```
---------------------------- MODULE StateAWSet ----------------------------
EXTENDS AWSet
---------------------------------------------------------------------------
VARIABLES
    aset,        aset[r]: set of active Instance(s) maintained by r ∈ Replica
    tset,        tset[r]: set of tombstone Instance(s) maintained by r ∈ Replica

   network variables
    incoming,    incoming[r]: incoming messages at replica r ∈ Replica
    lmsg,
   SEC variables
    updateset,
    uincoming

Nvars   ≜ ⟨incoming, lmsg⟩
SECvars ≜ ⟨updateset, uincoming⟩
vars ≜ ⟨aset, tset, seq, Nvars, SECvars⟩
---------------------------------------------------------------------------
Msg ≜ [r : Replica, seq : Nat, A : SUBSET Element, T : SUBSET Element]   message type
Network ≜ INSTANCE BasicNetwork                                           instantiate basic network

Read(r, s) ≜ {ele.d : ele ∈ s}                                            read the state of r ∈ Replica
SEC ≜ INSTANCE StateSEC WITH data ← aset                                  instantiate SEC module
---------------------------------------------------------------------------
TypeOK ≜       check types
    ∧  aset ∈ [Replica → SUBSET Element]
    ∧  tset ∈ [Replica → SUBSET Element]
    ∧  IntTypeOK
    ∧  Network!SMTypeOK
    ∧  SEC!SECTypeOK
---------------------------------------------------------------------------
Init ≜        initial state
    ∧ aset = [r ∈ Replica ↦ {}]
    ∧ tset = [r ∈ Replica ↦ {}]
    ∧ Network!NInit
    ∧ SEC!StateSECInit
    ∧ IntInit
---------------------------------------------------------------------------
 1
 1
 1
 1
 1
 1
 1
 1
```

```
$Send(r) \triangleq$        $r \in Replica$ sends a message
    $\land\ Network!NBroadcast(r,\ [r \mapsto r,\ seq \mapsto seq[r],\ \ A \mapsto aset[r],\ T \mapsto tset[r]])$
    $\land\ IntSend(r)$
    $\land\ SEC!StateSECSend(r,\ seq[r])$
    $\land\ \text{UNCHANGED}\ \langle aset,\ tset \rangle$

$Deliver(r) \triangleq$        $r \in Replica$ delivers a $message(lmsg')$
    $\land\ \ Network!NDeliver(r)$
    $\land\ \ IntDeliver(r)$
    $\land\ \ SEC!StateSECDeliver(r,\ [r \mapsto lmsg'.r,\ seq \mapsto lmsg'.seq])$
    $\land\ \ tset' = [tset\ \text{EXCEPT}\ ![r] = @ \cup lmsg'.T]$
    $\land\ \ aset' = [aset\ \text{EXCEPT}\ ![r] = (@ \cup lmsg'.A) \setminus tset'[r]]$
    $\land\ \ \text{UNCHANGED}\ \langle \rangle$
```

```
$Add(d,\ r) \triangleq$        $r \in Replica$ adds $d \in Data$
    $\land\ aset' = [aset\ \text{EXCEPT}\ ![r] = @ \cup \{[d \mapsto d,\ r \mapsto r,\ k \mapsto seq[r]]\}]$
    $\land\ IntDo(r)$
    $\land\ Network!NDo$
    $\land\ SEC!StateSECUpdate(r,\ seq[r])$
    $\land\ \text{UNCHANGED}\ \langle tset \rangle$

$Remove(d,\ r) \triangleq$        $r \in Replica$ removes $d \in Data$
    $\land\ \text{LET}\ E\ \triangleq\ \{ele \in aset[r] : ele.d = d\}$
        $\text{IN}\ \ \land\ aset' = [aset\ \text{EXCEPT}\ ![r] = @ \setminus E]$
              $\land\ tset' = [tset\ \text{EXCEPT}\ ![r] = @ \cup E]$
    $\land\ IntDo(r)$
    $\land\ Network!NDo$
    $\land\ SEC!StateSECUpdate(r,\ seq[r])$

$Do(r) \triangleq$        update operations
    $\exists\, a \in Data : Add(a,\ r) \lor Remove(a,\ r)$
```

```
$Next \triangleq$        next-state relation
    $\exists\, r \in Replica : Deliver(r) \lor Send(r) \lor Do(r)$

$Spec \triangleq Init \land \Box[Next]_{vars}$
```
---

* Modification History
* Last modified *Thu Jun* 13 21:34:30 *CST* 2019 by *xhdn*
* Created *Fri* May 24 14:13:38 *CST* 2019 by *xhdn*