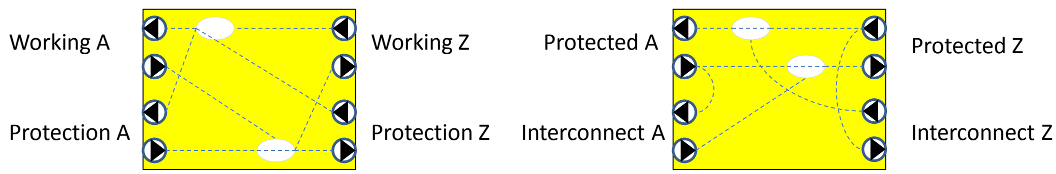


Figure 4-1 Two FCs with four bidirectional FcPorts

Considering the model described below, in the two cases in the figure above, the ForwardingSpec and supporting PortSetSpec describe the capabilities of the FC in terms of unidirectional flow elements, MultiSwitchedUniFlows, each of which has [1..*] IngressPortSets and [1..*] EgressPortSets. Each MultiSwitchedUniFlow may have [0..1] ingress switches and [0..1] egress switches, where the ingress switch may select only one member from the ingress set and the egress switch may select [1..*] members from the egress set. The ingress and egress switch selections are controlled by the ConfigurationAndSwitchController (see TR-512.5) that may be:

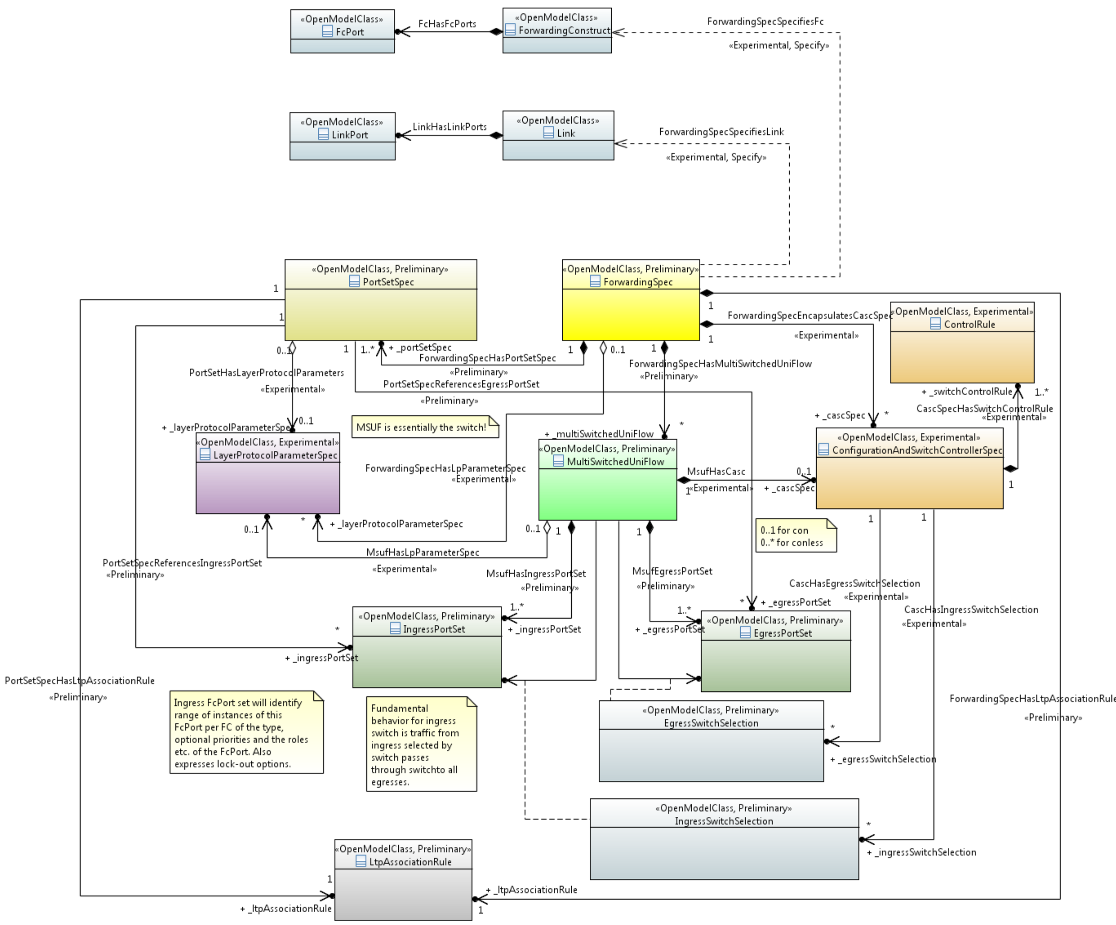
- embedded in the switch when there is no coordination of switches required
- embedded in the FC when the coordination of switches is only in the scope of the FC
- independent of the FC and described by the ConfigurationGroupSpec where there is multi-FC coordination required

The behavior of the ConfigurationAndSwitchController is described by the ConfigurationAndSwitchControlSpec and associated ControlRules.

The model has been exercised for a number of different cases (some illustrated later in this section). The figure below provides the class diagram of the Forwarding Spec fragment.

[28] This can be a run time augmentation. Consider a Controller running with a set of prototypical instances. A device with a capability beyond that set is to be controlled. The device provides a new prototypical instance (that describes internal flows etc.). The controller adds it to its set. When it interprets flows for validity, it now can interpret these new internal flows. It will be able to do loop detect etc. on existing FCs in the network that use this structure.

 © 2018 Open Networking Foundation



CoreModel diagram: Spec-CoreOfForwardingCapabilitySpec

Figure 4-2 Class Diagram of the core of the Spec Model of the FC and FcPort

Also shown in the figure above, are Link and ForwardingDomain, these will be discussed in a later section.

4.1.2 Class details

4.1.2.1 ConfigurationAndSwitchControllerSpec

Qualified Name:
CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::FcCapabilityCore::ConfigurationAndSwitchControllerSpec

The spec of a ConfigurationAndSwitchController.

Inherits properties from:

- GlobalClass

This class is Experimental.

 © 2018 Open Networking Foundation

4.1.2.2 ControlRule

Qualified Name:
CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::FcCapabilityCore::ControlRule

A rule describes the bounds of the behavior of a CASC.

Inherits properties from:

- LocalClass

This class is Experimental.

4.1.2.3 EgressPortSet

Qualified Name:
CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::FcCapabilityCore::EgressPortSet

The grouping of FC egress ports that have the same behavior and relationship to the switch etc.
Will carry rules for the grouping.

Inherits properties from:

- LocalClass

This class is Preliminary.

4.1.2.4 EgressSwitchSelection

Qualified Name:
CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::FcCapabilityCore::EgressSwitchSelection

Rules for the control of the state of the egress switch.
This class is Preliminary.

4.1.2.5 ForwardingSpec

Qualified Name:
CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::FcCapabilityCore::ForwardingSpec

The overall spec for the forwarding entity.

Inherits properties from:

- GlobalClass

This class is Preliminary.

4.1.2.6 IngressPortSet

Qualified Name:
CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::FcCapabilityCore::IngressPortSet

The grouping of FC ingress ports that have the same behavior and relationship to the switch etc.
Will carry rules for the grouping.

 © 2018 Open Networking Foundation

Inherits properties from:

- LocalClass

This class is Preliminary.

4.1.2.7 IngressSwitchSelection

Qualified Name:
CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::FcCapabilityCore::IngressSwitchSelection

Rules for the control of the state of the ingress switch.
This class is Preliminary.

4.1.2.8 MultiSwitchedUniFlow

Qualified Name:
CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::FcCapabilityCore::MultiSwitchedUniFlow

A switched unidirectional forwarding element that can take one or more inputs and switch to one or more outputs.
The switch can also be open (high impedance).

Inherits properties from:

- LocalClass

This class is Preliminary.

4.1.2.9 PortSetSpec

Qualified Name:
CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::FcCapabilityCore::PortSetSpec

The specification of a set of equivalent port of the forwarding entity.
For example, there may be a several ports with exactly the same behavior with respect to each other and with respect to all other ports. These can all reference one PortSetSpec.
In a symmetric FC this means one PortsSetSpec can be used for all ports.

Inherits properties from:

- LocalClass

This class is Preliminary.

4.1.3 Enhanced Forwarding Spec to cater for the photonic model

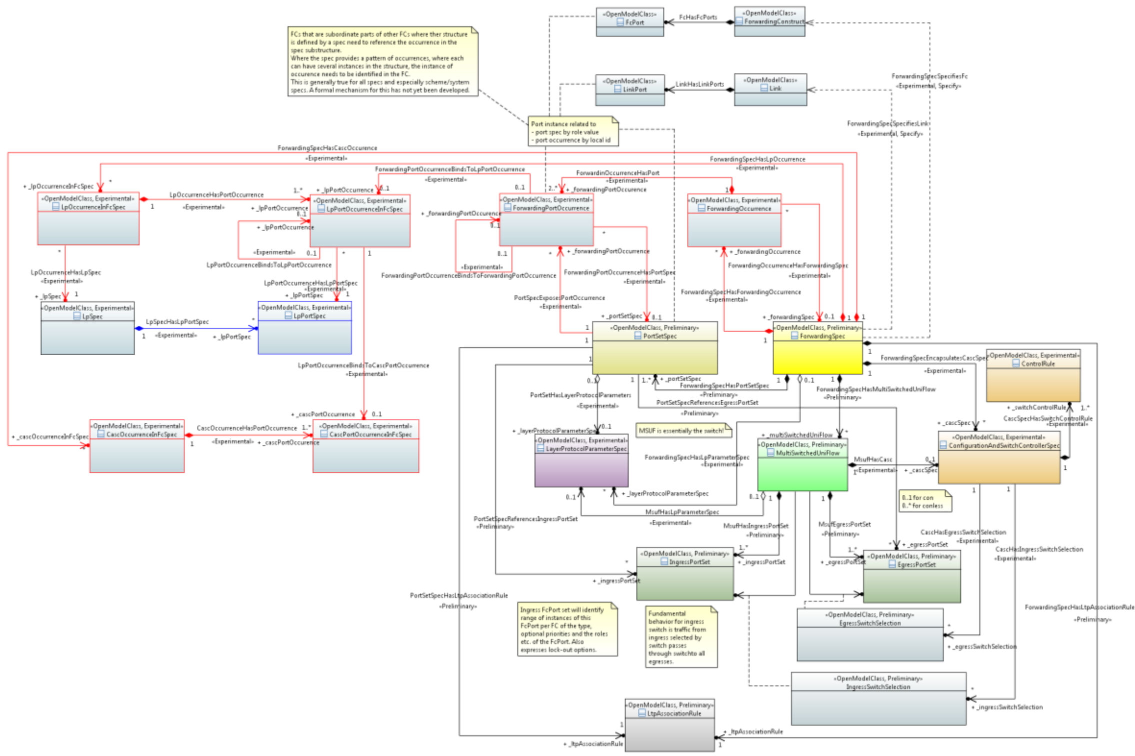
The next figure shows the full FC spec model with additions made in V1.4 highlighted in red. These additions:

- Enable specification of a recursion of ForwardingSpecs to describe a recursion of Forwarding via the ForwardingOccurrence:
  - A ForwardingSpec may contain many occurrences of Forwarding where:
    - Each ForwardingOccurrence may have two or more ForwardingPortOccurrences and where the ForwardingPortOccurrences may be:

 © 2018 Open Networking Foundation

        - bound together via ForwardingPortOccurrenceBindsToForwardingPortOccurrence
        - exposed at the containing level via PortSpecExposesPortOccurrence
      - Each ForwardingOccurrence will have a ForwardingSpec that may have a further decomposition
      - Any ForwardingOccurrences may have a definition in terms of MultiSwitchUniFlow etc. as for a basic FC
      - Each ForwardingOccurrence may have an instance representation as part of a recursion of FCs related via the occurrence id
  - To provide support for the complex FC assemblies described in TR-512.A.4
- Enable the specification of LPs at any level of the recursion such that part of the forwarding structure is terminated in some way
  - To enable support of termination in some of the complex FC assemblies described in TR-512.A.4
  - Where the termination is associated via a port to the ForwardingPortOccurrence via ForwardingPortOccurrenceBindsToLpPortOccurrence
  - Where the degree of termination is described in an LpSpec
    - Which itself allows for recursion of LpOccurrences and ForwardingOccurrences (see 4.2 LogicalTerminationPoint and LayerProtocol specification on page 38)
  - Where the structure is such that an FC cannot expose an LP other than as an effect via some Forwarding
    - i.e. the LP cannot directly access an FcPort
- Enable the specification of Casc related to the LpOccurrence

 © 2018 Open Networking Foundation



CoreModel diagram: Spec-FullForwardingCapabilitySpec

Figure 4-3 Class Diagram of the full Spec Model of the FC and FcPort

The LayerProtocolParameterSpec provides a vehicle to convey layer protocol specific parameters to the FC/Link where there is no benefit in using the sophisticated "occurrence" approach.

The various photonic structures described in TR-512.A.4 provide examples where the enhancements described above will be applied.

4.1.4 Class added to deal with Photonic FCs

4.1.4.1 CascOccurrenceInFcSpec

Qualified Name: CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::OccurrencesInFcSpec::CascOccurrenceInFcSpec

A CASC component that is part of a system of components that represents the behavior of an FC/ForwardingOccurrence.
This class is Experimental.

 © 2018 Open Networking Foundation

4.1.4.2 CascPortOccurrenceInFcSpec

Qualified Name:
CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::OccurrencesInFcSpec::CascPortOccurrenceInFcSpec

Port of a CascOccurrence where the port may be bound to a port of an LpOccurrence.
This port may NOT be exposed as a port on the containing FC/ForwardingOccurrence.
This class is Experimental.

4.1.4.3 ForwardingOccurrence

Qualified Name:
CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::OccurrencesInFcSpec::ForwardingOccurrence

A forwarding component that is part of a system of components that represents the behavior of an FC or a ForwardingOccurrence at a higher abstraction (leading to an FC).
This class is Experimental.

4.1.4.4 ForwardingPortOccurrence

Qualified Name:
CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::OccurrencesInFcSpec::ForwardingPortOccurrence

Port of a ForwardingOccurrence where the port may be bound to another port of another ForwardingOccurrence or to another component or may be exposed as a port on the containing FC/ForwardingOccurrence.
This class is Experimental.

4.1.4.5 LpOccurrenceInFcSpec

Qualified Name:
CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::OccurrencesInFcSpec::LpOccurrenceInFcSpec

A termination component that is part of a system of components that represents the behavior of an FC/ForwardingOccurrence.
This class is Experimental.

4.1.4.6 LpPortOccurrenceInFcSpec

Qualified Name:
CoreModel::CoreSpecificationModel::FcCapability::ObjectClasses::OccurrencesInFcSpec::LpPortOccurrenceInFcSpec

Port of a LpOccurrence where the port may be bound to another port of another LpOccurrence, a port of a ForwardingOccurrence or to another component.
This port may NOT be exposed as a port on the containing FC/ForwardingOccurrence.
This class is Experimental.

 © 2018 Open Networking Foundation

4.1.5 Pictorial representation of the Forwarding Spec

The diagrams below[29] show a pictorial view of some of the classes above (the colors used in the figure are consistent with those used in the model above).

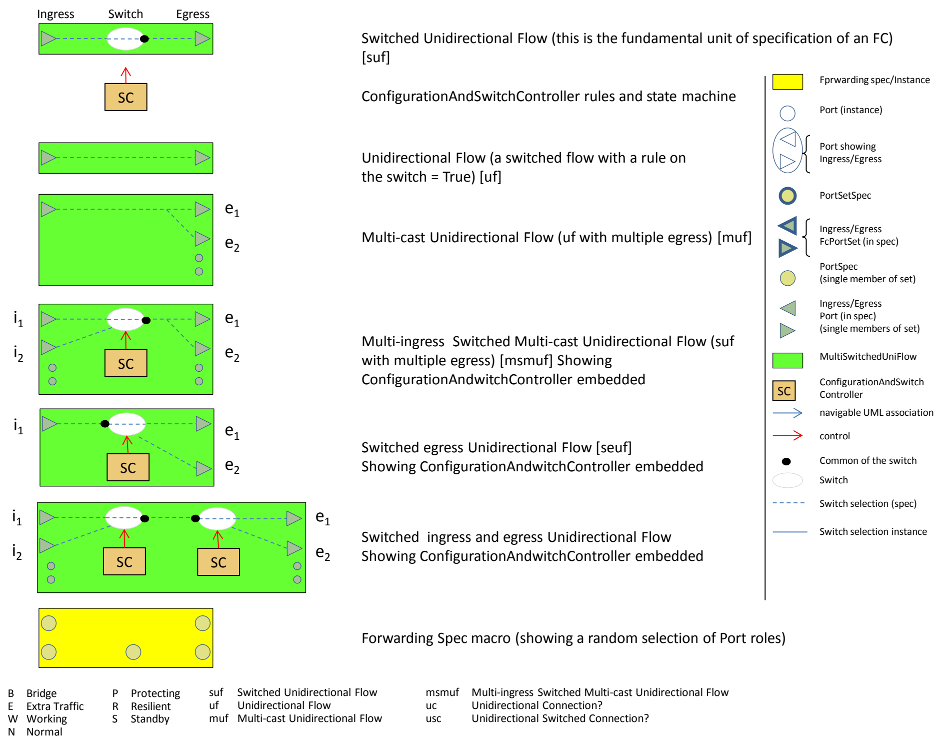


Figure 4-4 Pictorial view of the Spec Model of Configuration Control

4.1.6 Relating ForwardingSpec to FC and Link

The following figure shows the relationship between the ForwardingSpec and the Link and FC. The model form follows the general pattern for association between the Spec and the classes that are specified.

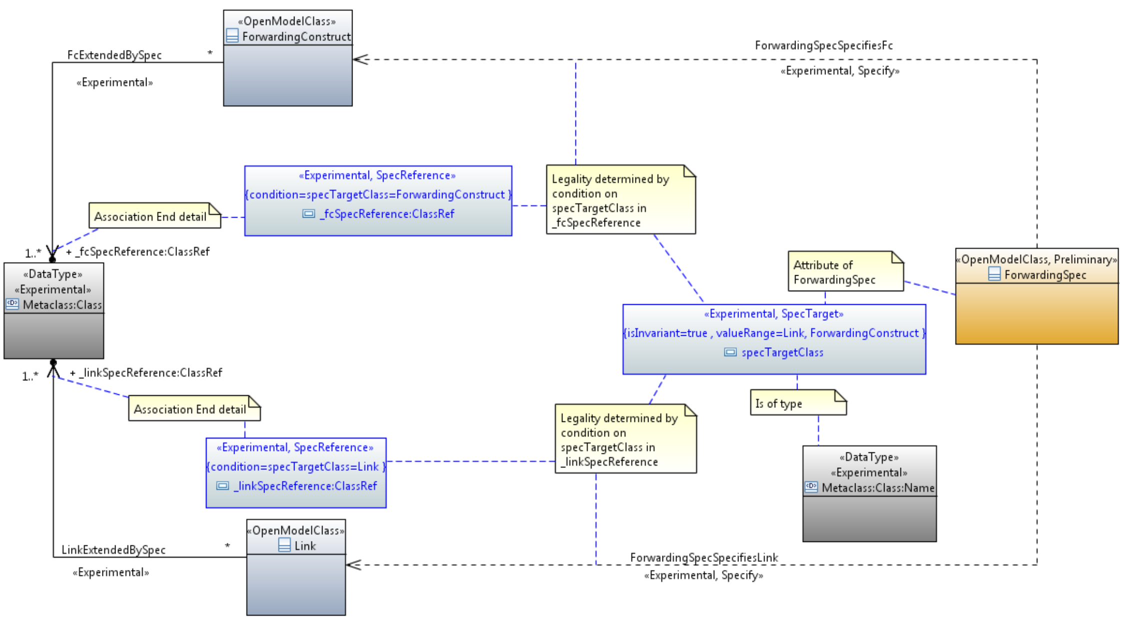
In general, a case of specification is used to refine an instance of a class. In the instance that has been refined it references the specification. The specification is a class model and hence the instance that has been refined points at a class. As a consequence, the class that the instance is defined by needs to reference the UML Metaclass "Class". As this is not formally supported by the tooling, a Data Type has been used as a proxy for the Metaclass. The two cases shown in the figure are _fcSpecReference and _linkSpecReference.

[29] Note that the Switched Unidirectional Flow is essentially a degenerate FC. Hence there is an opportunity to converge the FC and ForwardingSpec models. Also note that the point of transition from detailed FC layout to ForwardingSpec structure depends upon the case. The model of FC assembly and recursion can be used to the most detailed level of unidirectional point to point, but in cases where this FC detail is not desired, the spec can then be used to model the effect of the detail.

 © 2018 Open Networking Foundation

Essentially any class can be referenced. In practice, only appropriate spec classes will be applicable. The referenced class is valid if it defines an attribute "specTargetClass" that has a "valueRange" that includes the Name of the Class (in this case "ForwardingConstruct" or "Link"). The Name is the text name defined in the Metaclass "Class". The attribute "specTargetClass" is the representation of the Specify association shown in the figure. A specific case of a spec is likely to only apply to one class but the pattern, as is the case here, may apply to several classes (Link and FC here).

The spec reference in the instance that is being extended has a condition of "specTargetClass=" that is set to the name of the class that is being extended. It is this condition value that is used in the validation. The value of "specTargetClass" in the stereotype and in the attribute of the referenced class must match for the specification to be applied.



CoreModel diagram: Spec-RelatingToForwardingCapabilitySpec

Figure 4-5 Forwarding Spec relationship to FC and Link

4.1.7 Metaclasses

4.1.7.1 Metaclass:Class

Qualified Name: CoreModel::CoreSpecificationModel::TypeDefinitions::Metaclass:Class

This datatype represents the "<<Metaclass>> Class" from the UML metamodel.
An instance of the referencing Class (e.g. LTP) will reference a Class (not an instance).
This referenced Class will provide definition to extend the referencing instance.
So, for example, an LTP instance will have the attributes defined in the LTP class and also the attributes defined in the referenced Class (an LtpSpec).

© 2018 Open Networking Foundation

The referenced Class may:
(1) provide invariant properties (that are the same for many instances) that then are not conveyed with the referencing instance.
(2) provide definitions for attributes that are present in the instance that are not defined in the Class of the instance (these attribute may have been pruned and refactored from one or more external definition sources).
(3) apply constraints to attributes in the instance that were defined in the class of the referencing instance.
(4) replace attributes that were present in the class of the referencing instance by a new definition (same name).;

4.1.7.2 Metaclass:Class:Name

Qualified Name: CoreModel::CoreSpecificationModel::TypeDefinitions::Metaclass:Class:Name

4.1.8 ForwardingSpec in V1.3.1

As a result of the enhancements in V1.4 to handle photonics, some rudimentary modeling supported in V1.3.1 has been removed. The figure below, extracted from V1.3.1 highlights the model changes.

The elements marked with red crosses have been removed. The elements marked in blue are not relevant to show in the figures as there are not related directly to the Forwarding Spec.

 © 2018 Open Networking Foundation

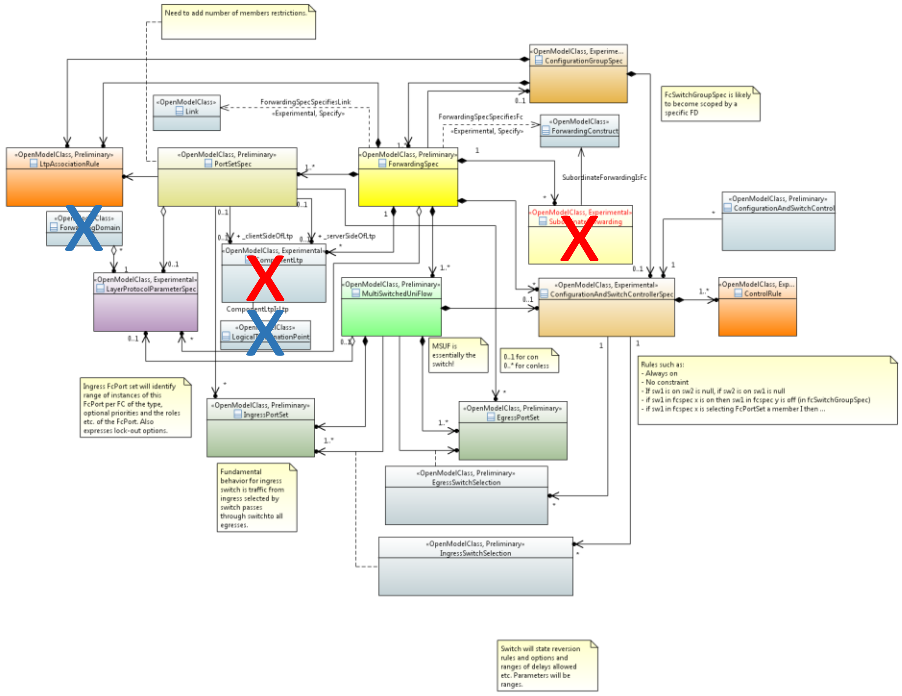


Figure 4-6 The Forwarding Spec from V1.3.1 highlighting changes

The ComponentLtp has been superseded by the LpOccurrence and the SubordinateForwarding has been superseded by the ForwardingOccurrence.

4.1.9 Use of the Forwarding Spec

The figure below shows a pictorial view of a case of ForwardingSpec. The lower element of the diagram shows specification class instances and the upper element shows an instance of FC abiding by the spec. As noted in the diagram key above, the blue arrows represent the UML associations direction and not the flow direction. The flow direction is conveyed by the orientation of the ingress/egress ports (trinagles) of the MultiSwitchedUniFlow (green) elements.

The MultiSwitchUniFlow elements represent the allowed flows across the FC and explain what each switch (in this case there is one switch only) related to the FC does. Omni-directional cases are covered by representation as unidirectional forwarding combinations. The spec explains where forwarding is intentionally possible. The spec also covers unintentional flow such as reflection characteristics of an omni-directional media.

 © 2018 Open Networking Foundation

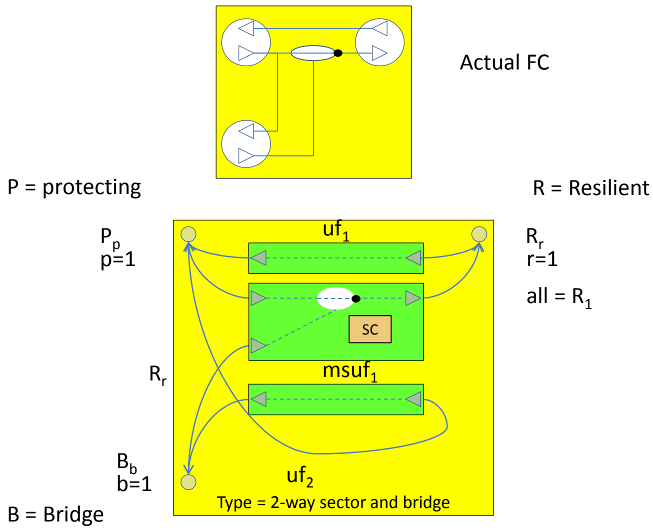


Figure 4-7 Pictorial view of spec model and resulting FC instance

The example above is a relatively complex switch (used in ladder protection schemes). A more straightforward case is shown below. It should be noted that the multiple ports on the left side of the actual FC are represented by one statement in the spec. The spec can provide a compact statement of the capability where there is a systematic structure.

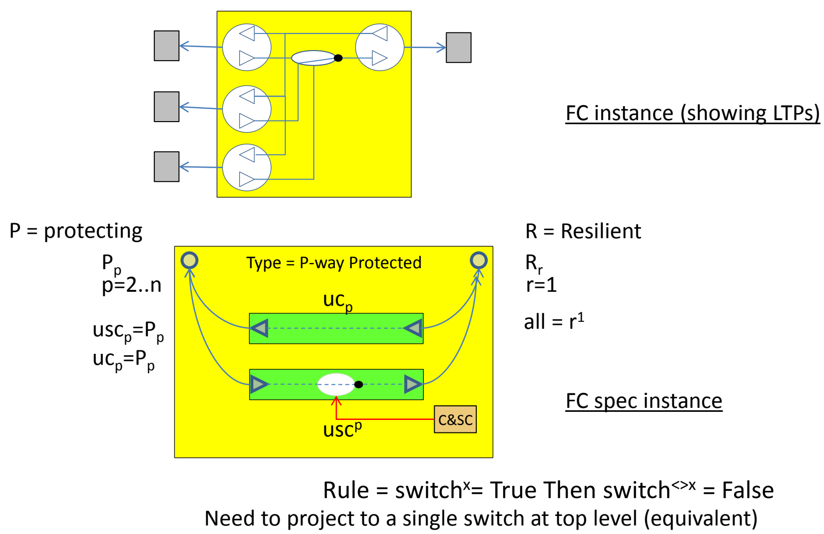


Figure 4-8 Compacting the Forwarding Spec rules

Where there is significant symmetry, a very compact form can be developed. In the figure below the FC and switch arrangement is highly symmetric. It is assumed that the switches do not offer any reversion (i.e. the switch will stay where it is until there is a failure on the signal it is receiving when it will then switch to the other port). Both sides of the FC behave in the same way.

 © 2018 Open Networking Foundation

The specification forms at the bottom of the figure all represent the FC at the top. The one on the left is very specific and verbose and the one on the right is most compact (and assumes that there is normal case of "no loopback allowed" which forces there to be at least two ports). Clearly there is a need to state the bounds on the number ports in the PortSet which in this case, assuming the FC figure is precisely what is supported, is precisely 4 ports in two groups of two.

Note that the very compact forms are not recommended other than for unprotected packet cases, as the method of constraining ports in the PortSetSpec has not been fully developed. Clearly it is important that a standard form of specification of constraints emerges so as to prevent unnecessary decoding complexity. Without a well-defined constraint, the compact spec on the left would allow all sorts of FCs to be created from a two port switched unidirectional FC upwards.

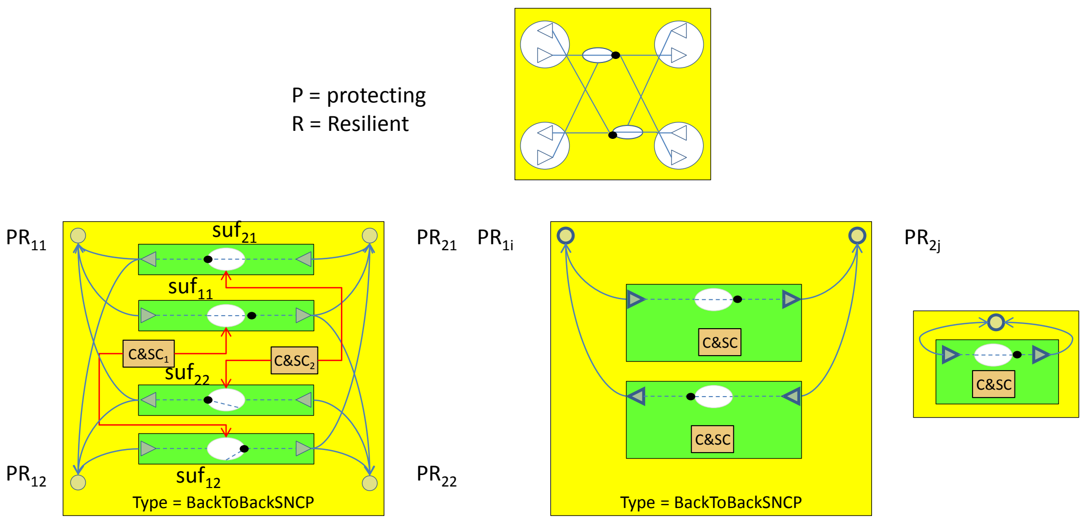


Figure 4-9 Highly compact form for symmetric FC

The figure below sets out some Forwarding Specs overlaid where the FCs corresponding to the specs would be in a 1:N protection scheme (see TR-512.5), where the spec layout represents the NE on the right in the figure in TR-512.5). The upper two FCs have the same spec. The lower FC has a different spec. The figure does not show the full arrangement of C&SCs (which would align with the protection scheme definition)[30].

[30] It has been recognized that there should be scheme specifications to deal with protection constraints on protection structures and that these should use the techniques set out in this document. This is for future development.

 © 2018 Open Networking Foundation

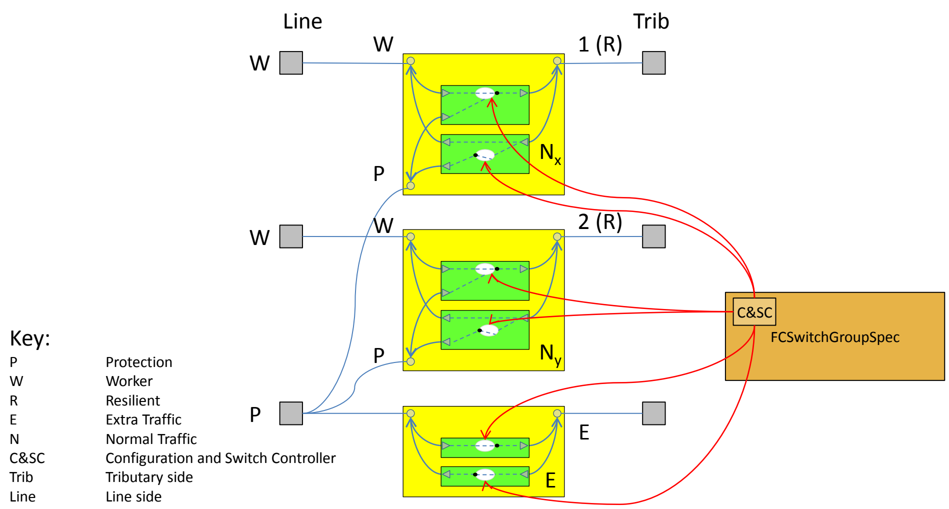


Figure 4-10 Forwarding Spec view for 1:N protection

4.1.10 Dealing with exceptional behavior on failure

The figure below shows the symmetric FC discussed earlier. Both the representation of the FC and the spec view on the left provide an abstraction that may be relevant to offer to a service oriented client. However the underlying implementation may allow a variety of undesired behaviors under various failure cases.

The spec view on the right (one direction shown only) provides a more accurate description of the opportunities for asymmetric flow but this may be considered too complex to present to the service oriented user.

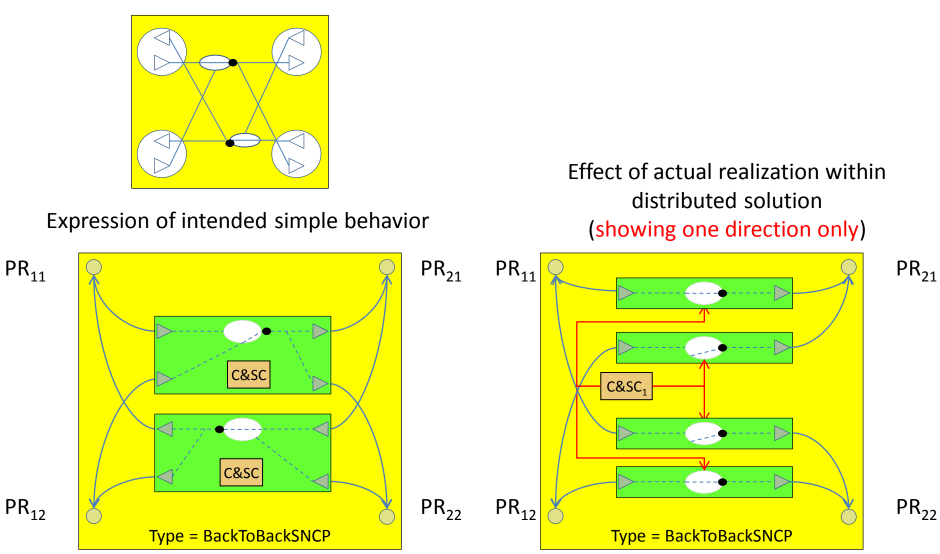


Figure 4-11 Desired abstraction and actual potential

 © 2018 Open Networking Foundation

But how then should the operator present obscure failure cases on the rare occasions that they do occur? Clearly the operator may want to temporarily express to the client what the actual current state of flow is. This will be especially relevant if the client wants to perform some engineering works on their local network that may involve disconnecting traffic from one or more of the ports.

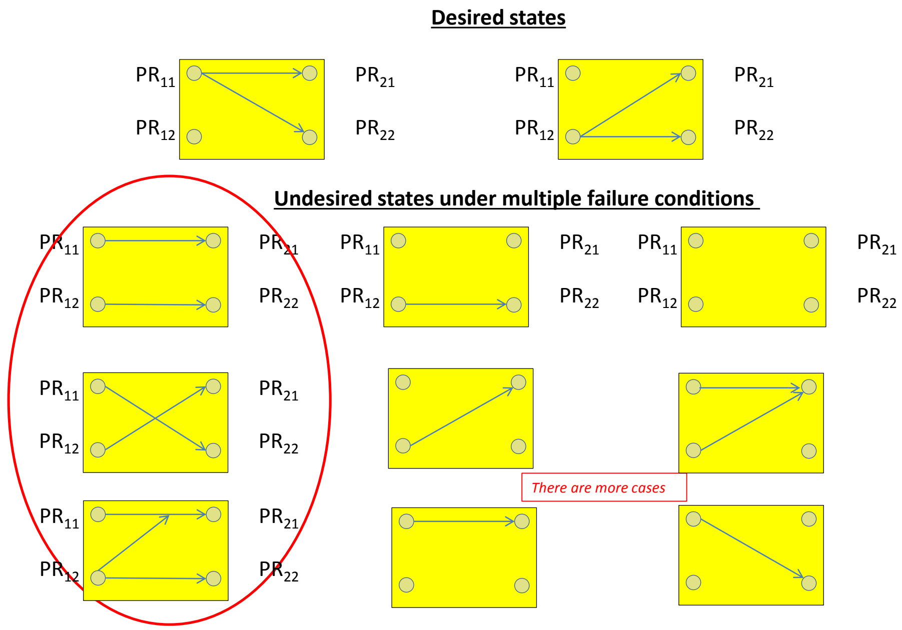


Figure 4-12 Various flow states under failure

To account for the issue highlighted in the figure above, an actual possible flow statement could be provided (along perhaps with an alert of non-normal internal flow state). The actual possible flow would be described in one or more specs that lay out the switched flows of the current snapshot of disjoint structures. This could use the normal spec structure.

The figure below shows a spec instance representing the undesired flow. Specs could be defined for each undesired flow pattern. It would be reasonable to define common cases of undesired flow where there is significant value in doing this but not more obscure cases. The more obscure cases could be represented by a single pattern indicating unknown flow.

 © 2018 Open Networking Foundation

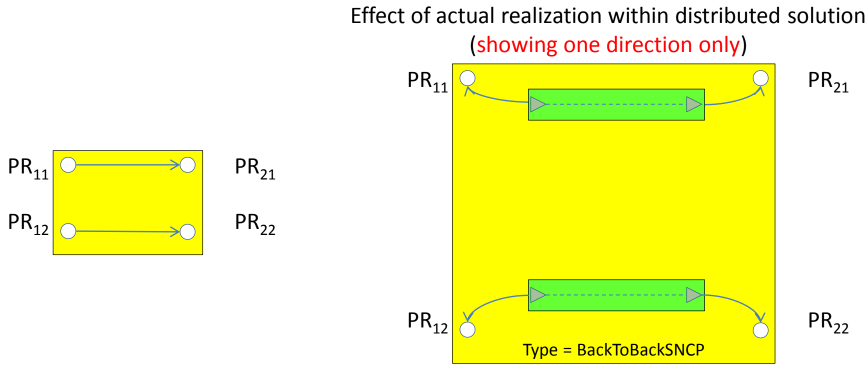


Figure 4-13 Using the Forwarding Spec model to show undesired flow

Further information is provided in TR-512.5.

4.1.11 Spec v Instance

The spec provides all invariant details, i.e., the type definition as well as the unchanging value (so that there is nothing to state in the instance), and identifies dynamic aspects, i.e., the type definition in terms of legal range, defaults etc. As the instance references the spec and the spec is available at runtime, the instance of the class need only identify the state of the dynamic aspects, i.e., the value of the dynamic attribute (all ranges, defaults etc. are conveyed in the spec)[31].

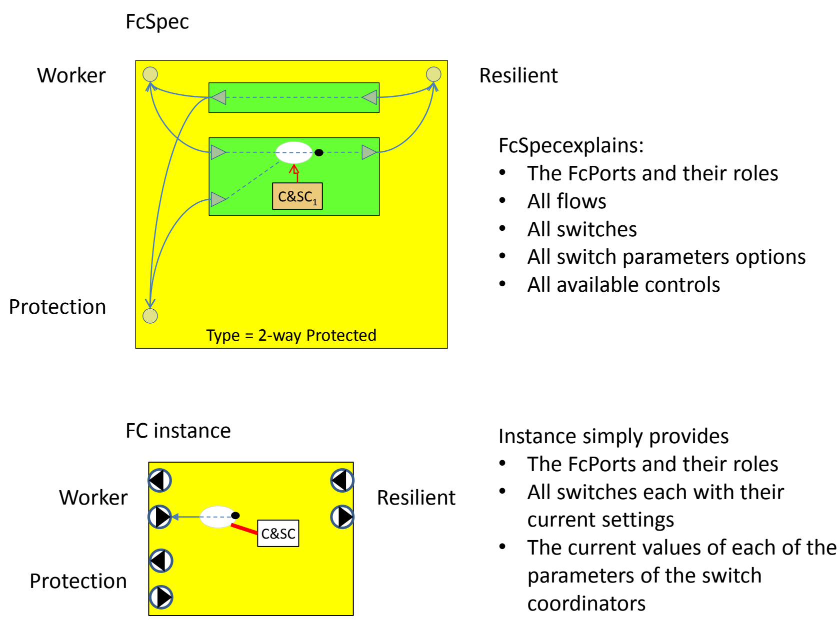


Figure 4-14 ForwardingSpec and FC instance details

[31] As the invariant aspects are carried by the spec, representation in each instance of the class would be unnecessary replication. This approach means that the information flowing over an interface run time can be kept to a minimum.

 © 2018 Open Networking Foundation

The figure below shows, at the top, an example of a 1+1 protection network layout and highlights the nodal FC (in the red dashed circle) for the right most NE (these diagrams were derived from TR-512.5). The figure also shows a view of the FC model and a view of ForwardingSpec model. These two models are related to diagram of the instance of the FC model representing the FC in the red dashed circle and the spec that would describe the FC highlighted in the red dashed circle.

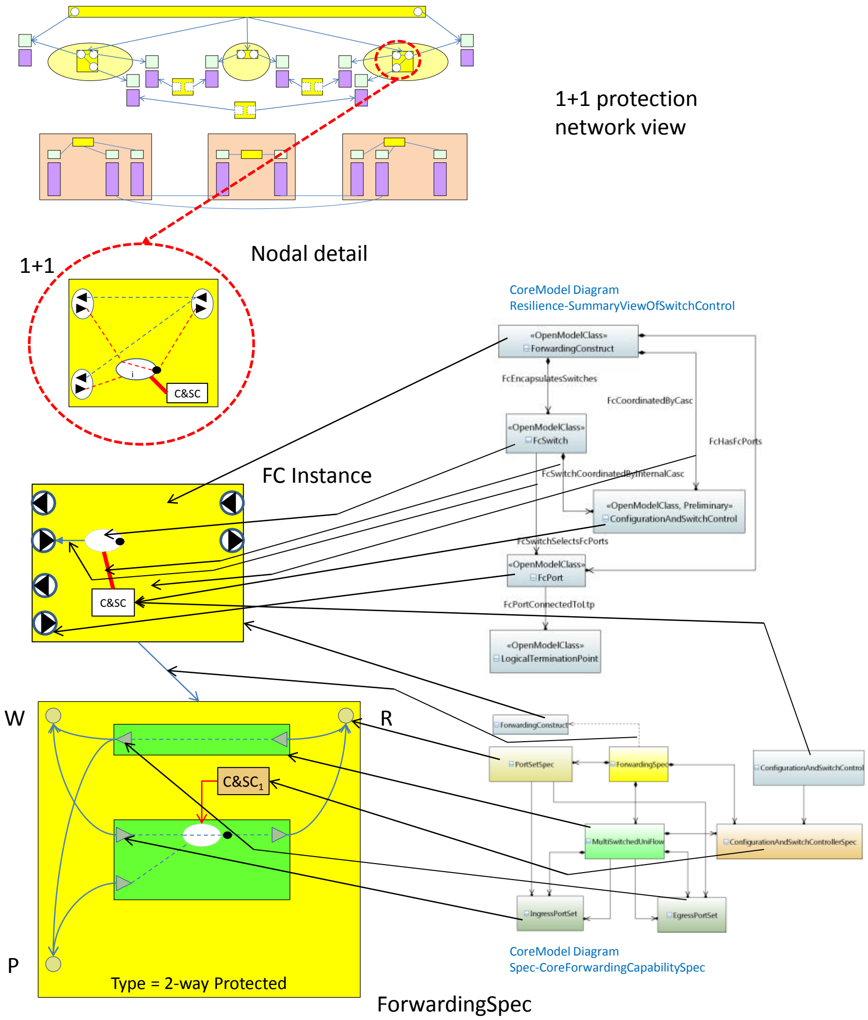


Figure 4-15 View of ForwardingSpec and FC model in the context of a 1+1 protection case

4.1.12 Role names and Port numbers

The spec also provides role Port role names and Port numbering rules.

- Where there are compact symmetric definitions the spec should provide naming rules for each dimension of unfolding of the spec (e.g. PR_{11}, PR_{21}, PR_{22} and PR_{12}).
- Role names relate to the flow and could be such Root/Leaf, Active/Standby, Worker/Protection, Balanced, Symmetric etc.

 © 2018 Open Networking Foundation

4.1.13 Mapping to Open Flow

As the Forwarding Spec model is designed to break the FC down into component flows, the Forwarding Spec provides a bridge to the flow aspect of the OpenFlow[32]. Some work was carried out as a proof of concept. This work has not been extended recently and is now somewhat dated in detail, but still provides a view of an appropriate route for mapping.

The figure below provides a sketch of the mapping between a pictorial view of ForwardingSpec statements and an older form of OpenFlow Table Type Patterns for a Root and Leaf form.

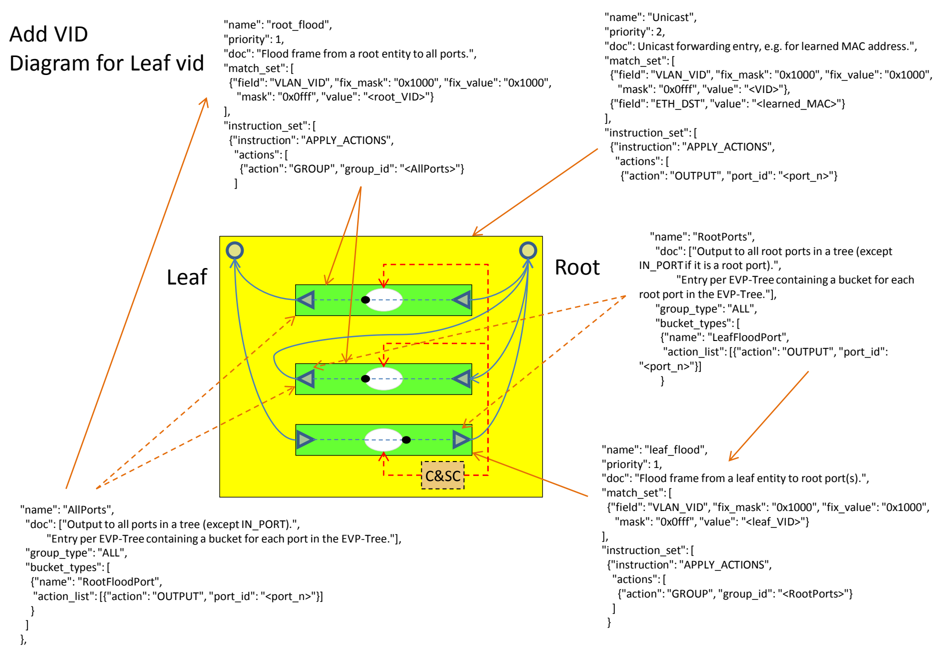


Figure 4-16 Sketch of mapping to OpenFlow using ForwardingSpec constructs

4.2 LogicalTerminationPoint and LayerProtocol specification

4.2.1 Rationale and requirements

The LTP/LP spec structure was developed to primarily support from a management-control perspective:

- The controlled introduction of layer protocol (network technology) specific attributes from sources external to ONF with minimum burden to ONF
    - ONF do not have a mandate for network technology (layer protocol) definition

[32] Whilst OpenFlow can provide flow only rules, it tends to be used in what is a hybrid mode from the IM perspective where definition of flow and termination is provided in single rule statements. The FC spec model only provides part of the mapping and the LTP spec model needs to be used in conjunction with the FC spec model to provide a full definition.

 © 2018 Open Networking Foundation

        - Lesson: Work in other standardization activities has suffered from the burden of maintaining alignment between their redefined technology properties and the properties defined by the technology specification body
    - A method that provides ready access to the appropriate work of the other bodies whilst decoupling that from the ONF modelling work was considered vital
        - This eliminates the possibility of use of simple conditional composition
- The reducing of the definition of a network technology to match a specific realization
    - Often a specific realization of a network technology will not support the full technology definition
        - In many cases this reduce support will manifest via attributes that have reduced ranges etc. compared to the standard
    - The method must support the narrowing of the definition of attributes within their original semantics[33]
        - This eliminates the possibility of use of simple inheritance
        - The Pruning and Refactoring approach developed to enable generation of an implementation specific view of the Core Model has been used
- Specific rules that define the layer protocol mapping opportunities and also set out the interactions between layer protocols
    - This will allow the definition of the port layer stacks and link capacity etc.
        - Layering is assumed and coded but to best enable innovative deployments layering should be explicitly stated
    - The method must provide a rich enough rule structure to allow definition of all currently understood layer protocol interactions
- Proprietary extensions to a network technology where the vendor has introduced a capability within that network technology
    - This maybe prior to standardization or for a niche application etc.
        - This may involve extending attribute definitions within their original semantics
    - The method must have no barriers to such innovation whilst preventing unnecessary variety and ensuring appropriate decoupling
        - Attribute extension must be done in the context of the network technology outside ONF such that the bringing into the ONF context is always carried out by pruning
- Definition of a completely new network technology (that maybe proprietary)
    - This maybe a new or emerging standard
    - The method must guide appropriately the positioning of information related to the new technology to maximize consistency with other network technologies
- Migration and upgrade of definition of the technologies
    - The method should allow for on-the-fly redefinition
- Definition of constrained usage of a specific type of LTP
    - Where a capability already express by a specification is to be reduced for a particular application so that that reduced capability can be expressed by another specification provided through the view exposed to the user of that particular application

[33] As defined by the specification authority.

 © 2018 Open Networking Foundation

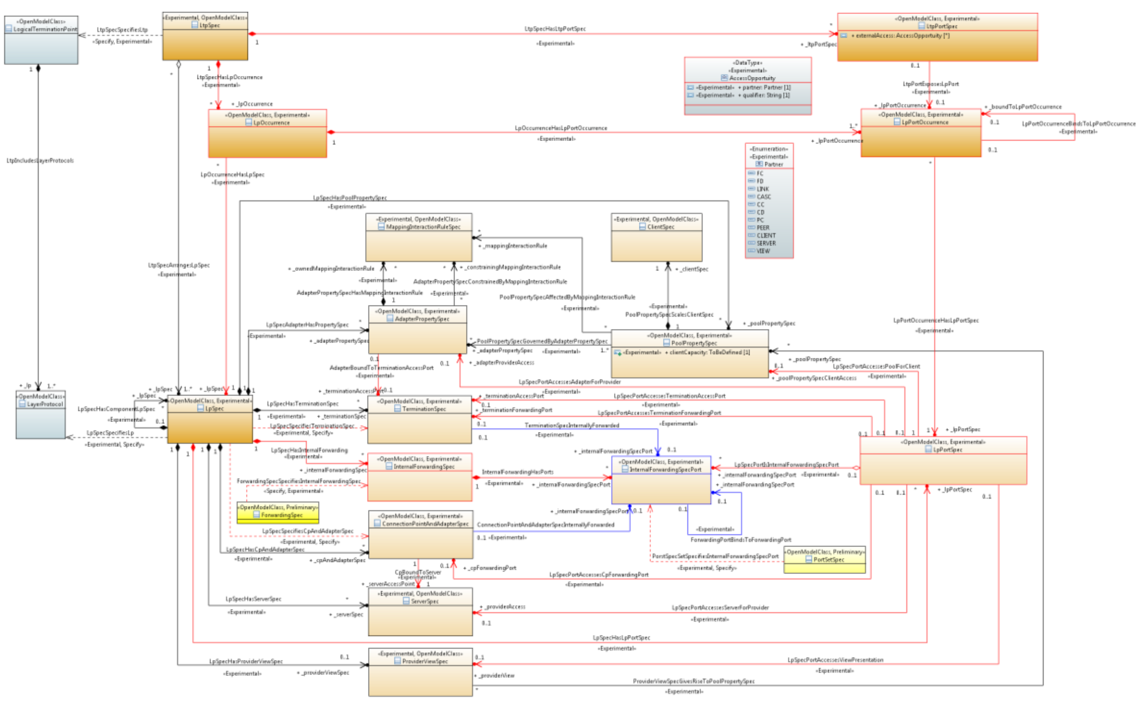
- The specific use of the LTP/LP in the PHOTONIC_MEDIA layer leads to the need to deal with exposure of additional complexity within the LTP/LP
    - This degree of complexity is also present for other layerProtocols but can normally be hidden
        - For a fully interpretable LTP/LP definition the complexity would need to be exposed explicitly, because in this case it is relevant for management purposes

In TR-512 V1.4 the LTP/LP spec has been enhanced to enable more complex structural statements to be made.

4.2.2 Model skeleton

The figure below provides a view of the structure of the LTP/LP spec model.

Once again, by modelling the internals of a LP, we can represent a 'prototypical instance' that can be used by LP and LTP instances.



CoreModel diagram: Spec-LtpCapabilitySpecWithLtp

Figure 4-17 Class Diagram of the Spec Model of LTP and LP

Enhancements have been made in V1.4 mainly to deal with the photonic media constructions described in TR-512.A.4 are discussed below:

- Major enhancements highlighted with red lines in the figure above (working from the top down):
    - Opportunity to model LtpPortSpec. In V1.4 the LTP does not have formal ports.
        - The LTP ports are essentially modelled via several explicit associations.

 © 2018 Open Networking Foundation

        - In a later release it is intended to add the LtpPort
        - The LtpPortSpec can be used to explain the behaviour visible through each explicit association
      - This enhancement along with the LpPortSpec will make it possible to encode the complex LTP/LP arrangements highlighted in TR-512.A.4
  - Opportunity to explicitly model occurrences of LP in the LpSpec
      - This allows for the repeated use of occurrences of the same structure in a spec.
      - The LpOccurence and its corresponding LpPortOccurences are essentially the Component and Port in the Component-System pattern and the LpSpec and its detail is the System of Components (see TR-512.A.2)
        - The LpOccurrences can be arranged by binding their ports using LpPortOccurrenceBindsToLpPortOccurence
  - Explicit structuring of the Spec elements[34]:
      - Via
        - AdapterBoundToTerminationAccessPort
        - CpBoundToServer
      - Which add to the existing ProviderViewSpecGivesRiseToPoolPropertySpec
      - Ensure the spec is directly interpretable.
  - Addition of LpPortSpec which allows explicit statement of the interconnection of LPs within an LTP and also which are exposed via LtpPortExposesLpPort
      - LpPortSpec expresses which part of the LP system it allows access to via the LpSpecPortAccessesXxx associations
  - Enrichment of the forwarding statements by addition of InternalForwardingSpec
      - Allows for expression the rich complex forwarding arrangements within an LP in a photonic media case
      - The InternalForwardingSpec has a port that can be used to interconnect
        - Spec parts in the LpSpec:
          - Terminations
          - ConnectionPointsAndAdapters
          - InternalForwarding
        - To ports of the LpSpec
          - Which then allow LPs within the LTP to be connected using LpPortOccurrenceBindsToLpPortOccurence
  - Application of the ForwardingSpec to the InternalForwardingSpec so that a full forwarding definition can be applied as appropriate
      - The PortSetSpec of the ForwardingSpec is added to the InternalForwardingSpecPort
- More subtle enhancements highlighted with blue lines:
  - The ConnectionSpec from the earlier versions has been renamed (and repurposed) as InternalForwardingSpecPort

[34] The spec model in V1.3 assumed an order and did not provide navigable associations. The structure is now set out explicitly via associations (and is hence adjustable).

 © 2018 Open Networking Foundation

- This actually allows backward compatibility with previous versions where the Port can be used essentially stand-alone (although this is not recommended)
    - The enhancement also fixes the composition issue with the previous version

The distinction between the model in this version and the model in V1.3 can be seen by comparing the following two figures.

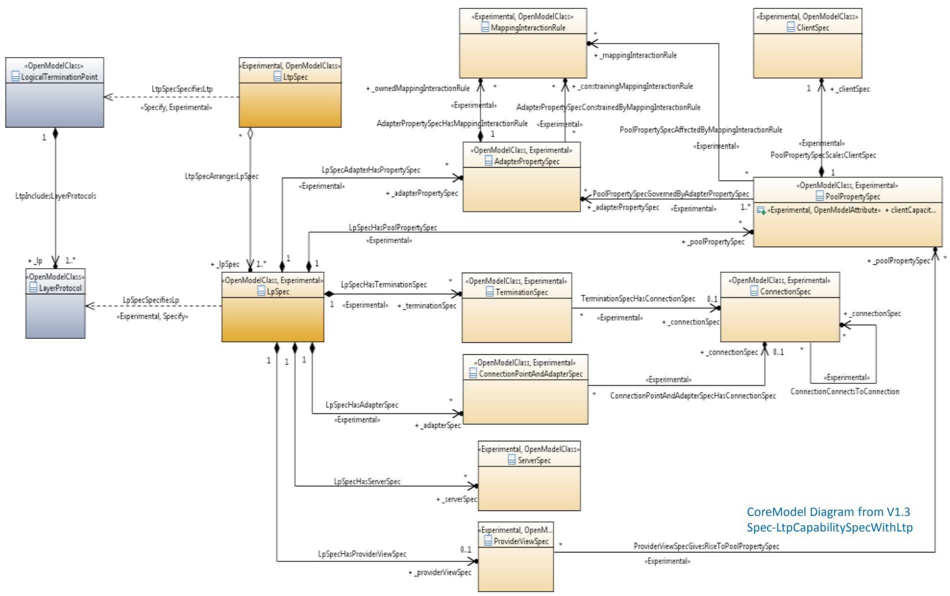
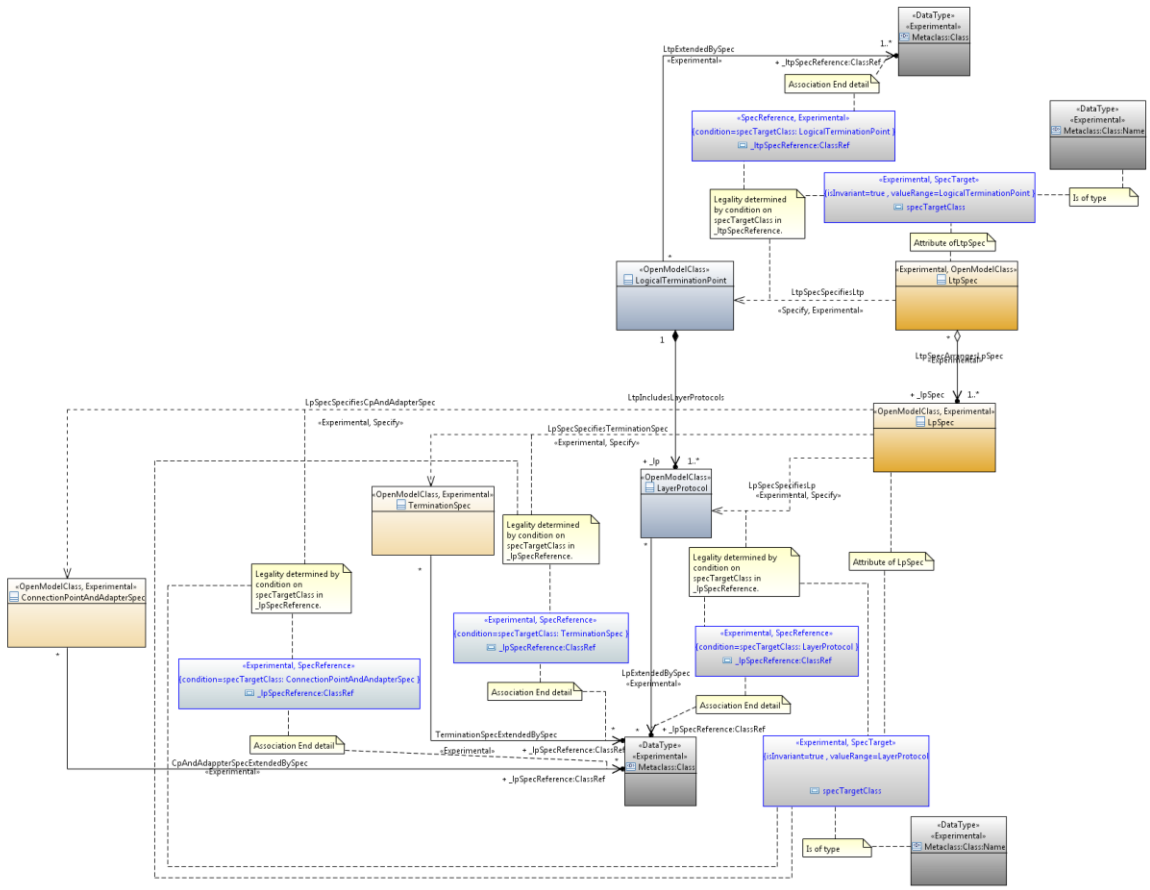


Figure 4-18 LtpSpec showing V1.3 model

As for the ForwardingSpec discussed in 4.1 Forwarding Specification on page 20, the general spec pattern is used.

 © 2018 Open Networking Foundation



CoreModel diagram: Spec-RelatingToLtpCapabilitySpec

Figure 4-19 LtpSpec/LpSpec relationship to LTP/LP

The elements of the spec model are related to the pictorial symbols and hence functional blocks in the essential layer protocol model. The areas of the specification that support each aspect of definition highlighted in section 4.2.1 Rationale and requirements on page 38 should be apparent.

 © 2018 Open Networking Foundation

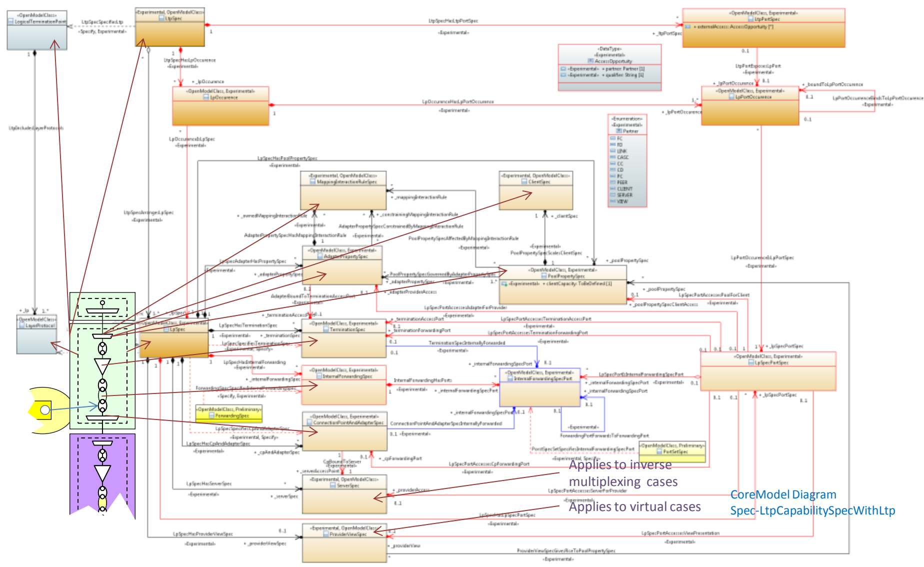


Figure 4-20 Relating LTP/LP spec elements to the pictorial symbols

The intention is that the LTP/LP spec model be capable of describing all cases of LTP/LP. Many cases are set out in TR-512.2 (two figures, "LP Cases" and "LTP Cases", show the cases that the LP and LTP spec needs to support).

4.2.3 Class details

4.2.3.1 AdapterPropertySpec

Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::AdapterPropertySpec

The specification of the properties of the client side adapter of an LP.
This class is Experimental.

4.2.3.2 ClientSpec

Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::ClientSpec

The specification of a client layer protocol supported by the adapter of an LP.
This class is Experimental.

4.2.3.3 ConnectionPointAndAdapterSpec

Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::ConnectionPointAndAdapterSpec

The specification of the server facing connection point and the adapter that deals with the transformation of a single signal of the layer protocol to/from the server.
Equivalent to an ITU-T CTP [ITU-T G.8052][G.874.1].
This class is Experimental.

 © 2018 Open Networking Foundation

4.2.3.4 InternalForwardingSpec
Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::InternalForwardingSpec

InternalForwardingSpec defines the encapsulated forwarding in the LP.
The InternalForwarding is essentially a ForwardingConstruct.
This class is Experimental.

4.2.3.5 InternalForwardingSpecPort
Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::InternalForwardingSpecPort

The specification of the flexibility of the association between the ConnectionPoint and the Termination of the LP.
This is the port of the Internal Forwarding and is the equivalent to the FcPort.
This class is Experimental.

4.2.3.6 LayerProtocolParameterSpec
Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::LayerProtocolParameterSpec

Offers the opportunity to define a list of layer-protocol related parameters.
Used to specify the extension a class.
This class is Experimental.

4.2.3.7 LpOccurrence
Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::LpOccurrence

An occurrence of a LP in the specified LTP where the LP occurrence will have an identifier.
The LP occurrence will have a spec.
This class is Experimental.

4.2.3.8 LpPortOccurrence
Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::LpPortOccurrence

The occurrence of a port on the LP occurrence.
The port occurrence will correspond to a port in the spec of the corresponding LpOccurrence.
This class is Experimental.

4.2.3.9 LpPortSpec
Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::LpPortSpec

The spec for the ports of the LP.
This class is Experimental.

4.2.3.10 LpSpec
Qualified Name: CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::LpSpec

 © 2018 Open Networking Foundation

The specification of the capabilities of a specific type of LP.
This class is Experimental.

4.2.3.11 LtpPortSpec

Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::LtpPortSpec

Spec for the LTP Port. Each LTP Port relates to an association end related to the LTP class.
This class is Experimental.

4.2.3.12 LtpSpec

Qualified Name: CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::LtpSpec

The specification of a specific type of LTP.
This class is Experimental.

4.2.3.13 MappingInteractionRuleSpec

Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::MappingInteractionRuleSpec

The specification of the interaction between the support for different client layer protocol signals.
For example an LP that supports 20 layer protocol X signals and 5 layer protocol Y signals may be such that a particular layer protocol X instance being used eliminates the possibility of using a particular layer protocol Y instance being used.
This class is Experimental.

4.2.3.14 PoolPropertySpec

Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::PoolPropertySpec

The specification for the properties of the pool of available instances of a particular client layer protocol.
This may cover numbering range, capacity, number of instances etc.
This class is Experimental.

4.2.3.15 ProviderViewSpec

Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::ProviderViewSpec

The specification of the properties of an LP at the base of a virtual/floating LTP that relate to the provider of capacity/capability for that floating LTP.
This class is Experimental.

4.2.3.16 ServerSpec

Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::ServerSpec

The specification of the server side of an LP at the base of an LTP that supports the creation of server LTPs for use in an inverse multiplexing scheme.
This class is Experimental.

 © 2018 Open Networking Foundation

4.2.3.17 TerminationSpec

Qualified Name:
CoreModel::CoreSpecificationModel::LtpCapability::ObjectClasses::TerminationSpec

The specification of the layer protocol termination (including framing, modulation etc.).
For example, the specification of the function that takes a MAC frame and extracts the content (removing the MAC address in the process).
This class is Experimental.

4.2.4 Assumptions

The specification approach assumes that:

- For any particular layer-protocol there is a standard definition of a set of capabilities and that definition provides details of entities, attributes/properties and their values
  - The standard definition either is in Papyrus UML or can be transformed into Papyrus UML
  - The standard definition is properly layered and provides suitable guidance on whether a property relates to termination, adaptation, connectivity etc. and whether it is a control property etc.
- For any well-formed definition:
  - The classes and attributes/properties can be associated with the LP spec
  - The attributes of a layer-protocol each have unique names within the scope of that layer-protocol
  - Where there are multiple instances of a particular property, then this property is contained in a class that has an association with an appropriate multiplicity to a class directly related to one or more parts of the specification

The above assumptions ease the work of ONF. Currently ITU-T provide definitions in Papyrus UML, other bodies do not at the time of publication of this document. Significant work to untangle some definitions will be required. It is highly likely that ONF will need to provide a service of migration of some layer protocol definitions, from other specification bodies, into well-formed UML.

It is assumed that text documents (such as this document), that set out in loose form the functional dependency graph and related flow semantics, will be required for a significant period of time to augment the specifications to enable development of code with significant behavior. The work here is for interface definition and driving of "simple" systematic behavior.

4.2.5 Broad usage

Just as the LTP/LP do, the LTP/LP spec model supports terminations that are:

- Ports bound to physical
- Floating in an NE
- Fully virtual in the network

The intention is that any view of any termination with any degree of abstraction is supported (see TR-512.2 and TR-512.4).

The figure below depicts the relationship between the various models and aspects.

 © 2018 Open Networking Foundation

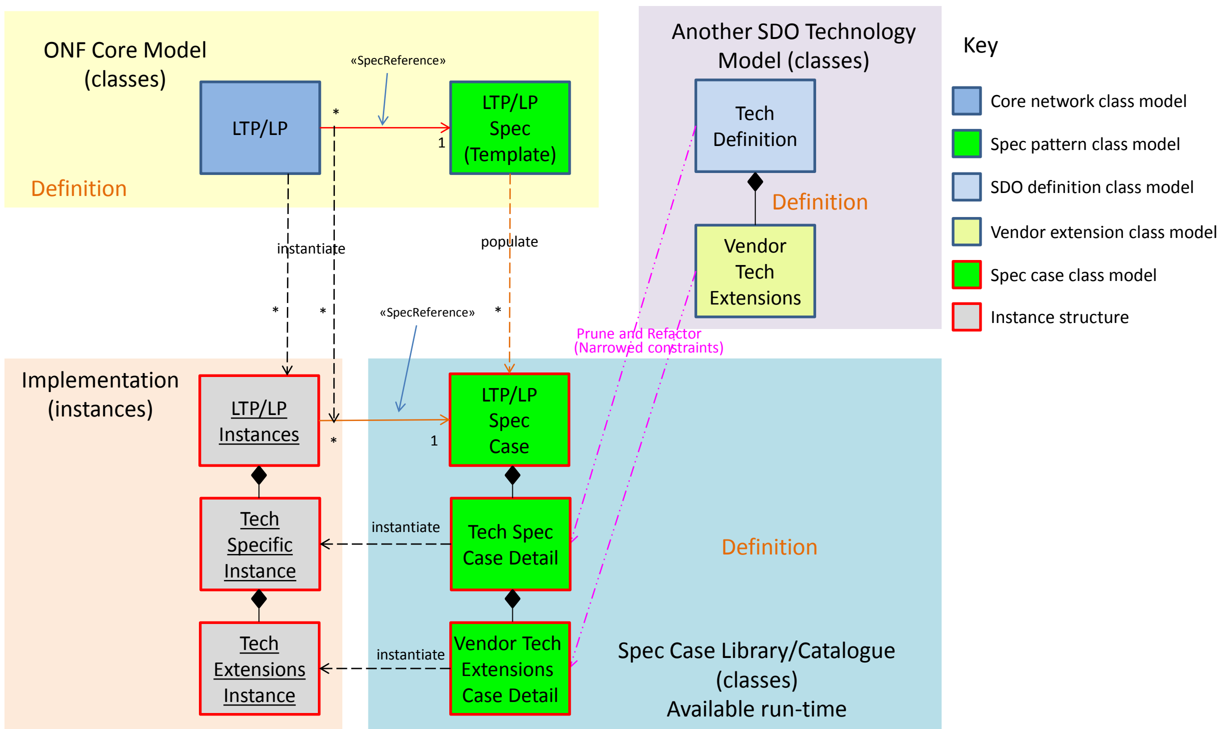


Figure 4-21 Relating LTP/LP spec with the class and instance models

In the figure above:

- The upper yellow area depicts the ONF core model.
  - The LTP and LP reference the spec model via an association with a stereotype to indicate that the association when instantiated will not point at instances but instead at a class model (schema)
- The upper purple area depicts a network technology model (e.g. as defined in [ITU-T G.8052]).
  - This model can be extended by vendors as necessary via a composition mechanism (perhaps via reverse navigation composition or technique favoured by the technology definition authority).
- The lower blue area depicts the integration of the specification model pattern and the appropriately pruned and refactored technology definition
  - The integration is achieved by using the spec model as a template
  - The spec model is cloned to provide a framework to fill in
    - The clone should be given a UUID to allow unique identification in the library
  - The appropriately extended technology model is duplicated (as a file copy to preserve the Papyrus IDs) and PruneAndRefactor associations are created

 © 2018 Open Networking Foundation

between both the original model and the copy for all classes, attributes, associations etc. in a mapping model[35]
  - The attributes from the model copy collapsed into classes that reflect the structure of the spec
    - Highlight the essential association between elements of the spec and the elements of the model considering the potential challenges
      - Sub-layering approach
      - Models from bodies that do **not** split termination aspects (TTP/CTP for example)
    - Preferable that the model offers extension opportunities but this can be done by reverse association
    - Inheritance is expanded and containment collapsed based on the [1] and [0..1] rules such that a set of classes that reflect the basic pattern of the specification model leaving the [n..*] classes intact and any residual cross class associations intact
    - Attach the collapsed classes to the corresponding spec model classes via a [1] compositions
      - Note that the [1] compositions could be collapsed but the approach above maintains the mapping relationships in the most recoverable form
    - Attributes can be removed or pruned at this point
      - Each has ONLY aspects relevant for the case with ranges relevant to the case defined as modified data types and as necessary with OCL
        - Dependencies between items should be modelled explicitly including between attribute values etc. – this should also be in the technology model and extensions
  - The prune and refactor mapping associations should be updated at this point[36]
  - The above process results in an LTP/LP spec case with appropriate UUIDs.
  - For the runtime environment the spec case will be encoded in an appropriate run time schema language (e.g. Yang or JSON schema)
  - The spec case is hence essentially a "class model" identifying all layering and attributes for a particular case of port etc.
- The lower brown area depicts the instance model (obviously instantiation from some encoded schema form of the class model e.g. in Yang or JSON schema)
  - The structure and key attributes are derived from the class model in the upper yellow area (e.g. globalId for the LTP and localId for the LP etc.)
  - Some LTP/LP instances reference the LTP/LP spec case constructed above using the UUID
    - Assuming that the LTP/LP corresponds to the specific case discussed above

[35] Ideally the Prune & Refactor tooling would be used for this. The tooling is in development.
[36] The Prune & Refactor tooling will provide some updates but at the point of delivery of this document the tooling is relatively immature.

 © 2018 Open Networking Foundation

- The LTP/LP have attributes present that defined in the LTP/LP spec case (both pruned tech spec attributes and tech extension attributes) in addition to those attributes defined in the class
- The controller can get the necessary specs from the library

The description above provides a general walk through of the operation of the LTP/LP spec model. The description was purposely simplified in some areas.

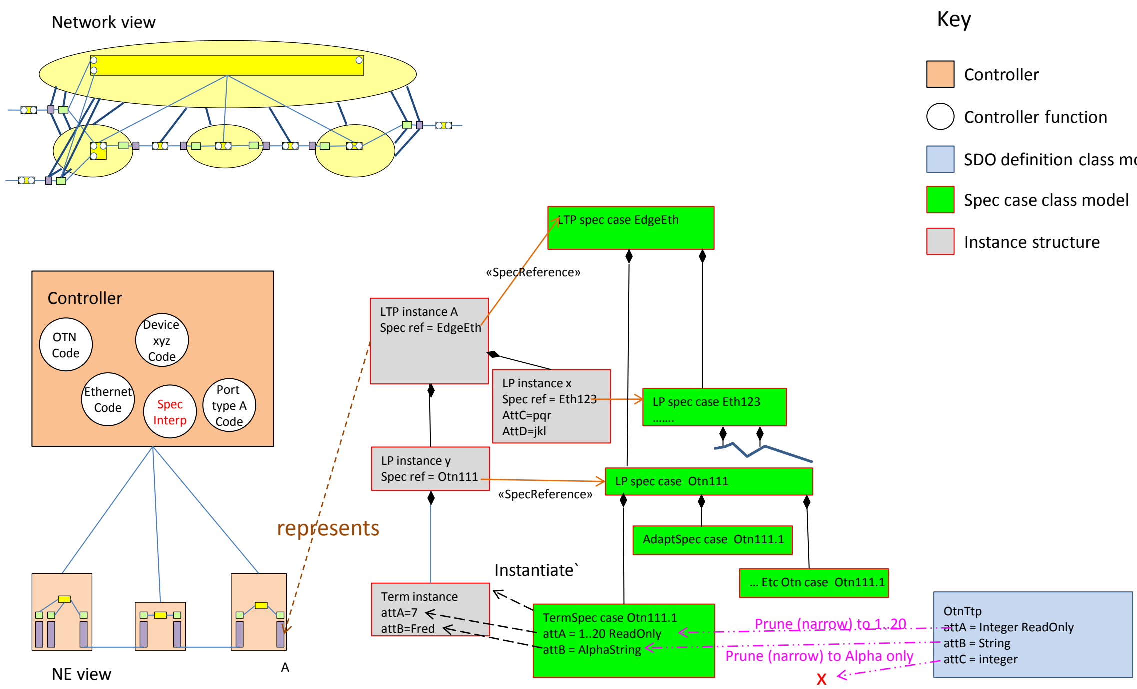


Figure 4-22 Sketch of use of LTP spec case

The figure above shows a sketch of the use of the LTP/LP spec (the color coding is as for the previous figure). To the left of the figure:

- At the bottom is a small network
- In the middle is a controller with various functions
- At the top is the controller view of the network

To the bottom right of the figure is a class of a technology spec with attributes attA, attB and attC. This technology class is pruned to form the TermSpec case Otn111.1 where attribute attA and attB are both narrowed in definition and attC is removed. The TermSPec case Otn111.1 is part of the LP spec case Otn111 which is itself part of the LTP spec case EdgeEth. The EdgeEth spec and its parts are available in the library.

Port A of the NE on the right is represented by the LTP instance A shown in grey and that has component parts defined by the LP spec Otn111. In the Term instance of the LP instance of LTP instance A there are attA and attB that are instantiations of the attributes in the TermSpec case Otn111.1 which hence abide by the definition of the attA and attB in the TermSpec.

 © 2018 Open Networking Foundation

4.2.6 Who constructs the spec?

The following roughly sets out responsibilities and ownerships.

- SDO X develops network technology model
  - Technology experts carry out the work
  - The model is structured as per the technology operation focussing on the specific layering of the technology
  - The work uses an approach:
    - That separates degrees of termination, adaptation and connectivity
  - The modeller identifies relevant optionality and what the rationale for support is
  - Note that any aspect of structure at this stage may be flattened in the implementation
- ONF provide examples of use of the spec model to describe the technology developed by SDO X
- Device vendors uses the process described in section 4.2.5 Broad usage on page 47 to construct necessary LTP and LP spec cases.
  - These cases are related appropriately per LTP/LP instance
- Service designer uses a process similar to that described in 4.2.5 to construct service endpoint definitions

4.2.7 Usage pattern

An LP composition per layer-protocol can be developed that has all multiplicity [0..1] and [1] held directly in the LP and that has all multiplicity [..*] attributes held in composed classes of the LP (essentially becoming a list form in a realization).

4.2.8 Spec example

The figure below provides a view of the detailed content of a partially formed spec case where the attributes shown were derived from [ITU-T G.8052].

 © 2018 Open Networking Foundation

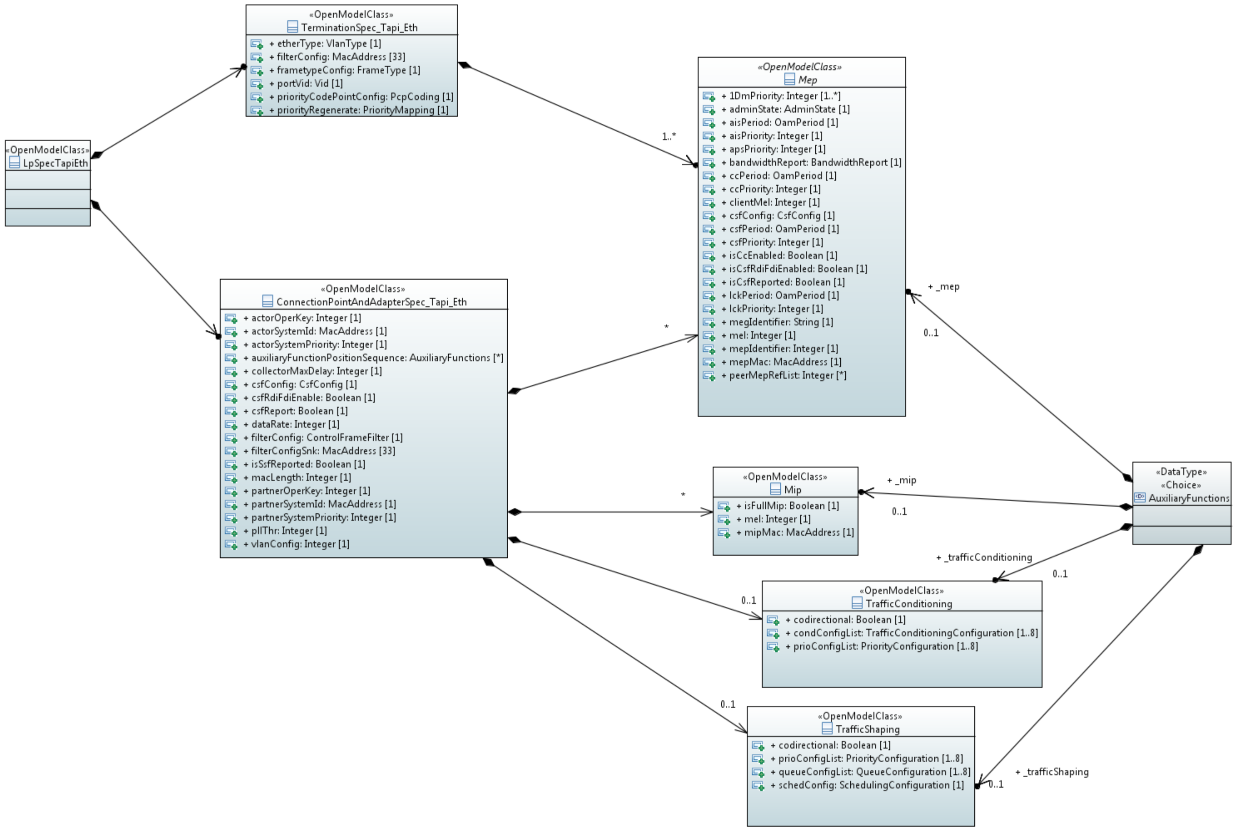


Figure 4-23 Sketch of Ethernet OAM spec case

The following figure shows the Ethernet spec case in context of the "Full Layer, Fixed" LP Case ([TR-512.2](TR-512.2)) and a fragment of the LP Spec model (see Figure 4-20 Relating LTP/LP spec elements to the pictorial symbols on page 44).

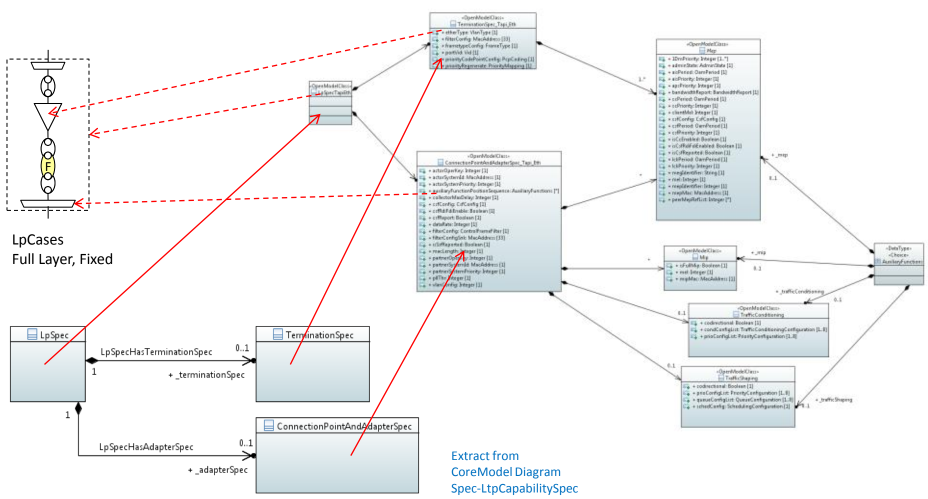


Figure 4-24 Sketch of Ethernet OAM spec case in context of LP Case and LP Spec model

 © 2018 Open Networking Foundation

4.2.9 Running system

The following sketches the application of spec cases in a running system.

- Vendor populates spec cases (including equipment cases, LP cases etc.)
- Vendor provides spec cases to the library
- Operator chooses to plan and prebuild intent for a particular type of "NE" (as part of a particular network fragment)
- "NE" spec identifies equipment spec which identify legal assemblies which identify instances of LTP cases supported
- Operator plans and prebuilds NE (expectation)
- NE is physically installed
- NE is discovered
- LTPs reported and validated against specs
- LTP specs are acquired where unexpected LTP types are found
- LTPs to be used are validated for the relevant parts to be used

4.2.10 Adoption migration

The following sequence explores adoption and migration for the LTP/LP spec. This section develops from section 3.4 Adoption and migration considerations on page 19.

- Step 1: (a) Null spec (traditional approach) and (b) empty spec (first step to spec model)
    - Spec reference attribute is supported in LTP and LP but is either set to empty string or provides the name of a spec that when opened is empty (this is a legal spec and could be taken ongoing as an indication that a coded solution or a basic discovery solution is required)
    - Assumes traditional coded solution
        - Capabilities are identified during enrolment of devices derived from current live equipment
        - Capability knowledge may be pre-coded from paper specs for the devices
        - Any planning will necessarily be done using coded solutions or handcrafted equivalents to the spec model (proto-spec)
    - Solution form is similar to that for an [TMF 612] realization
        - Attribute list is flat per layer-protocol (similar to layeredTransmissionParameters of [TMF 612])
    - Sophistication level: Very Low
- Step 2-n: Rudimentary spec through to full spec
    - The spec is referenced but some parts are intentionally empty
        - It is assumed that at this level of sophistication all parts of the spec will be included but where the spec aspect is not supported the part will be empty (rather than absent)
        - For the parts not present, traditional coding will be required
    - Expected growth path
        - Termination attributes included in the spec (connectivity and adaptation spec elements present but empty)

 © 2018 Open Networking Foundation

        - Connectivity attributes added (adaptation spec elements present but empty)
        - Adaptation attributes included (all properties are assumed to be independent)[37]
        - Inter-attribute and inter-adaptation rules included
    - Sophistication level: Low to Medium to High
- Indicating the degree of completeness/correctness
    - Use of lifecycle stereotypes in each spec case
        - Experimental/preliminary on an aspect of the spec means that that aspect in total will need code to support it and things will happen that are unexpected. Experimental means that definitions may be wrong as well as missing, preliminary means that what is in place should be correct (other than where specifically stated as experimental at the next level down) but there may be stuff missing
        - Example could be used in a general spec that provides suitable guidance but may have additional as well as missing attributes etc.
        - Faulty could be used when an error is found in a spec (the spec is republished with the faulty item marked)
- Naming a spec
    - The class name is the spec reference
    - The subordinate classes can be named systematically as extensions to the spec name where the conditional pac in the entity would be expected to simply have the extension name (and be a structural item in the encoded form)

4.2.11 Various applications of LTP spec

The following figure provides a view of how various interrelated bodies would use the spec approach. Note that the primary ONF responsibility (not shown in the figure) is the formulation of the spec model.

[37] May want to find a way to indicate that some trial and error expected

 © 2018 Open Networking Foundation

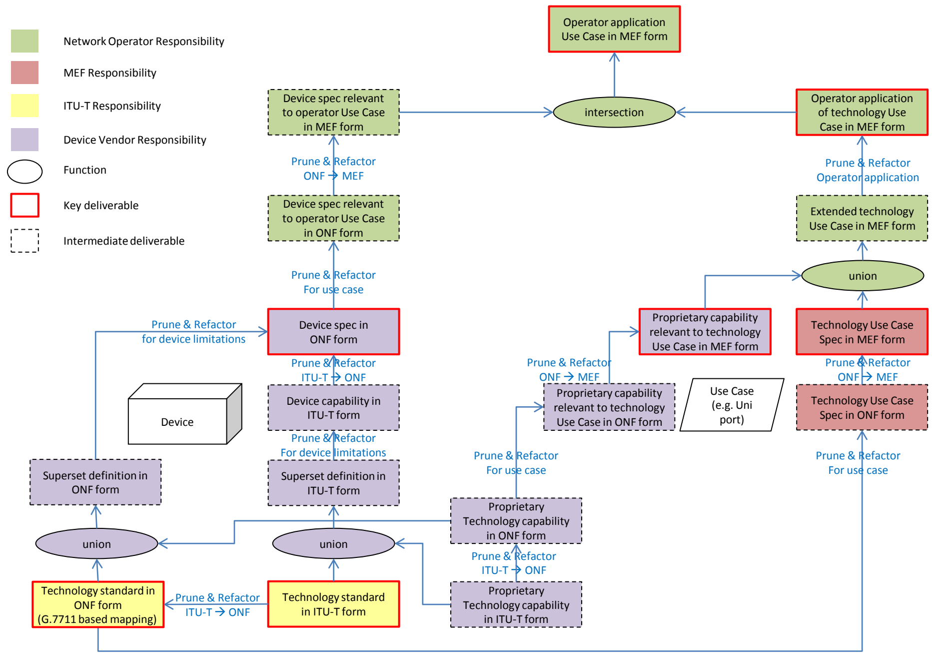


Figure 4-25 Sketch of spec usage

The flow starts with the Technology standard in ITU-T form at the bottom of the figure and via various transformations shows the production of:

- ONF forms of
  - Technology standard
  - Device spec
- MEF forms of
  - Technology Use Case spec
  - Proprietary capability relevant to Use Case
  - Operator application technology Use Case
  - Operator application Use Case

As the aim is convergence to the MEF and ONF forms (and removal of the ITU native flow), simplifications can be made to the diagram as in the figure below (where O/M form is the converged ONF/MEF model form).

 © 2018 Open Networking Foundation

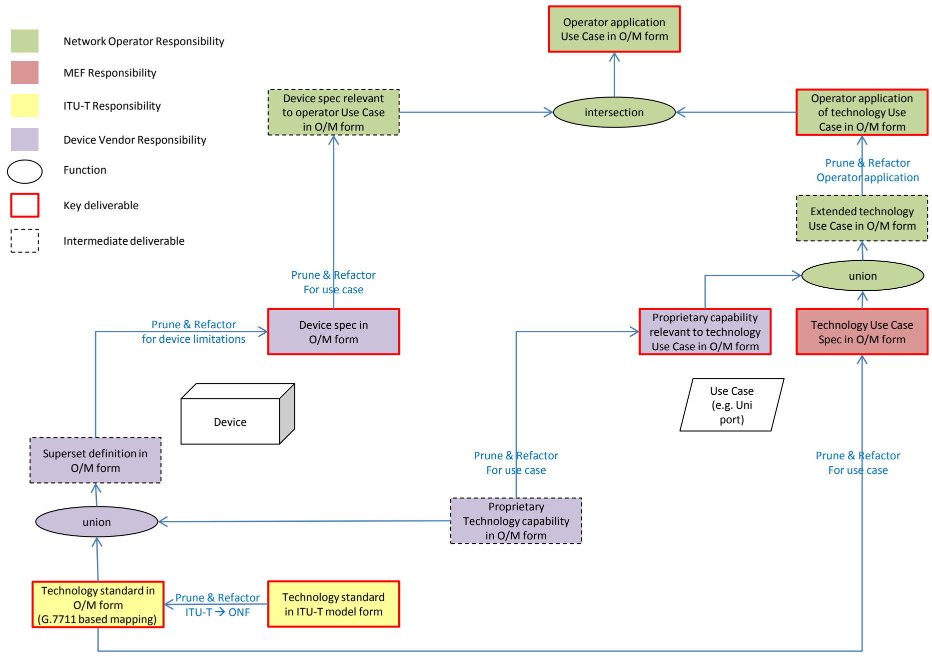


Figure 4-26 Simplified sketch of spec usage

4.3 ForwardingDomain (FD) and Link specification

The basic FD model assumes full flexibility such that there are no constraints on creation of FowardingConstructs) and such that the forwarding properties (such as cost, delay etc.) are equal for all FCs created in the FD. The common case of Link is point to point. Any multi-point Links are assumed to be fully flexible like the FD. In these simple uniform case (as discussed in TR-512.4)) the FD/Link may have directly applied cost properties etc. However for an asymmetric multi-pointed case, the properties will not be the same for all possible transits.

Where the properties differ for different transits and assuming that the FD/Link is "FC-opaque"[38] two options are available:

- Query per case: The client of the controller asks what specific properties would apply to a transit across the link/FD between specific bounding LTPs.
- Property model: The controller presents a property model for each FD/Link up front

This section discusses the "property model" approach. Presentation of a property model is assumed to be the most appropriate approach for general usage (including planning).

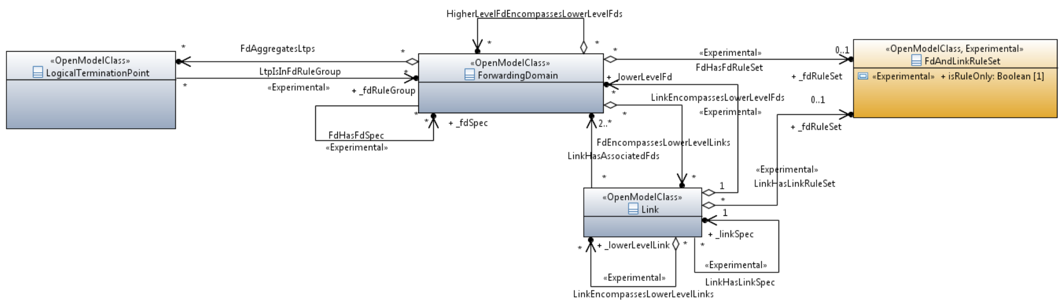
Unlike the two previous mechanisms, the FD rule mechanism uses simple instances. The rule instances may be shared between many instances of FD. For example where the rules correspond to the restrictions in a type of device, the FD instances that relate to instances of that type of device will all abide by the same rules and hence will all reference the same rule instances.

[38] The FD/Link does not decompose into subordinate parts for the purposes of creation of FCs.

 © 2018 Open Networking Foundation

4.3.1 FD/Link rule/property model pattern

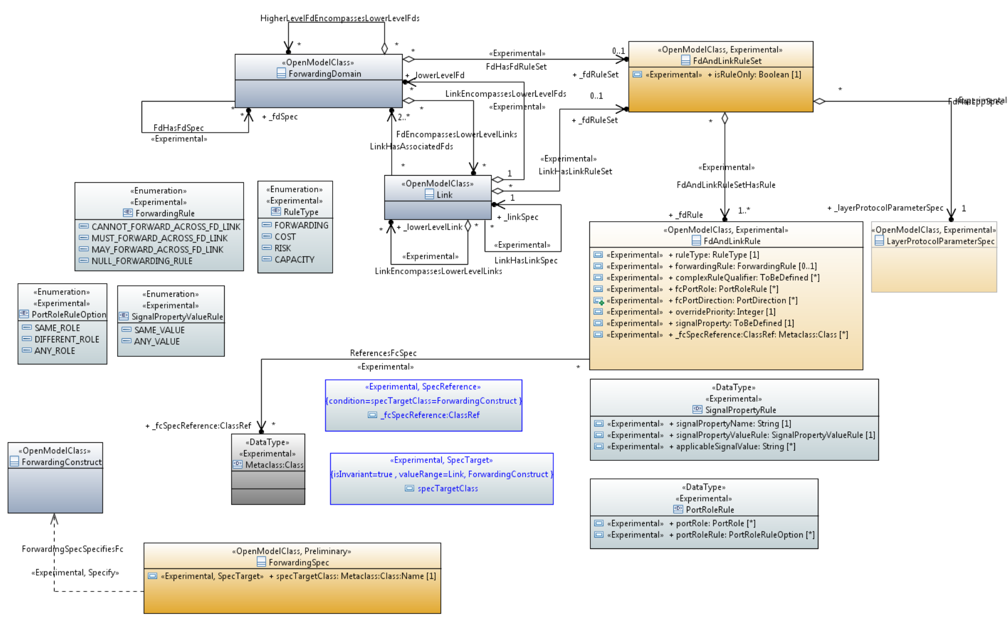
The following figure shows the FdAndLinkRuleSet in the context of the Link, FD and LTP.



CoreModel diagram: Spec-FdRulesInContext

Figure 4-27 Class Diagram of the context for the Spec Model of FD and Link

The following figure shows the details of the rules. Some rules relate to other classes in the model that are not shown in the figure.



CoreModel diagram: Spec-FdRuleDetail

Figure 4-28 Class Diagram showing the details of the FD rules structure

The figure above shows FD and Link possessing rule properties. Some of the rule properties may be per case and others may be per instance; the same essential structure applies for both. For the symmetric case, the properties apply to the FD/Link itself. Clearly the FD/Link can be broken

 © 2018 Open Networking Foundation

down into subordinate parts (as described in TR-512.4 and depicted below in a figure for FDs from that document).

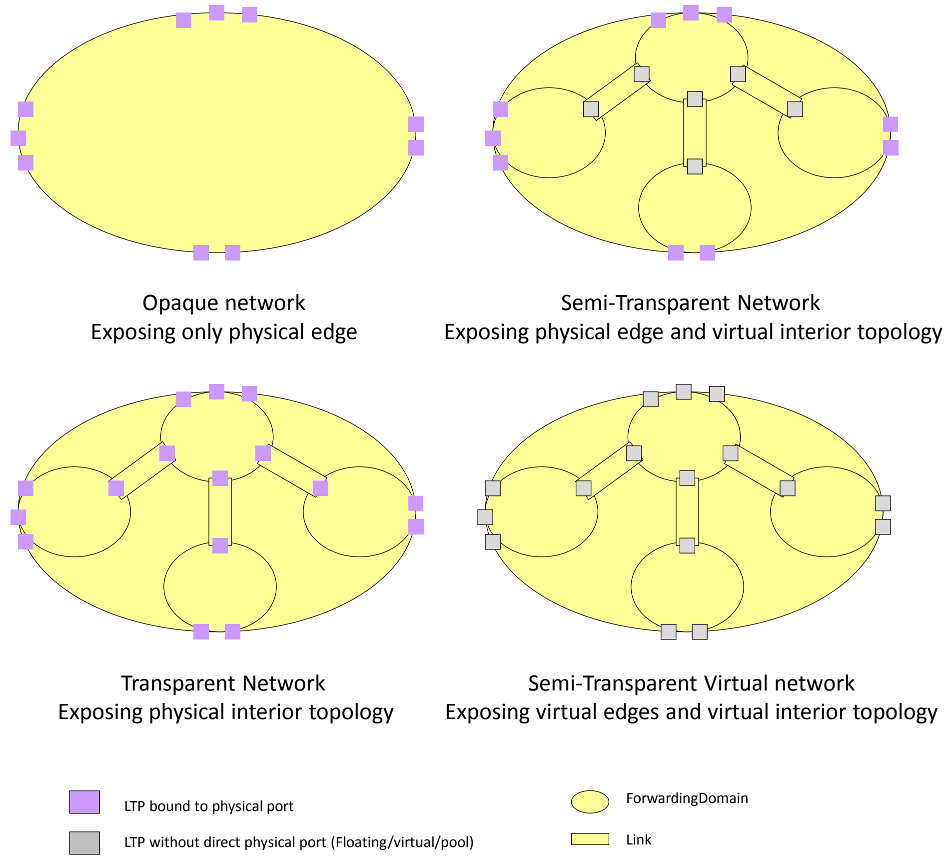


Figure 4-29 Various view boundaries

The subordinate FDs will have FCs present to represent the path of any top level FC across the subordinate FDs (as described in TR-512.5 and depicted below in a figure from that document).

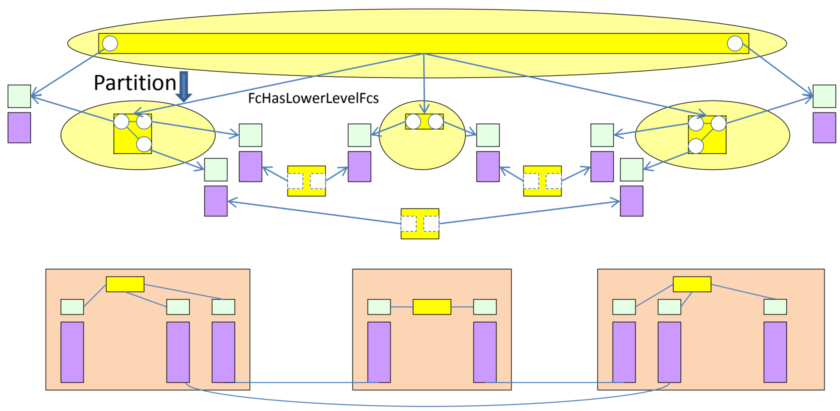


Figure 4-30 Simple summary example of 1?1 cases (represented via partition)

 © 2018 Open Networking Foundation

Clearly the rules related to building the top level FC in the figure above are provided by the partition FDs (where protection is allowed to be supported etc.). For an FC-opaque FD there is no visibility of FCs in subordinate FDs, but the FD could be partitioned to solely express constraints.

The constraint partitions need not be the same as the underlying FC partitions (in exactly the way that the apparent underlying FC partition depicted above need not reflect the real network as it may be an abstraction of the network). There is a brief discussion on views in TR-512.4.

4.3.2 Class details

4.3.2.1 FdAndLinkRule

Qualified Name:
CoreModel::CoreSpecificationModel::ForwardingDomainAndLinkCapability::ObjectClasses::FdAndLinkRule

Set of "AND" rules related to creation of FCs across the FD/Link (i.e., all rules have to be met for the FC creation to be allowed).
Embedded conditions all have to be met and hence are AND. Elements of the list attributes are ORs.
Absence fcSpec NOT valid for FORWARDING rules (only valid for cost/risk etc. rules).
Absence of FcPortRole means all roles for referenced spec.
Absence of direction means all directions.
This class is Experimental.

4.3.2.2 FdAndLinkRuleSet

Qualified Name:
CoreModel::CoreSpecificationModel::ForwardingDomainAndLinkCapability::ObjectClasses::FdAndLinkRuleSet

Set of "OR" rules related to creation of FCs across the FD/Link (i.e. only one of the rules have to be met for the FC creation to be allowed).
Absence of RuleSet means "Any", i.e. all points, all FcTypes etc.
Presence of RuleSet means possibilities must all be defined by rules
Absence of forwardingRule means no explicit stated possibilities.
No capacity statement means no capacity restrictions.

This class is Experimental.

4.3.3 FD/Link rule model detail

The figure below builds on the earlier figure showing normal partition and rule only partition (the virtual edge form is used but this would apply to any of the depicted cases). The FDs that provide only rules are considered as LtpGroups (not explicitly modeled – the FD by its very nature is an LtpGroup[39]) and the Links that provide only rules are considered InterGroupRuleLinks (again not explicitly modeled).

[39] An explicit class may be developed for LtpGroup rules although other insights suggest that this would not be the right direction.

 © 2018 Open Networking Foundation

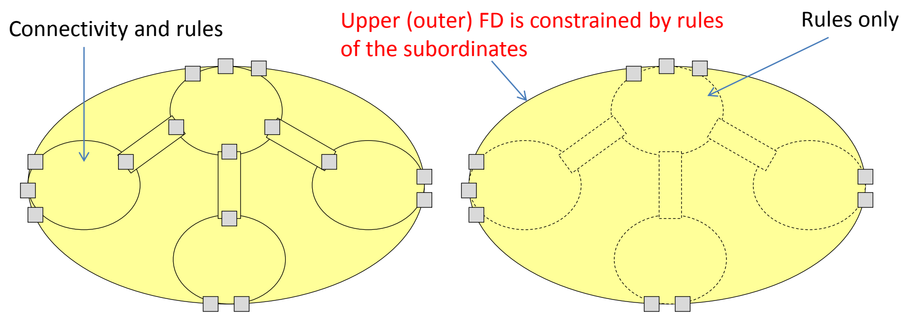


Figure 4-31 Normal FD/Link and rule-only FD/Link views

The method is based on the recognition that:

- At the lowest level of the real network there are deterministic elements, the assembly of which provide the abstract view.
- At some level of decomposition of the FD/Link structure there is a simple description that can be applied.

The rule system operates as follows

- No rule statement at a higher level means "refer to the next level down" such that rules at all levels are accumulated.
- It is assumed that a default of fully flexible would apply such that no rule statement at the lowest level means fully flexible, within the scope of that lowest level FD/Link.
- Rules within an FD correspond to LTPs bounding that FD
- Rules in Links correspond to LTPs in the bounding FDs only (i.e. FD-Link rules do not chain)
- Rules may be capability based or restriction based where a restriction overrides a capability statement
- Multiple different types of rules may be presented and should be analysed in combination, where each type of rule should be in a single FD/Link partition (i.e. a type of rule cannot spread across several partition views) and multiple types can be in one partition

The figure below sets out a rough example of rules in a single partition (that was depicted in the earlier figure).

 © 2018 Open Networking Foundation

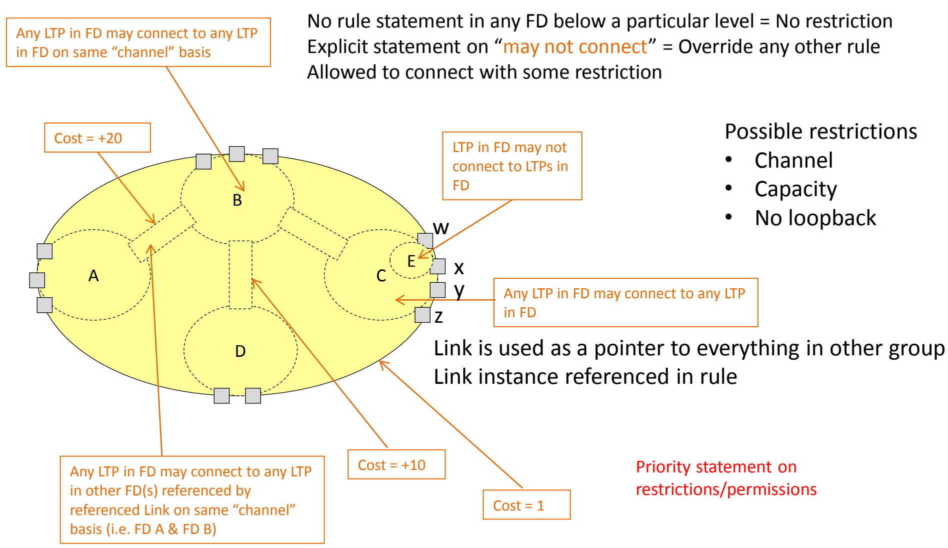


Figure 4-32 Rule example

Any FC created between exposed LTPs[40] must abide by the rules stated. The following assumes that bidirectional point to point FCs are to be created (there are rules related to orientation of more complex FCs that are not discussed here). So:

- An FC can be created between any LTPs in A with no restriction for a cost of 1
- An FC can be created between LTPs in B so long as the FC uses the same "channel"[41] on the LTPs again for cost of 1
- An FC can be created between an LTP in A and an LTP in B so long as the FC uses the same channel on the LTP in A and the LTP in B and in this case the cost is 21
- It is not possible to create an FC between LTPs in A and LTPs in D
- In C, an FC can be created between LTP w and LTP y, but not LTP w and LTP x (because E overrides other statements)
- An FC can be created between LTP x in C and any LTP in B with no restrictions for a cost of 1
- An FC can be created between an LTP in D and any LTP in B with no restrictions for a cost of 11

There may be other constraints related to the network shown above. The figure below considers risk statements for shared risk assessment and relates them to LTP combinations using the mechanism described. The grouping for one rule set need not be related to the groupings for another rule set.

[40] Where the server LTP is represented at the boundary of the FD (see TR-512.4)
[41] Any property of the layer-protocol could be required to be preserved, channel is just an example.

 © 2018 Open Networking Foundation

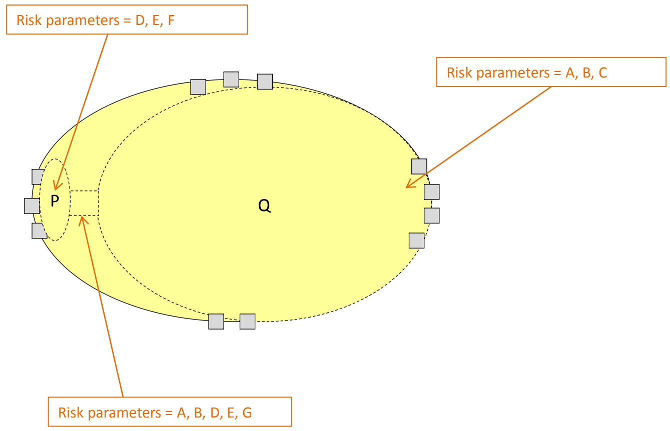


Figure 4-33 Rule example showing risk parameters

In the figure above, an FC:

- Between LTPs in P incurs risks D, E, F
- Between points in P and points in Q incurs risks A, B, D, E and G

4.3.4 Link asymmetries

The Link model follows the same pattern as the ForwardingConstruct and has the same need to deal with potential asymmetries, for example where a server layer FC offers dual homed protection to a client layer such that the server layer has four related ends. The figure below (adapted from a figure [TMF TR215]), shows such a case where the link (highlighted with {DAD}) is multi-pointed asymmetric (and has exactly the same asymmetries etc. as the supporting FC).

This complex case shows Interconnect ({IC}) protection with roles Resilient, Bridge and Protection and Double Add-Drop with roles (where the roles are in pairs (left M/S pair and right M/S pair)).

The lower diagram in the figure shows the HO path across the BLSR connecting the terminations in B, C, D and E that support the LO CTPs shown in the upper diagram in the figure below.

 © 2018 Open Networking Foundation

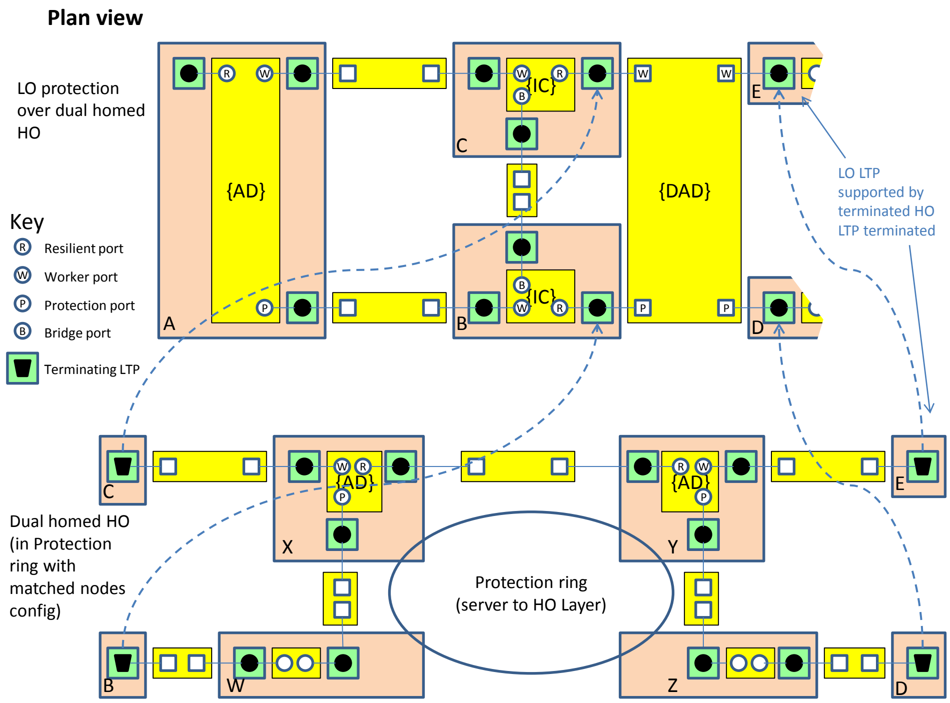


Figure 4-34 Multi-pointed flexible external FC scenario with Dual Homing [TMFTR21]

As shown in the earlier figure, the Link has access to the ForwardingSpec.

4.3.5 LayerProtocol parameters

Also shown in the earlier figure the Link and ForwardingDomain have access to the LayerProtocolParameterSpec. This is a rudimentary model at this point. It essentially provides a vehicle to convey layer protocol specific parameters to the FC (and Link/FD).

There is a complexity here that has not been fully developed in that the parameters are invariable abstractions of properties of the LTP/LP. This will be developed in more detail in the next release.

4.4 PC, ControlComponent and C&SC spec considerations

The ProcessingConstruct (PC), ControlComponent and ConfigurationAndSwitchController (C&SC) can be considered as using a common spec approach. All three can encapsulate complex functionality and all three can expose an abstraction of that functionality. This is essentially the Component-System pattern spec (see TR-512.A.2).

The C&SC can be considered as an example when discussing the general pattern. The general pattern uses layers of specification.

The specification method uses a combination of specification references, pruning & refactoring and scheme specs. The scheme spec has not yet been fully developed and hence the work here is early experimental.

 © 2018 Open Networking Foundation

The scheme spec can take any classes from the model and lay out a pattern of these classes. The pattern is generated using a pruning and refactoring process, where the pattern model is a pruned and refactored CIM. The pattern can include more than one case of each class, along with statements of rules (in OCL) that constrain the assembly. The attributes of the classes, as is the case for any spec, have extended type allowing for statements of range, effort etc.

An example of application of scheme spec could be considered for [ITU-T G.8032] protection. In a full model the scheme spec mechanism would apply as shown below.

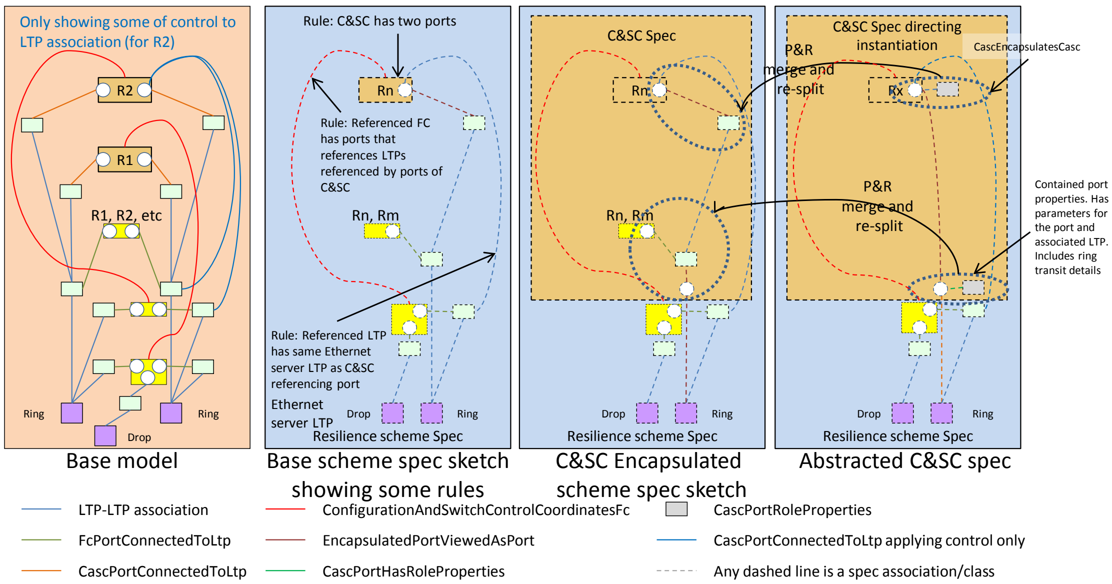


Figure 4-35 Scheme spec example

The "Base model" diagram shows an example layout, in an instance diagram form, of a single [ITU-T G.8032] node (see TR-512.5) that is involved in two protected rings (R1 and R2).

The "Base scheme spec sketch…" diagram shows a detailed representation of the nodal aspects of the [ITU-T G.8032] protection scheme spec. The elements of the spec are created from the ONF CIM using the "Prune and Refactor" (P&R) approach. This supports the construction of several cases of any class from the model with corresponding associations from the ONF CIM. The spec is augmented with rules that constrain the creation of instances of entities of the scheme so as to abide by the scheme definition. The scheme spec structure is aggregated by a ConstraintDomain (see TR-512.11), i.e. the scheme spec identity is that of a ConstraintDomain.

The "C&SC Encapsulated…" diagram shows the results of a second P&R stage where the scheme spec is taken and embedded in a C&SC shell. This intermediate step provides an aspect of the mapping of the raw scheme to the "Abstracted C&SC spec". As the FCs and LTP cannot be embedded in the C&SC, the model is somewhat of a hybrid but it allows the next step of construction.

The "Abstracted C&SC spec" diagram shows the results of a third P&R stage where the properties of the LTPs (including association ends) are merged into the corresponding C&SC ports, as are the port properties of the FC and the FC itself (the FC itself has no relevant properties).

 © 2018 Open Networking Foundation

In this case the "Base scheme spec…" is further backed up by a more detailed set of specs for the C&SC for [G.8032]. At this point the spec for [ITU-T G.8032] is not documented in a machine interpretable form. The longer term intention is that it would be machine interpretable.

As a consequence of the above steps, there is a formal path from the "Abstracted C&SC spec" to the definition of the detailed underlying mechanism. As a consequence, the representation from an implementation that uses the "Abstract C&SC spec" form can be transformed in a running solution to a view that follows the "Base scheme spec.." using a machine interpretable definition of the transformation.

As the detailed set of specs are moved to machine interpretable forms, an advanced controller will have the information to fully interpret the protection scheme and its data.

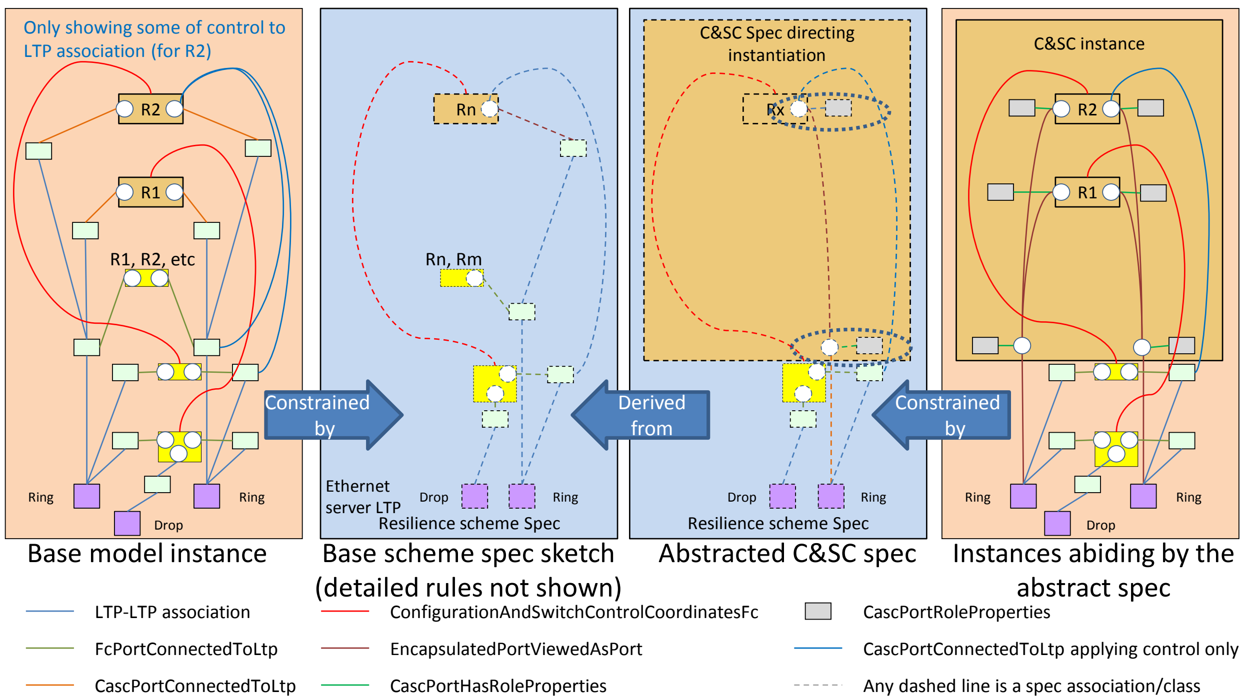


Figure 4-36 Scheme spec example showing derivation

The figure above shows the relationship between instance sketches and the corresponding specs, and highlights the relationship between the specs.

4.5 Acquiring the specifications run-time

The following sketches show how a controller (or other system requiring details of the specifications) could acquire the specification on discovery of a previously unknown controllable thing.

 © 2018 Open Networking Foundation

The figures are intentionally generalized. The controllable thing could be a network element or another controller.

4.5.1 Initial system arrangement

The figure below shows a controller that has been designed to understand control of a layered network but that does not yet know any specific layer-protocols or parameters associated etc. The Controller is not yet aware of the Controllable thing, its instance repository is empty. There is a schema repository[42] known to the Controller.

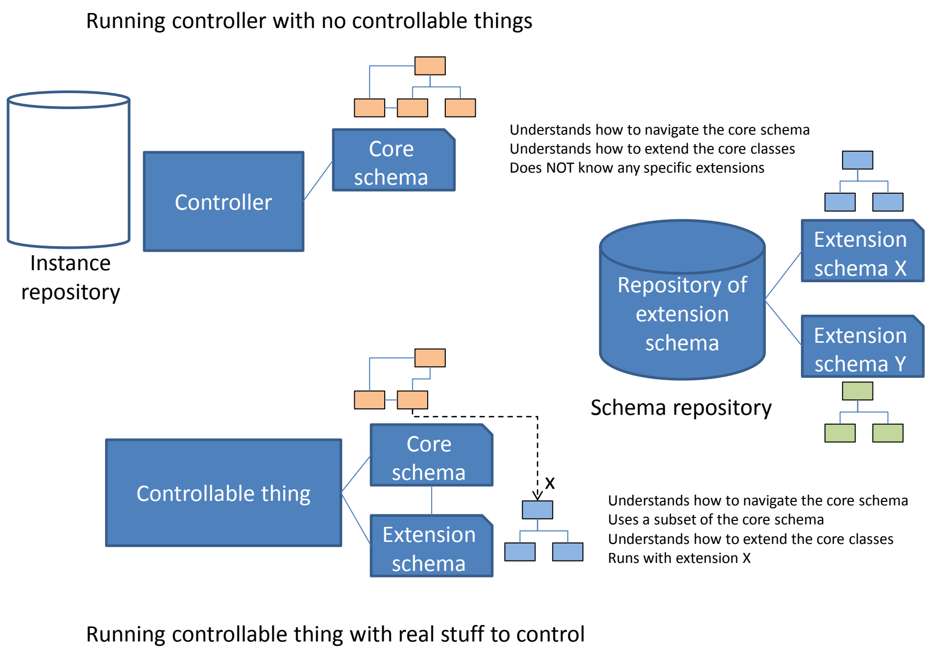


Figure 4-37 Controller and Controllable thing not yet connected

4.5.2 Learning to control the Controllable thing

In the following figure the Controller has been asked to control the Controllable thing. It has connected and is retrieving information.

[42] There may be many schema repositories (one per vendor) etc.

 © 2018 Open Networking Foundation

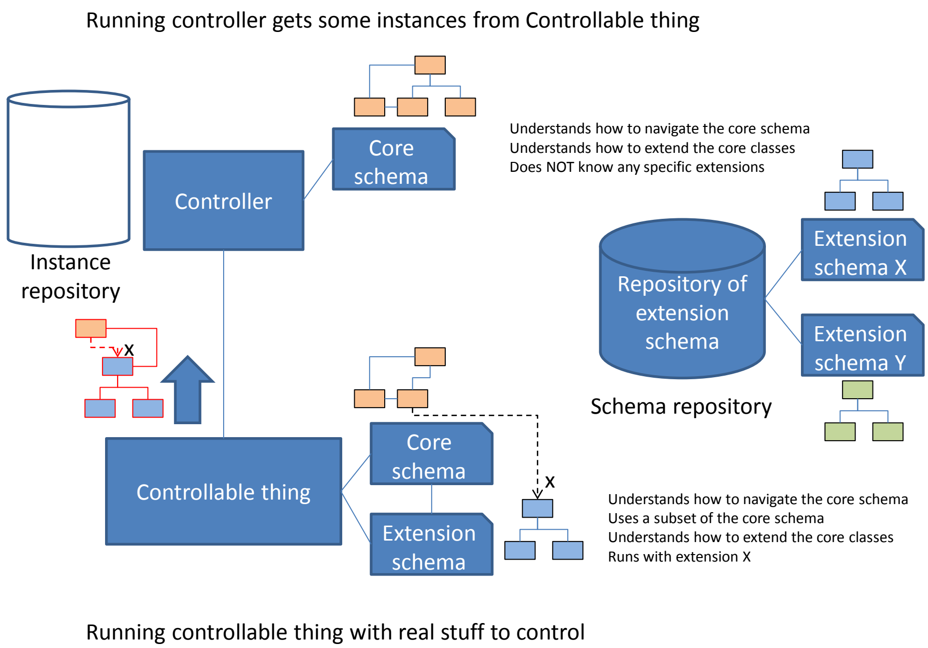


Figure 4-38 Controller connected to Controllable thing and retrieving information

The figure below shows that the Controller has stored the information it retrieved from the Controllable thing in its instance repository and has recognized that some information is not yet interpretable but that the information is related to a schema "X". The controller has asked the Schema repository if it has schema "X" and that schema is being sent to the controller. Schema "X" is a run-time version of a case of a specification (described earlier in this document).

 © 2018 Open Networking Foundation

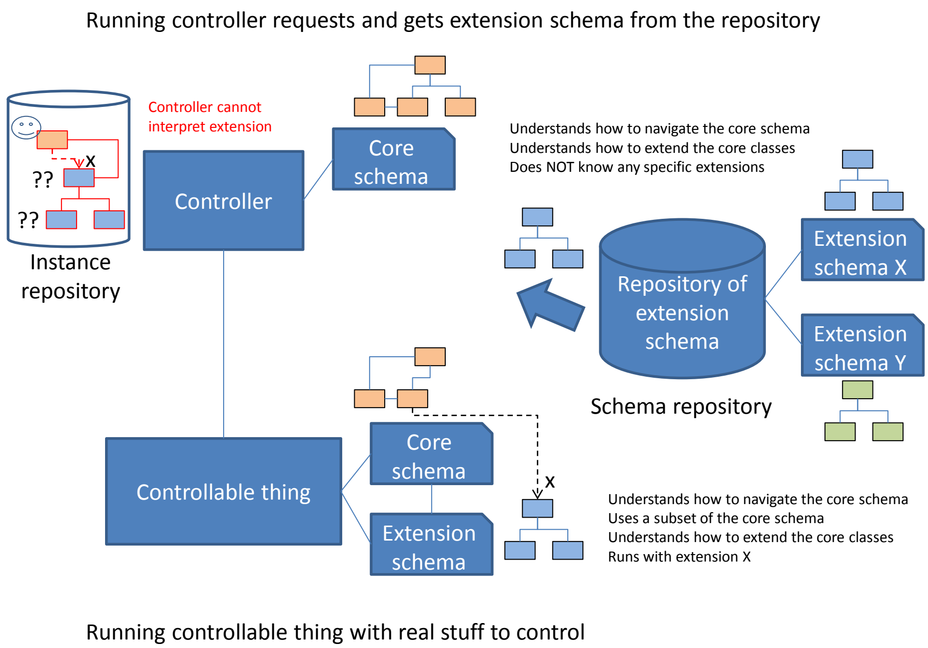


Figure 4-39 Controller acquires information on schema "X"

In the figure below the Controller now has schema "X" and hence it has:

- all invariant details for the discovered aspects of the Controllable thing (including information not known to the Controllable thing
    - These invariant details include simple information and rules
- the definitions of the dynamic properties
    - This includes read-only or read/write
    - Default values
    - Legal value ranges
    - Etc.

Hence the Controller can interpret the attributes and in a basic solution provide the user with the opportunity to display and, as appropriate, set the attributes.

 © 2018 Open Networking Foundation

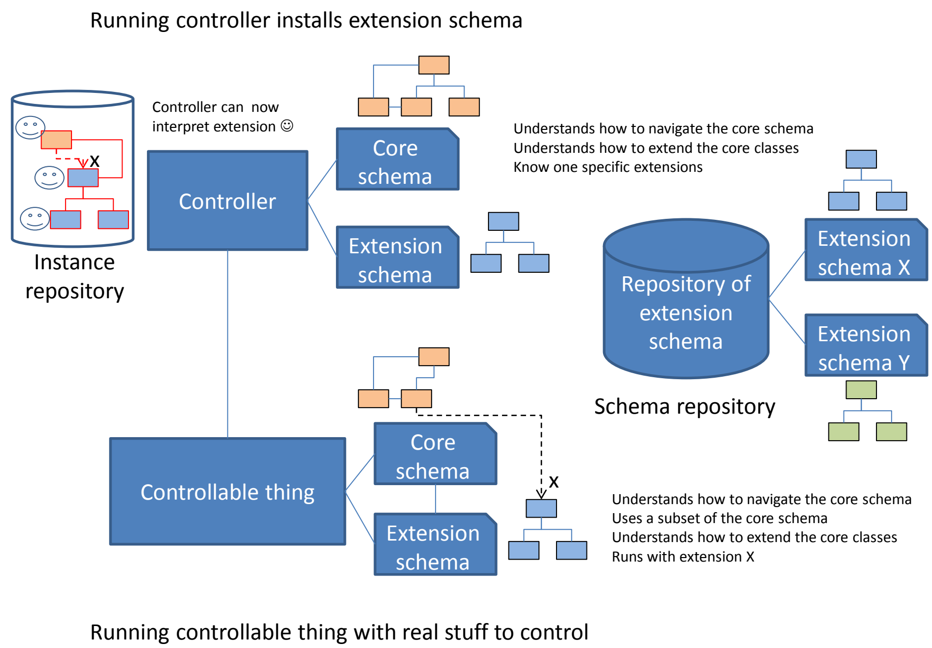


Figure 4-40 Controller can interpret and use attributes from schema "X"

4.5.3 Implications of the above

Clearly this is a very basic and simplified walkthrough and there are further system features required to fully integrate the Controllable thing.

- Controller API definition at start-up only includes the core schema
- Controller API definition is dynamically extended on arrival of a thing with an extension
- The controller may need to also acquire behaviour to deal with the new attributes etc. of the extension BUT it may be able to provide significant capability without any additional behaviour

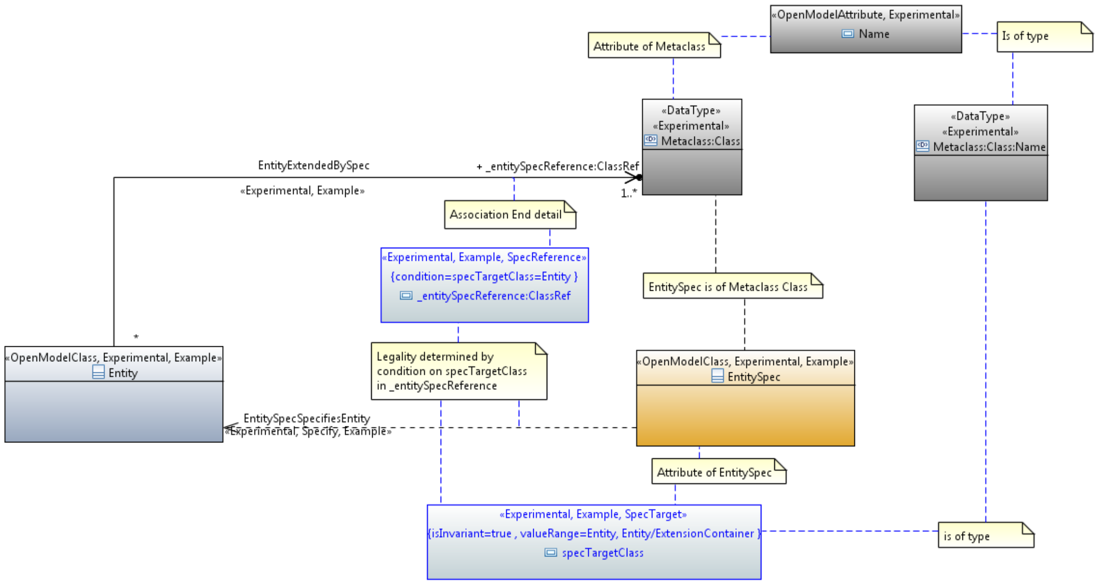
4.6 Work on the general pattern

This work is early draft experimental.

4.6.1 The Specification Model Pattern

The following figure shows the essential specification pattern.

 © 2018 Open Networking Foundation



CoreModel diagram: Spec-CoreOfPattern

Figure 4-41 Class Diagram of the spec model pattern

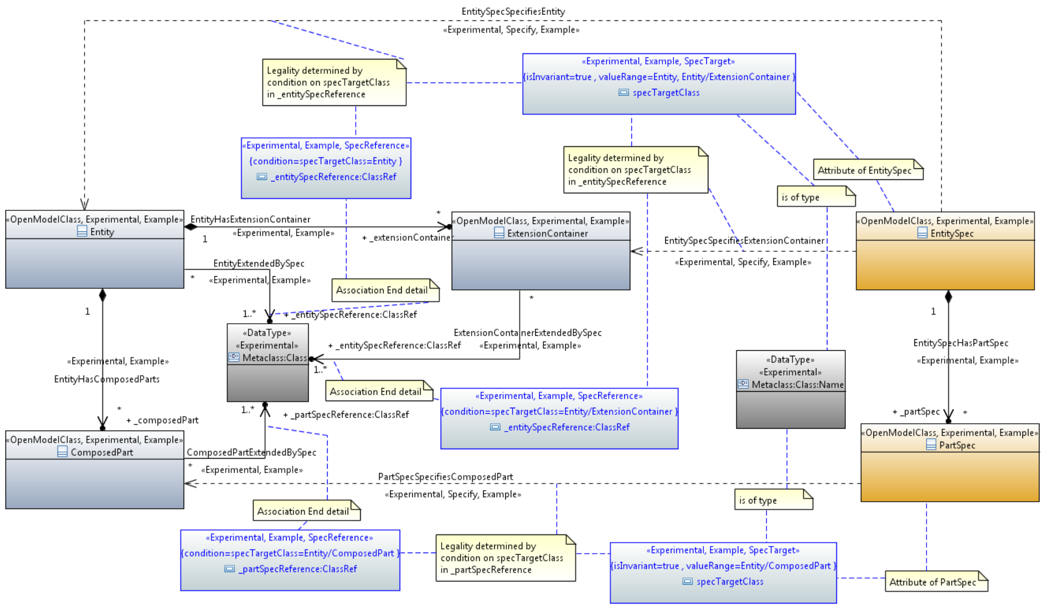
An instance of Entity will reference one or more Classes (in this figure the example class is EntitySpec). The referenced Class will be confirmed as appropriate it will have a specTargetClass attribute and one of the named items in the specTargetClass attribute will be the name of the class of the referencing instance (in this case Entity). The attributes that have been pulled out of the classes along with the comments in the figure highlight the validation.

Assuming that the referenced class is valid, then the definitions (attributes, rules etc.) will be assumed to apply to a subset of the attributes etc. of the Entity instance. The Entity instance may have attributes that are not defined in the Entity Class. In this case the definition for each should be found in the EntitySpec Class.

The spec reference is retained in the Extension instance and the Extension instance is augmented by the attributes defined in this Spec.

The figure below shows a more complex case, where the Entity has subordinate parts and the EntitySpec has corresponding subordinate parts.

 © 2018 Open Networking Foundation



CoreModel diagram: Spec-FullPattern

Figure 4-42 Spec pattern showing further detail of application

The figure above shows several validations for the expanded case.

4.6.2 The Scheme Specification approach (requires further development)

As discussed, a scheme specification is constructed by Pruning and Refactoring the core to produce a number of "scheme role classes" for each relevant class from the core in a structure that relates to the scheme to be represented.

An instance of a core class which participates in scheme will reference, via the specification approach defined above, a scheme role class as well as a capability spec class.

The scheme spec can use SubordinateForwarding to describe route rules.

It would appear that the all capability specs are essentially scheme specs and the scheme spec pattern appears to be the basis for enhancement of LTP and Forwarding specs. The Scheme spec will be developed further in future releases.

4.6.3 Attributes of the spec (requires further development)

 © 2018 Open Networking Foundation

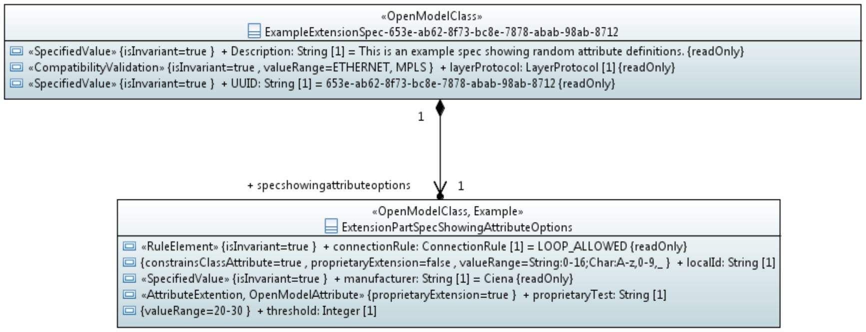


Figure 4-43 Example of types of properties in a spec

A specific runtime Entity instance has an associated spec. The attributes of the Entity abide by the referenced spec. The schema is defined in the associated spec which provides structure, attributes and rules.

The attributes defined in the Spec become attributes of the Extension where the attributes are NOT

- <<SpecifiedValue>> (where the attribute does not need to be repeated as it is fixed for all instances referecing the Spec),
- <<RuleElement>> (again fixed and also not directly meaningful for the entity itself)
- <<CompatibilityValidation>> (again fixed and not directly meaningful for the entity itself)

There is a basic mechanism suggested to allow the spec to modulate existing properties of a class.

4.6.4 Thoughts on Profiles (requires further development)

The profile will abide by a Union of specs. The Referencing StructuralUnit would be expected to reference at least one spec of the Union. Some cases will be rejected as the profile constraints may violate the specific Spec. Although multiple profiles can be applied, some combinations will not be allowed. There will need to be a precedence ordering of rules for the multiple profile cases. A profile is itself a structural unit and hence can have profiles applied to it.

When a profile is applied, the properties available through the StructuralUnit change (runtime/dynamic).The profile will ALWAYS constrain and never add capability.

Visibility:

- NotVisibleInTarget: Can only see the value in the Profile
- ReadOnlyInTarget: Can see the value in the target but it is not adjustable in the target
- OverridableInTarget: Can be written in the target so that the target does not align with the profile (profile should provide a realign option). The StructuralUnit must indicate that it is not aligned with the Profile.

EffortSettingTarget:

 © 2018 Open Networking Foundation

- BestEffort: If the attribute is not available then no problem. If the value chosen cannot be set then the target stays as is. Best effort attributes can be overridden without any flagging of misalignment
- MandatoryClosestHigher: The attribute must be available but if the range is reduced in the specific case, then the closest higher value should be chosen. Hence the profile can only be applied where the attribute is available.
- MandatoryClosestLower:
- MandatoryExact: The exact value must be applied, hence the profile can only be applied where the attribute is available with exactly the same value range.

InitialValue is a special case of Profile that has an association only present for an instant at creation of the StructuralUnit and where the content of the profile is only Fixed.

Implied context requires specific navigation per case. It may be that the navigation is inherently obvious for all cases. For example the LtpGroup referenced from an LTP/LP must be one related to an FD that the LP is a member of.

Note that a profile can be encapsulated in another StructuralUnit from the one it applies to.

4.6.5 Rules related to the pattern (required further development)

Note that this section has not been updated in this release. The figures are not precisely aligned with the current model pattern but they do provide some additional details that are still not absorbed into the formal model and hence have been left the document.

This model fragment is not in the ONF Core Model and is provided here solely for information on direction.

The figure below shows a sketch of an expanded view of the spec model pattern (Entity, Extension and Spec), adds a consideration of Profile and the relationship to the Spec and also provides a sketch of some potentially useful stereotypes, in the two extension specs to the top right of the figure, to deal with marking of properties that do not get instantiated in the entity instance.

The following figure shows a sketch of the rules related to specification usage.

 © 2018 Open Networking Foundation

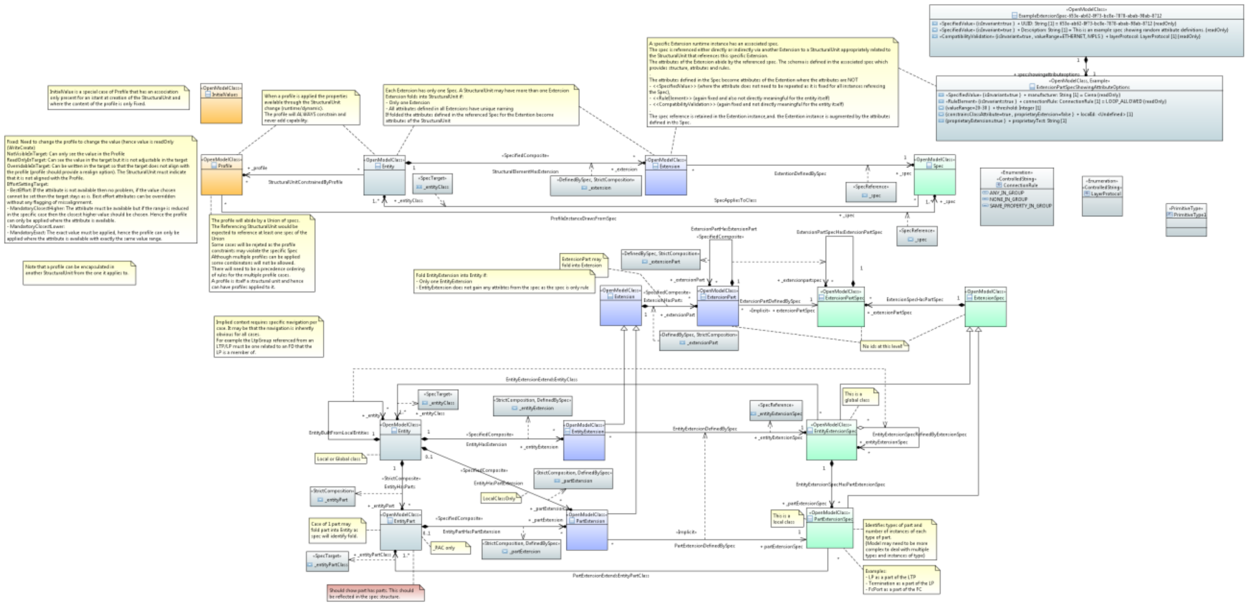


Figure 4-44 Basic spec pattern with rule sketch

4.6.6 Further vision considerations

Classification model becomes a rule pattern view of generalized (component-system) classes as per the sketch above

Any aspect of the rule pattern could be per instance or per case, invariant or dynamic.

Should the FC spec be rationalized to recognise that MSUF is essential an FC (implications etc.)?

Should spec structure such as Termination actually be structure in the LP model?

How should constraints on each class spec be expressed in the context of a generalized spec?

Relationship to dynamic APIs

End of document

 © 2018 Open Networking Foundation