* Authors: *Santhosh Raju* < *santhosh@byisystems.com* >
* Cherry *G. Mathew* < cherry@*byisystems.com* >
* Fransisca *Andriani* < *sisca@byisystems.com* >
*

MODULE *VoucherRedeem*

The description is based on the "Redeem" operation mentioned in *RFC* 3506. This specification describes the redemption of Voucher between a Holder and Collector. It is implemented over the Two-Phase Commit protocol, in which a Voucher Transaction Provider (*VTP*) coordinates the Voucher Holders (*Hs*) to redeem vouchers (Vs) to Voucher Collectors (*Cs*) described in the *VoucherLifeCycle* specification module. In this specification, *Hs* and *Cs* spontaneously issue *Prepared* messages. We ignore the Prepare messages that the *VTP* can send to the *Hs* and *Cs*.

For simplicity, we also eliminate *Abort* messages sent by an *Hs* / *Cs* 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.

CONSTANT

V,	The set of Vouchers
H,	The set of Voucher Holders
C	The set of Voucher Collectors

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.
$cState$,	$cState[c]$ is the state of voucher collector c.
$vtpState$,	The state of the voucher transaction provider.
$vtpRPrepared$,	The set of *Hs* and *Cs* from which the *VTP* has received "Prepared for Voucher *Redeem*" 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.

$Messages \triangleq$

The set of all possible messages. Messages of type "Prepared" are sent from the H indicated by the message's vh field to the VTP. Similar "Prepared" is also sent from C indicated by message's vc field to the VTP. Messages of type "Redeem" and "Abort" are broadcast by the $VTPs$, to be received by all Hs and Cs. The set $msgs$ contains just a single copy of such a message.

$[type : \{\text{"Prepared"}\},\ vh : H] \cup$
$[type : \{\text{"Prepared"}\},\ vc : C] \cup$
$[type : \{\text{"Redeem"},\ \text{"Abort"}\}]$

$VTPTypeOK \triangleq$

The type-correctness invariant

$\land\ vState \in [V \to \{\text{"valid"},\ \text{"redeemed"}\}]$
$\land\ vlcState \in [V \to \{\text{"working"},\ \text{"done"}\}]$
$\land\ hState \in [H \to \{\text{"holding"},\ \text{"prepared"},\ \text{"redeemed"},\ \text{"aborted"}\}]$
$\land\ cState \in [C \to \{\text{"waiting"},\ \text{"prepared"},\ \text{"redeemed"},\ \text{"aborted"}\}]$
$\land\ vtpState \in \{\text{"init"},\ \text{"done"}\}$
$\land\ vtpRPrepared \subseteq (H \cup C)$
$\land\ msgs \subseteq Messages$

$VTPInit \triangleq$

The initial predicate.

$\land\ vState = [v \in V \mapsto \text{"valid"}]$
$\land\ vlcState = [v \in V \mapsto \text{"working"}]$
$\land\ hState = [h \in H \mapsto \text{"holding"}]$
$\land\ cState = [c \in C \mapsto \text{"waiting"}]$
$\land\ vtpState = \text{"init"}$
$\land\ vtpRPrepared \quad = \{\}$
$\land\ msgs = \{\}$

We now define the actions that may be performed by the processes, first the VTP's actions, the Hs' actions, then the Cs' actions.

$VTPRcvPrepared(h,\ c) \triangleq$

The VTP receives a "Prepared" message from Voucher Holder h and the Voucher Collector c. We could add the additional enabling condition h,$c \notin vtpRPrepared$, which disables the action if the VTP has already received this message. But there is no need, because in that case the action has no effect; it leaves the state unchanged.

$\land\ vState = [v \in V \mapsto \text{"valid"}]$
$\land\ vlcState = [v \in V \mapsto \text{"working"}]$
$\land\ vtpState = \text{"init"}$
$\land\ [type \mapsto \text{"Prepared"},\ vh \mapsto h] \in msgs$
$\land\ [type \mapsto \text{"Prepared"},\ vc \mapsto c] \in msgs$
$\land\ vtpRPrepared' = vtpRPrepared \cup \{h,\ c\}$
$\land$ UNCHANGED $\langle vState,\ vlcState,\ hState,\ cState,\ vtpState,\ msgs\rangle$

$VTPRedeem(v) \triangleq$

The VTP Redeems the voucher; enabled iff the VTP is in its initial state and every H and C has sent a "Prepared" message.

$\land\ vState[v] = \text{“valid”}$
$\land\ vlcState[v] = \text{“working”}$
$\land\ vtpState = \text{“init”}$
$\land\ vtpRPrepared = H \cup C$
$\land\ vtpState' = \text{“done”}$
$\land\ vState' = [vState \text{ EXCEPT } ![v] = \text{“redeemed”}]$
$\land\ vlcState' = [vlcState \text{ EXCEPT } ![v] = \text{“done”}]$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{“Redeem”}]\}$
$\land$ UNCHANGED $\langle hState,\ cState,\ vtpRPrepared\rangle$

$VTPAbort(v) \triangleq$

The *VTP* spontaneously aborts the transaction.

$\land\ vState[v] = \text{“valid”}$
$\land\ vlcState[v] = \text{“working”}$
$\land\ vtpState = \text{“init”}$
$\land\ vtpState' = \text{“done”}$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{“Abort”}]\}$
$\land$ UNCHANGED $\langle vState,\ vlcState,\ hState,\ cState,\ vtpRPrepared\rangle$

$HPrepare(h) \triangleq$

Voucher holder h prepares.

$\land\ vState = [v \in V \mapsto \text{“valid”}]$
$\land\ vlcState = [v \in V \mapsto \text{“working”}]$
$\land\ hState[h] = \text{“holding”}$
$\land\ hState' = [hState \text{ EXCEPT } ![h] = \text{“prepared”}]$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{“Prepared”},\ vh \mapsto h]\}$
$\land$ UNCHANGED $\langle vState,\ vlcState,\ vtpState,\ cState,\ vtpRPrepared\rangle$

$HChooseToAbort(h) \triangleq$

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

$\land\ vState = [v \in V \mapsto \text{“valid”}]$
$\land\ vlcState = [v \in V \mapsto \text{“working”}]$
$\land\ hState[h] = \text{“holding”}$
$\land\ hState' = [hState \text{ EXCEPT } ![h] = \text{“aborted”}]$
$\land$ UNCHANGED $\langle vState,\ vlcState,\ vtpState,\ cState,\ vtpRPrepared,\ msgs\rangle$

$HRcvRedeemMsg(h) \triangleq$

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

$\land\ vState \in [V \to \{\text{“valid”},\ \text{“redeemed”}\}]$
$\land\ vlcState \in [V \to \{\text{“working”},\ \text{“done”}\}]$
$\land\ hState[h] = \text{“holding”}$
$\land\ [type \mapsto \text{“Redeem”}] \in msgs$
$\land\ hState' = [hState \text{ EXCEPT } ![h] = \text{“redeemed”}]$
$\land$ UNCHANGED $\langle vtpState,\ vState,\ vlcState,\ cState,\ vtpRPrepared,\ msgs\rangle$

$HRcvAbortMsg(h) \triangleq$

Voucher holder h is told by the VTP to abort.

$\land\ vState = [v \in V \mapsto \text{“valid”}]$
$\land\ vlcState = [v \in V \mapsto \text{“working”}]$
$\land\ hState[h] = \text{“holding”}$
$\land\ [type \mapsto \text{“Abort”}] \in msgs$
$\land\ hState' = [hState \text{ EXCEPT } ![h] = \text{“aborted”}]$
$\land$ UNCHANGED $\langle vState, vlcState, vtpState, cState, vtpRPrepared, msgs\rangle$

$CPrepare(c) \triangleq$

Voucher collector c prepares.

$\land\ vState = [v \in V \mapsto \text{“valid”}]$
$\land\ vlcState = [v \in V \mapsto \text{“working”}]$
$\land\ cState[c] = \text{“waiting”}$
$\land\ cState' = [cState \text{ EXCEPT } ![c] = \text{“prepared”}]$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{“Prepared”}, vc \mapsto c]\}$
$\land$ UNCHANGED $\langle vState, vlcState, vtpState, hState, vtpRPrepared\rangle$

$CChooseToAbort(c) \triangleq$

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

$\land\ vState = [v \in V \mapsto \text{“valid”}]$
$\land\ vlcState = [v \in V \mapsto \text{“working”}]$
$\land\ cState[c] = \text{“waiting”}$
$\land\ cState' = [cState \text{ EXCEPT } ![c] = \text{“aborted”}]$
$\land$ UNCHANGED $\langle vState, vlcState, vtpState, hState, vtpRPrepared, msgs\rangle$

$CRcvRedeemMsg(c) \triangleq$

Voucher collector c is told by the VTP to *Redeem*.

$\land\ vState \in [V \to \{\text{“valid”}, \text{“redeemed”}\}]$
$\land\ vlcState \in [V \to \{\text{“working”}, \text{“done”}\}]$
$\land\ cState[c] = \text{“waiting”}$
$\land\ [type \mapsto \text{“Redeem”}] \in msgs$
$\land\ cState' = [cState \text{ EXCEPT } ![c] = \text{“redeemed”}]$
$\land$ UNCHANGED $\langle vtpState, vState, vlcState, hState, vtpRPrepared, msgs\rangle$

$CRcvAbortMsg(c) \triangleq$

Voucher collector c is told by the VTP to abort.

$\land\ vState = [v \in V \mapsto \text{“valid”}]$
$\land\ vlcState = [v \in V \mapsto \text{“working”}]$
$\land\ cState[c] = \text{“waiting”}$
$\land\ [type \mapsto \text{“Abort”}] \in msgs$
$\land\ cState' = [cState \text{ EXCEPT } ![c] = \text{“aborted”}]$
$\land$ UNCHANGED $\langle vState, vlcState, vtpState, hState, vtpRPrepared, msgs\rangle$

$VTPNext \triangleq$
$\lor \exists\, v \in V :$
$VTPRedeem(v) \lor VTPAbort(v)$
$\lor \exists\, h,\ c \in H \cup C :$
$VTPRcvPrepared(h,\ c)$
$\lor \exists\, h \in H :$
$HPrepare(h) \lor HChooseToAbort(h)$
$\lor HRcvAbortMsg(h) \lor HRcvRedeemMsg(h)$
$\lor \exists\, c \in C :$
$CPrepare(c) \lor CChooseToAbort(c)$
$\lor CRcvAbortMsg(c) \lor CRcvRedeemMsg(c)$

$VTPConsistent \triangleq$

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

$$\land \forall\, h \in H,\ c \in C : \quad \begin{array}{r} \land \neg \land\ hState[h] = \text{“redeemed”} \\ \land\ cState[c] = \text{“aborted”} \\ \land \neg \land\ hState[h] = \text{“aborted”} \\ \land\ cState[c] = \text{“redeemed”} \end{array}$$

$VTPVars \triangleq \langle hState,\ cState,\ vState,\ vlcState,\ vtpState,\ vtpRPrepared,\ msgs \rangle$

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

The complete spec of the a Voucher *Redeem* 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 *Redeem* 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 13:35:49 *IST* 2018 by Fox
* Created *Fri Mar* 16 17:45:37 *SGT* 2018 by Fox