```
1  ---------- MODULE PaxosStore ----------
   Specification of the consensus protocol in PaxosStore.
   See [PaxosStore@VLDB2017](https://www.vldb.org/pvldb/vol10/p1730-lin.pdf) by Tencent.
   In this version:
   - Client-restricted config (Ballot)
   - Message types: "Prepare", "Accept", "ACK"
13 EXTENDS Integers, FiniteSets
14 ----------
15 Max(m, n) ≜ IF m > n THEN m ELSE n
16 Injective(f) ≜ ∀ a, b ∈ DOMAIN f : (a ≠ b) ⇒ (f[a] ≠ f[b])
17 ----------
18 CONSTANTS
19     Participant,   the set of partipants
20     Value          the set of possible input values for Participant to propose
22 None ≜ CHOOSE b : b ∉ Value
23 NP ≜ Cardinality(Participant)   number of p ∈ Participants
25 Quorum ≜ {Q ∈ SUBSET Participant : Cardinality(Q) * 2 ≥ NP + 1}
26 ASSUME QuorumAssumption ≜
27     ∧  ∀ Q ∈ Quorum : Q ⊆ Participant
28     ∧  ∀ Q1, Q2 ∈ Quorum : Q1 ∩ Q2 ≠ {}
30 Ballot ≜ Nat
32 PIndex ≜ CHOOSE f ∈ [Participant → 1 .. NP] : Injective(f)   TODO: (1) symmetry set? (2) model
33 Bals(p) ≜ {b ∈ Ballot : b%NP = PIndex[p] − 1}   allocate ballots for each p ∈ Participant
34 ----------
35 State ≜ [maxBal  : Ballot ∪ {−1},
36          maxVBal : Ballot ∪ {−1}, maxVVal : Value ∪ {None}]
38 InitState ≜ [maxBal ↦ −1, maxVBal ↦ −1, maxVVal ↦ None]
   For simplicity, in this specification, we choose to send the complete state of a participant each
   time. When receiving such a message, the participant processes only the "partial" state it needs.
44 Message ≜ [type  : {"Prepare", "Accept", "ACK"},
45            from : Participant, to : SUBSET Participant,   TODO: remove "to"
46            state : [Participant → State]]
47 ----------
48 VARIABLES
49     state,   state[p][q]: the state of q ∈ Participant from the view of p ∈ Participant
50     msgs     the set of messages that have been sent
52 vars ≜ ⟨state, msgs⟩
54 TypeOK ≜
55     ∧  state ∈ [Participant → [Participant → State]]
```

```
56      ∧    msgs ⊆ Message

58  Send(m) ≜ msgs′ = msgs ∪ {m}
59  ----------------------------------------------------------------
60  Init ≜
61      ∧ state = [p ∈ Participant ↦ [q ∈ Participant ↦ InitState]]
62      ∧ msgs = {}
```

$p \in Participant$ starts the prepare phase by issuing a ballot $b \in Ballot$.

```
66  Prepare(p, b) ≜
67      ∧  state[p][p].maxBal < b
68      ∧  b ∈ Bals(p)
69      ∧  state′ = [state EXCEPT ![p][p].maxBal = b]
70      ∧  Send([type ↦ "Prepare", from ↦ p, to ↦ Participant, state ↦ state′[p]])
```

$q \in Participant$ updates its own state $state[q]$ according to the actual state pp of $p \in Participant$ extracted from a message $m \in Message$ it receives. This is called by $OnMessage(q)$.

Note: pp is $m.state[p]$; it may not be equal to $state[p][p]$ at the time $UpdateState$ is called.

```
79  UpdateState(q, p, pp) ≜
80      state′ = [state EXCEPT
81                  ![q][p].maxBal = Max(@, pp.maxBal),
82                  ![q][p].maxVBal = Max(@, pp.maxVBal),
83                  ![q][p].maxVVal = IF state[q][p].maxVBal < pp.maxVBal
84                                    THEN pp.maxVVal ELSE @,
85                  ![q][q].maxBal = Max(@, pp.maxBal),
86                  ![q][q].maxVBal = IF state[q][q].maxBal ≤ pp.maxVBal
87                                    THEN pp.maxVBal ELSE @,    make promise
88                  ![q][q].maxVVal = IF state[q][q].maxBal ≤ pp.maxVBal  TODO: write-once?
89                                    THEN pp.maxVVal ELSE @]   accept
```

$q \in Participant$ receives and processes a message in $Message$.

```
93  OnMessage(q) ≜
94      ∃ m ∈ msgs :
95          ∧ m.type = "ACK" ⇒ m.to = {q}
96          ∧ LET p ≜ m.from
97            IN  UpdateState(q, p, m.state[p])
98          ∧ IF  ∨ m.state[q].maxBal < state′[q][q].maxBal   TODO: delete "if"?
99                ∨ m.state[q].maxVBal < state′[q][q].maxVBal
100             THEN Send([type ↦ "ACK", from ↦ q, to ↦ {m.from}, state ↦ state′[q]])
101             ELSE UNCHANGED msgs
```

$p \in Participant$ starts the accept phase by issuing the ballot $b \in Ballot$ with value $v \in Value$.

```
106 Accept(p, b, v) ≜
107     ∧ ¬∃ m ∈ msgs :   TODO: delete it? to allow repeating Phase2a?
108           m.type = "Accept" ∧ m.state[p].maxBal = b
109     ∧ b ∈ Bals(p)      TODO: delete it? to break "client-restricted config"?
110     ∧ ∃ Q ∈ Quorum : ∀ q ∈ Q : state[p][q].maxBal = b
111     ∧ ∨ ∀ q ∈ Participant : state[p][q].maxVBal = −1   free to pick its own value
```

```
112          ∨ ∃ q ∈ Participant :  v is the value with the highest maxVBal
113               ∧ state[p][q].maxVVal = v
114               ∧ ∀ r ∈ Participant : state[p][q].maxVBal ≥ state[p][r].maxVBal
115       ∧ state' = [state EXCEPT ![p][p].maxVBal = b,
116                                ![p][p].maxVVal = v]
117       ∧ Send([type ↦ "Accept", from ↦ p, to ↦ Participant, state ↦ state'[p]])
118 ├──────────────────────────────────────────────────────────────────────────┤
119  Next  ≜  ∃ p ∈ Participant : ∨ OnMessage(p)
120                               ∨ ∃ b ∈ Ballot : ∨ Prepare(p, b)
121                                                ∨ ∃ v ∈ Value : Accept(p, b, v)
122  Spec  ≜  Init ∧ □[Next]_vars
123 ├──────────────────────────────────────────────────────────────────────────┤
124  ChosenP(p)  ≜  the set of values chosen by p ∈ Participant
125      {v ∈ Value : ∃ b ∈ Ballot :
126                     ∃ Q ∈ Quorum : ∀ q ∈ Q : ∧ state[p][q].maxVBal = b
127                                              ∧ state[p][q].maxVVal = v}

129  chosen  ≜  UNION {ChosenP(p) : p ∈ Participant}

131  Consistency  ≜  Cardinality(chosen) ≤ 1

133  THEOREM Spec ⇒ □Consistency
134 └──────────────────────────────────────────────────────────────────────────┘
      \ * Modification History
      \ * Last modified Wed Aug 14 18:44:24 CST 2019 by hengxin
      \ * Last modified Mon Jul 22 13:59:15 CST 2019 by pure_
      \ * Last modified Mon Jun 03 21:26:09 CST 2019 by stary
      \ * Last modified Wed May 09 21:39:31 CST 2018 by dell
      \ * Created Mon Apr 23 15:47:52 GMT + 08:00 2018 by pure_
```