IEEE P802.1Qdd **R**esource **A**llocation **P**rotocol (**RAP**)

Reworking RAP Propagator

Feng Chen (Siemens AG)

Johannes Specht (Self, Siemens AG)

IEEE 802.1 TSN Weekly Meeting

Feb. 21, 2022

Contents

Introduction - From the Document

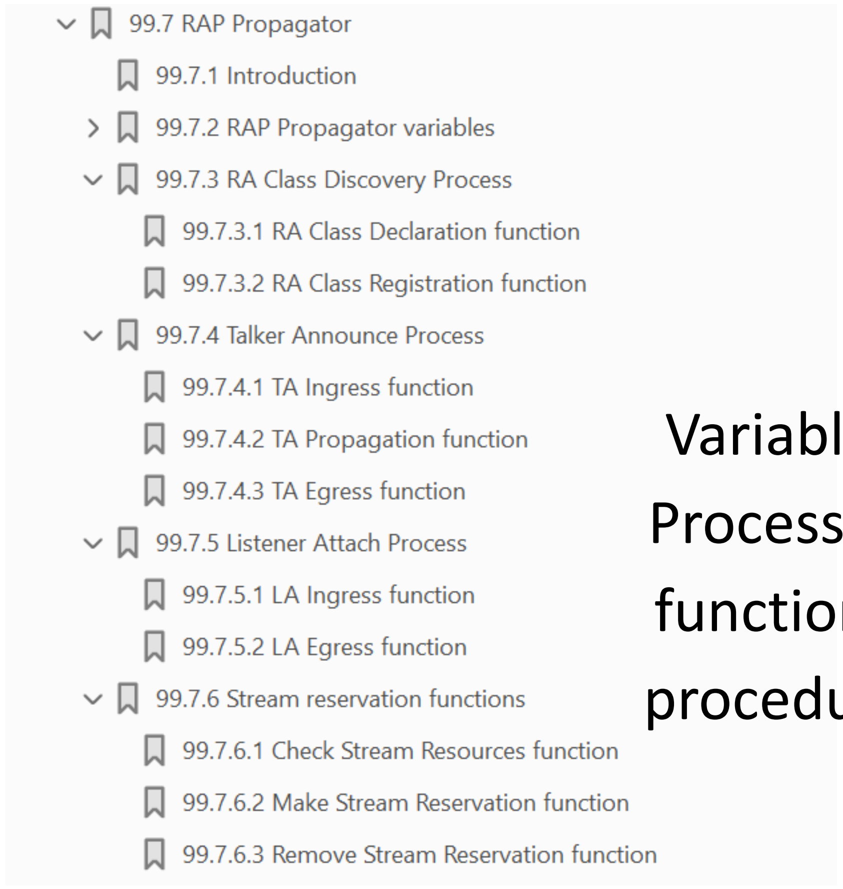
This document contains a re-worked version of the RAP Propagator defined in subclause 99.7 of IEEE P802.1Qdd draft D0.5. Based on review feedback, several aspects were identified where clarity and readability of the technical contents defined so far in D0.5 can be enhanced.

The enhancements in the re-worked version include the following:

- New subclause structure, simplifying finding particular contents
- Contents ordered for top-down reading
- Removal of several synonyms/ambiguously used terms (“functions” vs. “procedures”, etc.)
- More formal description of protocol mechanisms
- Removal of unnecessary concurrency
- Various enhancements in non-formal descriptions (including shortening)

The reworked contents are intended for incorporation into the next Qdd draft D0.6.

Restructured Subclause



- 99.7 RAP Propagator
  - 99.7.1 Introduction
  - 99.7.2 RAP Propagator variables
  - 99.7.3 RA Class Discovery Process
    - 99.7.3.1 RA Class Declaration function
    - 99.7.3.2 RA Class Registration function
  - 99.7.4 Talker Announce Process
    - 99.7.4.1 TA Ingress function
    - 99.7.4.2 TA Propagation function
    - 99.7.4.3 TA Egress function
  - 99.7.5 Listener Attach Process
    - 99.7.5.1 LA Ingress function
    - 99.7.5.2 LA Egress function
  - 99.7.6 Stream reservation functions
    - 99.7.6.1 Check Stream Resources function
    - 99.7.6.2 Make Stream Reservation function
    - 99.7.6.3 Remove Stream Reservation function

Variables

Processes,

functions,

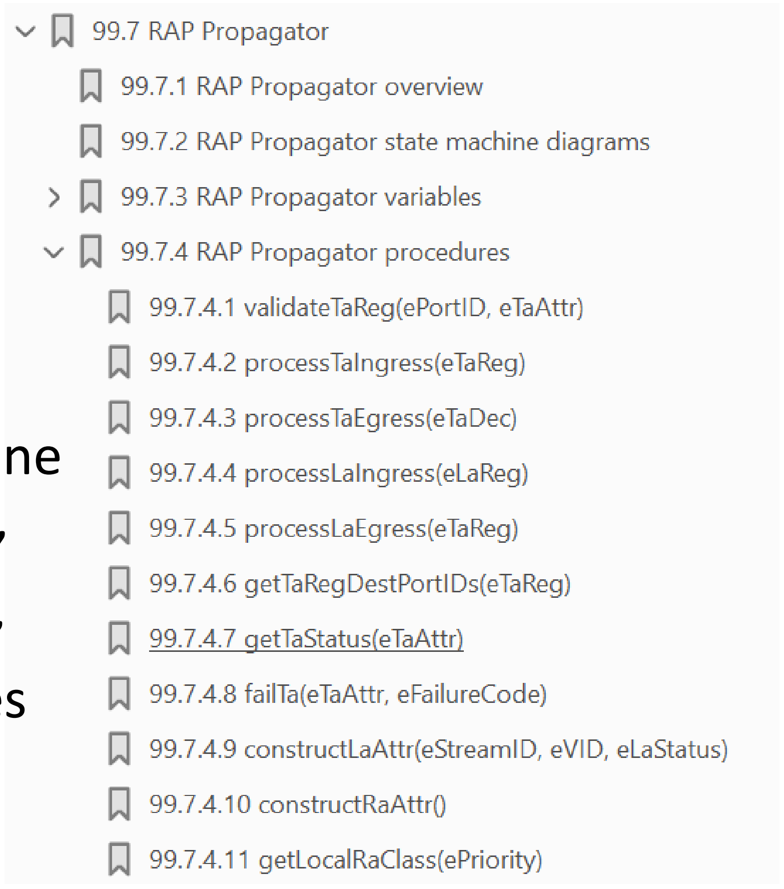
procedures

restructured

State machine diagrams,

variables,

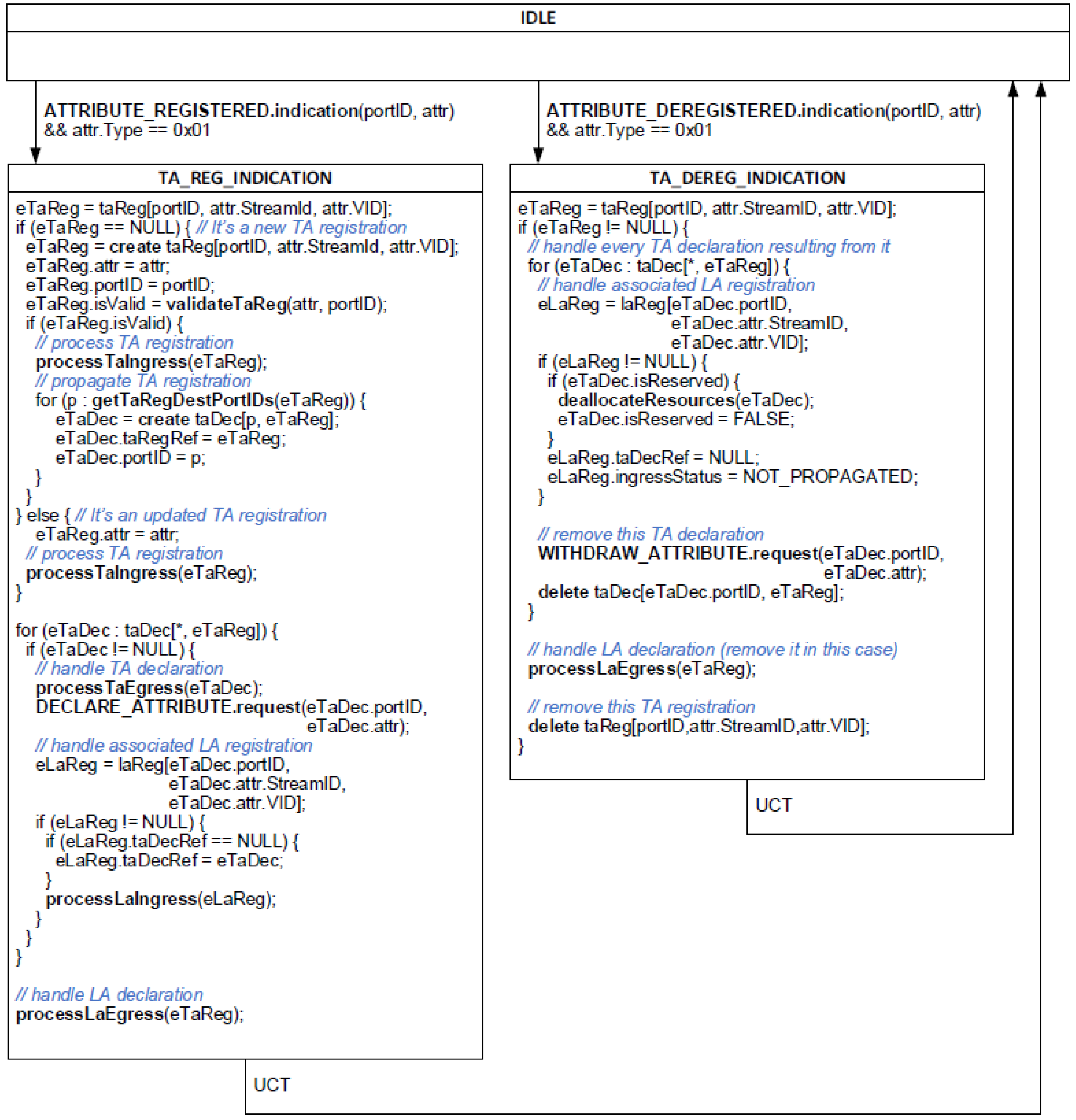
procedures



- 99.7 RAP Propagator
  - 99.7.1 RAP Propagator overview
  - 99.7.2 RAP Propagator state machine diagrams
  - 99.7.3 RAP Propagator variables
  - 99.7.4 RAP Propagator procedures
    - 99.7.4.1 validateTaReg(ePortID, eTaAttr)
    - 99.7.4.2 processTaIngress(eTaReg)
    - 99.7.4.3 processTaEgress(eTaDec)
    - 99.7.4.4 processLaIngress(eLaReg)
    - 99.7.4.5 processLaEgress(eTaReg)
    - 99.7.4.6 getTaRegDestPortIDs(eTaReg)
    - 99.7.4.7 getTaStatus(eTaAttr)
    - 99.7.4.8 failTa(eTaAttr, eFailureCode)
    - 99.7.4.9 constructLaAttr(eStreamID, eVID, eLaStatus)
    - 99.7.4.10 constructRaAttr()
    - 99.7.4.11 getLocalRaClass(ePriority)

State Machine Diagrams

- A single state machine defining the operation of RAP Propagator
  - Split into multiple diagrams
  - Based on Annex E conventions
- A "IDLE" state waiting for events
  - Transitions from this state triggered by primitives (i.e., interfaces to RAP Participants, etc.)
  - Transition to different event handling states
- Concurrent processes from D0.5 gone:
  - Not needed
  - Reduces complexity
  - Avoids potential concurrency issues



IDLE
ATTRIBUTE_REGISTERED.indication(portID, attr) && attr.Type == 0x01
ATTRIBUTE_DEREGISTERED.indication(portID, attr) && attr.Type == 0x01
TA_REG_INDICATION
TA_DEREG_INDICATION
UCT
UCT

TA_REG_INDICATION

```
eTaReg = taReg[portID, attr.StreamId, attr.VID];
if (eTaReg == NULL) { // It's a new TA registration
  eTaReg = create taReg[portID, attr.StreamId, attr.VID];
  eTaReg.attr = attr;
  eTaReg.portID = portID;
  eTaReg.isValid = validateTaReg(attr, portID);
  if (eTaReg.isValid) {
    // process TA registration
    processTaIngress(eTaReg);
    // propagate TA registration
    for (p : getTaRegDestPortIDs(eTaReg)) {
        eTaDec = create taDec[p, eTaReg];
        eTaDec.taRegRef = eTaReg;
        eTaDec.portID = p;
    }
  }
} else { // It's an updated TA registration
    eTaReg.attr = attr;
  // process TA registration
  processTaIngress(eTaReg);
}

for (eTaDec : taDec[*, eTaReg]) {
  if (eTaDec != NULL) {
    // handle TA declaration
    processTaEgress(eTaDec);
    DECLARE_ATTRIBUTE.request(eTaDec.portID,
                              eTaDec.attr);
    // handle associated LA registration
    eLaReg = laReg[eTaDec.portID,
                   eTaDec.attr.StreamID,
                   eTaDec.attr.VID];
    if (eLaReg != NULL) {
      if (eLaReg.taDecRef == NULL) {
        eLaReg.taDecRef = eTaDec;
      }
      processLaIngress(eLaReg);
    }
  }
}

// handle LA declaration
processLaEgress(eTaReg);
```

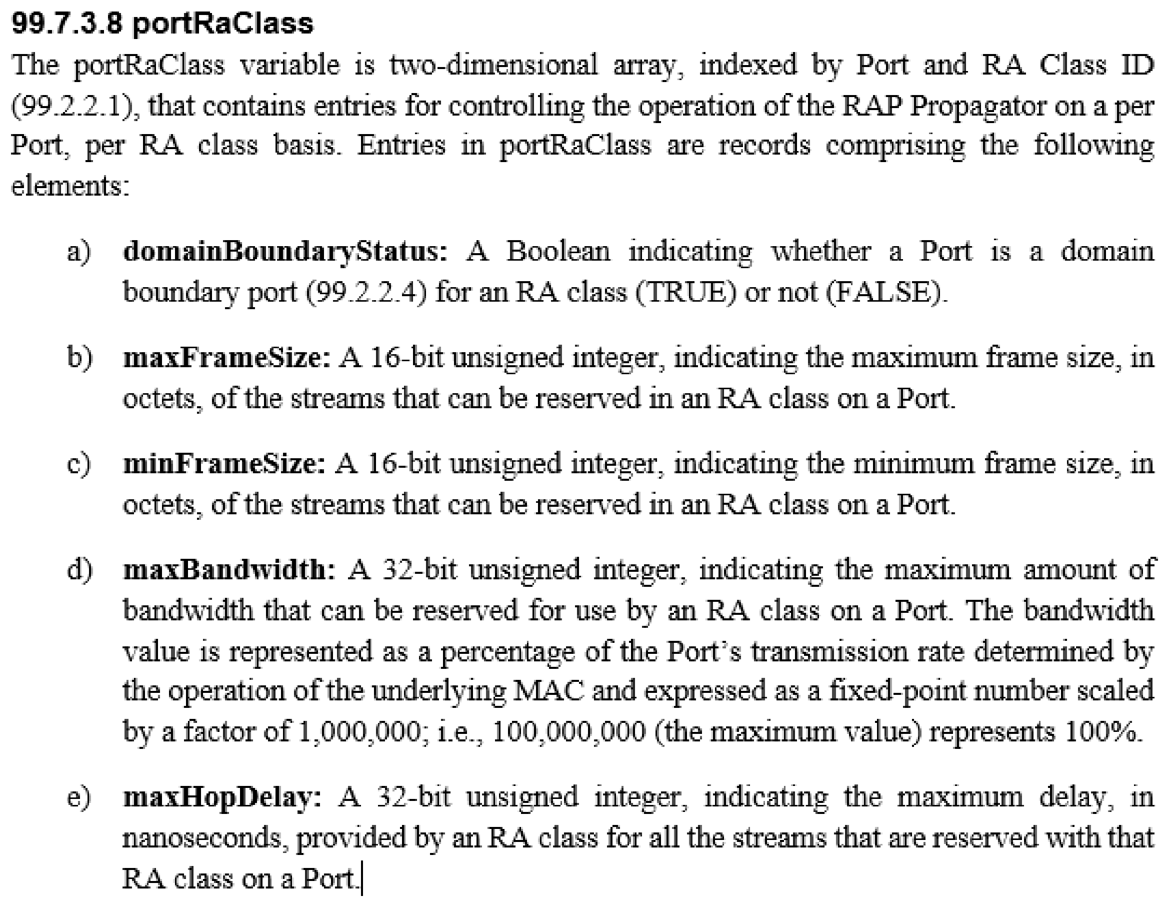
TA_DEREG_INDICATION

```
eTaReg = taReg[portID, attr.StreamID, attr.VID];
if (eTaReg != NULL) {
  // handle every TA declaration resulting from it
  for (eTaDec : taDec[*, eTaReg]) {
    // handle associated LA registration
    eLaReg = laReg[eTaDec.portID,
                   eTaDec.attr.StreamID,
                   eTaDec.attr.VID];
    if (eLaReg != NULL) {
      if (eTaDec.isReserved) {
        deallocateResources(eTaDec);
        eTaDec.isReserved = FALSE;
      }
      eLaReg.taDecRef = NULL;
      eLaReg.ingressStatus = NOT_PROPAGATED;
    }

    // remove this TA declaration
    WITHDRAW_ATTRIBUTE.request(eTaDec.portID,
                               eTaDec.attr);
    delete taDec[eTaDec.portID, eTaReg];
  }

  // handle LA declaration (remove it in this case)
  processLaEgress(eTaReg);

  // remove this TA registration
  delete taReg[portID,attr.StreamID,attr.VID];
}
```

Figure 99-25— Processing of Talker Announce registration and deregistration

Variables

- Organize variables with various scopes into arrays
  - Replacement for lists in D0.5
  - e.g., an array portRaClass containing all per-Port per RA class variables.
- Notational conventions for referencing array variables in pseudo-code
  - e.g. ***portRaClass [<portID>, <raClassID>].maxFrameSize*** refers to the maxFrameSize value of an entry in the array indexed by the given 2-tuple.
- Lifecycle management simplified, compared to the lifecycle management of lists in D0.5



99.7.3.8 portRaClass

The portRaClass variable is two-dimensional array, indexed by Port and RA Class ID (99.2.2.1), that contains entries for controlling the operation of the RAP Propagator on a per Port, per RA class basis. Entries in portRaClass are records comprising the following elements:

a) **domainBoundaryStatus:** A Boolean indicating whether a Port is a domain boundary port (99.2.2.4) for an RA class (TRUE) or not (FALSE).

b) **maxFrameSize:** A 16-bit unsigned integer, indicating the maximum frame size, in octets, of the streams that can be reserved in an RA class on a Port.

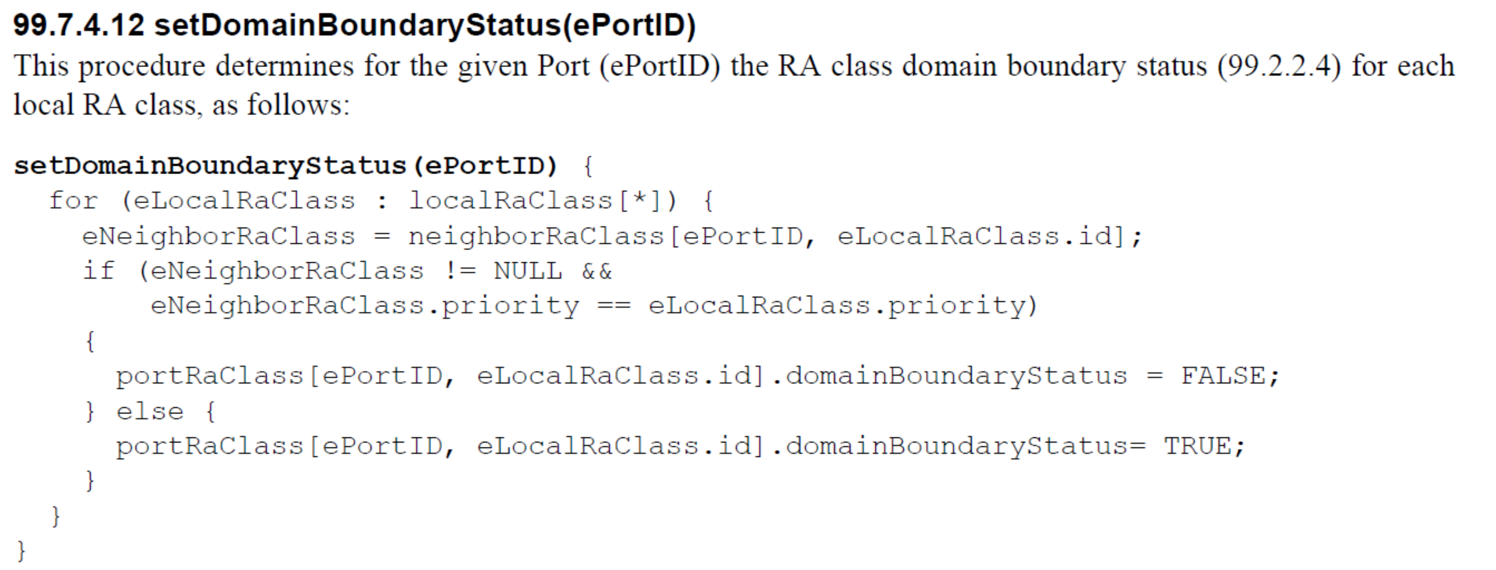
c) **minFrameSize:** A 16-bit unsigned integer, indicating the minimum frame size, in octets, of the streams that can be reserved in an RA class on a Port.

d) **maxBandwidth:** A 32-bit unsigned integer, indicating the maximum amount of bandwidth that can be reserved for use by an RA class on a Port. The bandwidth value is represented as a percentage of the Port's transmission rate determined by the operation of the underlying MAC and expressed as a fixed-point number scaled by a factor of 1,000,000; i.e., 100,000,000 (the maximum value) represents 100%.

e) **maxHopDelay:** A 32-bit unsigned integer, indicating the maximum delay, in nanoseconds, provided by an RA class for all the streams that are reserved with that RA class on a Port.

Procedures

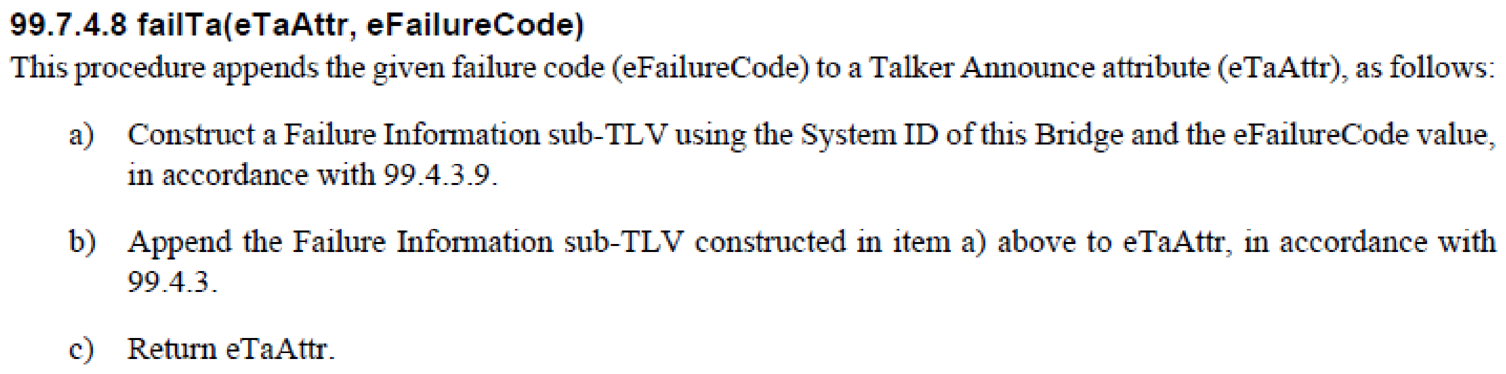
- Two co-existing styles
  - Pseudo-Code
    - C/C++ like
    - Weaker typed
  - Stds language



99.7.4.12 setDomainBoundaryStatus(ePortID)

This procedure determines for the given Port (ePortID) the RA class domain boundary status (99.2.2.4) for each local RA class, as follows:

```
setDomainBoundaryStatus(ePortID) {
  for (eLocalRaClass : localRaClass[*]) {
    eNeighborRaClass = neighborRaClass[ePortID, eLocalRaClass.id];
    if (eNeighborRaClass != NULL &&
        eNeighborRaClass.priority == eLocalRaClass.priority)
    {
      portRaClass[ePortID, eLocalRaClass.id].domainBoundaryStatus = FALSE;
    } else {
      portRaClass[ePortID, eLocalRaClass.id].domainBoundaryStatus= TRUE;
    }
  }
}
```



99.7.4.8 failTa(eTaAttr, eFailureCode)

This procedure appends the given failure code (eFailureCode) to a Talker Announce attribute (eTaAttr), as follows:

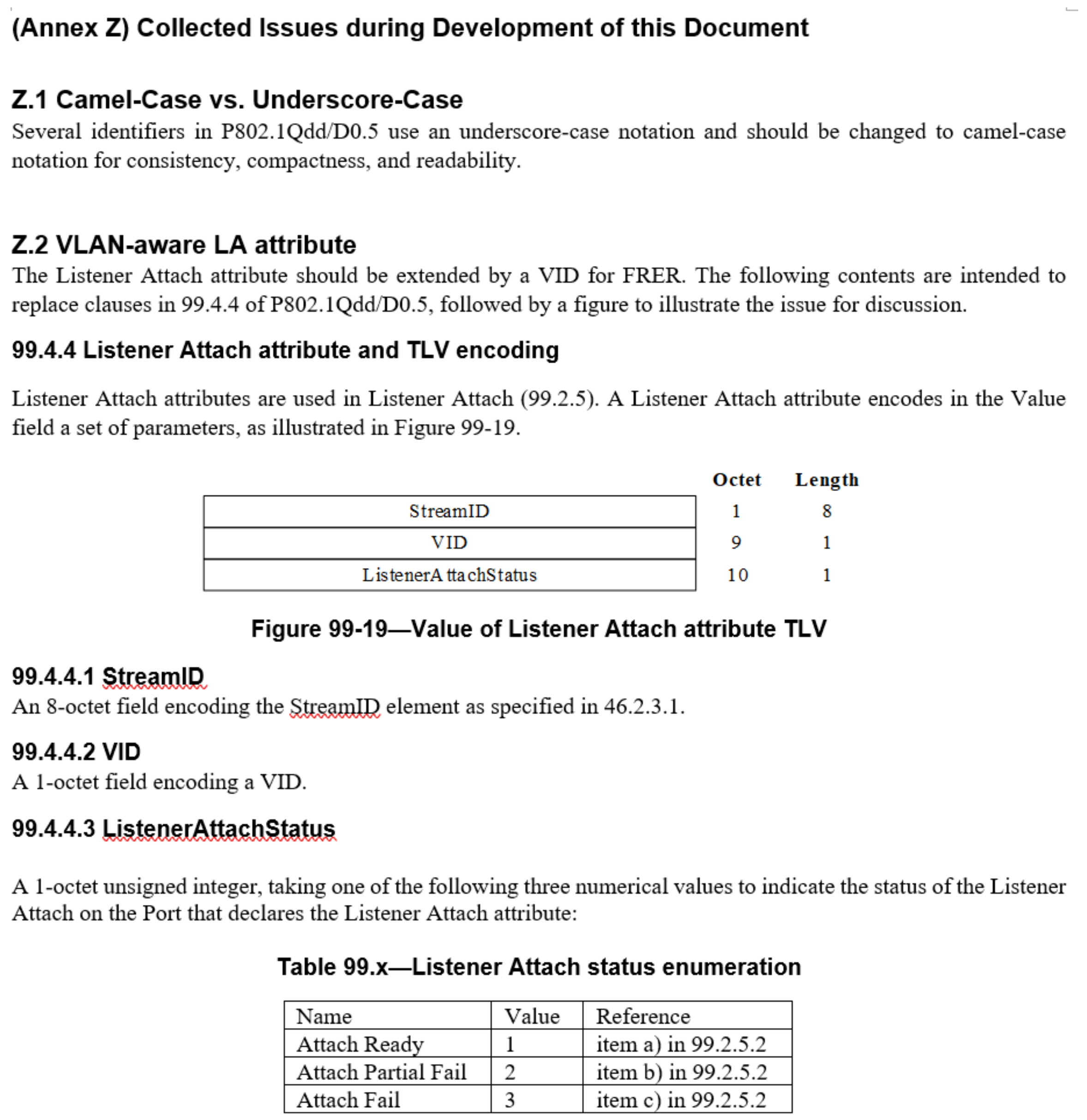
a) Construct a Failure Information sub-TLV using the System ID of this Bridge and the eFailureCode value, in accordance with 99.4.3.9.

b) Append the Failure Information sub-TLV constructed in item a) above to eTaAttr, in accordance with 99.4.3.

c) Return eTaAttr.

Annex Z

- Open technical issues discovered during (and as a result of) reworking
- To be incorporated into Annex Z of the next Qdd draft
- Intended for later discussion



(Annex Z) Collected Issues during Development of this Document

Z.1 Camel-Case vs. Underscore-Case

Several identifiers in P802.1Qdd/D0.5 use an underscore-case notation and should be changed to camel-case notation for consistency, compactness, and readability.

Z.2 VLAN-aware LA attribute

The Listener Attach attribute should be extended by a VID for FRER. The following contents are intended to replace clauses in 99.4.4 of P802.1Qdd/D0.5, followed by a figure to illustrate the issue for discussion.

99.4.4 Listener Attach attribute and TLV encoding

Listener Attach attributes are used in Listener Attach (99.2.5). A Listener Attach attribute encodes in the Value field a set of parameters, as illustrated in Figure 99-19.

	Octet	Length
StreamID	1	8
VID	9	1
ListenerAttachStatus	10	1

Figure 99-19—Value of Listener Attach attribute TLV

99.4.4.1 StreamID

An 8-octet field encoding the StreamID element as specified in 46.2.3.1.

99.4.4.2 VID

A 1-octet field encoding a VID.

99.4.4.3 ListenerAttachStatus

A 1-octet unsigned integer, taking one of the following three numerical values to indicate the status of the Listener Attach on the Port that declares the Listener Attach attribute:

Table 99.x—Listener Attach status enumeration

Name	Value	Reference
Attach Ready	1	item a) in 99.2.5.2
Attach Partial Fail	2	item b) in 99.2.5.2
Attach Fail	3	item c) in 99.2.5.2

Next Steps

- Document uploaded for offline reading & comparison with D0.5
  - Comments on the document are welcome!
  - However, keep in mind that several types of comments will be more appropriate after incorporation into D0.6 according to the balloting process.
- Timeslot during the March 2022 Electronic Plenary

Thank You!

Questions, Comments,
Opinions, Ideas?