

```

\ * Copyright (c) 2018, Backyard Innovations Pte. Ltd., Singapore.
\ *
\ * Released under the terms of the Apache License 2.0
\ * See: file LICENSE that came with this software for details.
\ *
\ * This file contains Intellectual Property that belongs to
\ * Backyard Innovations Pte Ltd., Singapore.
\ *
\ * 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

<i>V</i> ,	The set of Vouchers
<i>H</i> ,	The set of Voucher Holders
<i>C</i>	The set of Voucher Collectors

##### VARIABLES

<i>vState</i> ,	<i>vState</i> [ <i>v</i> ] is the state of voucher <i>v</i> .
<i>vlcState</i> ,	<i>vlcState</i> [ <i>v</i> ] is the state of the voucher life cycle machine.
<i>hState</i> ,	<i>hState</i> [ <i>h</i> ] is the state of voucher holder <i>h</i> .
<i>cState</i> ,	<i>cState</i> [ <i>c</i> ] is the state of voucher collector <i>c</i> .
<i>vtpState</i> ,	The state of the voucher transaction provider.
<i>vtpRPrepared</i> ,	The set of <i>Hs</i> and <i>Cs</i> from which the <i>VTP</i> has received “Prepared for Voucher <i>Redeem</i> ” 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  $VTP$ s, to be received by all  $H$ s and  $C$ s. 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

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

$VTPInit \triangleq$

The initial predicate.

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

---

We now define the actions that may be performed by the processes, first the  $VTP$ ’s actions, the  $H$ s’ actions, then the  $C$ s’ 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.

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

$VTPRedeem(v) \triangleq$

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

$$\begin{aligned}
&\wedge vState[v] = \text{"valid"} \\
&\wedge vlcState[v] = \text{"working"} \\
&\wedge vtpState = \text{"init"} \\
&\wedge vtpRPrepared = H \cup C \\
&\wedge vtpState' = \text{"done"} \\
&\wedge vState' = [vState \text{ EXCEPT } ![v] = \text{"redeemed"}] \\
&\wedge vlcState' = [vlcState \text{ EXCEPT } ![v] = \text{"done"}] \\
&\wedge msgs' = msgs \cup \{[type \mapsto \text{"Redeem"}]\} \\
&\wedge \text{UNCHANGED } \langle hState, cState, vtpRPrepared \rangle
\end{aligned}$$

$VTPAbort(v) \triangleq$

The *VTP* spontaneously aborts the transaction.

$$\begin{aligned}
&\wedge vState[v] = \text{"valid"} \\
&\wedge vlcState[v] = \text{"working"} \\
&\wedge vtpState = \text{"init"} \\
&\wedge vtpState' = \text{"done"} \\
&\wedge msgs' = msgs \cup \{[type \mapsto \text{"Abort"}]\} \\
&\wedge \text{UNCHANGED } \langle vState, vlcState, hState, cState, vtpRPrepared \rangle
\end{aligned}$$

$HPrepare(h) \triangleq$

Voucher holder  $h$  prepares.

$$\begin{aligned}
&\wedge vState = [v \in V \mapsto \text{"valid"}] \\
&\wedge vlcState = [v \in V \mapsto \text{"working"}] \\
&\wedge hState[h] = \text{"holding"} \\
&\wedge hState' = [hState \text{ EXCEPT } ![h] = \text{"prepared"}] \\
&\wedge msgs' = msgs \cup \{[type \mapsto \text{"Prepared"}, vh \mapsto h]\} \\
&\wedge \text{UNCHANGED } \langle vState, vlcState, vtpState, cState, vtpRPrepared \rangle
\end{aligned}$$

$HChooseToAbort(h) \triangleq$

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

$$\begin{aligned}
&\wedge vState = [v \in V \mapsto \text{"valid"}] \\
&\wedge vlcState = [v \in V \mapsto \text{"working"}] \\
&\wedge hState[h] = \text{"holding"} \\
&\wedge hState' = [hState \text{ EXCEPT } ![h] = \text{"aborted"}] \\
&\wedge \text{UNCHANGED } \langle vState, vlcState, vtpState, cState, vtpRPrepared, msgs \rangle
\end{aligned}$$

$HRcvRedeemMsg(h) \triangleq$

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

$$\begin{aligned}
&\wedge vState \in [V \rightarrow \{\text{"valid"}, \text{"redeemed"}\}] \\
&\wedge vlcState \in [V \rightarrow \{\text{"working"}, \text{"done"}\}] \\
&\wedge hState[h] = \text{"holding"} \\
&\wedge [type \mapsto \text{"Redeem"}] \in msgs \\
&\wedge hState' = [hState \text{ EXCEPT } ![h] = \text{"redeemed"}] \\
&\wedge \text{UNCHANGED } \langle vtpState, vState, vlcState, cState, vtpRPrepared, msgs \rangle
\end{aligned}$$

$HRcvAbortMsg(h) \triangleq$

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

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

$CPrepare(c) \triangleq$

Voucher collector  $c$  prepares.

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

$CChooseToAbort(c) \triangleq$

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

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

$CRcvRedeemMsg(c) \triangleq$

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

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

$CRcvAbortMsg(c) \triangleq$

Voucher collector  $c$  is told by the *VTP* to abort.

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

$$\begin{aligned}
VTPNext &\triangleq \\
&\vee \exists v \in V : \\
&\quad VTPRedeem(v) \vee VTPAbort(v) \\
&\vee \exists h, c \in H \cup C : \\
&\quad VTPRcvPrepared(h, c) \\
&\vee \exists h \in H : \\
&\quad HPrepare(h) \vee HChooseToAbort(h) \\
&\quad \vee HRcvAbortMsg(h) \vee HRcvRedeemMsg(h) \\
&\vee \exists c \in C : \\
&\quad CPrepare(c) \vee CChooseToAbort(c) \\
&\quad \vee CRcvAbortMsg(c) \vee CRcvRedeemMsg(c)
\end{aligned}$$


---

$VTPConsistent \triangleq$

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

$$\begin{aligned}
&\wedge \forall h \in H, c \in C : \quad \wedge \neg \wedge hState[h] = \text{"redeemed"} \\
&\quad \wedge cState[c] = \text{"aborted"} \\
&\quad \wedge \neg \wedge hState[h] = \text{"aborted"} \\
&\quad \wedge cState[c] = \text{"redeemed"}
\end{aligned}$$


---

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

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

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

THEOREM  $VTPSpec \Rightarrow \Box(VTPTtypeOK \wedge VTPConsistent)$

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