```
1  ---------- MODULE AJupiterExtended ----------
   AJupiter extended with JupiterCtx. This is used to show that AJupiter implements XJupiter.
6  EXTENDS JupiterCtx
7  ----------
8  VARIABLES cbuf, crec, sbuf, srec
10 varsEx ≜ ⟨intVars, ctxVars, cbuf, crec, sbuf, srec⟩
12 Msg ≜ [ack : Int, cop : Cop, oid : Oid]
13 ----------
14 TypeOKEx ≜
15     ∧ TypeOKInt
16     ∧ TypeOKCtx
17     ∧ Comm(Msg)!TypeOK
18     ∧ crec ∈ [Client → Int]
19     ∧ srec ∈ [Client → Int]
20     ∧ cbuf ∈ [Client → Seq(Cop)]
21     ∧ sbuf ∈ [Client → Seq(Cop)]
22 ----------
23 InitEx ≜
24     ∧ InitInt
25     ∧ InitCtx
26     ∧ Comm(Msg)!Init
27     ∧ crec = [c ∈ Client ↦ 0]
28     ∧ srec = [c ∈ Client ↦ 0]
29     ∧ cbuf = [c ∈ Client ↦ ⟨⟩]
30     ∧ sbuf = [c ∈ Client ↦ ⟨⟩]
31 ----------
   Client c ∈ Client issues an operation op.
35 DoOp(c, op) ≜
36     LET cop ≜ [op ↦ op, oid ↦ [c ↦ c, seq ↦ cseq′[c]], ctx ↦ ds[c]]
37     IN  ∧ crec′ = [crec EXCEPT ![c] = 0]
38         ∧ cbuf′ = [cbuf EXCEPT ![c] = Append(@, cop)]
39         ∧ state′ = [state EXCEPT ![c] = Apply(op, @)]
40         ∧ Comm(Msg)!CSend([ack ↦ crec[c], cop ↦ cop, oid ↦ cop.oid])
42 DoIns(c) ≜
43     ∃ ins ∈ {op ∈ Ins : op.pos ∈ 1 .. (Len(state[c]) + 1) ∧ op.ch ∈ chins ∧ op.pr = Priority[c]} :
44         ∧ DoOp(c, ins)
45         ∧ chins′ = chins \ {ins.ch}
47 DoDel(c) ≜
48     ∃ del ∈ {op ∈ Del : op.pos ∈ 1 .. Len(state[c])} :
49         ∧ DoOp(c, del)
50         ∧ UNCHANGED chins
```

```
DoEx(c) ≜
        ∧ DoCtx(c)
        ∧ ∨ DoIns(c)
          ∨ DoDel(c)
        ∧ UNCHANGED ⟨sbuf, srec⟩
```

Client $c \in Client$ receives a message from the *Server*.

```
RevEx(c) ≜
    ∧ Comm(Msg)!CRev(c)
    ∧ crec' = [crec EXCEPT ![c] = @ + 1]
    ∧ LET m ≜ Head(cincoming[c])
          cBuf ≜ cbuf[c]
          cShiftedBuf ≜ SubSeq(cBuf, m.ack + 1, Len(cBuf))
          xcop ≜ XformOpOps(COT, m.cop, cShiftedBuf)
          xcBuf ≜ XformOpsOp(COT, cShiftedBuf, m.cop)
      IN  ∧ cbuf' = [cbuf EXCEPT ![c] = xcBuf]
          ∧ state' = [state EXCEPT ![c] = Apply(xcop.op, @)]
    ∧ RevCtx(c)
    ∧ UNCHANGED ⟨chins, sbuf, srec⟩
```

The *Server* receives a message.

```
SRevEx ≜
    ∧  Comm(Msg)!SRev
    ∧  LET m ≜ Head(sincoming)
           c ≜ ClientOf(m.cop)
           cBuf ≜ sbuf[c]
           cShiftedBuf ≜ SubSeq(cBuf, m.ack + 1, Len(cBuf))
           xcop ≜ XformOpOps(COT, m.cop, cShiftedBuf)
           xcBuf ≜ XformOpsOp(COT, cShiftedBuf, m.cop)
       IN  ∧ srec' = [cl ∈ Client ↦
                        IF cl = c
                        THEN srec[cl] + 1
                        ELSE 0]
           ∧ sbuf' = [cl ∈ Client ↦
                        IF cl = c
                        THEN xcBuf
                        ELSE Append(sbuf[cl], xcop)]
           ∧ state' = [state EXCEPT ![Server] = Apply(xcop.op, @)]
           ∧ Comm(Msg)!SSend(c, [cl ∈ Client ↦ [ack ↦ srec[cl], cop ↦ xcop, oid ↦ xcop.oid]])
    ∧  SRevCtx
    ∧  UNCHANGED ⟨chins, cbuf, crec⟩
```

```
NextEx ≜
    ∨  ∃ c ∈ Client : DoEx(c) ∨ RevEx(c)
```

```
100         ∨ SRevEx

102  FairnessEx ≜  There is no requirement that the clients ever generate operations.
103      WF_varsEx(SRevEx ∨ ∃ c ∈ Client : RevEx(c))

105  SpecEx ≜ InitEx ∧ □[NextEx]_varsEx  ∧ FairnessEx
106 ├──────────────────────────────────────────────────────────────────────────┤
107  QC ≜  Quiescent Consistency
108         Comm(Msg)!EmptyChannel ⇒ Cardinality(Range(state)) = 1

110  THEOREM SpecEx ⇒ □QC
111 └──────────────────────────────────────────────────────────────────────────┘
      \* Modification History
      \* Last modified Sat Dec 29 18:55:12 CST 2018 by hengxin
      \* Created Thu Dec 27 21:15:09 CST 2018 by hengxin
```