Orchestration Server Developer's Guide

Detect Consult Call

5/1/2025

Detect Consult Call

The following example illustrates the case when we are trying to detect whether the call that started the session is a `consult` call. The assumption is that the following SCXML file is configured on a Routing Point and the SCXML session is started when a call is made to the Routing Point.

Here are two scenarios:

1. Direct call to Routing Point
   - Customer makes a call and is connected to a Routing Point.
   - This is considered the primary call and the only active interaction.
   - All actions (`<queue:submit>`, `<dialog:playsound>`, etc) are applied to this interaction.
2. Consult call to Routing Point
   - Customer makes a call and is connected to an agent X.
   - This is considered the primary call and is not being monitored by Orchestration Server (the interaction is ownerless).
   - Agent X does a consult call to the Routing Point. This starts a SCXML session which is monitored by Orchestration.
   - The consult call is considered the `effective call` until the primary and consult calls are merged (which happens if agent X completes the transfer to the Routing Point). At that time, the consult call is no longer valid and the primary call is the `effective call`.
   - All actions (`<queue:submit>`, `<dialog:playsound>`, etc) are applied to the `effective call`.

Assumptions:

- At any time during the session, if the primary call is dead, the SCXML session will be terminated, regardless of the status of the consult call. This assumes the primary call and consult calls have not been merged.
- At any time during the session, if the effective call is dead, the SCXML session will be terminated.

```
<scxml version="1.0" xmlns="http://www.w3.org/2005/07/scxml"
        xmlns:queue="www.genesyslab.com/modules/queue"
        xmlns:dialog="www.genesyslab.com/modules/dialog"
        xmlns:ixn="http://www.genesyslab.com/modules/interaction"
        initial="global">
        <script>
                var reqid;
                var consult_ixn_id;
                var primary_ixn_id;
                var effective_ixn_id;
                var sessionStarted = false;
        </script>
        <!--**************************************************************************-->
        <state id="global" initial="initial">
                <!--**************************************************************************-->
                <state id="initial">
                        <!--This ensures the session terminates after 10 minutes-->
```

```
                        <onentry>
                                <send event="'toExit'" delay="'600s'" />
                        </onentry>
                        <transition event="interaction.added" cond="sessionStarted == false">
                                <script>
                                        /* To avoid catching another 'interaction.added' event
                                        (caused by 'attach') in the same state again, set
sessionStarted to
                                        true. 'Attach' action could be done in a separate
state, but for
                                        the sake of simplicity and to minimize number of
states it is done
                                        here in initial state...*/
                                        sessionStarted = true;
                                        /* Assign interaction IDs that will be needed later
on ... */
                                        if(
_genesys.ixn.interactions[_event.data.interactionid].voice.type == 'consult' )
                                        {
                                                consult_ixn_id = _event.data.interactionid;
                                                primary_ixn_id =
_genesys.ixn.interactions[consult_ixn_id].parentid;
                                                effective_ixn_id = consult_ixn_id;
                                        }
                                        else
                                        {
                                                consult_ixn_id = undefined;
                                                primary_ixn_id = _event.data.interactionid;
                                                effective_ixn_id = primary_ixn_id;
                                        }
                                </script>
                                <log expr="'CONSULT_EXAMPLE: consult_ixn_id = ' +
consult_ixn_id" />
                                <log expr="'CONSULT_EXAMPLE: primary_ixn_id = ' +
primary_ixn_id" />
                                <log expr="'CONSULT_EXAMPLE: effective_ixn_id = ' +
effective_ixn_id" />
                                <if cond="consult_ixn_id ;!= undefined">
                                        <log expr="'CONSULT_EXAMPLE: Consult call started
strategy. Attaching primary call...'" />
                                        <ixn:attach requestid="reqid"
interactionid="primary_ixn_id" />
                                <else />
                                        <log expr="'CONSULT_EXAMPLE: Normal call started
strategy. Proceeding with session ...'" />
                                        <send event="'toProceed'" />
                                </if>
                        </transition>
                        <transition event="interaction.attach.done"
cond="_event.data.requestid == reqid" target="prewaiting_state" />
                        <!-- error.interaction.attach event (if happened) will be caught in
global state -->
                        <transition event="toProceed" target="CUSTOM_WORKING_STATE" />
                </state>
                <!--*********************************************************************-->
                <state id="prewaiting_state">
                        <onentry>
                                <!--This illustrates the case when the session is started by
a consult
                                        call (and that call is still alive here), sometimes
it makes sense
                                        to wait for some short amount of time. This time
could depend on
```

```
                                        how fast TServer completes transfer, or could be done
to avoid
                                        routing consult call during mute transfer, etc.-->
                                <log expr="'CONSULT_EXAMPLE: Continuing session with some
short delay...'" />
                                <send event="'toProceed'" delay="'1s'" />
                        </onentry>
                        <transition event="toProceed" target="CUSTOM_WORKING_STATE" />
                </state>
                <!--*************************************************************************-->
                <!--************ This is where your main logic goes *********************-->
                <!--*************************************************************************-->
                <state id="CUSTOM_WORKING_STATE" initial="route_to_agent">
                        <!--This will try to route the call to agent 703_sip. If it is not
                                successful within 3 seconds, it will transition to state
"dialog"
                                and play music. The attribute "clearontimeout" is set to
false so
                                router will continue trying to route to the agent while the
music is
                                playing.-->
                        <state id="route_to_agent">
                                <onentry>
                                        <queue:submit requestid="reqid"
interactionid="effective_ixn_id"
                                                priority="5" timeout="3"
clearontimeout="false">
                                                <queue:targets>
                                                        <queue:target type="agent"
name="'703_sip'" />
                                                </queue:targets>
                                        </queue:submit>
                                </onentry>
                                <transition event="error.queue.submit" target="dialog">
                                        <log expr="'ERROR WITH QUEUE SUBMIT: ' + uneval(
_event )" />
                                </transition>
                        </state>
                        <!-- This plays music for 60 seconds. -->
                        <state id="dialog">
                                <onentry>
                                        <dialog:playsound requestid="reqid"
interactionid="effective_ixn_id"
                                                type="'music'" resource="'music/on_hold'"
duration="60" />
                                </onentry>
                                <transition event="dialog.playsound.done.timeout" />
                                <transition event="dialog.playsound.done" target="exit" />
                                <transition event="error.dialog.playsound" target="error">
                                        <log expr="'ERROR PLAYING MUSIC: ' + uneval(_event)"
/>
                                </transition>
                        </state>
                        <transition event="queue.submit.done" target="exit">
                                <log expr="'QUEUE SUBMIT DONE.  Ending Session.'" />
                        </transition>
                        <transition event="interaction.partystatechanged"
cond="effective_ixn_id == _event.data.interactionid">
                                <log expr="'CONSULT_EXAMPLE: Got partystatechanged event: ' +
uneval(_event.data)" />
                        </transition>
                </state>
                <!--*************************************************************************-->
```

```
                <!--************************************************************************-->
                <!--************************************************************************-->
                <transition event="interaction.onmerge"
cond="_event.data.frominteractionid == consult_ixn_id && _event.data.tointeractionid ==
primary_ixn_id">
                        <script>
                                consult_ixn_id = undefined;
                                effective_ixn_id = primary_ixn_id;
                        </script>
                        <log expr="'CONSULT_EXAMPLE: Effective call ID changed because of
transfer completion: ' + uneval(_event)" />
                        <log expr="'CONSULT_EXAMPLE: consult_ixn_id = ' + consult_ixn_id" />
                        <log expr="'CONSULT_EXAMPLE: primary_ixn_id = ' + primary_ixn_id" />
                        <log expr="'CONSULT_EXAMPLE: effective_ixn_id = ' + effective_ixn_id"
/>
                </transition>
                <transition event="interaction.deleted"
                        cond="_event.data.interactionid == effective_ixn_id" target="exit">
                        <log expr="'CONSULT_EXAMPLE: Effective call is dead. Exiting...: ' +
uneval(_event)" />
                </transition>
                <transition event="interaction.deleted"
                        cond="_event.data.interactionid == primary_ixn_id &&
consult_ixn_id != undefined"
                        target="exit">
                        <log expr="'CONSULT_EXAMPLE: Primary call is dead, consult call is
alive and useless. Exiting...: ' + uneval(_event)" />
                </transition>
                <!--In case none of the other events are triggered, this will end the
                        session after number of minutes specified at the strategy beginning-->
                <transition event="toExit" target="exit">
                        <log expr="'CONSULT_EXAMPLE: Possibly stuck session is self-
destructing. Exiting...: ' + uneval(_event)" />
                </transition>
                <!--This will catch all the errors that are not processed elsewhere-->
                <transition event="error.*" target="error">
                        <log expr="'CONSULT_EXAMPLE: ERROR AT GLOBAL LEVEL'" />
                        <log expr="'CONSULT_EXAMPLE: Got error event: ' + uneval( _event )" />
                </transition>
        </state>
        <final id="exit" />
        <final id="error" />
</scxml>
```

- When agent X initiates a transfer or consult to the Routing Point, it will trigger a SCXML session to be created and will wait for the `interaction.added` event.
- After the `interaction.added` event is received, it will set the `consult_ixn_id`, `primary_ixn_id`, and `effective_ixn_id` depending on whether the session was started by a regular call, or a consult call to the Route Point.
- If the SCXML application detects that the call from Agent X to the Routing Point is of type `consult`, we attach the parent interaction (the primary call which is ownerless) to the current session (see interaction attach for more details about ownership).
- The `interaction.attach.done` event will trigger a transition to the `prewaiting_state`, where we put in a delay. This delay is needed depending on how fast TServer completes the transfer, or is sometimes done to avoid routing a consult call during a mute transfer.
- The `CUSTOM_WORKING_STATE` is where you would put your main logic. In this example, we first try to route the call to agent `703_sip`. If this is not successful within 3 seconds, we transition to the `dialog` state and play music for 60 seconds.

- At any time during the session, if agent X decides to complete the transfer to the Routing Point or to agent Y (if the consult call was routed from the Routing Point to agent Y), the primary and consult calls are merged, and the event `interaction.onmerge` is raised. This event triggers a transition in the SCXML application and redefines the variables `consult_ixn_id`, and `effective_ixn_id` since the consult interaction is deleted during the merge. The `consult_ixn_id` will no longer be valid and is set to `undefined`. The `effective_ixn_id` is changed from the consult call to the primary call and should be used from this point forward for all functions and actions that require an interaction ID.
- Exiting the session is triggered by any of the following situations:
  - The call is successfully routed to agent `703_sip`.
  - Music has been played for 60 seconds.
  - There was a problem playing the file `music/on_hold`.
  - The effective call is deleted (effective call is the consult call until the consult or transfer is complete, at which time, it is the only call left).
  - The primary call is deleted before the consult or transfer is complete (the consult call can still be alive but is useless at this point).
  - Any `error.*` events that are raised during the session.
  - The session may be stuck and self-destucts 10 minutes after it was created.