MODULE *NewSpec*

EXTENDS	*Bags*, *FiniteSets*, *Naturals*, *TLC*	
CONSTANTS	*MessageId*,	The set of all possible message identifiers that can be used in the negotiation.
	NullId,	The empty message identifier.
	MaxDuplicates	The maximum number of duplicate messages allowed on the network at any one point in time.
VARIABLES	*provContracted*, *custContracted*,	Boolean flags denoting whether the provider and customer are in a contracted state (or not).
	provSent, *custSent*,	The sets of messages sent by the provider and customer.
	provInQ, *custInQ*,	The bags of messages waiting to be received by the provider and customer.
	provRcvd, *custRcvd*,	The sets of messages received by the provider and customer.
	ackdOffer,	The set of acknowledged but unprocessed offers.
	usedId	The set of used *MessageIds*.

Definitions of the allowed messages in the protocol.

$$QuoteRequestMessage \triangleq [messageType : \{\text{“QuoteRequest”}\}, messageId : MessageId, correlationId : MessageId \cup \{NullId\}]$$

$$OfferMessage \triangleq [messageType : \{\text{“Offer”}\}, messageId : MessageId, correlationId : MessageId \cup \{NullId\}]$$

$$RevokeRequestMessage \triangleq [messageType : \{\text{“RevokeRequest”}\}, messageId : MessageId, correlationId : MessageId \cup \{NullId\}]$$

$$QuoteMessage \triangleq [messageType : \{\text{“Quote”}\}, messageId : MessageId, correlationId : MessageId \cup \{NullId\}]$$

$$OfferAckMessage \triangleq [messageType : \{\text{“OfferAck”}\}, messageId : MessageId, correlationId : MessageId]$$

$$
\begin{array}{ll}
\mathit{AcceptMessage} & \triangleq [\mathit{messageType} : \{\text{“Accept”}\}, \\
& \quad \mathit{messageId} : \mathit{MessageId}, \\
& \quad \mathit{correlationId} : \mathit{MessageId}] \\
\mathit{RejectMessage} & \triangleq [\mathit{messageType} : \{\text{“Reject”}\}, \\
& \quad \mathit{messageId} : \mathit{MessageId}, \\
& \quad \mathit{correlationId} : \mathit{MessageId}] \\
\mathit{RevokeAcceptMessage} & \triangleq [\mathit{messageType} : \{\text{“RevokeAccept”}\}, \\
& \quad \mathit{messageId} : \mathit{MessageId}, \\
& \quad \mathit{correlationId} : \mathit{MessageId}] \\
\mathit{ProvMessage} & \triangleq \text{UNION } \{\mathit{QuoteMessage}, \mathit{OfferAckMessage}, \\
& \qquad \mathit{AcceptMessage}, \mathit{RejectMessage}, \\
& \qquad \mathit{RevokeAcceptMessage}\} \\
\mathit{CustMessage} & \triangleq \text{UNION } \{\mathit{QuoteRequestMessage}, \mathit{OfferMessage}, \\
& \qquad \mathit{RevokeRequestMessage}\}
\end{array}
$$

A function to return a message identifier that hasn't been used before in the negotiation. This ensures that each new message has a unique identifier.

$$
\mathit{NewId} \triangleq \text{CHOOSE } \mathit{id} \in \mathit{MessageId} : \mathit{id} \notin \mathit{usedId}
$$

Definition of the messaging channels. $p2c$ is provider to customer, $c2p$ is customer to provider.

$$
\begin{array}{ll}
c2p & \triangleq \text{INSTANCE } \mathit{ChannelB} \text{ WITH } \mathit{sent} \leftarrow \mathit{custSent}, \\
& \qquad \mathit{received} \leftarrow \mathit{provRcvd}, \\
& \qquad \mathit{network} \leftarrow \mathit{provInQ}, \\
& \qquad \mathit{Message} \leftarrow \mathit{CustMessage}, \\
& \qquad \mathit{MaxDuplicates} \leftarrow \mathit{MaxDuplicates} \\
p2c & \triangleq \text{INSTANCE } \mathit{ChannelB} \text{ WITH } \mathit{sent} \leftarrow \mathit{provSent}, \\
& \qquad \mathit{received} \leftarrow \mathit{custRcvd}, \\
& \qquad \mathit{network} \leftarrow \mathit{custInQ}, \\
& \qquad \mathit{Message} \leftarrow \mathit{ProvMessage}, \\
& \qquad \mathit{MaxDuplicates} \leftarrow \mathit{MaxDuplicates}
\end{array}
$$

PROVIDER SERVICE SPECIFICATION.

A function to check if a type of message has been sent in response to a particular offer.

$$
\begin{array}{l}
\mathit{AlreadySent}(\mathit{offer}, \mathit{type}) \triangleq \\
\qquad \land \{\mathit{msg} \in \mathit{BagToSet}(\mathit{provSent}) \cap \mathit{type} :
\end{array}
$$

$$offer.messageId = msg.correlationId\} \neq \{\}$$

Message resend actions for the provider.

$$\begin{array}{l}
ResendRevokeAccept(offer) \triangleq \\
\quad \text{LET } matches \triangleq \{msg \in BagToSet(provSent) \\
\qquad \cap\, RevokeAcceptMessage : \\
\qquad\quad offer.messageId = msg.correlationId\} \\
\quad \text{IN} \quad \wedge \text{LET } revokeAccept \triangleq \text{CHOOSE } msg \in matches : \\
\qquad\quad msg.messageType = \text{“RevokeAccept”} \\
\qquad \text{IN} \quad \wedge\, p2c!Send(revokeAccept) \\
\qquad\quad \wedge \text{UNCHANGED } \langle provContracted,\, provRcvd,\, provInQ, \\
\qquad\qquad\qquad\qquad custContracted,\, custSent, \\
\qquad\qquad\qquad\qquad custRcvd,\, usedId\rangle
\end{array}$$

$$\begin{array}{l}
ResendReject(offer) \triangleq \\
\quad \text{LET } matches \triangleq \{msg \in BagToSet(provSent) \\
\qquad \cap\, RejectMessage : \\
\qquad\quad offer.messageId = msg.correlationId\} \\
\quad \text{IN} \quad \wedge \text{LET } reject \triangleq \text{CHOOSE } msg \in matches : \\
\qquad\quad msg.messageType = \text{“Reject”} \\
\qquad \text{IN} \quad \wedge\, p2c!Send(reject) \\
\qquad\quad \wedge \text{UNCHANGED } \langle provContracted,\, ackdOffer, \\
\qquad\qquad\qquad\qquad custContracted,\, custSent, \\
\qquad\qquad\qquad\qquad custRcvd,\, usedId\rangle
\end{array}$$

$$\begin{array}{l}
ResendAccept \triangleq \text{LET } accept \triangleq \text{CHOOSE } msg \in BagToSet(provSent) \\
\qquad\qquad \cap\, AcceptMessage : msg.messageType = \text{“Accept”} \\
\qquad \text{IN} \quad \wedge\, p2c!Send(accept) \\
\qquad\quad \wedge \text{UNCHANGED } \langle provSent,\, provContracted, \\
\qquad\qquad\qquad\qquad provInQ,\, provRcvd,\, ackdOffer, \\
\qquad\qquad\qquad\qquad custContracted,\, custSent, \\
\qquad\qquad\qquad\qquad custRcvd,\, usedId\rangle
\end{array}$$

Definition of the *SendQuote* action. The first action sends a *Quote* message without correlation to a received message. That is, the correlation identifier is null. The second action sends a Quote in response to a *QuoteRequest* message already received. In this case the correlation identifier references the message received.

$$\begin{array}{l}
SendQuote1 \triangleq \text{LET } id \triangleq NewId \\
\qquad \text{IN} \quad \wedge\, p2c!Send([messageType \mapsto \text{“Quote”}, \\
\qquad\qquad\qquad\qquad\quad messageId \mapsto id, \\
\qquad\qquad\qquad\qquad\quad correlationId \mapsto NullId]) \\
\qquad\quad \wedge\, usedId' = usedId \cup \{id\} \\
\qquad\quad \wedge \text{UNCHANGED } \langle custSent,\, custRcvd,\, custContracted, \\
\qquad\qquad\qquad\qquad ackdOffer,\, provRcvd,\, provInQ, \\
\qquad\qquad\qquad\qquad provContracted\rangle
\end{array}$$

$$
\begin{array}{lll}
SendQuote2 & \triangleq & \exists\, message \in BagToSet(provInQ) \cap QuoteRequestMessage : \\
& & \quad \land\ c2p!Receive(message) \\
& & \quad \land\ \text{LET}\ id \triangleq NewId \\
& & \qquad \text{IN} \quad \land\ p2c!Send([messageType \mapsto \text{“Quote”}, \\
& & \qquad\qquad\qquad\qquad messageId \mapsto id, \\
& & \qquad\qquad\qquad\qquad correlationId \mapsto message.messageId]) \\
& & \qquad\qquad \land\ usedId' = usedId \cup \{id\} \\
& & \qquad\qquad \land\ \text{UNCHANGED}\ \langle custSent,\ custRcvd,\ custContracted, \\
& & \qquad\qquad\qquad\qquad ackdOffer,\ provContracted\rangle
\end{array}
$$

The full *SendQuote* action is composed of the two specifications above. The action is only enabled if the provider is not contracted.

$$
\begin{array}{lll}
SendQuote & \triangleq & \land\ \neg provContracted \\
& & \land\ (SendQuote1 \lor SendQuote2)
\end{array}
$$

Definition of the *SendOfferAck* action. The action checks for an offer in the inbound network channel. If an offer is present, we receive the offer. We then attempt to find the offer acknowledgement with the same correlation identifier as the message identifier of the offer. If an acknowledgement exists then we resend it. If no acknowledgement exists we send a new acknowledgement using the *AckOffer* action (the *AckOffer* action creates and sends a new offer acknowledgement, remembering to add its message identifier to the set of used identifiers used so that it cannot be reused). The action then checks to see what state the provider is in. If the provider is “not contracted” the offer is added to the set of offers waiting to be processed, otherwise the original accept message that was used to form the contract is resent.

$$
\begin{array}{lll}
ProcessOffer(offer) & \triangleq & \\
& & \text{IF}\ provContracted \\
& & \text{THEN}\ ResendAccept \\
& & \text{ELSE}\ ackdOffer' = ackdOffer \cup \{offer\}
\end{array}
$$

$$
\begin{array}{lll}
AckOffer(offer) & \triangleq & \text{LET}\ id \triangleq NewId \\
& & \text{IN} \quad \land\ p2c!Send([messageType \mapsto \text{“OfferAck”}, \\
& & \qquad\qquad\qquad\qquad messageId \mapsto id, \\
& & \qquad\qquad\qquad\qquad correlationId \mapsto offer.messageId]) \\
& & \qquad\qquad \land\ usedId' = usedId \cup \{id\}
\end{array}
$$

$$
\begin{array}{lll}
SendOfferAck & \triangleq & \land\ \exists\, offer \in BagToSet(provInQ) \cap OfferMessage : \\
& & \quad \land\ c2p!Receive(offer) \\
& & \quad \land\ \text{LET}\ offerAck \triangleq \{msg \in BagToSet(provSent) \cap \\
& & \qquad\qquad OfferAckMessage : offer.messageId = msg.correlationId\} \\
& & \qquad \text{IN} \quad \land\ \text{IF}\ offerAck = \{\} \\
& & \qquad\qquad\qquad \text{THEN}\ \land\ AckOffer(offer) \\
& & \qquad\qquad\qquad\qquad\ \land\ ProcessOffer(offer) \\
& & \qquad\qquad\qquad\qquad\ \land\ \text{UNCHANGED}\ \langle provContracted,\ custContracted, \\
& & \qquad\qquad\qquad\qquad\qquad\qquad custSent,\ custRcvd\rangle \\
& & \qquad\qquad\qquad \text{ELSE}\ \ \land\ p2c!Send(offerAck) \\
& & \qquad\qquad\qquad\qquad\ \land\ ProcessOffer(offer)
\end{array}
$$

Definition of the *SendAccept* action. The *SendAccept* action first checks to see if there is an offer in the set of offers acknowledged but not processed. If an offer exists, we remove the offer from the set of ackd offers and we attempt to resend the reject message initially sent for this offer. If *ResendReject* returns TRUE (*i.e.* a reject existed and was resent) we return, else we attempt to resend the revoke accept for this offer. If *ResendRevokeRequest* returns TRUE (*i.e.* a revoke accept existed and was resent) we return, otherwise we accept the offer using the *AcceptOffer* action.

$$
\begin{array}{l}
AcceptOffer(offer) \triangleq \\
\quad \text{LET } id \triangleq NewId \\
\quad \text{IN} \quad \wedge\, p2c!Send([messageType \mapsto \text{“Accept”}, \\
\qquad\qquad\qquad\qquad messageId \mapsto id, \\
\qquad\qquad\qquad\qquad correlationId \mapsto offer.messageId]) \\
\qquad\quad \wedge\, usedId' = usedId \cup \{id\} \\
\qquad\quad \wedge\, provContracted' = \text{TRUE} \\
\qquad\quad \wedge\, \text{UNCHANGED } \langle provInQ,\, provRcvd, \\
\qquad\qquad\qquad\qquad\qquad custContracted,\, custSent, \\
\qquad\qquad\qquad\qquad\qquad custRcvd\rangle
\end{array}
$$

$$
\begin{array}{lll}
SendAccept & \triangleq & \exists\, offer \in ackdOffer : \\
& & \quad \wedge\, ackdOffer' = ackdOffer \setminus \{offer\} \\
& & \quad \wedge\, \text{IF } AlreadySent(offer,\, RejectMessage) \\
& & \qquad \text{THEN } ResendReject(offer) \\
& & \qquad \text{ELSE} \quad \wedge\, \text{IF } AlreadySent(offer,\, RevokeAcceptMessage) \\
& & \qquad\qquad\qquad \text{THEN} \quad \wedge\, ResendRevokeAccept(offer) \\
& & \qquad\qquad\qquad \text{ELSE} \quad \wedge\, AcceptOffer(offer) \\
& & \qquad\qquad\qquad\qquad\quad \wedge\, ackdOffer' = \{\}
\end{array}
$$

Definition of the *SendReject* action. The *SendReject* action first checks to see if there is an offer in the set of offers acknowledged but not processed. If an offer exists, we remove the offer from the set of ackd offers and we attempt to resend the reject message initially sent for this offer. If *ResendReject* returns TRUE (*i.e.* a reject existed and was resent) we return, else we attempt to resend the revoke accept for this offer. If *ResendRevokeRequest* returns TRUE (*i.e.* a revoke accept existed and was resent) we return, else we reject the offer using the *RejectOffer* action.

$$
\begin{array}{l}
RejectOffer(offer) \triangleq \\
\quad \text{LET } id \triangleq NewId \\
\quad \text{IN} \quad \wedge\, p2c!Send([messageType \mapsto \text{“Reject”}, \\
\qquad\qquad\qquad\qquad messageId \mapsto id, \\
\qquad\qquad\qquad\qquad correlationId \mapsto offer.messageId]) \\
\qquad\quad \wedge\, usedId' = usedId \cup \{id\} \\
\qquad\quad \wedge\, \text{UNCHANGED } \langle provContracted,\, provInQ, \\
\qquad\qquad\qquad\qquad\qquad provRcvd,\, custContracted, \\
\qquad\qquad\qquad\qquad\qquad custSent,\, custRcvd\rangle
\end{array}
$$

$$
\begin{array}{lll}
SendReject & \triangleq & \exists\, offer \in ackdOffer : \\
& & \quad \wedge\, ackdOffer' = ackdOffer \setminus \{offer\} \\
& & \quad \wedge\, \text{IF } AlreadySent(offer,\, RejectMessage)
\end{array}
$$

```
                THEN ResendReject(offer)
                ELSE ∧ IF AlreadySent(offer, RevokeAcceptMessage)
                       THEN ResendRevokeAccept(offer)
                       ELSE RejectOffer(offer)
```

Definition of the *SendRevokeAccept* action.

```
RevokeOffer(offer) ≜
    LET id ≜ NewId
    IN  ∧ p2c!Send([messageType ↦ "RevokeAccept",
                    messageId ↦ id,
                    correlationId ↦ offer.messageId])
        ∧ usedId' = usedId ∪ {id}
        ∧ UNCHANGED ⟨provContracted, custContracted,
                     custSent, custRcvd,
                     ackdOffer⟩

SendRevokeAccept ≜ ∃ revokeRequest ∈ BagToSet(provInQ) ∩
                     RevokeRequestMessage :
                   ∧ LET match ≜ {msg ∈ provRcvd ∩ OfferMessage :
                                  msg.messageId = revokeRequest.correlationId}
                     IN  ∧ match ≠ {}  Have received the offer the customer wants to revoke
                         ∧ c2p!Receive(revokeRequest)
                         ∧ LET offer ≜ CHOOSE msg ∈ match : msg.messageType = "Offer"
                           IN  ∧ IF AlreadySent(offer, RejectMessage)
                                 THEN ResendReject(offer)
                                 ELSE ∧ IF AlreadySent(offer, RevokeAcceptMessage)
                                        THEN ResendRevokeAccept(offer)
                                        ELSE RevokeOffer(offer)
```

Definition of the *CustMessageWithNoAction*. In some circumstances when a provider receives a customer message they may wish not to take any further action. For example, when a provider receives a quote request message from the customer they are not obliged to respond to it. This action allows this case by simply receiving the message and taking no further action.

```
CustMessageWithNoAction ≜
    ∃ message ∈ BagToSet(provInQ) ∩
      QuoteRequestMessage :
      ∧ c2p!Receive(message)
      ∧ UNCHANGED ⟨provContracted, provSent,
                   custInQ, custRcvd,
                   ackdOffer, custContracted,
                   usedId, custSent⟩
```

CUSTOMER SERVICE SPECIFICATION.

Definition of the *SendQuoteRequest* action. The first action sends a *QuoteRequest* message without correlation to a received message. That is, the correlation identifier is null. The second action sends a *QuoteRequest* in response to a message waiting on the network. In this case the correlation identifier references that message.

$$
\begin{array}{lll}
SendQuoteRequest1 & \triangleq & \text{LET } id \triangleq NewId \\
 & & \text{IN} \quad \land\ c2p!Send([messageType \mapsto \text{“QuoteRequest”}, \\
 & & \qquad\qquad\qquad messageId \mapsto id, \\
 & & \qquad\qquad\qquad correlationId \mapsto NullId]) \\
 & & \qquad \land\ usedId' = usedId \cup \{id\} \\
 & & \qquad \land\ \text{UNCHANGED } \langle ackdOffer,\ provSent, \\
 & & \qquad\qquad\qquad\qquad custInQ,\ provRcvd, \\
 & & \qquad\qquad\qquad\qquad custRcvd,\ custContracted, \\
 & & \qquad\qquad\qquad\qquad provContracted\rangle
\end{array}
$$

$$
\begin{array}{lll}
SendQuoteRequest2 & \triangleq & \exists\, message \in BagToSet(custInQ) \cap \\
 & & \qquad \text{UNION } \{QuoteMessage,\ RejectMessage\} : \\
 & & \quad \land\ p2c!Receive(message) \\
 & & \quad \land\ \text{LET } id \triangleq NewId \\
 & & \qquad \text{IN} \quad \land\ c2p!Send([messageType \mapsto \text{“QuoteRequest”}, \\
 & & \qquad\qquad\qquad\qquad messageId \mapsto id, \\
 & & \qquad\qquad\qquad\qquad correlationId \mapsto message.messageId]) \\
 & & \qquad\qquad \land\ usedId' = usedId \cup \{id\} \\
 & & \qquad\qquad \land\ \text{UNCHANGED } \langle ackdOffer,\ provSent, \\
 & & \qquad\qquad\qquad\qquad\qquad provRcvd,\ custContracted, \\
 & & \qquad\qquad\qquad\qquad\qquad provContracted\rangle
\end{array}
$$

The full *SendQuoteRequest* action is composed of the two specifications above. The action is only enabled if the customer is not contracted.

$$
\begin{array}{lll}
SendQuoteRequest & \triangleq & \land\ \neg custContracted \\
 & & \land\ (SendQuoteRequest1 \lor SendQuoteRequest2)
\end{array}
$$

Definition of the *SendOffer* action. The first action sends an *Offer* message without correlation to a received message. That is, the correlation identifier is null. The second action sends an *Offer* message in response to a received *Quote* or *Reject* message. In this case the correlation identifier references that message.

$$
\begin{array}{lll}
SendOffer1 & \triangleq & \text{LET } id \triangleq NewId \\
 & & \text{IN} \quad \land\ c2p!Send([messageType \mapsto \text{“Offer”}, \\
 & & \qquad\qquad\qquad messageId \mapsto id, \\
 & & \qquad\qquad\qquad correlationId \mapsto NullId]) \\
 & & \qquad \land\ usedId' = usedId \cup \{id\} \\
 & & \qquad \land\ \text{UNCHANGED } \langle ackdOffer,\ provSent, \\
 & & \qquad\qquad\qquad\qquad custInQ,\ provRcvd, \\
 & & \qquad\qquad\qquad\qquad custRcvd,\ custContracted, \\
 & & \qquad\qquad\qquad\qquad provContracted\rangle
\end{array}
$$

$$
\begin{array}{l}
SendOffer2 \triangleq \exists\, message \in BagToSet(custInQ) \cap \\
\qquad \text{UNION } \{QuoteMessage,\ RejectMessage\} : \\
\qquad \land\ p2c!Receive(message) \\
\qquad \land\ \text{LET } id \triangleq NewId \\
\qquad\quad \text{IN} \quad \land\ c2p!Send([messageType \mapsto \text{“Offer”}, \\
\qquad\qquad\qquad\qquad\qquad messageId \mapsto id, \\
\qquad\qquad\qquad\qquad\qquad correlationId \mapsto message.messageId]) \\
\qquad\qquad\qquad \land\ usedId' = usedId \cup \{id\} \\
\qquad\qquad\qquad \land\ \text{UNCHANGED } \langle ackdOffer,\ provSent, \\
\qquad\qquad\qquad\qquad\qquad\qquad provRcvd,\ custContracted, \\
\qquad\qquad\qquad\qquad\qquad\qquad provContracted\rangle
\end{array}
$$

The full *SendOffer* action is composed of the two specifications above. The action is only enabled if the customer is not contracted.

$$
\begin{array}{l}
SendOffer \triangleq\ \land\ \neg custContracted \\
\qquad\qquad\qquad\ \land\ (SendOffer1 \lor SendOffer2)
\end{array}
$$

Definition of the *SendRevokeRequest* action. This action is only enabled if the customer is not contracted. If this is the case, for an offer in the set of sent offers we check to see if the offer is either revoked or rejected. If it is not, we use its message identifier as the correlation identifier of a new *RevokeRequest* message.

$$
\begin{array}{l}
NotRejectedOffer(offer) \triangleq \\
\qquad \land\ \{msg \in custRcvd \cap RejectMessage : \\
\qquad\quad msg.correlationId = offer.messageId\} = \{\}
\end{array}
$$

$$
\begin{array}{l}
NotRevokedOffer(offer) \triangleq \\
\qquad \land\ \{msg \in custRcvd \cap RevokeAcceptMessage : \\
\qquad\quad msg.correlationId = offer.messageId\} = \{\}
\end{array}
$$

$$
\begin{array}{l}
RequestRevoke(offer) \triangleq \\
\qquad \text{LET } match \triangleq \{msg \in BagToSet(custSent) \cap RevokeRequestMessage : \\
\qquad\quad msg.correlationId = offer.messageId\} \\
\qquad \text{IN} \quad \land\ \text{IF } match \neq \{\} \text{ Found an already sent revoke request} \\
\qquad\qquad \text{THEN } \land\ \text{LET } rr \triangleq \text{CHOOSE } msg \in match : msg.messageType = \text{“RevokeReq} \\
\qquad\qquad\qquad\quad \text{IN} \quad \land\ c2p!Send(rr) \\
\qquad\qquad\qquad\qquad\quad \land\ \text{UNCHANGED } \langle provContracted,\ provSent, \\
\qquad\qquad\qquad\qquad\qquad\qquad provRcvd,\ ackdOffer, \\
\qquad\qquad\qquad\qquad\qquad\qquad custContracted,\ custInQ, \\
\qquad\qquad\qquad\qquad\qquad\qquad custRcvd,\ usedId\rangle \\
\qquad\qquad \text{ELSE } \text{LET } id \triangleq NewId \text{ Send new revoke request} \\
\qquad\qquad\qquad\quad \text{IN} \quad \land\ c2p!Send([messageType \mapsto \text{“RevokeRequest”}, \\
\qquad\qquad\qquad\qquad\qquad\qquad messageId \mapsto id, \\
\qquad\qquad\qquad\qquad\qquad\qquad correlationId \mapsto offer.messageId]) \\
\qquad\qquad\qquad\qquad\quad \land\ usedId' = usedId \cup \{id\} \\
\qquad\qquad\qquad\qquad\quad \land\ \text{UNCHANGED } \langle provContracted,\ provSent,
\end{array}
$$

$provRcvd$, $ackdOffer$,
$custContracted$, $custInQ$,
$custRcvd\rangle$

$$
\begin{array}{lcl}
SendRevokeRequest & \triangleq & \land \neg custContracted \\
& & \land \exists\, offer \in BagToSet(custSent) \cap OfferMessage : \\
& & \quad \land NotRejectedOffer(offer) \\
& & \quad \land NotRevokedOffer(offer) \\
& & \quad \land RequestRevoke(offer)
\end{array}
$$

Definition of the *ReceiveAccept* action. If an accept message is present in the customers inbound queue of unprocessed messages, we receive the message. As a result, the customer is contracted.

$$
\begin{array}{lcl}
ReceiveAccept & \triangleq & \exists\, accept \in BagToSet(custInQ) \cap AcceptMessage : \\
& & \land p2c!Receive(accept) \\
& & \land custContracted' = \text{TRUE} \\
& & \land \text{UNCHANGED } \langle provContracted,\ provSent, \\
& & \qquad\qquad provInQ,\ provRcvd, \\
& & \qquad\qquad ackdOffer,\ custSent, \\
& & \qquad\qquad usedId\rangle
\end{array}
$$

Definition of the *ProvMessageWithNoAction*. In some circumstances a customer may receives a provider message and not wish to take any further action. For example, when a customer receives a quote they don't like they may take the choice not to pursue the negotiation further. This action allows this case by receiving the message and taking no further action.

$$
\begin{array}{lcl}
ProvMessageWithNoAction & \triangleq & \\
& & \exists\, message \in BagToSet(custInQ) \cap \\
& & \quad \text{UNION } \{QuoteMessage,\ OfferAckMessage, \\
& & \qquad\qquad RejectMessage,\ RevokeAcceptMessage\} : \\
& & \quad \land p2c!Receive(message) \\
& & \quad \land \text{UNCHANGED } \langle provContracted,\ provSent, \\
& & \qquad\qquad\qquad provInQ,\ provRcvd, \\
& & \qquad\qquad\qquad ackdOffer,\ custContracted, \\
& & \qquad\qquad\qquad custSent,\ usedId\rangle
\end{array}
$$

$$
\begin{array}{lcll}
Init & \triangleq & \land c2p!Init & \\
& & \land p2c!Init & \\
& & \land usedId = \{\} & \text{No message ids have been used.} \\
& & \land ackdOffer = \{\} & \text{No offers waiting to be processed.} \\
& & \land provContracted = \text{FALSE} & \text{Both parties are not contracted,} \\
& & \land custContracted = \text{FALSE} & i.e., \text{ no agreement has been made.}
\end{array}
$$

$$
\begin{array}{lcll}
ProvAction & \triangleq & \lor SendQuote & \text{The allowed provider actions.} \\
& & \lor SendOfferAck & \\
& & \lor SendAccept &
\end{array}
$$

$\lor$ *SendReject*
$\lor$ *SendRevokeAccept*
$\lor$ *CustMessageWithNoAction*

CustAction $\triangleq$ $\lor$ *SendQuoteRequest* — The allowed customer actions.
$\lor$ *SendOffer*
$\lor$ *ReceiveAccept*
$\lor$ *SendRevokeRequest*
$\lor$ *ProvMessageWithNoAction*

Next $\triangleq$ $\lor$ *ProvAction* — The next state is found through taking either a provider or customer action
$\lor$ *CustAction*

TypeInvariant $\triangleq$ $\land$ $IsABag(provSent)$
$\land$ $BagToSet(provSent) \subseteq ProvMessage$
$\land$ $IsABag(provInQ)$
$\land$ $BagToSet(provInQ) \subseteq CustMessage$
$\land$ $IsFiniteSet(provRcvd)$
$\land$ $provRcvd \subseteq CustMessage$
$\land$ $IsABag(custSent)$
$\land$ $BagToSet(custSent) \subseteq CustMessage$
$\land$ $IsABag(custInQ)$
$\land$ $BagToSet(custInQ) \subseteq ProvMessage$
$\land$ $IsFiniteSet(custRcvd)$
$\land$ $custRcvd \subseteq ProvMessage$
$\land$ $IsFiniteSet(ProvMessage)$
$\land$ $IsFiniteSet(CustMessage)$
$\land$ $provContracted \in$ BOOLEAN
$\land$ $custContracted \in$ BOOLEAN
$\land \forall\, msg \in provRcvd : msg \in CustMessage$
$\land \forall\, msg \in custRcvd : msg \in ProvMessage$
$\land \forall\, msg \in BagToSet(provSent) : msg \in ProvMessage$
$\land \forall\, msg \in BagToSet(custSent) : msg \in CustMessage$
$\land \forall\, msg \in BagToSet(provInQ) : msg \in CustMessage$
$\land \forall\, msg \in BagToSet(custInQ) : msg \in ProvMessage$

MsgConstraint $\triangleq$ $BagCardinality(custSent) + BagCardinality(provSent) < Cardinality(MessageId)$

Liveness $\triangleq$ $\land$ $c2p!Liveness$
$\land$ $p2c!Liveness$

Safety $\triangleq$ $\land$ $c2p!Safety$

$\land\ p2c!Safety$
$\land\ Cardinality(BagToSet(provSent) \cap AcceptMessage) \in \{0, 1\}$

Tuples of variables.

$provVars \triangleq \langle provContracted,\ provSent,\ provInQ,\ provRcvd,\ ackdOffer\rangle$
$custVars \triangleq \langle custContracted,\ custSent,\ custInQ,\ custRcvd\rangle$
$vars \triangleq \langle provVars,\ custVars,\ usedId\rangle$

$NewSpec \triangleq \land\ Init$
$\land\ \Box[Next]_{\langle vars\rangle}$

THEOREM $NewSpec \Rightarrow \Box\, TypeInvariant$
THEOREM $NewSpec \Rightarrow Liveness \land Safety$