thanks to Antonio de la Oliva and Lily Lv for review and constructive comments
Summary

• P802.1CQ/D0.5 was reviewed in TG Ballot.
• Comment resolution was completed in November.
  - Address Blocks were introduced for address claiming
• In March, Editor presented “Block Address Registration and Claiming (BARC)”
  - cq-marks-BARC-0321-v00.pdf
  - address blocks used for registrar-managed addresses as well
  - Address Registration and Claiming (ARC)
    - address blocks, and also claiming address ranges using MAAP
      - presented also to IEEE 1722 Working Group
• Main issues raised in March concerned VLAN operation
• This contribution previews P802.1CQ/D0.6
  - refinements and details since March presentation
  - discussion on improved VLAN support
  - v01 presented to TSN at May 802.1 Interim; this version (v03) adds detail
BARC assigns MAC Addresses in Address Blocks

1) Address Blocks (ABs) are sets of local addresses.

2) An AB includes equal-sized unicast and multicast address subblocks.

3) No BARC address falls within more than one AB.

4) An Address Block Designation (ABD) is a CABA or a RABI.

5) Claimable AB Address (CABA) is claimable by a Claimant without using a Registrar.
   - identifies Claimable Address Blocks (CABs) holding Claimable Addresses (CAs)
   - CABA is a multicast MAC address, not in any AB, and used as a DA.

6) RABI
   - identifies a Registrable Address Block (RAB) holding Registrable Addresses (RAs)
   - Registrable Address Block Indicators (RABIs): held in inventory of a Registrar
     - may be assigned to Claimants
     - may be claimed by Registrants

7) A large set of Temporary Unicast Addresses (TUAs) is specified
   - useful for initial discovery by Claimant lacking a unicast address
## MAC Address Categorization

<table>
<thead>
<tr>
<th>determinable via inspection:</th>
<th>Expanded name</th>
<th>I/G</th>
<th>indicates, by inspection</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>claimable address, in claimable address block (CAB)</td>
<td>U,M</td>
<td>CABA, CAB Size, CAB (including all other CAs in CAB)</td>
</tr>
<tr>
<td>[ICA=unicast GCA=multicast]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CABA</td>
<td>CAB Address</td>
<td>M</td>
<td>CAB Size, CAB</td>
</tr>
<tr>
<td>RA</td>
<td>registrable address, in registrable address block (RAB)</td>
<td>U,M</td>
<td>BABI Size [Basic ABI Size]</td>
</tr>
<tr>
<td>[IRA=unicast GRA=multicast]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TUA</td>
<td>temporary unicast address</td>
<td>U</td>
<td>note: ~6.9E10 to choose among</td>
</tr>
</tbody>
</table>

## Address Block Designation (ABD) Categorization

<table>
<thead>
<tr>
<th>ABD</th>
<th>Address Block Designation (CABA or RABI)</th>
<th>not an address</th>
<th>AB (including all RAs in AB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CABA</td>
<td>CAB address</td>
<td>M</td>
<td>CAB Size, CAB</td>
</tr>
<tr>
<td>RABI</td>
<td>RAB Identifier</td>
<td>not an address</td>
<td>RAB Size, BABI Size, RAB, MABI Size [Multiple ABI Size]</td>
</tr>
<tr>
<td>CABA</td>
<td>CAB address</td>
<td>M</td>
<td>CAB Size, CAB</td>
</tr>
</tbody>
</table>
### BARC MAC Address Structure

<table>
<thead>
<tr>
<th>N11</th>
<th>r</th>
<th>i</th>
<th>jk</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>i</td>
<td>jk</td>
<td>m</td>
<td>(for registrable addresses, r=1; for claimable addresses, r=0)</td>
</tr>
<tr>
<td>N10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>m is the usual multicast (I/G) bit; 111 is local “SAI” range per IEEE Std 802c</td>
</tr>
<tr>
<td>N9</td>
<td></td>
<td></td>
<td></td>
<td>(0000 for claimable addresses)</td>
</tr>
</tbody>
</table>

12 nibbles per 48-bit address

- address block includes subblocks of
  - $16^j k$ claimable addresses, or
  - $16^j k$ registrable addresses (or aggregated into larger blocks)
- for claimable addresses, $i$ distinguishes
  - Claimable Addresses (CAs) from
  - CABAs
    - identifiers that are also used as addresses
- see Appendix for details

<table>
<thead>
<tr>
<th>$r$</th>
<th>$i$</th>
<th>$jk$</th>
<th>$m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA</td>
<td>0</td>
<td>1</td>
<td>CAB I/G</td>
</tr>
<tr>
<td>CABA</td>
<td>0</td>
<td>0</td>
<td>Size 1</td>
</tr>
<tr>
<td>TUA</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>RA</td>
<td>1</td>
<td>1</td>
<td>Option I/G</td>
</tr>
</tbody>
</table>

Option BABI Size
Claimant of CABA_X AB listens to CABA_X multicast address.

1. **CABA_1**: DISCOVER state
2. (unicast) **CABA_1**: CLAIMED state

**LAN**

(1) **CABA_1**: DISCOVER state
(2) (unicast) **CABA_1**: CLAIMED state

**LAN**

(1) **CABA_6**: DISCOVER state

**LAN**

(4) **CABA_6**: CLAIMED state
start listening to CABA_6
Registrar

- Claimant need not be aware of Registrar when initiating a claim.

- Registrar maintains an inventory of RABIs.
  - a protocol specifies how Registrars acquire RABIs.
  - set of RABs is disjoint from the set of CABs
    - AB is either claimable (CAB) or registrable (RAB); not both

- Registrar listens for all messages to a CABA.
  - r=0, i=0, m=1, i.e. DA begins 00**-1111
    - [MMRP NumberOfValues field is 13 bits]

- Registrar can respond to a DISCOVER with an offer of a RABI in its inventory.
  - The offer can also defend the DISCOVER’s CABA.
  - Registrar confirms registration of request for offered RABI.

- Pre-claim Inquiry lets Claimant reach Registrar or Advisor.
  - Client can learn of Registrars and received Claim proposals.
Operation with Registrars

- **Registrar RABI$_5$**
  - (2) (unicast) RABI$_5$: OFFERED state

- **Claimant**
  - (1) CABA$_1$: DISCOVER state
  - (2) (unicast) RABI$_1$: OFFERED state
  - (3) (unicast) RABI$_1$: REQUESTED state
  - (4) (unicast) RABI$_1$: REGISTERED state

- **LAN**
  - (1) CABA$_1$: DISCOVER state
  - (2) (unicast) RABI$_1$: OFFERED state

- **Registrar**
Inquiry to (anticipated) Registrar or Advisor

Claimant

RABI

Registrar

CABA

(1) ABD: INQUIRY

(2) RABI₁: OFFER

(3a) CABAₚ: DISCOVER

(3b) RABIₚ: INQUIRY

(4b) RABI₁: OFFER (based on RABIₚ)

Advisor cannot register ABDs; can propose ABD and Registrar address

Registrar

BD is CABA or RABI
DA could be, for example:
• “Nearest Customer Bridge” (NCB) address
  01-80-C2-00-00-00 (non-forwarding)
• stored unicast Registrar Address
• the null CABA (CABA₀)

• multicast

unicast
BARC Design

• A BARC architecture follows, with details including state machines.
  - additional details in Appendix

• BARC (Block Address Registration and Claiming) is put into the broader context of Address Registration and Claiming (ARC), which supports both:
  - address blocks (ABs), identified by Address Block Identifiers (ABIs)
  - address ranges (ARs), excluding addresses specified by BARC

• ARC is the general protocol
  - BARC handles ABI Registration and CABA Claiming
  - existing MAAP handles AR Claiming
ARC Claimant sets ingress MAC address for:
- Claimed CABA
Application also enabled to set filter to pass:
- AB addresses as adopted for receiving
Filter always set to pass MAAP multicast address

**-** Claimed CABA

**Ingress MAC address filter**
### BARCPDU Summary

<table>
<thead>
<tr>
<th>field name</th>
<th>purpose</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA</td>
<td>dest address</td>
<td>DA</td>
</tr>
<tr>
<td>SA</td>
<td>source address</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Ethertype</td>
<td>[tbd; could be 22F0 (MAAP Ethertype)]</td>
</tr>
<tr>
<td>t</td>
<td>subtype</td>
<td>[tbd, per 1722 WG; see IEEE 1722 Table 6]</td>
</tr>
<tr>
<td>S1</td>
<td>State 1</td>
<td>D (Discover), C (Claimed), V (Vacant), R (Registered), I (Inquiry), P (Proposal), A (address), RD, RC, RV, RX, N(null)</td>
</tr>
<tr>
<td>I1</td>
<td>Identifier 2</td>
<td>48-bit address or ABI</td>
</tr>
<tr>
<td>S2</td>
<td>State 2</td>
<td>O (Offered), A (address), N (null)</td>
</tr>
<tr>
<td>I2</td>
<td>Identifier 2</td>
<td>48-bit address or ABI</td>
</tr>
<tr>
<td>S3</td>
<td>State 3</td>
<td>A (address), T (token)</td>
</tr>
<tr>
<td>I3</td>
<td>Identifier 3</td>
<td>48-bit address or token</td>
</tr>
</tbody>
</table>

### AVTPDU Summary

<table>
<thead>
<tr>
<th>field name</th>
<th>purpose</th>
<th>content</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA</td>
<td>dest address</td>
<td>91:E0:F0:00:FF:00 for MAAP multicast</td>
</tr>
<tr>
<td>E</td>
<td>Ethertype</td>
<td>22F0 (MAAP Ethertype)</td>
</tr>
<tr>
<td>t</td>
<td>subtype</td>
<td>FE per IEEE 1722 Table 6</td>
</tr>
</tbody>
</table>
CABA Claim Procedure

ABD Claimant

Existing CABA Claimant

CABA State goes to D (Discover)

DC timer starts

CABA State goes from C to V (Vacant) (state machine destroyed)

decision to claim CABA

CABA:D

uni

CABA:D

uni:A

DC

CABA:V

CABA State

goes to D

(Discover)

ABD:state

uni

DA

SA; could be TUA

CABA:C

0:N

receives PDU

sends PDU

CABA State

is C

(Claimed)

not listening
to CABA1, so
PDU not received

not listening
to CABA1, so
PDU not received

CABA1 State
goes to D

DC

CABA1:D

uni

CABA1:D

uni:A

CABA1 State
goes to D

(Discover)

CABA1 State
goes from D to C
(Claimed)

adopt CABA1 AB
start listening to CABA1

CABA1

uni

sa

CABA1:C

sa:A

CABA1 State
goes from D to C
(Claimed)

CABA1 State
goes to D

(Discover)

uni

SA; could be TUA

SA; could be ICA

uni

a unicast address

sa

source address

TUA

temporary unicast address

ICA

individual (unicast)
claimable address
ABD Claimant/Registrar Procedure

begins without knowledge of Registrar

CABA State goes from C to V (Vacant)

RABI state machine changes from state V to state O (Offered); state machine stores RegA.

Even after CABA state machine goes to V state, further offers can arrive from other Registrars and transition other RABI state machines to O state.

select RABI with traits (such as AB size) of received CABA

RABI is controlled via ABD state machine in Claimant and a RABI state machine in Registrar.

new RABI State goes to O

offer times out

offer times out

Registrar's unicast address
RABI Registration

- RABI State goes from O to Q (Request)
- IRA selected from RABI IRA and token T stored in RABI State
- RABI R State Renew timer reset
  - adopt AB per RABI
  - start listening to IRA unicast address

**ABD Claimant**

- decision to register claim to RABI

**RABI Registrar**

- IRA & token stored in RABI State
- IRA & token T stored in RABI State
- RABI State goes from O to R

**IRA** individual (unicast) registrable address
Renewal and Withdrawal of a Registration

Claimant-initiated withdrawal

- decision to renew claim to RABI
- repeated claim (e.g. to renew)

Registrar-initiated revocation

- decision to withdraw claim to RABI
- RABI State goes from R to V
- decision to revoke claim to RABI
- RABI State goes from R to V

Note 1: The PDU will be rejected if the enclosed token and IRA do not match those in the RABI State Machine. This interferes with counterfeit PDUs.

Note 2: IRA and token are included only in unicast messages, limiting their distribution in the network.

Note: PDU rejected unless RegA, IRA, and token match the RABI State Machine. PDU not delivered if IRA is incorrect.

IRA | individual (unicast) registrable address
DA is an address of an expected Registrar or Advisor. This could be a configured or previously learned unicast address. It could also be a non-forwarding Nearest Customer Bridge address, if a response is anticipated from an immediately connected bridge of unknown address. DA could also be the specified null CABA (CABA₀).

Select RABI with traits (such as AB size) of received ABD (PRABI or CABA).

Register response to Inquiry is identical to its response to Discover, except that no “C” identifier is included since there is no claim to defend.

Register’s unicast address

New RABI State goes to O

Offer times out

Offer times out

RABI State goes from V to O

No state.

State indicated as I (Inquiry)

DA unicast

ABD:I

uni:A

uni:A

RegA:A

RegA

RABI:O

RABI:O

RABI:V

A decision to Inquire based on PRABI or CABA.

ABD (PRABI or CABA) is not claimed for use, only to indicate traits, including AB size.

Inquiry followed by Registrar Offer
DA is an address of an expected Registrar or Advisor. This could be a configured or previously learned unicast address. It could also be a non-forwarding Nearest Customer Bridge address, if a response is anticipated from an immediately connected bridge of unknown address. DA could also be the specified null CABA (CABA<sub>0</sub>).

ABD (PRABI or CABA) is a not claimed for use, only to indicate traits, including AB size.

ABD, State goes from V to P

An "A" state indicator is used to denote RegA, the address of a Registrar, proposed as a Registrar for the Claimant. Set to 0 for no recommendation. Registrar can propose its own address.

ABD<sub>1</sub> is the Advisor's ABD proposal (state indicated as "P"). This could be a CABA (possibly null CABA) or PRABI (possibly null PRABI).
ARC Claimant: cBARCPDU_in

BARCPDU (I1,S1, I2,S2 [,I3,S3])

null

S3
{Note: S1 is A}

C

if State(I3) is not V then
BARC(C,I1)! to I3

if I2 is not (null RABI) then
BARC(O,I1)! to I2

S2
{Note: S1 is D,C,R,Q,V,I}

O

if State(I1) is not V then
BARC(S1, I2 [,I3])! to I1

A

if State(I2) is V or E then
BARC(P, I1)! to I2

P

{Note: S1 is A}

{Note: square-bracketed parameters are sometimes absent.}

END

if I2 is not (null RABI) then
BARC(O,I1)! to I2

{Note: S1 is A}

ARC Claimant: cBARCPDU_out

cBARC (i1:s1, i2:s2 [,i3:s3]) {da[,sa]}

send BARCPDU with I1=i1,S1=s1, I2=i2,S2=s2 [,i3=i3,S3=s3] to da from sa

if sa is null, use default

could repeat, with repetition count possibly based on S1 and S2

END
AR Claimant Procedure

AR Claimant

AR:V

MAAP DA
uni
AVTPDU Probe/v2
AR

AR:D

AVTPDU Defend

Existing AR Claimant either MAAP or BARC

AR:D

AR Claimant and (legacy) MAAP Claimant respond identically to AVTPDU Probe/v2.

AR:V

AR State remains as D

Device can optionally register the Offer via BARC, while the AR may independently be claimed if not defended by a MAAP Claimant.

AR State goes to D

(Discovery)

AR State V (Vacant)

AR State goes to V

(Vacant)

AR:V

AR State V

(Vacant)

oblivious to Registrar

new RABI state machine in state O

(Offered)

RABI:O

MAAP AVPDUs are received by BARC Registrar. If the AVTPDU is a MAAP Probe/v2, then the Registrar responds just as it does to a Targeted Claim.

RABI Registrar

RABI:O

select RABI with traits (such as size) of AR

RABI State goes to O

MAAP DA
uni
AVTPDU Probe/v2
AR

 uni
RegA
RegA:A
RABI:O

AR address range
BARC Registrar: AVTPDU Processor

AVTPDU (maap_version, message_type, AR) at MAAP DA from SA

maap_version

not 2

message_type

not 1

Disco("M", AR, SA, [VID])

END

not 2

not 1

message_type

2

1

# AR State Transition Table

<table>
<thead>
<tr>
<th>Event</th>
<th>VACANT (V)</th>
<th>DISCOVERY (D)</th>
<th>ACQUIRED (A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add(sa)!</td>
<td>sMAAP(Begin(AR,sa)!)</td>
<td>DISCOVERY</td>
<td></td>
</tr>
<tr>
<td>rMAAP(AR:Defend)!</td>
<td></td>
<td>Outcome(A,AR)</td>
<td>ACQUIRED</td>
</tr>
<tr>
<td>rMAAP(AR:Initial)!</td>
<td></td>
<td>Outcome(F,AR)</td>
<td>Outcome(X)[AR]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VACANT</td>
<td>VACANT</td>
</tr>
</tbody>
</table>

rMAAP(AR:State!) invokes an event at the state machine when the MAAP state changes to State

sMAAP(Action!) invokes Action! event at MAAP state machine
BARC Architecture – Registrar

Ingress MAC address filter accepts BARCPDUs addressed to any CABA and MAAP multicast address for AVTPDUs.
BARC Registrar Application: Disco Process

DISCO (State, Range, SA, [VID])
from AVTPDU Processor

Discriminator

D

M

ABD_out=ABD to defend ABD;
otherwise ABD_out=null CABA
State_out=C

ABD_out=0
State_out=N

ABD_out=0
State_out=N

select RABI to offer, based on:
• Disco State (BARC: D or I; MAAP:M)
• Disco ABD request, including:
  - size
  - CABA or PRABI
• MAAP Range request (including size)
• VID
• BARC ABD Registrar States
set RABI=null for no offer

Invite(ABD_out,State_out,SA)! to RABI

Proposal

State

RegA = address of proposed Registrar
ABD=proposed CABA or PRABI

sBARC(RegA:A, ABD:P)(SA)

Offer

Respond

I

RegA = address of proposed Registrar
ABD=proposed CABA or PRABI

sBARC(RegA:A, ABD:P)(SA)
A RABI Registrant could be specified. Alternatively, a Primary RABI Claimant could be configured to hold and defend many RABIs in reserve, ensuring that RABIs are not excessively claimed. This would suffice in many cases.
BARC Registrar: rBARCPDU_in

BARCPDU (I1,S1, I2,S2) [I3,S3 [VID]]

S1

Disco(S1,I1,I2[VID])

if I1 is active then
BARC(S1,I2,S3[VID])! to RABI I1

Q or V

D or I

N or C

RD,RC,RV,RX

ClaimCheck(S1,I1,I2)

END

BARC Registrar: rBARCPDU_out

sBARC (i1:s1 [i2:s2]) [i3,s3] {da}

send BARCPDU with I1=i1,S1=s1,
[I2=i2,S2=s2]
[I3=i3,S3=s3]
to da from default sa
(if i2 is null, then
I2=default sa)

could repeat,
with repetition count based on S1

END
VLANs

- All address assignments are specific to the VLAN (or VLANs) in which messaging is communicated and under which the assignment was completed.

- Usage of any address may need to be limited to the VLAN or VLANs under which it was obtained.

- Due to the possibility that the same unicast address may be assigned in different VLANs, Independent VLAN Learning (IVL) may be needed in bridges, per IEEE Std 802.1Q Annex F (F.1.2).
  - This requirement could be relaxed in some cases
  - e.g. when assigned unicast addresses are declared via MMRP (instead of learning)

- This issue is common to claiming protocols generally.

- Some approaches follow.
IEEE Std 802-2014 says “Local MAC addresses need to be unique on a LAN or bridged LAN unless the bridges support VLANs with independent learning.”

With IVL, each VLAN has an independent forwarding table.
- but IVL is not always possible

BARC claiming on each VLAN is independent
  a duplicate address may occur in more than one VLAN; that is not harmful if managed carefully

A claimant with multiple VLANs needs to claim in each VLAN.
Claimed address is usable only in claimed VLAN:
  Claimant needs to bind address to VLAN
For convenience, Claimant may claim the same address in each of its VLANs
  -Still, requires multiple claim messages and multiple forwarding table entries.
  -Device needing many VLANs should consider an EUI
Claiming with VLAN: SVL

With SVL, VLANs share a forwarding table.

BARC claiming on each VLAN is independent
an address could become a duplicate, existing in more than one VLAN
forwarding table is limited to one entry per address, so duplication is catastrophic.
Network is configured with Registrar on all active VLANs on which BARC is used.

BARC claim from any VLAN is delivered to Registrar.
- Offer delivered on Claimant’s VLAN

Registrar ensures that registered address is unique across all (or perhaps only some) of its VLANs.
- SVL or IVL will work

Registrar needs to remember over which VLANs the address was assigned.
- Should be retained in State Machine
SVL is used for Asymmetric VLAN (IEEE Std 802.1Q Annex F.1.3)

Registrar can assign address to be unique across all VLANs available to the Registrar.
Summary

• Claimants operate with or without Registrars.

• Multiple registrars are supported, holding claims of disjoint RABIs.

• The block discretization provides:
  – a vast set of addresses to a LAN
  – a large set of temporary unicast addresses
  – operational efficiency and simplicity
  – both unicast and multicast addresses to Claimant
    – unicast and multicast subblocks share the same range, except for the m bit
      – could be exploited
    – devices needing both unicast and multicast addresses need make only one claim

• Could integrate with MMRP to limit propagation and eliminate learning of unicast AB content.
  – MMRP needs to efficiently handle address ranges
  – BARP could be specified as alternative MRP application
    (e.g. would understand an ABD)
Appendix 1

- additional details on BARC addresses and identifiers
BARC Address Parsing

<table>
<thead>
<tr>
<th></th>
<th>r</th>
<th>i</th>
<th>jk</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CABA</td>
<td>0</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>TUA</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**BARC address is CA:** Claimable Address

**within CAB:** Claimable Address Block identified by ABI: CAB Size jk

**MAC address**

**N10**

**Claimable Address** within **CAB**: Claimable Address Block identified by **ABI**: CAB Size jk

**N9**

**reserved**

**N11**

**0**

**not 111**

**1**

**m**

**1**

**0**

**IRA:** Individual CA (Unicast)

**GRA:** Group CA (Multicast)

**CABA of Size jk**
- Claimable ABD
- 16^th ICAs
- 16^th GCAs

**TUA:** temporary unicast address

**CA indicates CABA**

*lsn= least significant nibbles
# CABA and CA, CAB Size 0-3

## CAB Size C=0

<table>
<thead>
<tr>
<th>CAB Size C=0</th>
<th>CAB Size C=1</th>
<th>CAB Size C=2</th>
<th>CAB Size C=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CABA</td>
<td>CAB</td>
<td>CABA</td>
<td>CAB</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 1 0 0</td>
<td>0 0 0 1</td>
<td>0 1 0 1</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>1 1 1 *</td>
<td>1 1 1 1</td>
<td>1 1 1 *</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

| X8 | X8 | X8 | X8 | X8 | X8 | X8 | X8 |
| X7 | X7 | X7 | X7 | X7 | X7 | X7 | X7 |
| X6 | X6 | X6 | X6 | X6 | X6 | X6 | X6 |
| X5 | X5 | X5 | X5 | X5 | X5 | X5 | X5 |
| X4 | X4 | X4 | X4 | X4 | X4 | X4 | X4 |
| X3 | X3 | X3 | X3 | X3 | X3 | X3 | X3 |
| X2 | X2 | X2 | X2 | X2 | X2 | X2 | X2 |
| X1 | X1 | X1 | X1 | 0 | * | 0 | * |
| X0 | X0 | X0 | X0 | 0 | * | 0 | * |

## Notes

- 6.9E10 Size 0 CABAs
- 1 CA/subblock
- 4.3E9 Size 1 CABAs
- 16 CAs/subblock
- 2.7E8 Size 2 CABAs
- 256 CA/subblock
- 1.7E7 Size 3 CABAs
- 4096 CAs/subblock

* indicates wildcard (any value)
CABA/CAB Math

- CABA indicates the CAB
- CA indicates the CABA
- C=jk indicates the CAB Size
- each CAB subblock includes 16^C contiguous addresses

C(X) = (X&0x300000000000) / 0x100000000000 [extracts CAB Size C when X is CABA or CA]
Cmask(C) = ~(0x410000000000 + 0x10**C – 1) [CABA mask, per Size; used to create CABA from CA]
CABA(CA) = CA&Cmask(C(CA)) + 0x010000000000

Example: [Note: Underlining on the middle four nibbles is shown only as a reading aid.]
- CA = 0x6F0123456789 = 0110-1111-0000-0001-0010-0011-0100-0101-0110-0111-1000-1001
- C(CA) = 0x200000000000 / 0x100000000000 = 2
- Cmask(0x2) = ~(0x410000000000 + 0x0100 - 1) = ~(0x4100000000FF) = 0xBEFFFFFFFFFF000
- CABA(CA) = CA&0xBEFFFFFFFFFF00 + 010000000000 = 0x2F0123456700

A CA is within CABA_x if and only if CABA(CA) = CABA_x
- this requires identical CAB Size

The CAB of CABA_x is the set of all CAs that satisfy CABA(CA) = CABA_x

Lowest ICA in CABA:  ICAmin = (CABA-0x010000000000) | 0x400000000000 (example: 0x6E0123456700)
Highest ICA in CABA:  ICAmax = ICAmin + 0x10**C(CABA) – 1 (example: 0x6E01234567FF)

Lowest GCA in CABA:  GCAmn = CABA | 0x410000000000 (example: 0x6F0123456700)
Highest GCA in CABA:  GCAmx = GCAmn + 0x10**C(CABA) – 1 (example: 0x6F01234567FF)
### ABD (CABA/RABI) Format

<table>
<thead>
<tr>
<th>ABD</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>CABA</td>
<td>CAB</td>
</tr>
<tr>
<td>N11</td>
<td>0 0 j k</td>
</tr>
<tr>
<td>N10</td>
<td>1 1 1 1</td>
</tr>
</tbody>
</table>

- CABA is both an ABD (indicating CAB) and a MAC address
- CAB indicates the CABA
- jk indicates the CABA size C
- the C least significant nibbles of the CABA are 0
- each CAB subblock includes 16^C contiguous addresses
- each CAB includes a unicast subblock and a multicast subblock
- no CA within a CAB is within any other CAB (that is, a CAB with a different CABA)

<table>
<thead>
<tr>
<th>RABI</th>
<th>RAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>N11</td>
<td>1 i j k</td>
</tr>
<tr>
<td>N10</td>
<td>a b c d</td>
</tr>
</tbody>
</table>

- RABI is an ABD (indicating RAB) and never used as a MAC address
- RAB does not fully indicate the RABI
- i indicates RABI Option (Options 0 and 1 provide independent RABIs/RABs)
- jk indicates the BABI Size B
- “BABI” for “Basic Address Block Indicator”
- abcd indicates MABI Size M; those bits are not in the RAB addresses
- “MABI” for “Multiple Address Block Indicator”
- RAB Size R=B+M
- the R least significant nibbles of the RABI are 0
- each RAB subblock includes 16^R contiguous addresses
- each RAB includes a unicast subblock and a multicast subblock

### RABI Aggregation

RAB Size $R=B+M$
- for 48-bit addresses, set $a=0$; then $R = jk + bcd \leq 3 + 7 = 10$, matching the 10 available nibbles N0-N9
- could use $ijk$ as the BABI Size, and/or the full $abcd$ as the MABI Size; e.g., for 64-bit addresses

- A RABI may aggregate other RABIs.
- A RABI of RAB Size $R$ and MABI Size $M$ can be disaggregated into:
  - 16 RABIs of RAB Size $R-1$ (MABI Size $M-1$), or
  - $16^2$ RABIs of RAB Size $R-2$ (MABI Size $M-1$), or
  - $16^n$ RABIs of RAB Size $R-n$ (MABI Size $M-n$), or
  - ... $16^M$ RABIs of RAB Size $B$ (MABI Size $0$), or
- A RABI of RAB Size $B$ (MABI Size $0$) cannot be disaggregated.
- An RA appears in one and only RABI of each $M$. 
RABI and RA, MABI Size 0, BABI Size 0-3

<table>
<thead>
<tr>
<th>BABI Size 0</th>
<th>BABI Size 1</th>
<th>BABI Size 2</th>
<th>BABI Size 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>RABI</td>
<td>RAB</td>
<td>RABI</td>
<td>RAB</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>1 1 0 1</td>
<td>1 1 1 0</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>X9</td>
<td>X9</td>
<td>X9</td>
<td>X9</td>
</tr>
<tr>
<td>X8</td>
<td>X8</td>
<td>X8</td>
<td>X8</td>
</tr>
<tr>
<td>X7</td>
<td>X7</td>
<td>X7</td>
<td>X7</td>
</tr>
<tr>
<td>X6</td>
<td>X6</td>
<td>X6</td>
<td>X6</td>
</tr>
<tr>
<td>X5</td>
<td>X5</td>
<td>X5</td>
<td>X5</td>
</tr>
<tr>
<td>X4</td>
<td>X4</td>
<td>X4</td>
<td>X4</td>
</tr>
<tr>
<td>X3</td>
<td>X3</td>
<td>X3</td>
<td>X3</td>
</tr>
<tr>
<td>X2</td>
<td>X2</td>
<td>X2</td>
<td>X2</td>
</tr>
<tr>
<td>X1</td>
<td>X1</td>
<td>X1</td>
<td>X1</td>
</tr>
<tr>
<td>X0</td>
<td>X0</td>
<td>0</td>
<td>*</td>
</tr>
</tbody>
</table>

2 contiguous subblocks per RABI (one unicast, one multicast)

- 1.1E12 Size 0 RABIs
- 1 RA/subblock
- 6.9E10 Size 1 RABIs
- 16 RAs/subblock
- 4.3E9 Size 2 RABIs
- 256 RA/subblock
- 2.7E8 Size 3 RABIs
- 4096 RAs/subblock

* indicates wildcard (any value)
Aggregation Example: BABI Size 3, various MABI Sizes

<table>
<thead>
<tr>
<th>MABI Size 0</th>
<th>MABI Size 1</th>
<th>MABI Size 2</th>
<th>MABI Size 7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RABI</strong></td>
<td><strong>RABI</strong></td>
<td><strong>RABI</strong></td>
<td><strong>RABI</strong></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X9</td>
<td>X9</td>
<td>X9</td>
<td>X9</td>
</tr>
<tr>
<td>X8</td>
<td>X8</td>
<td>X8</td>
<td>X8</td>
</tr>
<tr>
<td>X7</td>
<td>X7</td>
<td>X7</td>
<td>X7</td>
</tr>
<tr>
<td>X6</td>
<td>X6</td>
<td>X6</td>
<td>X6</td>
</tr>
<tr>
<td>X5</td>
<td>X5</td>
<td>X5</td>
<td>X5</td>
</tr>
<tr>
<td>X4</td>
<td>X4</td>
<td>X4</td>
<td>X4</td>
</tr>
<tr>
<td>X3</td>
<td>X3</td>
<td>X3</td>
<td>X3</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td>0</td>
<td>*</td>
</tr>
</tbody>
</table>

* indicates wildcard (any value)  
# indicates wildcard (any value)
**Example:**
Hierarchical RABI Addressing with common Registrar

<table>
<thead>
<tr>
<th>Held by Registrar</th>
<th>Assigned to Bridge 1 by Registrar</th>
<th>Assigned by Registrar to bridges, end stations, etc. connected to Bridge 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MABI Size 2</td>
<td>MABI Size 1</td>
<td>MABI Size 1</td>
</tr>
<tr>
<td>RABI</td>
<td>RAB</td>
<td>RABI</td>
</tr>
<tr>
<td>1111</td>
<td>1111</td>
<td>1111</td>
</tr>
<tr>
<td>0010</td>
<td>0001</td>
<td>0001</td>
</tr>
<tr>
<td>X9</td>
<td>X9</td>
<td>X9</td>
</tr>
<tr>
<td>X8</td>
<td>X8</td>
<td>X8</td>
</tr>
<tr>
<td>X7</td>
<td>X7</td>
<td>X7</td>
</tr>
<tr>
<td>X6</td>
<td>X6</td>
<td>X6</td>
</tr>
<tr>
<td>X5</td>
<td>X5</td>
<td>X5</td>
</tr>
<tr>
<td>0</td>
<td>#</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>#</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td>0</td>
</tr>
</tbody>
</table>

Bridge serves as Advisor to connected station; proposes PRABI, and Registrar Address, in response to an Inquiry.
Example:
Hierarchical CABA Addressing

<table>
<thead>
<tr>
<th>Claimed by Bridge 1</th>
<th>Claimed by bridges, end stations, etc. connected to Bridge 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CAB Size C=3</strong></td>
<td><strong>CAB Size C=2</strong></td>
</tr>
<tr>
<td>CABA</td>
<td>CAB</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>0 1 1 1</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>1 1 1 *</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>X8</td>
<td>X8</td>
</tr>
<tr>
<td>X7</td>
<td>X7</td>
</tr>
<tr>
<td>X6</td>
<td>X6</td>
</tr>
<tr>
<td>X5</td>
<td>X5</td>
</tr>
<tr>
<td>X4</td>
<td>X4</td>
</tr>
<tr>
<td>X3</td>
<td>X3</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
</tr>
</tbody>
</table>

Bridge serves as Advisor to connected stations; proposes CABA in response to an Inquiry.
RABI registered as a Claimant could be disaggregated and reassigned by a Registrar function managed jointly with the Claimant.

A RABI in the “REGISTERED” state of the ABD Claimant State Machine could be considered to be in the Inventory of the RABI Registrar State Machine (along with Claimed RABIs) and could be disaggregated, Offered and Registered by that Registrar.
Example:
Hierarchical RABI Disaggregation with Tiered Registrars

Assigned to Bridge 1 by Registrar

<table>
<thead>
<tr>
<th>MABI Size 2</th>
<th>RABI</th>
<th>RAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>1111</td>
<td>0010</td>
</tr>
<tr>
<td>X9</td>
<td>X9</td>
<td></td>
</tr>
<tr>
<td>X8</td>
<td>X8</td>
<td></td>
</tr>
<tr>
<td>X7</td>
<td>X7</td>
<td></td>
</tr>
<tr>
<td>X6</td>
<td>X6</td>
<td></td>
</tr>
<tr>
<td>X5</td>
<td>X5</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>#</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>#</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

Used by Bridge 1 directly

<table>
<thead>
<tr>
<th>MABI Size 1</th>
<th>RABI</th>
<th>RAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>1111</td>
<td>0001</td>
</tr>
<tr>
<td>X9</td>
<td>X9</td>
<td>1111</td>
</tr>
<tr>
<td>X8</td>
<td>X8</td>
<td>1111</td>
</tr>
<tr>
<td>X7</td>
<td>X7</td>
<td>1111</td>
</tr>
<tr>
<td>X6</td>
<td>X6</td>
<td>1111</td>
</tr>
<tr>
<td>X5</td>
<td>X5</td>
<td>1111</td>
</tr>
<tr>
<td>0</td>
<td>8-15</td>
<td>8-15</td>
</tr>
<tr>
<td>0</td>
<td>#</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>#</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

Reassigned by Bridge 1 as secondary Registrar to connected bridges, end stations, etc.

<table>
<thead>
<tr>
<th>MABI Size 1</th>
<th>RABI</th>
<th>RAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>1111</td>
<td>0001</td>
</tr>
<tr>
<td>X9</td>
<td>X9</td>
<td>1111</td>
</tr>
<tr>
<td>X8</td>
<td>X8</td>
<td>1111</td>
</tr>
<tr>
<td>X7</td>
<td>X7</td>
<td>1111</td>
</tr>
<tr>
<td>X6</td>
<td>X6</td>
<td>1111</td>
</tr>
<tr>
<td>X5</td>
<td>X5</td>
<td>1111</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>#</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

...
i(X) = \( (X \& 0x4000000000000000) / 0x4000000000000000 \) [RABI Option, when X is RABI or RA]

B(X) = \( (X \& 0x3000000000000000) / 0x1000000000000000 \) [BABI Size, when X is RABI or RA]

M(X) = \( (X \& 0x0700000000000000) / 0x0100000000000000 \) [MABI Size when X is a RABI]

R(X) = B(X) + M(X) [RAB Size, when X is a RABI]

Rmask(N) = ~\( (0x0F000000000 + 0x10^{N-1}) \) [mask, used below]

An RA is within RABIx if and only if RA&Rmask(R(RABIx)) = RABIx&Rmask(R(RABIx))
   • this requires identical RABI Option and BABI Size
The RAB of RABIx is the set of all RAs that satisfy RA&Rmask(R(RABIx)) = RABIx&Rmask(R(RABIx))

Example:
   • RABI = 0xF20123400000 = 1111-0010-0000-0001-0010-0011-0100-0000-0000-0000-0000-0000
   • B(RABI) = 0x3000000000000000 / 0x1000000000000000 = 3
   • M(RABI) = 0x0200000000000000 / 0x0100000000000000 = 2
   • R(RABI) = 3+2 = 5
   • Rmask(5) = ~(0x0F000000000 + 0x10^{5-1}) = ~(0x0F00000FFFFF) = 0xF0FFFFF0000
   • R ABI&Rmask(5) = 0xF0123400000

   • RA = 0xFE0123456789 = 1111-1110-0000-0001-0010-0011-0100-0101-0110-0111-1000-1001
   • RA&Rmask(5) = 0xF00123400000 = RABI&Rmask(5)
   • so RA is within RABI
RABI Math 102

RABIcheck(RABI1, RABI2) determines whether RABI RABI1 overlaps RABI2.
RABIcheck(R1, R2) = R1 & Rmask(R(R2))
RABI1 shares RAs with RABI2 if and only if:
RABIcheck(RABI1, RABI2) = RABIcheck(RABI2, RABI1)
• Note: This can be ruled out by inspection if the two N11 nibbles differ

Example:
• RABI2 = 0xF50100000000 = 1111 - 0101 - 0000 - 0001 - 0000 - 0000 - 0000 - 0000 - 0000 - 0000 - 0000 - 0000 - 0000 - 0000 - 0000
  • B(RABI2) = 0x300000000000 / 0x100000000000 = 3
  • M(RABI2) = 0x070000000000 / 0x100000000000 = 5
  • R(RABI2) = 3 + 5 = 8
  • Rmask(8) = 0xF0FF00000000
  • RABI1 = 0xF20123400000
  • RABI1 & Rmask(8) = 0xF00100000000
  • RABI2 & Rmask(5) = 0xF00100000000
  • so RABI1 and RABI2 have RAs in common

Lowest IRA in RABI: IRAmin = RABI & 0xF0FFFFFF0000 + 0x0E0000000000
  Example: 0xFE0123400000)
Highest IRA in RABI: IRAmax = IRAmin + 0x10**(R(RABI)) – 1
  Example: 0xFE01234FFFFF)
Lowest GRA in RABI: GRAmin = RABI & 0xF0FFFFFF0000 + 0x0F0000000000
  Example: 0xFF0123400000)
Highest GRA in RABI: GRAmax = GRAmin + 0x10**(R(RABI)) – 1
  Example: 0xFF01234FFFFF)
An RA exists in one and only one RABI (called RABI\(_W\)) of each MABI Size \(W\).
Given the MABI Size \(W\), what is RABI\(_W\)?

A RABI exists in one and only one aggregated RABI called RABI of each larger MABI Size \(W\).
Given the MABI Size \(W\), what is RABI\(_W\)?

\[
RABI_W(X, W) = X \& \text{Rmask}(W+B(X)) + W \times 0x010000000000 \quad [X \text{ is an RA, or a RABI with } M(X)<W]
\]

Example:
\begin{itemize}
  \item RA\(_1\) = 0xF20123400000; B(RA\(_1\))=3; M(RA\(_1\))=2
  \item \(W = 5\); \(W + B = 8\)
  \item RA\(_1\)&Rmask(8) = 0xF20123400000&0xF0FF00000000 = 0xF00100000000
  \item RABIm(RA\(_1\),5) = 0xF00100000000 + 0x050000000000 = 0xF50100000000
\end{itemize}

Example:
\begin{itemize}
  \item RA = 0xFE0123456789; [B(RA)=3]
  \item \(W = 5\); \(W + B = 8\)
  \item RA&Rmask(8) = 0xFE0123456789&0xF0FF00000000 = 0xF00100000000
  \item RABIm(RA,8) = 0xF00100000000 + 0x050000000000 = 0xF50100000000
\end{itemize}

Example:
\begin{itemize}
  \item RA = 0xFE0123456789; [B(RA)=3]
  \item \(W = 2\); \(W + B = 5\)
  \item RA&Rmask(5) = 0xFE0123456789&0xF0FF00000000 = 0xF00123400000
  \item RABIm(RA,5) = 0xF00123400000 + 0x020000000000 = 0xF20123400000
\end{itemize}
### Temporary Unicast Address (TUA)

- For temporary use

- device without a source address selects a random temporary unicast address for initial discovery only
  - protocol then assigns at least one persistent unicast address

- simultaneous duplicate temporary addresses may lead to message loss in some circumstances
  - network learns route to source as initial message crosses the network
  - before response is returned, another initial message with duplicate source address crosses the path and rewrites the route
  - unlikely to be disastrous
  - loss of initial message will be corrected eventually

- nevertheless, need to consider the likelihood of duplication

- Temporary address range includes 9 full nibbles of 16 values each (0–F)
  - \(16^9 = 68,719,476,736\) temporary addresses in the pool
  - chance of no duplicates with \(k\) randomly selected addresses is approximated
    \[
    \exp\left(-\frac{k^2}{2N}\right)
    \]
  - with \(k=1000\) devices simultaneously using a temporary address, chance of no duplicates is \(~0.99988\)
  - can add \(jk\) bits to the pool if more entropy is needed
null CABA (CABA₀)

- Null CABA (CABA₀) is not an assignable address.
- Registrar listens to CABA₀.
- No Claimant listens to CABA₀.
- Can be used as the DA of BARC Inquiry; e.g., when Registrar address is unknown.
Proposed RABI (PRABI) and null RABI/PRABI

• Usable only as the content of a BARC Inquiry or BARC Proposal message.

• Indicates a set of RABIs characterized by RABI Option \(i\), BABI Size B \(jk\), and MABI Size M \(bcd\).

• If the PRABI has the form of a RABI (with 0 in the B+M least significant nibbles), then the PRABI set is only that RABI.

• Non-zero values in the lower B+M nibbles can signify that some bits of the higher nibbles are “don’t care.”

• For example, the lower B+B nibbles of the PRABI could form a bitmask of the higher nibbles.

Proposed RABIs are those that satisfy:

\[
\text{RABI} \mid \text{Pmask} = (\text{PRABI} \& \sim (0x10^{R(\text{PRABI})} \sim 1)) \mid \text{Pmask}
\]

where \( \text{Pmask}(\text{PRABI}) = 10^{R(\text{PRABI})} \times \text{PRABI} \& (0x10^{R(\text{Rcap})} \sim 1) \)

and \( \text{Rcap} = \min(R(\text{PRABI}), 10 \sim R(\text{PRABI})) \)

null RABI/null PRABI

• As a PRABI in a BARC Inquiry or Proposal message, indicates a set of RABIs characterized by RABI Option \(i\), BABI Size B \(jk\), and MABI Size M \(bcd\), without expressing a preference for the RABI values of the 0 nibbles.

• As a RABI in a BARC Offer message, indicates to Claimant that no RABI is offered.
Appendix 2

- additional procedural details
BARC Address Propagation with MMRP

The ARC Claimant Application AddABD Process includes “declare with MMRP”. This entails declaring, to MMRP (when available), MMRP attributes, using an MMRPDU per IEEE Std 802.1Q § 10.12.1.6:

- The multicast address represented by the CABA
  FirstValue field = CABA/NumberOfValues=1
- The unicast address subblock indicated by the ABD (CABA or RABI)
  FirstValue field = first address in unicast subblock
  NumberOfValues = $16^{\text{Size}}$ per ABD Size
  - limited to Size up to 3: $16^3=4096$, and MRP provides 13 bits of NumberOfValues ($2^{13}=8192$)

The ARC Claimant Application AddABD Process includes (“select CABA”); this selection should consider any local MMRP registration database to avoid selecting a registered CABA.

Unicast MMRP declaration can be useful because:
(1) A one-step declaration covers a contiguous range of self-assigned unicast addresses.
(2) Eliminates flooding for all the unicast addresses in the assignment.
(3) Eliminates the need for learning of each unicast address when used.
(4) Precludes erroneous re-learning of an address when a false duplicate is used elsewhere in the network.
  - Could be a way to control duplication.
  - Security issues to study.

BARC could alternatively specify “BARP,” a new MRP application. This could entail the following changes:

(a) the BARP application would be enabled to Join and Leave with the ABD as the declared attribute
(b) the BARP application would be specified to understand the semantics of the ABD and extract from it the indicated ABD multicast address and the indicated unicast address set, then use it to populate the FDB
(c) In the BARC Claimant ABD State Machine, the CABA claim ["sBARC(CABA:C)"] might not be needed, since the a BARP declaration could convey the claim to the CABA as well as the declaration of interest in receiving at the CABA multicast address

BARP might be better suited to specification within IEEE Std 802.1Q instead of 802.1CQ.
ARC Claimant Application Process:
BARC Management

START: Receive BARC FYI(status [,value])

- condition
- Alert

- Offered
  - action to add AB?
    - yes: go to: Initiate Adding
    - no: Expiring

- Expiring
  - action to resolve duplicate
    - yes: go to: Initiate Adding
    - no: delete

- delete
  - BARC Delete()|ABD|
  - update MMRP

- replacement ABD?
  - yes: go to: Initiate Adding
  - no: renewal
ARC Claimant Application Process: Drop Claim

START: Initiate Dropping

AB or AR

ABD Claimant Delete()[ABD]

END

AR Claimant Drop()[AR]
ARC Claimant Application Process: MAAP Management

START: Receive Outcome(X) [AR]

replacement AB or AR?

yes -> go to: Initiate Adding

no ->

END
BARC Registrar Application: ClaimCheck Process

- For RABI state machines in RC State, if RABIcheck(RABI,I1) then:
  - take action to resolve the identified existing assignment conflict
    - if deleting RABI, revoke any of its RABIs in Registered state

- For RABI state machines in RC State, if RABIcheck(RABI,I1) then:
  - sBARC(I1:RX){CABA_0}
    - take action to resolve the identified existing assignment conflict
      - if deleting RABI, revoke any of its RABIs in Registered state
  else
    - Occupied()! to RABI

- For RABI state machines in Registered State, if any RABIcheck(RABI,I1) then:
  - sBARC(I1:RC){I2}
BARC Registrar Application:
RABI Claiming

START: Initiate Adding or Deleting

Adding

Deleting

Remove()[RABI]

END

select RABI considering:
• BARC RABI Registrar States
• most recent Outcome content

Try()[RABI]

result of Outcome(result)[RABI]

C

U,V

yes

try again?

no

END

END

try again? no

END
RABI Claimant Application Process: BARC Management

START: Receive BARC FYI (condition)

1. Renewed
2. Alert
3. Condition
4. Action to resolve duplicate
5. Renew
6. BARC Renew(][RABI]!
7. END

- BARC Remove(][RABI]
- Replacement RABI?
  - Yes: go to: RABI Claiming/Initiate Adding
  - No: Action to resolve duplicate

- Delete

Alert: BARC Renew(][RABI)!