UML-like Diagrams Description

The following description is intended to be published on the YANGsters page in the https://1.ieee802.org/yangsters/yangsters-guidelines/yangsters-faq/ and/or as an appendix in any standard that contains UML-like diagrams.

The IEEE UML-like diagrams provide an easy to understand visualization of a model even for those that do not understand the formal UML modeling syntax.

A box is used to represent a class. A box can have multiple compartments. In most cases there will be a single-compartment box that represents a class or a two-compartment box that has a class name in the top box and the attributes for that class in the bottom box.

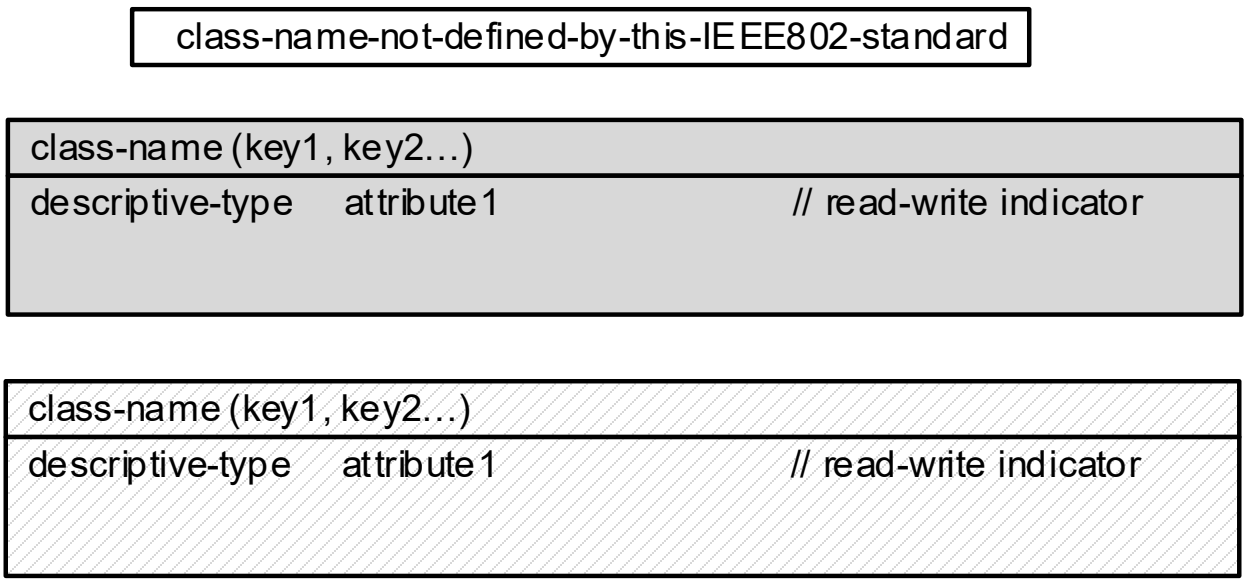

Figure 1 - example class/attribute boxes

In Figure 1, the white box with “class-name-not-defined-by-this-IEEE802-standard” is used to represent a class that is imported from another standard that is from outside the IEEE 802 standards committee. The shaded example in Figure 1 shows a grey box that has a class (with keys listed if applicable) and a compartment for attributes, that represents something defined by IEEE 802. The bottom example in Figure 1 which is filled with a grey-hatched pattern is used to represent something that is being added by the augment or modification being introduced in the standard. Figure 2 provides a small example. In Figure 2 This shows port information that is being added by an IEEE standard.

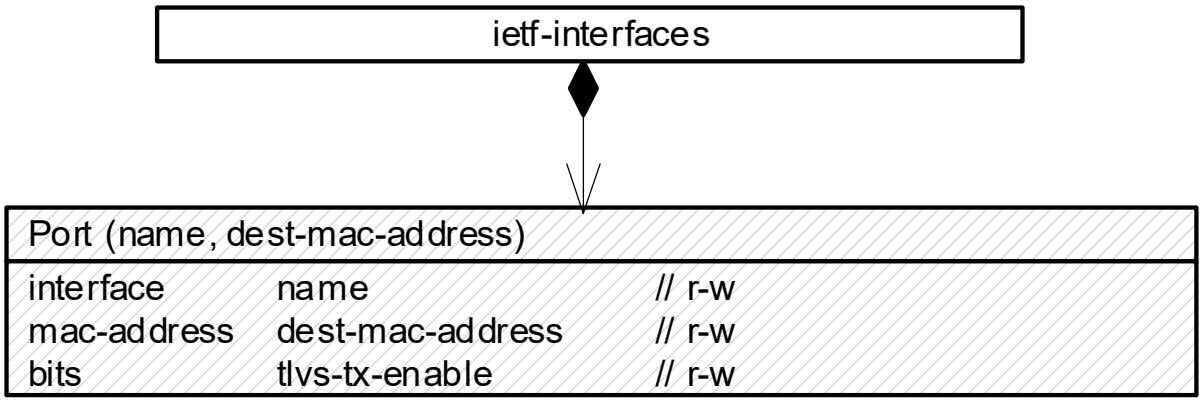

Figure 2 - class/attribute example

Another example shows the situation where the base functionality defined by an IEEE standard is being enhanced by another IEEE standard. The white statistics box is defined outside the IEEE, the grey box is defined as part of the dot1q-bridge YANG module. The striped/shaded box augments bridge-port-statistics with tpmr statistics.

statistics		
date-time	discontinuity-time;	// r
counter64	in-octets, in-unicast-pkts, in-broadcast-pkts, in-multicast-pkts;	// r
counter64	in-discards, in-errors, in-unknown-protos, ;	// r
counter64	out-octets, out-unicast-pkts, out-broadcast-pkts, out-multicast-pkts;	// r
counter64	out-discards, out-errors;	// r
bridge-port-statistics		
counter64	delay-exceeded-discards, mtu-exceeded-discards;	// (12.6.1.1.3) r
counter64	frame-rx, octets-rx, frame-tx, octets-tx;	// (12.6.1.1.3) r
counter64	discard-inbound, forward-outbound, discard-lack-of-buffers;	// (12.6.1.1.3) r
counter64	discard-transit-delay-exceeded, discard-on-error;	// (12.6.1.1.3) r
counter64	discard-on-ingress-filtering;	// (12.6.1.1.3) r
counter32	acks-tx, add-notification-tx, loss-notification-tx;	// (12.19.4.1.3.3) r
counter32	loss-confirmation-tx, acks-rx, add-notifications-rx	// (12.19.4.1.3.3) r
counter32	loss-notification-rx, loss-confirmation-rx, add-events;	// (12.19.4.1.3.3) r
counter32	loss-events, mac-status-notifications;	// (12.19.4.1.3.3) r

Figure 3 - counters augmenting statistics

The arrows/diamonds/lines provide for composition, aggregation, or association relationships.

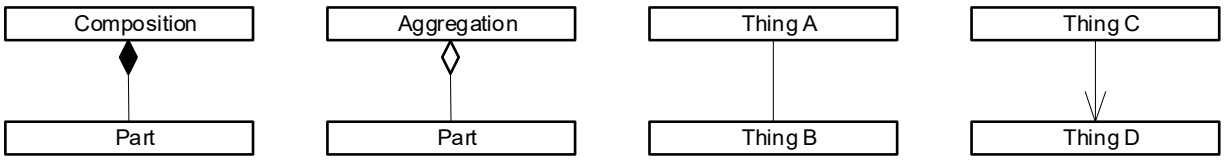

Figure 4 – relationships

In Figure 3, the “Composition” example shows a “has a” relationship. Composition is indicated by a filled diamond shape which is connected to the “parent” of the relationship. The Part is the “child” of the

relationship. When the class (parent) that contains the part (child) is deleted, the part is deleted as well. For example, if I delete a folder from my computer, all the files are deleted too. The “Aggregation” example is a “has a” relationship as well, but in this case the parts can exist independently of the class that was aggregating the parts. For example, if a school course is deleted, the students still exist.

The other two diagrams in Figure 3 show associations. Thing A is associated with Thing B, meaning there is (some kind of) structural relationship, but no further detail is given. Thing C is associated with Thing D and it is possible to navigate to Thing D through Thing C. The arrows can be on both ends of the association if bi-directional navigation is desired.

Relationship to YANG

The UML-like diagram provides a great deal of latitude to provide the most concise information in a graphical way. There is no requirement to match the data types in YANG in the UML-like diagram. There is no formal mapping between the structure of the YANG and the structure of the UML-like diagram.