Core Information Model (CoreModel)

TR-512.A.10 Appendix – Specification Examples

Version 1.4
November 2018

ONF Document Type: Technical Recommendation
ONF Document Name: Core Information Model version 1.4

Disclaimer

Open Networking Foundation
1000 El Camino Real, Suite 100, Menlo Park, CA 94025
www.opennetworking.org

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

List of Figures

Document History

Version	Date	Description of Change
		Appendix material was not published prior to Version 1.3
1.3	September 2017	Version 1.3 [Published via wiki only]
1.3.1	January 2018	Addition of text related to approval status.
1.4	November 2018	No change.

1 Introduction

This document is an appendix of the 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%) or open the associated image file (and zoom appropriately) or open the corresponding UML diagram via Papyrus (for each figure with a UML diagram the UML model diagram name is provided under the figure or within the figure).

1.5 Understanding the figures

Figures showing fragments of the model using standard UML symbols and also figures illustrating application of the model are provided throughout this document. Many of the application-oriented figures also provide UML class diagrams for the corresponding model fragments (see TR-512.1 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.

1.6 Appendix Overview

This document is part of the Appendix to TR-512. An overview of the Appendix is provided in TR-512.A.1.

2 Introduction to this Appendix document

This document provides various examples of the use of the CIM specification model to express constraints in various real contexts.

The examples in this document are built from descriptions in earlier referenced works.

3 General examples

3.1 The FD and FC Spec

This section focusses on the use of the FD spec to state restrictions in a network or NE/device wrt creation of FCs. The FD spec is used in conjunction with the FC spec for this purpose. A "fabricated example" NE/device is used where the rules and restrictions do not represent any particular device/NE type known. It is likely that those familiar with various devices from various vendors will be able to recognize where rules of the sort described could be used.

3.1.1 A basic NE/device example

3.1.1.1 FC spec used for the basic NE/device

The following figure provides an FC spec for a NE/device.

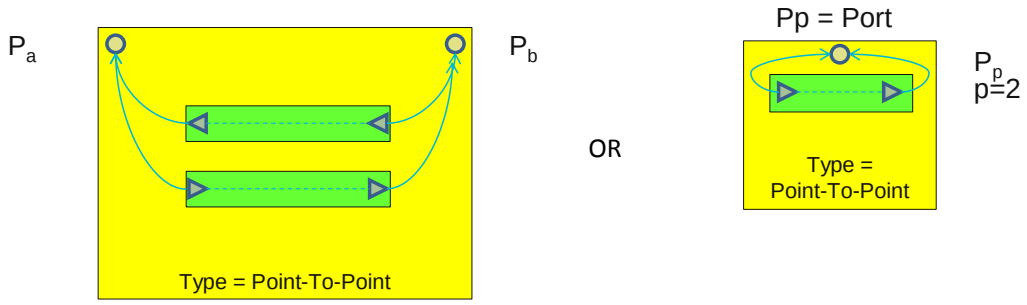

Figure 3-1 FC spec for unprotected FC

3.1.1.2 The NE rules

The NE has the following characteristics:

- The NE supports bidirectional "Point-To-Point" FCs (specs shown below) with "traditional" restrictions described below.
  - All ports on the NE are multi-channel
- The NE has 4 card types
  - Type 1 which has a single port which does not allow connectivity between channels on the port
  - Type 2 which has 8 ports that cannot do any port to port connectivity within the card

  - Type 3 which has 8 ports that can do arbitrary port to port within the card including from a channel within a port to another channel within the same port
  - Type 4 which has 8 ports that can do arbitrary port to port but cannot do connect two channels in the same port

- A port on a Type 1 card can connect to a port on
  - Another card of Type 1 on a same channel basis (or no vid swapping etc)
  - A Type 2/3/4 card with no channel restriction
- There is a backplane capacity limit between trib and Line port cards of one Line port capacity
  - A traditional NE might call the port on the Type 1 card a Line port and a port on the Type 2/3/4 card a trib port

3.1.1.3 Representing the NE restrictions

The NE can be represented using FD rules as set out in the figure below.

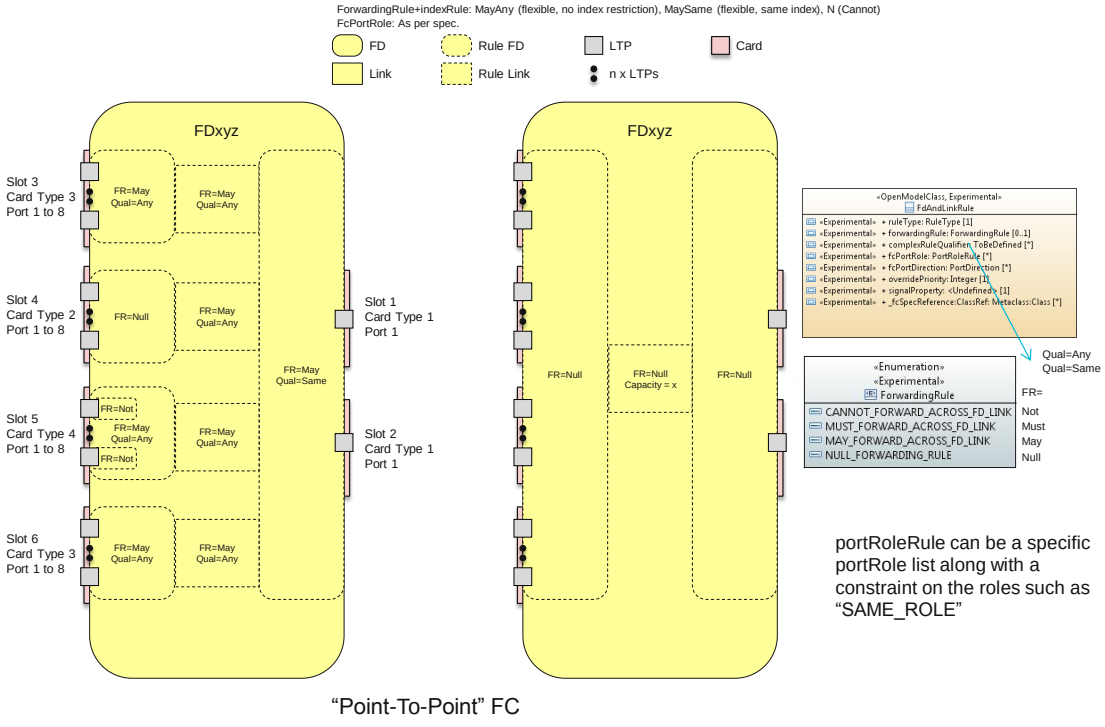

Figure 3-2 FD spec for a "fabricated" NE/device type

From the figure it can be seen, for example, that:

- Any "channel" on a port on an instance of card type 3 (in slot 3 and slot 6) can connect to any "channel" on any port (including the same port) on the same card instance.

- A "channel" on the port of the instance of card type 1 in slot 1 can connect to the same "channel" on the port of the instance of card type 1 in slot 2.
- Any "channel" on the port on an instance of card type 3 (in slot 3) can connect to any "channel" on any port on the same card instance.
- Any "channel" on a port on an instance of card type 4 (in slot 5) can connect to any "channel" on any port (but excluding the same port) on the same card instance.
- There is a capacity limit between the ports on the left and the ports on the right such that FC can be constructed between those ports until the capacity is reached.

The above rules are encoded in data following the model shown in the figure and discussed in more detail in TR-512.7. Instances of class of the model can be arranged to show the restrictions of a device or a network. If there are many instances of device with the same restrictions the same instances of rules can be used.

3.1.2 Enhancements to the basic NE

In the figure below the device discussed above is enhanced such that it is possible to set up FCs between ports in slot 4 and ports in slot 5 on a same "channel" basis.

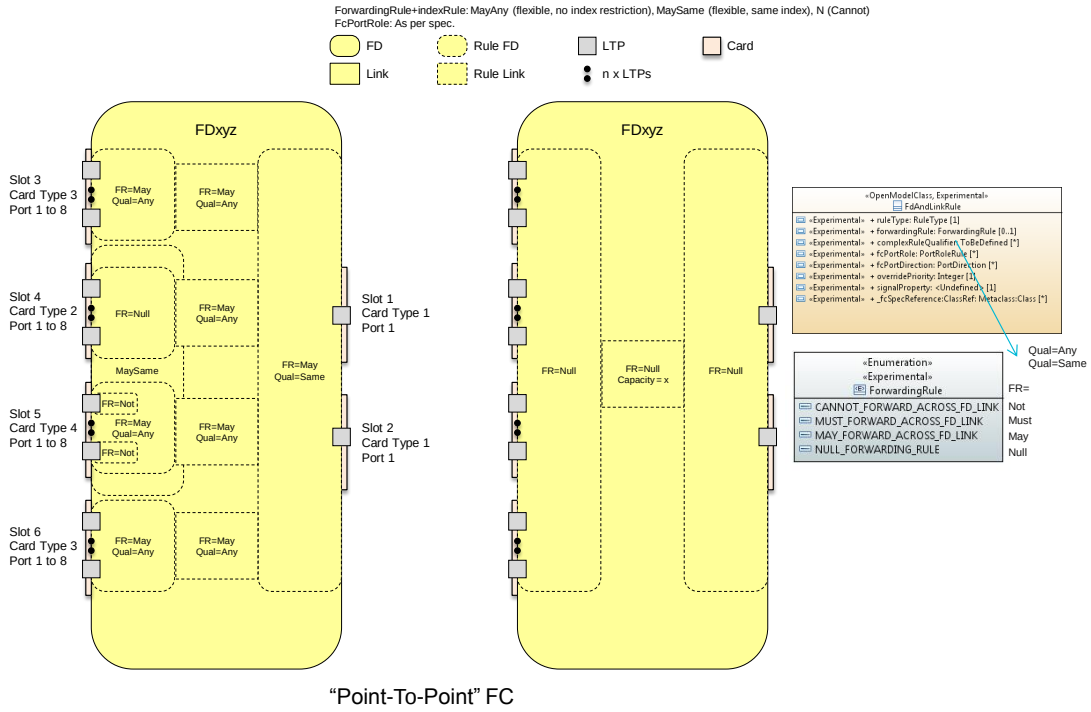

Figure 3-3 Simple FD Spec showing further overlay of rules

- As before the NE supports "Point-To-Point" FCs with traditional restrictions
- The NE also supports "Protected" FCs with similar (related) restrictions

  - A trib port can be protected by the Line ports where there is the same channel between Line ports but the trib port may be any channel

- And the NE supports "Back-To-BackSNCP" FCs
  - Two trib ports on any cards can be protected by and can protect two Line ports where there is a same channel rule between Line ports but no channel restriction on the trib ports
- Finally the NE supports "Root-leaf" FCs
  - Where the Line ports are the roots and the tribs the leaves
  - Leaves can be interconnected on some cards
  - Some cards can only support a single leaf
- With all of the above there is a total backplane capacity limit between Trib and Line port cards of one Line port capacity (x)

3.1.3 NE supporting more FC types

3.1.3.1 FC specs for more sophisticated NE example

The figure below shows FC specs for protected and root/leaf FCs.

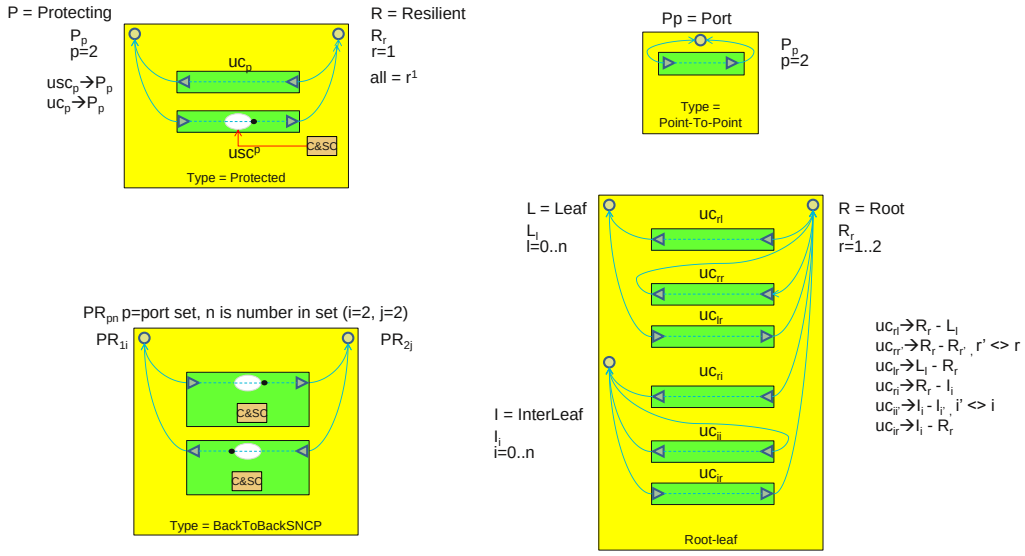

Figure 3-4 FC spec for protected and root/leaf FCs

3.1.3.2 Representing the NE restrictions

A device could support the FC types set out above as shown below. As overlaying all the rules would make the figure impossible to read the overlays are set out side by side. The "fabricated" device has a single FD that supports all capabilities set out in the figure below.

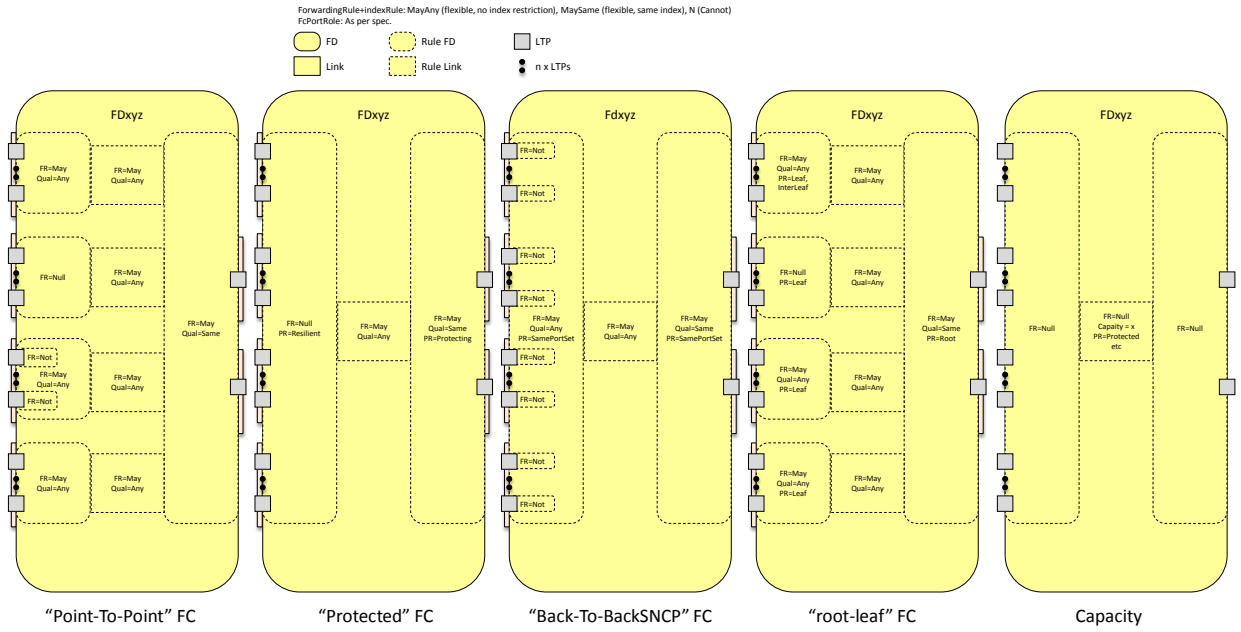

Figure 3-5 Simple FD Spec showing additional capabilities

3.1.4 Alternative representations of the same capability

The rule system allows various representations of the same capability. It is not intended that the patterns of rules be standardized. It is expected that the rules will be interpreted and hence as long as the rules conform with the rule model then an interpreter should be able to follow them. Some rule combinations will be more efficient than others at stating the viability of a particular FC. It is up to the coder of the rules to choose the rule form.

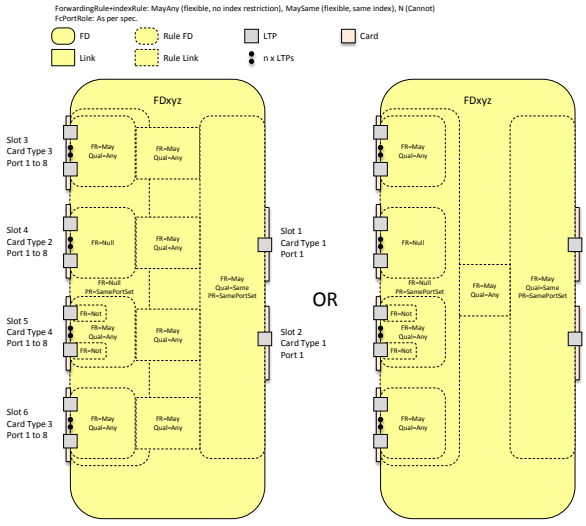

Figure 3-6 Simple FD Spec showing additional capabilities

It should be noted that the rule system can be used to create a connectivity/connectability matrix if every channel of every port has a dedicated rule statement to every channel of every other port in the device. Clearly this yields an extremely verbose solution where the construction of the rule structure would be particularly prone to error. Unless the forwarding restrictions of the device/network are extremely complicate and are not patterned it is recommended that a connectivity/connectability matrix approach is not used.

3.1.5 FD rule precedence

The overlaid FDs interact both at an FD level and at a rule level

3.1.5.1 Rule FD precedence

Overlaid FDs interact such that:

- An FD with no rule FDs has no restrictions and is essentially May/Any
- LTPs in an FD that are also in a rule FD contained in the FD
  - Abide by the rules of the rule FD
  - Are interconnectable to any LTP in the FD not in any contained rule FDs on a May/Any basis
- An FD that has no rule FDs but that is contained in an FD with contained rule FDs is essentially May/Any but FCs will be indirectly restricted by the superior FD rules
  - It is expected that in any view the rule FDs will be at the lowest levels of decomposition

3.1.5.2 Rule precedence

Rules interact such that:

- Priority n rules override priority n+1 rules
- Within a priority the rules override as follows (lower number overrides higher number)
  1. Not (MUST_NOT_CONNECT)
  2. Must (MUST_CONNNECT)
  3. May (MAY_CONNECT)
  4. Null (NULL_FORWARDING_RULE (default))
     - Which overrides nothing regardless of the override priority
- Within a rule the flexibility rules override as follow:
  1. Any
  2. Same X
  3. Same X and Same Y rule will form an intersection such that both need to be met

3.2 Applying rules

This section explores, via some abstract examples, the application of specifications to instances of classes.

3.2.1 Basic application of specs to a class and instances

The figure below shows the basic use of a spec, via an example, where each instance references both the class, X (by definition), and a schema defined by one or more other classes P, Q or R. The instance gains the attributes of the Class X and the relevant spec classes (in the case of Instance X.1 the attributes are acquired from spec P (i.e. attribute P)). Class X indicates in its definition that instances of it will reference one or more specs. Each spec indicates the classes that it can apply to and in this case P, Q and R can all apply to Class X. Clearly there will be other classes in the model that P does not apply, there may be other classes that P does apply to and there may be other specs that do not apply to Class X.

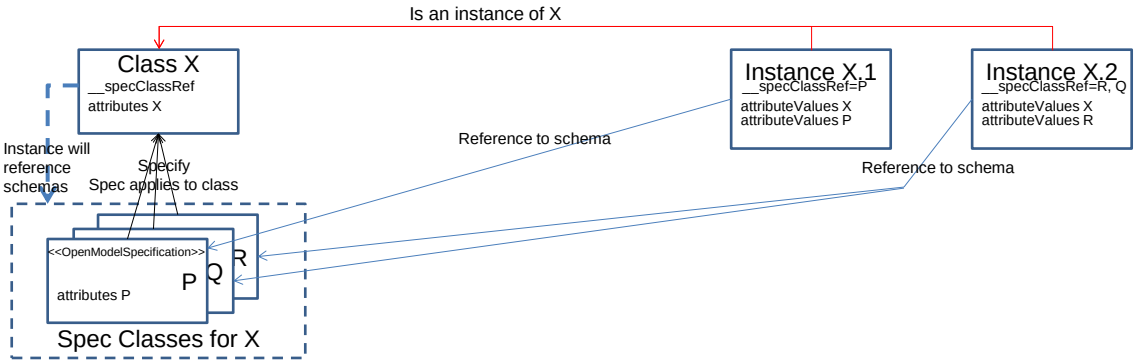

Figure 3-7 Basic spec application

3.2.2 Spec with outgoing pointer

The figure below builds on the figure in the previous section and adds a simple reference from a spec class to a class, Class Y (assumed for this example to be in the same model as class X). There is nothing special about the relationship from P to Class Y. The instance of X that reference spec class P acquire the attribute pointer to Class Y and hence can reference an instance of Y. In the example shown below the Instance X.1 reference Instance Y.6 via the attribute acquired from spec class P.

As Instance X.2 refers to a spec class R that does not refer to Class Y, Instance X.2 cannot reference an instance of Class Y. So Instance X.2 references Instance Y.9 is not allowed.

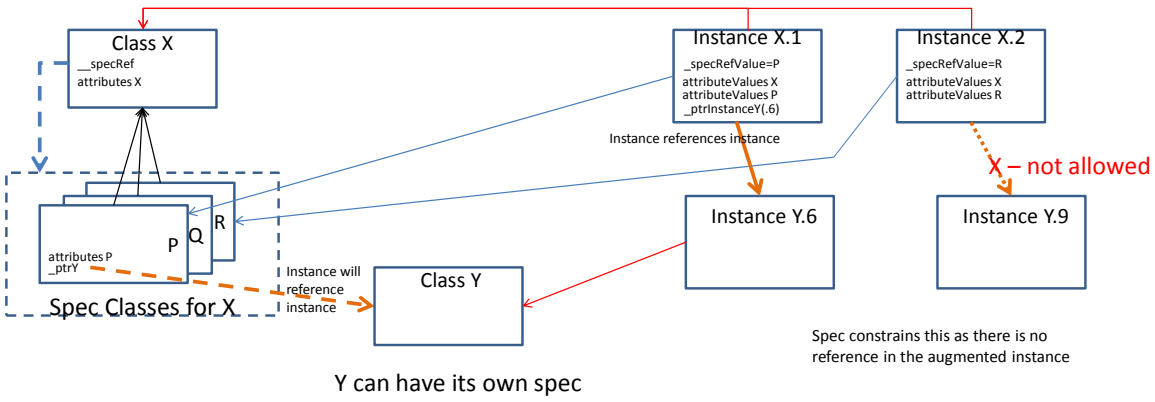

Figure 3-8 Outgoing pointer from spec

3.2.3 Spec with outgoing pointer from spec to spec

A spec can be composed of other specs. The figure below shows how a spec class J is a composed part of spec class P. In the example, Instance X.1 refers to spec class P and hence acquires the attributes of P but it also acquires the composed part Jx which is locally identified in Instance X.1 where Jx conforms to the spec class J. J may be used in the specs of many classes in the model.

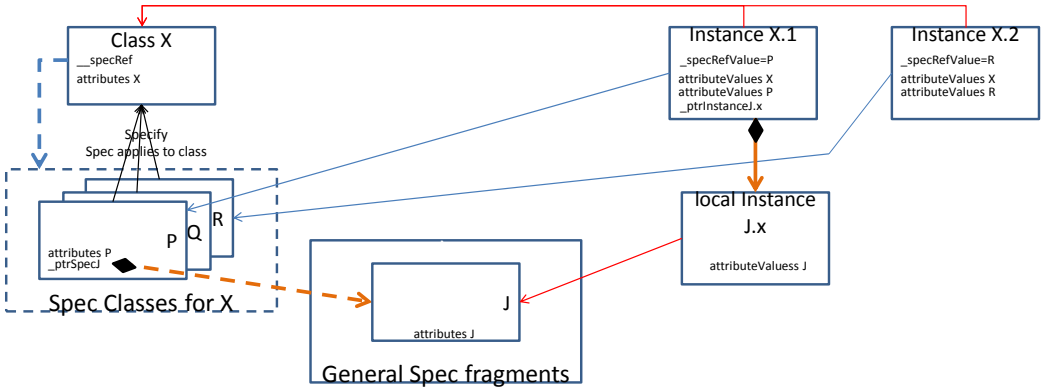

Figure 3-9 Outgoing pointer from spec to spec

3.2.4 Incoming pointer to spec

The previous cases were somewhat normal in form. The case described here is more complex. An instance of Class Y can have an association to an instance of Class X only where the instance of Class X references spec class P.

Class Y has a normal association to Class X but the association end attribute has a constraint that references spec class P. As Instance X.2 does not reference spec class P, Instance Y.6 is not allowed to reference Instance X.2 (note that in the diagram Instance Y.6 is shown twice).

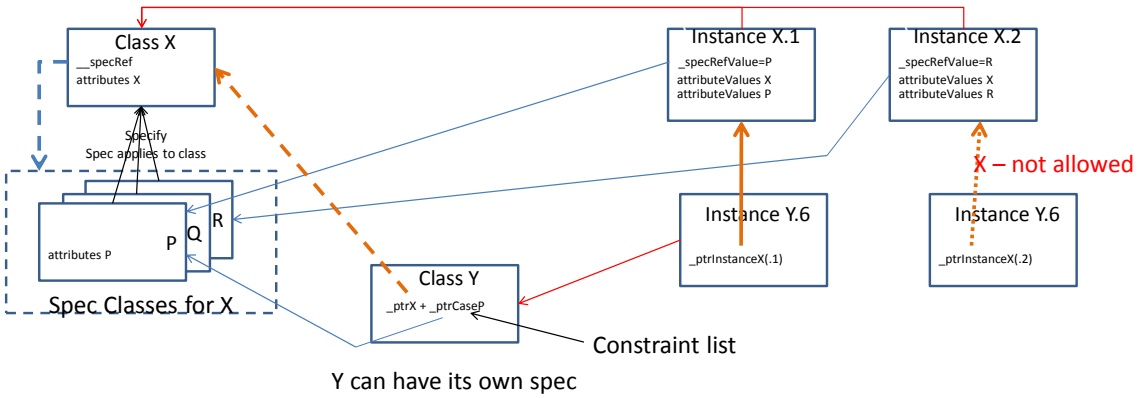

Figure 3-10 Incoming pointer to spec

In the following figure, Class Y simply has a constraint list entry of spec class P. The specific mechanism is that Class Y has an association to spec class P. In the instantiation of Class Y the association is transformed to an attribute of an unspecified class that takes that can reference any Instance.

The only instances that are allowed to be referenced are those that reference spec class P. In this case Instance X.1 references spec class P and so can be referenced by Instance Y.6 but Instance X.2 does not reference spec class P and so cannot be referenced by Instance Y.6 (note that in the diagram Instance Y.6 is shown twice).

In this form, unlike the previous form, any class that spec class P references can have instances that can be referenced by Instances of Class Y (i.e. not just instances of Class X).

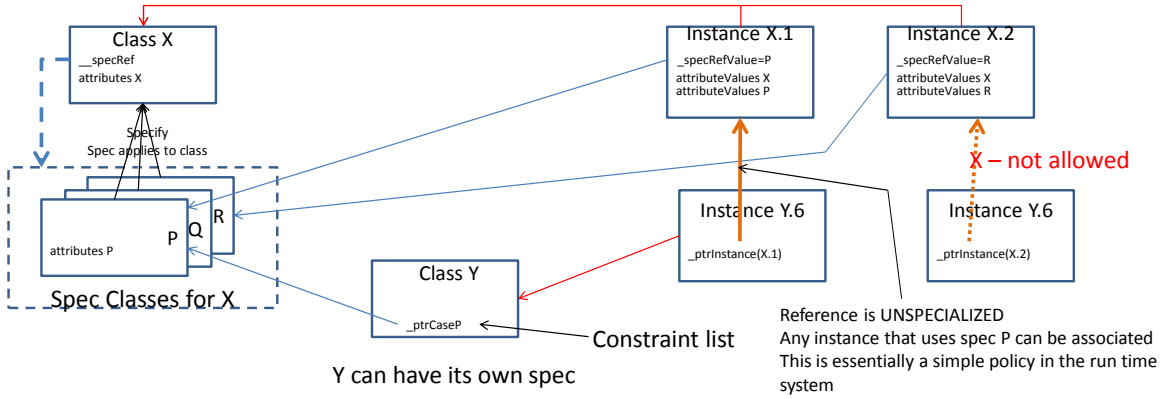

Figure 3-11 Incoming pointer to spec – sparse form

3.2.5 Outgoing pointer from spec to spec

In this case, it is necessary for some but not all cases of one class (in this example Class X) to be allowed to relate to some but not all cases of another class (in this example Class Y). To achieve this, a spec class, P, that specifies Class X references a spec class, F, that specifies Class Y.

In the instantiation, just as in the previous case, the association end attribute is of unspecified class. So, whilst the association end attribute in spec class P is specific to spec class F, the attribute in Instance X.4 is unspecialized, i.e. can reference an instance of any class.

There needs to be sufficient information in the spec to guide the tooling to form the correct ptr structure.

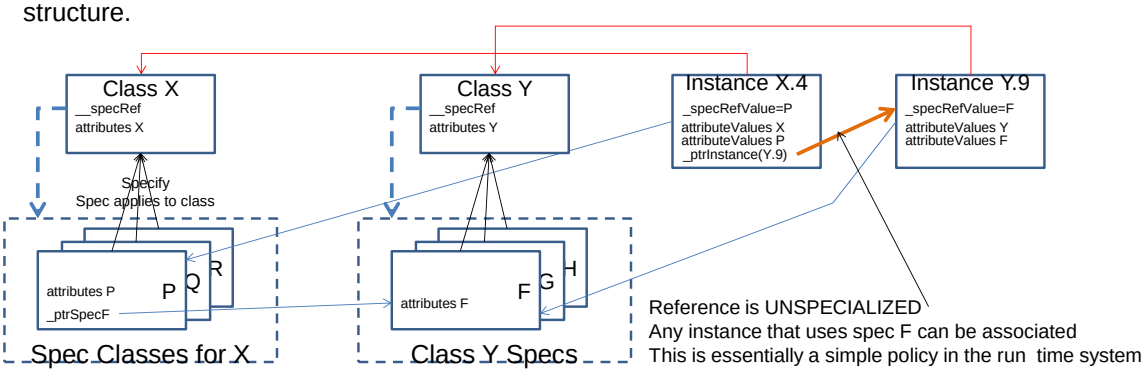

Figure 3-12 Outgoing pointer from spec to spec

3.2.6 Incoming pointer from spec extension to spec

In some cases, a spec class may be extended after it has been defined (i.e. independently of its construction) without changing the spec class. The extension is composition but the spec class to be extended does not carry the attribute reference. The extension provides a reference that is essentially reverse navigable.

In the example an Instance X.1 references spec class P and acquires those attributes, it also has a composition navigable from Instance X.1 to local Instance Jx, where Instance Jx reference spec class J. This is valid against the specs due to the reverse navigable composition.

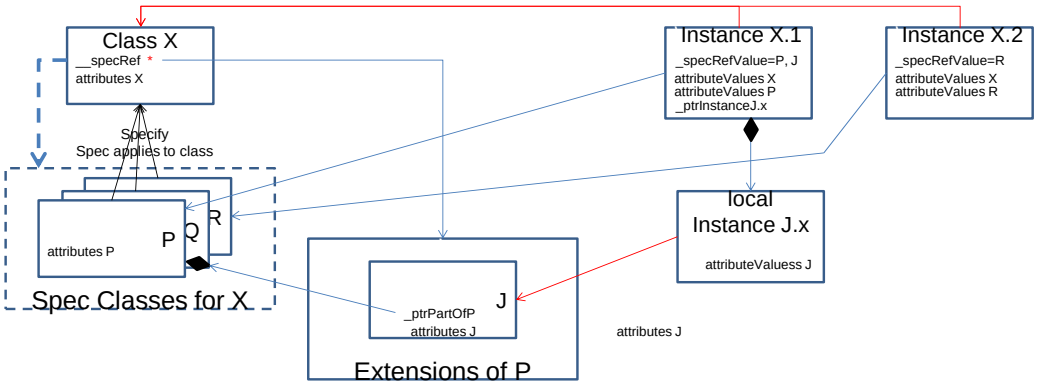

Figure 3-13 Incoming pointer from spec extension to spec

The figure below provides a sketch of the realization of an instance of X. The "Instantiation Policy" guides the "Instantiation Engine" to extend instances of Class X that are specified by spec class P with spec class J. If, as is probably normally the case, all instances of Class X that are specified by spec class P should also be extend by spec class J, then the policy can be automatically constructed by exploring the spec repository for all spec that have appropriate

references to spec class P and ensuring that the policy includes the extension. In this case it is assumed that only spec class J exists as an extension of spec class P.

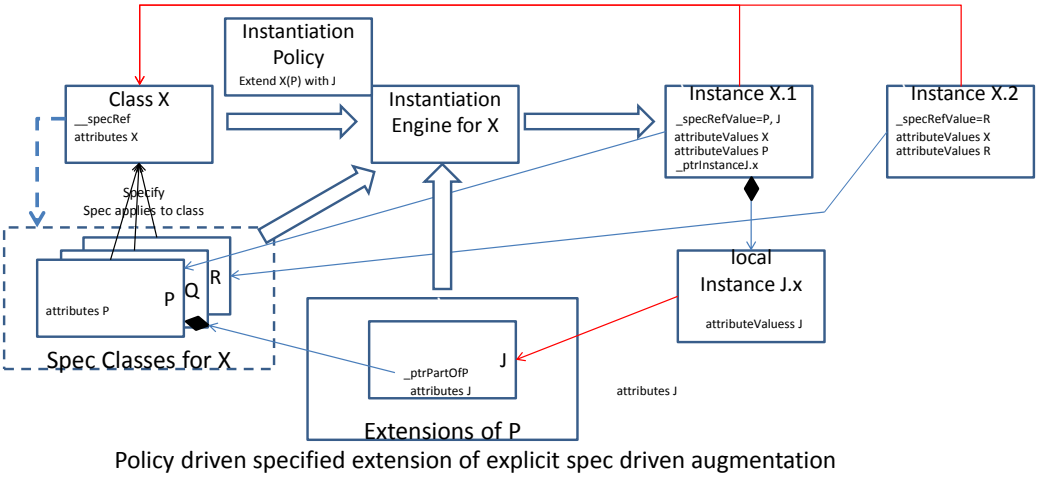

Figure 3-14 Incoming pointer from spec extension to spec with policy

3.3 LTP spec examples

Examples of LTP/LP spec will be provided in a later release. Work is proceeding on [ONF TAPI] on LTP/LP Specs.

End of Document