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

MODULE *VoucherIssue*

The description is based on the "Issue" operation mentioned in *RFC* 3506. This specification describes the issue 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 issue 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: We use the "phantom" state of a voucher before issuing a voucher. Once the voucher is issued it goes to "valid" state.

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.
vtpIPrepared, The set of *Hs* and Is from which the *VTP* has received "Prepared for Voucher *Issue*" 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 vc field to the *VTP*. Messages of type “Issue” and “Abort” are broadcast by the *VTPs*, to be received by all *Hs* and Is. The set *msgs* contains just a single copy of such a message.

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

$VTPTypeOK\ \triangleq$

The type-correctness invariant

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

$VTPInit\ \triangleq$

The initial predicate.

$\land\ vState = [v \in\ V \mapsto \text{“phantom”}]$
$\land\ vlcState = [v \in\ V \mapsto \text{“init”}]$
$\land\ hState = [h \in H \mapsto \text{“waiting”}]$
$\land\ iState\ = [i\ \in I\ \ \mapsto \text{“waiting”}]$
$\land\ vtpState = \text{“init”}$
$\land\ vtpIPrepared\ \ \ = \{\}$
$\land\ msgs = \{\}$

We now define the actions that may be performed by the processes, first the *VTP*’s actions, the *Hs′* actions, then the Is’ 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,$i \notin vtpIPrepared$, 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.

$\land\ vState = [v \in\ V \mapsto \text{“phantom”}]$
$\land\ vlcState\ = [v \in\ V \mapsto \text{“init”}]$
$\land\ vtpState = \text{“init”}$
$\land\ [type \mapsto \text{“Prepared”},\ vh \mapsto h] \in msgs$
$\land\ [type \mapsto \text{“Prepared”},\ vi\ \mapsto i]\ \in msgs$
$\land\ vtpIPrepared' = vtpIPrepared \cup \{h,\ i\}$
$\land\ \textsc{unchanged}\ \langle vState,\ vlcState,\ hState,\ iState,\ vtpState,\ msgs\rangle$

$VTPIssue(v)\ \triangleq$

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

$\land\ vState[v] = \text{“phantom”}$
$\land\ vlcState[v] = \text{“init”}$
$\land\ vtpState = \text{“init”}$
$\land\ vtpIPrepared = H \cup I$
$\land\ vtpState' = \text{“done”}$
$\land\ vState' = [vState \text{ EXCEPT } ![v] = \text{“valid”}]$
$\land\ vlcState' = [vState \text{ EXCEPT } ![v] = \text{“working”}]$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{“Issue”}]\}$
$\land\ \text{UNCHANGED } \langle hState, iState, vtpIPrepared \rangle$

$VTPAbort(v) \triangleq$

The *VTP* spontaneously aborts the transaction.

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

$HPrepare(h) \triangleq$

Voucher holder *h* prepares.

$\land\ vState = [v \in V \mapsto \text{“phantom”}]$
$\land\ vlcState = [v \in V \mapsto \text{“init”}]$
$\land\ hState[h] = \text{“waiting”}$
$\land\ hState' = [hState \text{ EXCEPT } ![h] = \text{“prepared”}]$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{“Prepared”}, vh \mapsto h]\}$
$\land\ \text{UNCHANGED } \langle vState, vlcState, vtpState, iState, vtpIPrepared \rangle$

$HChooseToAbort(h) \triangleq$

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

$\land\ vState = [v \in V \mapsto \text{“phantom”}]$
$\land\ vlcState = [v \in V \mapsto \text{“init”}]$
$\land\ hState[h] = \text{“waiting”}$
$\land\ hState' = [hState \text{ EXCEPT } ![h] = \text{“aborted”}]$
$\land\ \text{UNCHANGED } \langle vState, vlcState, vtpState, iState, vtpIPrepared, msgs \rangle$

$HRcvIssueMsg(h) \triangleq$

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

$\land\ vState \in [V \rightarrow \{\text{“phantom”}, \text{“valid”}\}]$
$\land\ vlcState \in [V \rightarrow \{\text{“init”}, \text{“working”}\}]$
$\land\ hState[h] = \text{“waiting”}$
$\land\ [type \mapsto \text{“Issue”}] \in msgs$
$\land\ hState' = [hState \text{ EXCEPT } ![h] = \text{“holding”}]$
$\land\ \text{UNCHANGED } \langle vtpState, vState, vlcState, iState, vtpIPrepared, msgs \rangle$

$HRcvAbortMsg(h) \triangleq$

Voucher holder h is told by the VTP to abort.

$\land\ vState = [v \in V \mapsto \text{“phantom”}]$
$\land\ vlcState = [v \in V \mapsto \text{“init”}]$
$\land\ hState[h] = \text{“waiting”}$
$\land\ [type \mapsto \text{“Abort”}] \in msgs$
$\land\ hState' = [hState\ \textsc{except}\ ![h] = \text{“aborted”}]$
$\land\ \textsc{unchanged}\ \langle vState, vlcState, vtpState, iState, vtpIPrepared, msgs\rangle$

$IPrepare(i) \triangleq$

Voucher issuer i prepares.

$\land\ vState = [v \in V \mapsto \text{“phantom”}]$
$\land\ vlcState = [v \in V \mapsto \text{“init”}]$
$\land\ iState[i] = \text{“waiting”}$
$\land\ iState' = [iState\ \textsc{except}\ ![i] = \text{“prepared”}]$
$\land\ msgs' = msgs \cup \{[type \mapsto \text{“Prepared”}, vi \mapsto i]\}$
$\land\ \textsc{unchanged}\ \langle vState, vlcState, vtpState, hState, vtpIPrepared\rangle$

$IChooseToAbort(i) \triangleq$

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

$\land\ vState = [v \in V \mapsto \text{“phantom”}]$
$\land\ vlcState = [v \in V \mapsto \text{“init”}]$
$\land\ iState[i] = \text{“waiting”}$
$\land\ iState' = [iState\ \textsc{except}\ ![i] = \text{“aborted”}]$
$\land\ \textsc{unchanged}\ \langle vState, vlcState, vtpState, hState, vtpIPrepared, msgs\rangle$

$IRcvIssueMsg(i) \triangleq$

Voucher issuer i is told by the VTP to $Issue$.

$\land\ vState \in [V \rightarrow \{\text{“phantom”}, \text{“valid”}\}]$
$\land\ vlcState \in [V \rightarrow \{\text{“init”}, \text{“working”}\}]$
$\land\ iState[i] = \text{“waiting”}$
$\land\ [type \mapsto \text{“Issue”}] \in msgs$
$\land\ iState' = [iState\ \textsc{except}\ ![i] = \text{“issued”}]$
$\land\ \textsc{unchanged}\ \langle vtpState, vState, vlcState, hState, vtpIPrepared, msgs\rangle$

$IRcvAbortMsg(i) \triangleq$

Voucher issuer i is told by the VTP to abort.

$\land\ vState = [v \in V \mapsto \text{“phantom”}]$
$\land\ vlcState = [v \in V \mapsto \text{“init”}]$
$\land\ iState[i] = \text{“waiting”}$
$\land\ [type \mapsto \text{“Abort”}] \in msgs$
$\land\ iState' = [iState\ \textsc{except}\ ![i] = \text{“aborted”}]$
$\land\ \textsc{unchanged}\ \langle vState, vlcState, vtpState, hState, vtpIPrepared, msgs\rangle$

$$
\begin{array}{l}
\mathit{VTPNext} \triangleq \\
\quad \lor \exists\, v \in V : \\
\qquad \mathit{VTPIssue}(v) \lor \mathit{VTPAbort}(v) \\
\quad \lor \exists\, h,\, i \in H \cup I : \\
\qquad \mathit{VTPRcvPrepared}(h,\, i) \\
\quad \lor \exists\, h \in H : \\
\qquad \mathit{HPrepare}(h) \lor \mathit{HChooseToAbort}(h) \\
\qquad \lor \mathit{HRcvAbortMsg}(h) \lor \mathit{HRcvIssueMsg}(h) \\
\quad \lor \exists\, i \in I : \\
\qquad \mathit{IPrepare}(i) \lor \mathit{IChooseToAbort}(i) \\
\qquad \lor \mathit{IRcvAbortMsg}(i) \lor \mathit{IRcvIssueMsg}(i)
\end{array}
$$

$$\mathit{VTPConsistent} \triangleq$$

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

$$
\begin{array}{ll}
\land \forall\, h \in H,\, i \in I : & \land \lnot \land \mathit{hState}[h] = \text{“holding”} \\
& \qquad \land \mathit{iState}[i] \;= \text{“aborted”} \\
& \land \lnot \land \mathit{hState}[h] = \text{“aborted”} \\
& \qquad \land \mathit{iState}[i] \;= \text{“issued”}
\end{array}
$$

$$\mathit{VTPVars} \triangleq \langle \mathit{hState},\, \mathit{iState},\, \mathit{vState},\, \mathit{vlcState},\, \mathit{vtpState},\, \mathit{vtpIPrepared},\, \mathit{msgs} \rangle$$

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

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

THEOREM $\mathit{VTPSpec} \Rightarrow \Box(\mathit{VTPTypeOK} \land \mathit{VTPConsistent})$

This theorem asserts the truth of the temporal formula whose meaning is that the state predicate $\mathit{VTPTypeOK} \land \mathit{VTPConsistent}$ 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 *Issue* 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 $\mathit{VTPSpec} \Rightarrow \mathit{VSpec}$

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 13:33:03 *IST* 2018 by Fox
\* Created *Fri Mar* 16 17:45:37 *SGT* 2018 by Fox