```
──────────────── MODULE 2PCwithBTM ────────────────
EXTENDS Integers, Sequences, FiniteSets, TLC
CONSTANT RM,          The set of participating resource managers RM = 1 .. 3
         RMMAYFAIL,
         TMMAYFAIL  Whether TM may fail MAYFAIL = TRUE or FALSE
 ************************************************************************
A modified version of P2TCommit at http://lamport.azurewebsites.net/tla/two-phase.html
Transaction manager (TM) is added.
--algorithm TransactionCommit{
  variable rmState = [rm ∈ RM ↦ "working"],
           tmState = "init" ;
  define {
    canCommit ≜     ∀ rmc ∈ RM : rmState[rmc] ∈ {"prepared"}
                 ∨   ∃ rm ∈ RM  : rmState[rm] ∈ {"committed"}  for when BTM takes over
    canAbort ≜      ∃ rm  ∈ RM  : rmState[rm] ∈ {"aborted", "failed"}
                 ∧ ¬∃ rmc ∈ RM : rmState[rmc] = "committed"   inconsistent if commented
    }
  macro Prepare( p ) {
    await rmState[p] = "working" ;
    rmState[p] := "prepared" ;  }

  macro Decide( p ) {
    either { await tmState = "commit" ;
             rmState[p] := "committed" ;  }

    or     { await rmState[p] = "working" ∨ tmState = "abort" ;
             rmState[p] := "aborted" ;  }
    }

  macro Fail( p ) {
    if ( RMMAYFAIL ) rmState[p] := "failed" ;
    }

  fair  process ( RManager ∈ RM ) {
   RS: while ( rmState[self] ∈ {"working", "prepared"} ) {
         either Prepare(self)or Decide(self)or Fail(self) }
    }

  fair process ( TManager = 0 ) {
 TS: either { await canCommit ;
          TC:  tmState := "commit" ;
          F1:  if ( TMMAYFAIL ) tmState := "hidden" ;  }

      or { await canAbort ;
       TA: tmState := "abort" ;
       F2: if ( TMMAYFAIL ) tmState := "hidden" ;  }
    }
  fair process ( BTManager = 10 ) {
BTS: either { await canCommit ∧ tmState = "hidden" ;
         BTC: tmState := "commit" ;  }

       or {   await canAbort ∧ tmState = "hidden" ;
         BTA: tmState := "abort" ;  }
     }
 }
  .'
************************************************************************
 BEGIN TRANSLATION
VARIABLES rmState, tmState, pc

 define statement
canCommit ≜      ∀ rmc ∈ RM : rmState[rmc] ∈ {"prepared"}
               ∨   ∃ rm ∈ RM   : rmState[rm] ∈ {"committed"}
canAbort ≜       ∃ rm    ∈ RM   : rmState[rm] ∈ {"aborted", "failed"}
             ∧ ¬∃ rmc ∈ RM : rmState[rmc] = "committed"


vars ≜ ⟨rmState, tmState, pc⟩

ProcSet ≜ (RM) ∪ {0} ∪ {10}

Init ≜   Global variables
           ∧ rmState = [rm ∈ RM ↦ "working"]
           ∧ tmState = "init"
           ∧ pc = [self ∈ ProcSet ↦ CASE self ∈ RM → "RS"
                                      □   self = 0 → "TS"
                                      □   self = 10 → "BTS"]

RS(self) ≜  ∧ pc[self] = "RS"
            ∧ IF rmState[self] ∈ {"working", "prepared"}
                 THEN ∧ ∨ ∧ rmState[self] = "working"
                          ∧ rmState' = [rmState EXCEPT ![self] = "prepared"]
                        ∨ ∧ ∨ ∧ tmState = "commit"
                                ∧ rmState' = [rmState EXCEPT ![self] = "committed"]
                              ∨ ∧ rmState[self] = "working" ∨ tmState = "abort"
                                ∧ rmState' = [rmState EXCEPT ![self]       = "aborted"]
                        ∨ ∧ IF RMMAYFAIL ∧ ¬∃ rm ∈ RM : rmState[rm] = "failed"
                                THEN ∧ rmState' = [rmState EXCEPT ![self] = "failed"]
                                ELSE ∧ TRUE
                                     ∧ UNCHANGED rmState
                      ∧ pc' = [pc EXCEPT ![self] = "RS"]
                 ELSE ∧ pc' = [pc EXCEPT ![self] = "Done"]
                      ∧ UNCHANGED rmState
```

```
                  ∧ UNCHANGED tmState

RManager(self) ≜ RS(self)

TS ≜ ∧ pc[0] = "TS"
     ∧ ∨ ∧ canCommit
         ∧ pc′ = [pc EXCEPT ![0] = "TC"]
       ∨ ∧ canAbort
         ∧ pc′ = [pc EXCEPT ![0] = "TA"]
     ∧ UNCHANGED ⟨rmState, tmState⟩

TC ≜ ∧ pc[0] = "TC"
     ∧ tmState′ = "commit"
     ∧ pc′ = [pc EXCEPT ![0] = "F1"]
     ∧ UNCHANGED rmState

F1 ≜ ∧ pc[0] = "F1"
     ∧ IF TMMAYFAIL
          THEN ∧ tmState′ = "hidden"
          ELSE ∧ TRUE
               ∧ UNCHANGED tmState
     ∧ pc′ = [pc EXCEPT ![0] = "Done"]
     ∧ UNCHANGED rmState

TA ≜ ∧ pc[0] = "TA"
     ∧ tmState′ = "abort"
     ∧ pc′ = [pc EXCEPT ![0] = "F2"]
     ∧ UNCHANGED rmState

F2 ≜ ∧ pc[0] = "F2"
     ∧ IF TMMAYFAIL
          THEN ∧ tmState′ = "hidden"
          ELSE ∧ TRUE
               ∧ UNCHANGED tmState
     ∧ pc′ = [pc EXCEPT ![0] = "Done"]
     ∧ UNCHANGED rmState

TManager ≜ TS ∨ TC ∨ F1 ∨ TA ∨ F2

BTS ≜ ∧ pc[10] = "BTS"
      ∧ ∨ ∧ canCommit ∧ tmState = "hidden"
          ∧ pc′ = [pc EXCEPT ![10] = "BTC"]
        ∨ ∧ canAbort ∧ tmState = "hidden"
          ∧ pc′ = [pc EXCEPT ![10] = "BTA"]
      ∧ UNCHANGED ⟨rmState, tmState⟩

BTC ≜ ∧ pc[10] = "BTC"
      ∧ tmState′ = "commit"
```

```
              ∧ pc′ = [pc EXCEPT ![10] = "Done"]
              ∧ UNCHANGED rmState

BTA ≜ ∧ pc[10] = "BTA"
      ∧ tmState′ = "abort"
      ∧ pc′ = [pc EXCEPT ![10] = "Done"]
      ∧ UNCHANGED rmState

BTManager ≜ BTS ∨ BTC ∨ BTA

Next ≜ TManager ∨ BTManager
          ∨ (∃ self ∈ RM : RManager(self))
          ∨   Disjunct to prevent deadlock on termination
            ((∀ self ∈ ProcSet : pc[self] = "Done") ∧ UNCHANGED vars)

Spec ≜ ∧ Init ∧ □[Next]_vars
       ∧ ∀ self ∈ RM : WF_vars(RManager(self))
       ∧ WF_vars(TManager)
       ∧ WF_vars(BTManager)

Termination ≜ ◇(∀ self ∈ ProcSet : pc[self] = "Done")

 END TRANSLATION

The invariants:
TypeOK ≜
  The type-correctness invariant
 ∧ rmState ∈ [RM → {"working", "prepared", "committed", "aborted", "failed"}]
 ∧ tmState ∈ {"init", "commit", "abort", "hidden"}

Consistency ≜
  A state predicate asserting that two RMs have not arrived at conflicting decisions.
 ∀ rm1, rm2 ∈ RM : ¬ ∧ rmState[rm1] = "aborted"
                     ∧ rmState[rm2] = "committed"


NotCommitted ≜ ∀ rm ∈ RM : rmState[rm] ≠ "committed"

└──────────────────────────────────────────────────────────────────────────┘
\* Modification History
\* Last modified Wed Dec 13 14:34:34 EST 2017 by mad
\* Last modified Fri Nov 17 12:18:24 EST 2017 by murat
\* Last modified Tue Oct 11 08:14:15 PDT 2011 by lamport
\* Created Mon Oct 10 05:31:02 PDT 2011 by lamport
```