─── MODULE *VoucherCancel* ───

This specification describes the cancellation of Voucher between an Issuer and a Holder. It is implemented over the Two-Phase Commit protocol, in which a Voucher Transaction Provider (*VTP*) coordinates the Voucher Issuers (Is) to cancel vouchers (Vs) to Voucher Holders (*Hs*) as described in the *VoucherLifeCycle* specification module. In this specification, *Hs* and Is spontaneously issue *Prepared* messages. We ignore the Prepare messages that the *VTP* can send to the *Hs* and Is.

For simplicity, we also eliminate *Abort* messages sent by an *Hs* / Is when it decides to abort. Such a message would cause the *VTP* to abort the transaction, an event represented here by the *VTP* spontaneously deciding to abort.

Note: This operation is an addendum to the operations described in *RFC* 3506. This operation is not described in the *RFC*.

CONSTANT

V,	The set of Vouchers
H,	The set of Voucher Holders
I	The set of Voucher Issuers

VARIABLES

$vState$,	$vState[v]$ is the state of voucher v.
$vlcState$,	$vlcState[v]$ is the state of the voucher life cycle machine.
$hState$,	$hState[h]$ is the state of voucher holder h.
$iState$,	$iState[i]$ is the state of voucher issuer i.
$vtpState$,	The state of the voucher transaction provider.
$vtpCPrepared$,	The set of *Hs* and Is from which the *VTP* has received "Prepared for Voucher *Cancel*" messages.
$msgs$	

In the protocol, processes communicate with one another by sending messages. For simplicity, we represent message passing with the variable *msgs* whose value is the set of all messages that have been sent. A message is sent by adding it to the set *msgs*. An action that, in an implementation, would be enabled by the receipt of a certain message is here enabled by the presence of that message in *msgs*. For simplicity, messages are never removed from *msgs*. This allows a single message to be received by multiple receivers. Receipt of the same message twice is therefore allowed; but in this particular protocol, that's not a problem.

Messages $\triangleq$

$[type : \{$ "Prepared" $\}, vh : H] \cup$
$[type : \{$ "Prepared" $\}, vi : I] \cup$
$[type : \{$ "Cancel", "Abort" $\}]$

$VTPTypeOK \triangleq$

$\wedge vState \in [V \rightarrow \{$ "valid", "cancelled" $\}]$
$\wedge vlcState \in [V \rightarrow \{$ "working", "done" $\}]$
$\wedge hState \in [H \rightarrow \{$ "holding", "prepared", "cancelled", "aborted" $\}]$
$\wedge iState \in [I \rightarrow \{$ "waiting", "prepared", "cancelled", "aborted" $\}]$
$\wedge vtpState \in \{$ "init", "done" $\}$
$\wedge vtpCPrepared \subseteq (H \cup I)$
$\wedge msgs \subseteq Messages$

$VTPInit \triangleq$

$\wedge vState = [v \in V \mapsto$ "valid" $]$
$\wedge vlcState = [v \in V \mapsto$ "working" $]$
$\wedge hState = [h \in H \mapsto$ "holding" $]$
$\wedge iState = [i \in I \mapsto$ "waiting" $]$
$\wedge vtpState =$ "init"
$\wedge vtpCPrepared = \{\}$
$\wedge msgs = \{\}$

$VTPRcvPrepared(h, i) \triangleq$

$\wedge vState = [v \in V \mapsto$ "valid" $]$
$\wedge vlcState = [v \in V \mapsto$ "working" $]$
$\wedge vtpState =$ "init"
$\wedge [type \mapsto$ "Prepared", $vh \mapsto h] \in msgs$
$\wedge [type \mapsto$ "Prepared", $vi \mapsto i] \in msgs$
$\wedge vtpCPrepared' = vtpCPrepared \cup \{h, i\}$
$\wedge$ UNCHANGED $\langle vState, vlcState, hState, iState, vtpState, msgs \rangle$

$VTPCancel(v) \triangleq$

$\wedge\ vState[v] =$ "valid"
$\wedge\ vlcState[v] =$ "working"
$\wedge\ vtpState =$ "init"
$\wedge\ vtpCPrepared = H \cup I$
$\wedge\ vtpState' =$ "done"
$\wedge\ vState' = [vState$ EXCEPT $![v] =$ "cancelled"$]$
$\wedge\ vlcState' = [vlcState$ EXCEPT $![v] =$ "done"$]$
$\wedge\ msgs' = msgs \cup \{[type \mapsto$ "Cancel"$]\}$
$\wedge$ UNCHANGED $\langle hState,\ iState,\ vtpCPrepared\rangle$

$VTPAbort(v)\ \triangleq$

The VTP spontaneously aborts the transaction.

$\wedge\ vState[v] =$ "valid"
$\wedge\ vlcState[v] =$ "working"
$\wedge\ vtpState =$ "init"
$\wedge\ vtpState' =$ "done"
$\wedge\ msgs' = msgs \cup \{[type \mapsto$ "Abort"$]\}$
$\wedge$ UNCHANGED $\langle vState,\ vlcState,\ hState,\ iState,\ vtpCPrepared\rangle$

$HPrepare(h)\ \triangleq$

Voucher holder h prepares.

$\wedge\ vState = [v \in V \mapsto$ "valid"$]$
$\wedge\ vlcState = [v \in V \mapsto$ "working"$]$
$\wedge\ hState[h] =$ "holding"
$\wedge\ hState' = [hState$ EXCEPT $![h] =$ "prepared"$]$
$\wedge\ msgs' = msgs \cup \{[type \mapsto$ "Prepared",$\ vh \mapsto h]\}$
$\wedge$ UNCHANGED $\langle vState,\ vlcState,\ vtpState,\ iState,\ vtpCPrepared\rangle$

$HChooseToAbort(h)\ \triangleq$

Voucher holder h spontaneously decides to abort. As noted above, h does not send any message in our simplified spec.

$\wedge\ vState = [v \in V \mapsto$ "valid"$]$
$\wedge\ vlcState = [v \in V \mapsto$ "working"$]$
$\wedge\ hState[h] =$ "holding"
$\wedge\ hState' = [hState$ EXCEPT $![h] =$ "aborted"$]$
$\wedge$ UNCHANGED $\langle vState,\ vlcState,\ vtpState,\ iState,\ vtpCPrepared,\ msgs\rangle$

$HRcvCancelMsg(h)\ \triangleq$

Voucher holder h is told by the VTP to $Cancel$.

$\wedge\ vState \in [V \rightarrow \{$ "valid",\ "cancelled"$\}]$
$\wedge\ vlcState \in [V \rightarrow \{$ "working",\ "done"$\}]$
$\wedge\ hState[h] =$ "holding"
$\wedge\ [type \mapsto$ "Cancel"$] \in msgs$
$\wedge\ hState' = [hState$ EXCEPT $![h] =$ "cancelled"$]$
$\wedge$ UNCHANGED $\langle vtpState,\ vState,\ vlcState,\ iState,\ vtpCPrepared,\ msgs\rangle$

3

$HRcvAbortMsg(h) \triangleq$

Voucher holder h is told by the *VTP* to abort.

  $\wedge\ vState = [v \in V \mapsto \text{"valid"}]$
  $\wedge\ vlcState = [v \in V \mapsto \text{"working"}]$
  $\wedge\ hState[h] = \text{"holding"}$
  $\wedge\ [type \mapsto \text{"Abort"}] \in msgs$
  $\wedge\ hState' = [hState \text{ EXCEPT } ![h] = \text{"aborted"}]$
  $\wedge\ \text{UNCHANGED } \langle vState,\ vlcState,\ vtpState,\ iState,\ vtpCPrepared,\ msgs \rangle$

$IPrepare(i) \triangleq$

Voucher issuer i prepares.

  $\wedge\ vState = [v \in V \mapsto \text{"valid"}]$
  $\wedge\ vlcState = [v \in V \mapsto \text{"working"}]$
  $\wedge\ iState[i] = \text{"waiting"}$
  $\wedge\ iState' = [iState \text{ EXCEPT } ![i] = \text{"prepared"}]$
  $\wedge\ msgs' = msgs \cup \{[type \mapsto \text{"Prepared"},\ vi \mapsto i]\}$
  $\wedge\ \text{UNCHANGED } \langle vState,\ vlcState,\ vtpState,\ hState,\ vtpCPrepared \rangle$

$IChooseToAbort(i) \triangleq$

Voucher issuer i spontaneously decides to abort. As noted above, i does not send any message in our simplified spec.

  $\wedge\ vState = [v \in V \mapsto \text{"valid"}]$
  $\wedge\ vlcState = [v \in V \mapsto \text{"working"}]$
  $\wedge\ iState[i] = \text{"waiting"}$
  $\wedge\ iState' = [iState \text{ EXCEPT } ![i] = \text{"aborted"}]$
  $\wedge\ \text{UNCHANGED } \langle vState,\ vlcState,\ vtpState,\ hState,\ vtpCPrepared,\ msgs \rangle$

$IRcvCancelMsg(i) \triangleq$

Voucher issuer i is told by the *VTP* to *Cancel*.

  $\wedge\ vState \in [V \to \{\text{"valid"},\ \text{"cancelled"}\}]$
  $\wedge\ vlcState \in [V \to \{\text{"working"},\ \text{"done"}\}]$
  $\wedge\ iState[i] = \text{"waiting"}$
  $\wedge\ [type \mapsto \text{"Cancel"}] \in msgs$
  $\wedge\ iState' = [iState \text{ EXCEPT } ![i] = \text{"cancelled"}]$
  $\wedge\ \text{UNCHANGED } \langle vtpState,\ vState,\ vlcState,\ hState,\ vtpCPrepared,\ msgs \rangle$

$IRcvAbortMsg(i) \triangleq$

Voucher issuer i is told by the *VTP* to abort.

  $\wedge\ vState = [v \in V \mapsto \text{"valid"}]$
  $\wedge\ vlcState = [v \in V \mapsto \text{"working"}]$
  $\wedge\ iState[i] = \text{"waiting"}$
  $\wedge\ [type \mapsto \text{"Abort"}] \in msgs$
  $\wedge\ iState' = [iState \text{ EXCEPT } ![i] = \text{"aborted"}]$
  $\wedge\ \text{UNCHANGED } \langle vState,\ vlcState,\ vtpState,\ hState,\ vtpCPrepared,\ msgs \rangle$

$VTPNext \triangleq$
$\quad \lor \exists\, v \in V :$
$\qquad VTPCancel(v) \lor VTPAbort(v)$
$\quad \lor \exists\, h,\, i \in H \cup I :$
$\qquad VTPRcvPrepared(h,\, i)$
$\quad \lor \exists\, h \in H :$
$\qquad HPrepare(h) \lor HChooseToAbort(h)$
$\qquad \lor HRcvAbortMsg(h) \lor HRcvCancelMsg(h)$
$\quad \lor \exists\, i \in I :$
$\qquad IPrepare(i) \lor IChooseToAbort(i)$
$\qquad \lor IRcvAbortMsg(i) \lor IRcvCancelMsg(i)$

$VTPConsistent \triangleq$

A state predicate asserting that a H and an I have not reached conflicting decisions. It is an invariant of the specification.

$\quad \land \forall\, h \in H,\, i \in I : \quad \land \lnot\, \land hState[h] = \text{"cancelled"}$
$\qquad\qquad\qquad\qquad\qquad\qquad\quad \land iState[i]\ = \text{"aborted"}$
$\qquad\qquad\qquad\qquad\quad \land \lnot\, \land hState[h] = \text{"aborted"}$
$\qquad\qquad\qquad\qquad\qquad\quad \land iState[i]\ = \text{"cancelled"}$

$VTPVars \triangleq \langle hState,\, iState,\, vState,\, vlcState,\, vtpState,\, vtpCPrepared,\, msgs \rangle$

$VTPSpec \triangleq VTPInit \land \Box[VTPNext]_{VTPVars}$

The complete spec of the a Voucher *Cancel* using Two-Phase Commit protocol.

THEOREM $VTPSpec \Rightarrow \Box(VTPTypeOK \land VTPConsistent)$

This theorem asserts the truth of the temporal formula whose meaning is that the state predicate $VTPTypeOK \land VTPConsistent$ is an invariant of the specification $VTPSpec$. Invariance of this conjunction is equivalent to invariance of both of the formulas $VTPTypeOK$ and $VTPConsistent$.

We now assert that the Voucher *Cancel* specification implements the Voucher Life Cycle specification of a voucher mentioned in module *VoucherLifeCycle*. The following statement imports all the definitions from module *VoucherLifeCycle* into the current module.

INSTANCE $VoucherLifeCycle$

THEOREM $VTPSpec \Rightarrow VSpec$

This theorem asserts that the specification $VTPSpec$ of the Two-Phase Commit protocol implements the specification $VSpec$ of the Voucher life cycle specification.

\ * Modification History
\ * Last modified *Tue Jun* 12 13:03:21 *IST* 2018 by Fox
\ * Created *Fri Mar* 16 17:45:37 *SGT* 2018 by Fox