



This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Orchestration Server Developer's Guide

Work With E-Mail Or SMS

Work With E-Mail Or SMS

The following SCXML strategy receives an inbound e-mail or SMS from a customer. It then sends an acknowledgement to the customer and calls submit to find an available agent. The e-mail or SMS is redirected to the agent and an e-mail response from the agent is sent to the customer. Any inbound e-mail delivery confirmation is also handled.

```
<scxml version="1.0" xmlns="http://www.w3.org/2005/07/scxml"
  xmlns:queue="http://www.genesyslab.com/modules/queue"
  xmlns:ixn="http://www.genesyslab.com/modules/interaction"
  xmlns:ws="http://www.genesyslab.com/modules/ws"
  xmlns:dialog="http://www.genesyslab.com/modules/dialog"
  _persist="false" >
  <datamodel>
    <data ID="reqid" expr="''"/>
    <data ID="childIxn" expr="''"/>
    <data ID="routetarget" expr="''"/>
  </datamodel>
  <initial>
    <transition target="listening">
    </transition>
  </initial>
  <state id="main" initial="listening">
    <state id="listening">
      <onentry>
        <log expr="'Listening ...'" />
      </onentry>
      <!-- This script handles email and sms interactions -->
      <transition event="interaction.present"
cond="_genesys.ixn.interactions[_event.data.interactionid].category == 'msgbased'"
target="arriving_ixn">
        <script>
          _data.ixnid = _event.data.interactionid;
        </script>
      </transition>
    </state>
    <state id="arriving_ixn">
      <onentry>
        <log expr="uneval( global )" />
        <if
cond="_genesys.ixn.interactions[_data.ixnid].msgbased.media == 'TMediaEMail'">
          <if
cond="_genesys.ixn.interactions[_data.ixnid].udata.FromAddress ==
'postmaster@genesyslab.com'">
            <!-- This looks to be a delivery
confirmation. -->
            <event name="handle.email.confirmation"/>
          <else/>
            <event name="send.email.ack"/>
          </if>
        <elseif
cond="_genesys.ixn.interactions[_data.ixnid].msgbased.media == 'TMediaNativeSMS'">
          <event name="send.sms.ack"/>
        </if>
      </onentry>
      <transition event="handle.email.confirmation" target="exit">
        <!-- Add logic here to handle the delivery confirmation... -->
        <!-- Terminate the confirmation interaction -->
      </transition>
    </state>
  </state>
</scxml>
```

```

        <ixn:terminate interactionid="_data.ixnid" />
    </transition>
    <transition event="send.email.ack">
        <!-- Send an Email acknowledgment -->
        <!-- Acknowledgment text is a "Standard Response" with the
specified, 16-digit name. -->
        <ixn:createmessage requestid="_data.reqid"
type="acknowledgement"
media="_genesys.ixn.mediaType.TMediaEmail"
to="'_origin'" subject="'$USEURL'"
msgsrc="'gdata:config\\SR.0000Na5C1F3500JW'"
delivery="false" />
    </transition>
    <transition event="send.sms.ack">
        <!-- Send an SMS acknowledgment -->
        <ixn:createmessage requestid="_data.reqid"
type="acknowledgement"
media="_genesys.ixn.mediaType.TMediaNativeSMS"
to="'_udata\\_smsSrcNumber'" from="'8881234567'"
msgsrc="'Your request has been received and is being
processed'" />
    </transition>
    <transition event="msgbased.createmessage.done" target="find_agent"/>
    <transition event="error.msgbased.createmessage" target="ack_failed"/>
</state>
<state id="ack_failed">
    <transition target="find_agent">
        <log expr="Acknowledgement request failed and should be
handled here." />
    </transition>
</state>
<state id="find_agent">
    <onentry>
        <!-- Submit URS request to find an agent. -->
        <queue:submit route="false">
            <queue:targets>
                <queue:target name="'5'" type="agent"
statserver="'statserver'" />
                <queue:target name="'6'" type="agent"
statserver="'statserver'" />
                <queue:target name="'7'" type="agent"
statserver="'statserver'" />
            </queue:targets>
        </queue:submit>
    </onentry>
    <transition event="queue.submit.done" target="deliver_to_agent">
        <log expr="uneval( _event )" />
        <assign location="_data.routetarget"
expr="_event.data.resource" />
    </transition>
    <transition event="error.queue.submit" target="error">
        <log expr="uneval( _event )" />
    </transition>
</state>
<state id="deliver_to_agent">
    <onentry>
        <log expr="'Routing ...'" />
        <script>
            var hint = new Object();
            hint.outqueues = new Object();
            hint.outqueues.Orchestration_queue = "queue for
outbound response";
        </script>

```

```

                                <!-- For redirect, the 'to' object is gotten from data
returned from queue:submit -->                                <ixn:redirect requestid="_data.reqid"
interactionid="_data.ixnid"                                from="'scxml'" to="_data.routetarget" hints="hint"/>
                                </onentry>
                                <transition event="interaction.redirect.done" target="responding">
                                    <log expr="uneval( _event )" />
                                </transition>
                                <transition event="error.interaction.redirect" target="error">
                                    <log expr="uneval( _event )" />
                                </transition>
                            </state>
                            <parallel id="responding">
                                <onentry>
                                    <log expr="'Process Agent response...'" />
                                </onentry>

                                <state id="responding_outbound">
                                    <transition event="interaction.present"
cond="_genesys.ixn.interactions[_event.data.interactionid].msgbased.type == 'OutboundReply'"
                                <!-- This present occurs if agent sends outbound
response to queue which this ORS instance manages -->
                                    <log expr="'Event Interaction Present...'" />
                                    <assign location="_data.childIxn"
expr="_event.data.interactionid" />
                                    <!-- Send agent's response to customer -->
                                    <ixn:sendmessage interactionid="_data.childIxn"
delivery="true"/>
                                    </transition>
                                    <transition event="msgbased.sendmessage.done"
target="cleanup">
                                        <log expr="'Sending of outbound response
successful.'" />
                                        <ixn:terminate interactionid="_data.childIxn"
/>
                                    </transition>
                                    <transition event="error.msgbased.sendmessage"
target="requesterror">
                                        <log expr="'Error in sending outbound response.'" />
                                        <log expr="uneval( _event )" />
                                    </transition>
                                </state>
                                <state id="responding_initial_interaction">
                                    <transition event="interaction.notcontrolled">
                                        <log expr="'Initial interaction is no longer
controlled by this scxml.'" />
                                    </transition>
                                    <transition event="interaction.deleted">
                                        <!-- Initial interaction is going away -->
                                        <!-- Wait up to 60 seconds for any response to be
processed (in parallel state). -->
                                        <send delay="'60s'" type="'scxml'"
event="'wait_is_over'" />
                                    </transition>
                                    <transition event="wait_is_over" target="cleanup" />
                                </state>
                            </parallel>
                            <state id="cleanup">
                                <onentry>
                                    <log expr="'Cleanup logic goes here...'" />
                                </onentry>
                                <transition target="exit"/>

```

```
        </state>
        <state id="requesterror">
            <onentry>
                <log expr="'Request error processing goes here...'" />
            </onentry>
            <transition target="error"/>
        </state>
    </state>
    <final id="error"/>
    <final id="exit"/>
</scxml>
```