

802.3 Ethernet Interface YANG Task Force (802.3cf) and RMON MIB (RFC 3635) (Also Power over Ethernet YANG)

Rob Wilton

Cisco

2017 Jan 30, IETF/IEEE liaison meeting
Updated 27 Feb 2017

Ethernet 802.3 YANG Task Force

- IEEE 802.3cf is defining Ethernet YANG models:
E.g. basic Ethernet interface, Power over Ethernet, PON, Physical Link OAM (Ethernet in the first mile)
- A lot of content was previously defined in various MIBs (e.g. Etherlike MIB - RFC 2665)
- These MIBs were originally defined in IETF, but subsequently transitioned to IEEE 802.3 (via RFC 7448). Now defined in IEEE 802.3.1.

Ethernet 802.3 Clause 30 Background

- Clause 30 of IEEE 802.3 is an internal management API for all 802.3 standards compliant Ethernet devices.
- Hardware implementations of 802.3 must implement all of the appropriate parts of clause 30 (some parts are optional).
- Standard Ethernet management models (e.g. SNMP, YANG) are written to clause 30.
 - IEEE 802.3 standards models **must** only relate to fields defined in clause 30.
 - They can be combined, and not all fields must be used.

Ethernet 802.3 YANG – RMON MIB

- Some Ethernet related statistics are defined in the RMON MIB (RFC 2819)
 - Of particular interest is the Ethernet-Statistics group
- This MIB doesn't only define Ethernet related data, and ownership was not transitioned to IEEE 802.3
- 802.3cf would like to define YANG for some of the Ethernet counters previously defined in the RMON MIB

Ethernet 802.3 YANG – RMON MIB

- Don't plan on defining 802.3 YANG for all of RMON MIB Ethernet counters:
 - Only those that are supported by underlying Ethernet 802.3 clause 30 definitions (which may be extended)
 - Only those that are still relevant on modern hardware
 - 802.3cf proposes that some Ethernet related fields are defined in IETF RFCs (i.e. those that are not, or cannot, be tied back to IEEE 802.3 clause 30)
 - The proposal is to add them to the **interfaces-ethernet-like module**, this is being defined in **draft-ietf-netmod-intf-ext-yang** (I'm an author of this draft)

Ethernet 802.3 YANG – Questions

- Is anyone aware of any plans to convert the RMON MIB to YANG?
 - If so which WG, NETMOD?
 - If this work was ever done, then the Ethernet related counters could just be left out.
- Does anyone (particularly from IETF) have any comments or concerns on this approach?
 - I intend to run this approach via the NETMOD WG as well.

Ethernet 802.3 YANG

Power over Ethernet (PoE)

- Similar issue, but in the reverse direction!
- 802.3.1 currently owns the PoE MIB (originally RFC 3621), but doesn't have the underlying clause 30 definitions to support all of it.
 - The PSE/PD port information (pethPsePortTable) should be defined in 802.3cf
 - The power supply information (pethMainPseObjects, pethNotificationControlTable) should not be defined in 802.3cf
- It looks like some of the information being reported is quite closely aligned to the Entity MIB, and the associated Entity YANG model currently being developed in NETMOD.
- Propose talking with the authors of the Entity YANG draft and NETMOD to see if appropriate parts of the PoE YANG model could be aligned with it (and possibly also be standardized in NETMOD).

Thank you!

Backup Slides

IETF interface YANG statistics

(For reference. Every Ethernet interface always has these)

`+-ro statistics`

`+-ro discontinuity-time` `yang:date-and-time`

<code>+-ro in-octets?</code>	<code>yang:counter64</code> = (total good bytes, inc fcs chars)
<code>+-ro in-unicast-pkts?</code>	<code>yang:counter64</code> = good uni pkts (not drop/error/
<code>+-ro in-broadcast-pkts?</code>	<code>yang:counter64</code> = good bcast pkts unknown)
<code>+-ro in-multicast-pkts?</code>	<code>yang:counter64</code> = good mcast pkts “
<code>+-ro in-discards?</code>	<code>yang:counter32</code> = e.g. QoS/ACL drops
<code>+-ro in-errors?</code>	<code>yang:counter32</code> = e.g. Frame errors
<code>+-ro in-unknown-protos?</code>	<code>yang:counter32</code> = e.g. Unknown proto drops.

<code>+-ro out-octets?</code>	<code>yang:counter64</code>
<code>+-ro out-unicast-pkts?</code>	<code>yang:counter64</code>
<code>+-ro out-broadcast-pkts?</code>	<code>yang:counter64</code>
<code>+-ro out-multicast-pkts?</code>	<code>yang:counter64</code>
<code>+-ro out-discards?</code>	<code>yang:counter32</code>
<code>+-ro out-errors?</code>	<code>yang:counter32</code>

Existing RMON MIB Ethernet counters

(For reference purposes only, defined in RFC 2819)

etherStatsDropEvents	Counter32, // Drop due to lack of resources
etherStatsOctets	Counter32, // Total bytes (good + bad)
etherStatsPkts	Counter32, // Total pkts (good + bad)
etherStatsBroadcastPkts	Counter32, // Total good bcast pkts
etherStatsMulticastPkts	Counter32, // Total good mcast pkts
etherStatsCRCAlignErrors	Counter32, // 64 <= pkt <= 1518, bad CRC/align
etherStatsUndersizePkts	Counter32, // pkt < 64, good CRC
etherStatsOversizePkts	Counter32, // pkt > 1518, good CRC
etherStatsFragments	Counter32, // pkt < 64, bad CRC
etherStatsJabbers	Counter32, // pkt > 1518, bad CRC
etherStatsCollisions	Counter32, // Collision estimate
etherStatsPkts64octets	Counter32, // 64 byte pkts
etherStatsPkts65to127octets	Counter32, // 65 - 127 byte pkts
etherStatsPkts128to255octets	Counter32, // 128 - 255 byte pkts
etherStatsPkts256to511octets	Counter32, // 256 - 511 byte pkts
etherStatsPkts512to1023octets	Counter32, // 512 - 1023 byte pkts
etherStatsPkts1024to1518octets	Counter32, // 1024 - 1518 byte pkts

802.3 Ethernet Frame/Phy Counters

(Combined Etherlike MIB and RMON MIB)

This counters are in addition to the ietf-interfaces statistics.

interfaces-state/interface/ethernet/frame-statistics:

```
in-total-octets      counter64, // Total received bytes (good + bad)
in-total-pkts        counter64, // Total received pkts (good + bad)
in-pkts-errors-fcs   counter64, // 64 <= pkt <= 1518, bad CRC or alignment
in-pkts-errors-runt  counter64, // pkt < 64
in-pkts-errors-giant counter64, // pkt > MRU
out-total-octets      counter64, // Total transmitted bytes (good + bad)
out-total-pkts        counter64, // Total transmitted pkts (good + bad)
// May still be some generic input/output errors missing.
```

interfaces-state/interface/ethernet/phy-statistics:

```
in-errors-symbol      counter64, // symbol errors
lpi { <- TODO, make LPI a feature.
  in-lpi-transitions   counter64, // lpi transitions
  in-lpi-time           decimal64, // lpi time (seconds, 6 d.p.)
  out-lpi-transitions   counter64, // lpi transitions
  out-lpi-time           decimal64, // lpi time (seconds, 6 d.p.)
}
```