

```

\ * Copyright (c) 2018, Backyard Innovations Pte. Ltd., Singapore.
\ *
\ * Released under the terms of the Apache License 2.0
\ * See: file LICENSE in root directory 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 *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 <i>VTP</i> has received “Prepared for Voucher <i>Cancel</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 I indicated by message’s vi field to the VTP . Messages of type “Cancel” and “Abort” are broadcast by the VTP s, to be received by all H s and I s. The set $msgs$ contains just a single copy of such a message.

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

$VTPTypeOK \triangleq$

The type-correctness invariant

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

$VTPInit \triangleq$

The initial predicate.

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

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

$VTPRcvPrepared(h, i) \triangleq$

The VTP receives a “Prepared” message from Voucher Holder h and the Voucher Issuer i . We could add the additional enabling condition $h, i \notin vtpCPrepared$, 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.

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

$VTPCancel(v) \triangleq$

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

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

$VTPAbort(v) \triangleq$

The *VTP* spontaneously aborts the transaction.

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

$HPrepare(h) \triangleq$

Voucher holder h prepares.

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

$HChooseToAbort(h) \triangleq$

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

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

$HRcvCancelMsg(h) \triangleq$

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

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

$HRcvAbortMsg(h) \triangleq$

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

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

$IPrepare(i) \triangleq$

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*.

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

$IRcvAbortMsg(i) \triangleq$

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

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

$$\begin{aligned}
VTPNext &\triangleq \\
&\vee \exists v \in V : \\
&\quad VTPCancel(v) \vee VTPAbort(v) \\
&\vee \exists h, i \in H \cup I : \\
&\quad VTPRcvPrepared(h, i) \\
&\vee \exists h \in H : \\
&\quad HPrepare(h) \vee HChooseToAbort(h) \\
&\quad \vee HRcvAbortMsg(h) \vee HRcvCancelMsg(h) \\
&\vee \exists i \in I : \\
&\quad IPrepare(i) \vee IChooseToAbort(i) \\
&\quad \vee IRcvAbortMsg(i) \vee IRcvCancelMsg(i)
\end{aligned}$$

$$VTPConsistent \triangleq$$

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

$$\begin{aligned}
&\wedge \forall h \in H, i \in I : \quad \wedge \neg \wedge hState[h] = \text{"cancelled"} \\
&\quad \wedge iState[i] = \text{"aborted"} \\
&\quad \wedge \neg \wedge hState[h] = \text{"aborted"} \\
&\quad \wedge iState[i] = \text{"cancelled"}
\end{aligned}$$

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

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

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

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

This theorem asserts the truth of the temporal formula whose meaning is that the state predicate $VTPTYPEOK \wedge VTPConsistent$ 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