Genesys Mobile Services API Reference

request-inbound-delay

4/23/2025

request-inbound-delay

```
<scxml version="1.0" xmlns="http://www.w3.org/2005/07/scxml"
        xmlns:queue="http://www.genesyslab.com/modules/queue"
        xmlns:dialog="http://www.genesyslab.com/modules/dialog"
        xmlns:session="http://www.genesyslab.com/modules/session"
        xmlns:ixn="http://www.genesyslab.com/modules/interaction"
        xmlns:ws="http://www.genesyslab.com/modules/ws"
        xmlns:yahoo_placefinder="http://www.genesyslab.com/modules/dfm/yahoo_placefinder/v1"
        xmlns:gsgNotification="http://www.genesyslab.com/modules/dfm/gsgNotification/v1"
        xmlns:service="http://www.genesyslab.com/modules/dfm/gsgBasedServices/v1"
        xmlns:storage="http://www.genesyslab.com/modules/dfm/gsgStorage/v1"
        initial="advanced_async">
        <!--
        This is an advanced service which helps an application/end user contact the contact
center.
        It has the following characteristics:
        It supports customer initiated voice contacts.
        It supports waiting for the appropriate available agent before providing the
necessary access information.
        It stores and maintains application data with the service.
        It supports more advanced access number allocation:
        It can reserve the access information for a configurable period of time
        Simple random or round robin allocation
        It sends event and status notifications back to the requesting application.
        When the agent is available, it will return the access information in the status
event.
        It support the following types of access information:
        Access Number (DNIS) which is to be called by the application
        Access code which is to be supplied by the customer/application when the contact is
being established. This provides an extra level of authentication.
        -->
        <datamodel>
                <!-- globals -->
                <data id="reqid" />
                <data id="startSendId" />
                <data id="service_TYPE" expr="'ors'"/>
                <!-- TODO: The following is used for custom reserve service - it is not used
in this example -->
                <data id="service_NAME" expr="'http://138.120.72.47:8080/gsg-web/gms/1/
service/reserve'"/>
                <data id="service_TTL" expr="'3600'"/>
                <data id="_resource_group" _type="parameter"/>
                <data id="_provide_code" _type="parameter"/>
                <data id="reserve_provide_code" expr="'false'"/>
                <data id="_access_info_via_event" _type="parameter"/>
                <data id="_device_os" _type="parameter"/>
                <data id="_device_notification_id" _type="parameter"/>
                <data id="_data_id" _type="parameter"/>
                <data id="reserveSendId" />
                <data id="APIVersion" expr="'1'"/>
                <!-- TODO: This sample will route the call to the agent group "Billing". -->
                <data id="defaultAgentGroup" expr="'Billing'" />
                <data id="queueSubmitTimeout" expr="60" />
        </datamodel>
        <state id="advanced_async" initial="waitForStart">
                <state id="waitForStart">
                        <!-- wait for "gms.start" event from GMS before processing the
session -->
```

```
                        <transition event="gms.start" target="handleCheckRequiredParms">
                                <script>
                                        <!-- Save this event id so we can respond to it -->
                                        _data.startSendId = _event.sendid;
                                        <!-- Override defaults with params if available -->
                                        if ( _data.hasOwnProperty( '_type' ) )
                                        _data.service_TYPE = _data._type;
                                        if ( _data.hasOwnProperty( '_name' ) )
                                        _data.service_NAME = _data._name;
                                        if ( _data.hasOwnProperty( '_ttl' ) )
                                        _data.service_TTL = _data._ttl;
                                </script>
                        </transition>
                </state>
                <!-- Check required parameters are passed in -->
                <state id="handleCheckRequiredParms">
                        <onentry>
                                <script>
                                        var checkParmsOK = 'false';
                                        if ( _data.hasOwnProperty( '_device_os' )  &&
                                        _data.hasOwnProperty( '_device_notification_id' )  &&
                                        _data.hasOwnProperty( '_resource_group' ) )  {
                                        checkParmsOK = 'true'
                                        }
                                        var result = { "error": "Missing parameters one of
_type, _name and _ttl"};
                                </script>
                        </onentry>
                        <transition target="checkRequiredParmsOK" cond="checkParmsOK ==
"true""  />
                        <transition target="error" cond="checkParmsOK == "false"" >
                                <!-- Send error response to the gms.start event -->
                                <ws:response requestid="_data.startSendId"
resultcode="JSON.stringify(result)" />
                        </transition>
                </state>
                <!-- All parms provided notify GMS -->
                <state id="checkRequiredParmsOK">
                        <onentry>
                                <script>
                                        var result = new Object();
                                        result._id = _sessionid;
                                        if ( _data.hasOwnProperty( '_provide_code' ) )
                                        _data.reserve_provide_code = _data._provide_code;
                                </script>
                                <!-- Send session id after successfully creating the session
-->
                                <ws:response requestid="_data.startSendId"
resultcode="JSON.stringify( result )" />
                        </onentry>
                        <transition target="handleRegisterDeviceStatusChanged">
                                <!-- TODO: May want to start timer here instead of after
registrations -->
                        </transition>
                </state>
                <!-- Register on behalf of device for status changed with notification
service -->
                <state id="handleRegisterDeviceStatusChanged">
                        <onentry>
                                <!-- Register with notification service -->
                                <script>
                                        var deviceSubscriptionStatusChanged = {
"subscriberId": _sessionid,
```

```
                                        "notificationDetails":
{"deviceId":_data._device_notification_id, "type":_data._device_os},
"expire":_data.service_TTL,

"filter":"a2c.advanced.service.statuschanged."+_sessionid};
                                </script>
                                <!--  using notification DFM to -->
                                <!--  create subscription for device to receive service
status changed events -->
                                <gsgNotification:create apiVersion="_data.APIVersion"
content="deviceSubscriptionStatusChanged" />
                        </onentry>
                        <transition event="gsgNotification.create.done"
target="handleRegisterDeviceAgentAvailable">
                                <!-- TODO: no one needs to be notified of subscription -->
                        </transition>
                        <transition event="error.gsgNotification.create" target="error">
                                <!-- TODO: Log error message-->
                        </transition>
                </state>
                <!-- Register on behalf of device for agent available with notification
service -->
                <state id="handleRegisterDeviceAgentAvailable">
                        <onentry>
                                <!-- Register with notification service -->
                                <script>
                                        var deviceSubscriptionAgentAvailable =
{"subscriberId":
                                        _sessionid,
                                        "notificationDetails":
                                        {"deviceId":_data._device_notification_id,
"type":_data._device_os}, "expire":_data.service_TTL,

"filter":"a2c.advanced.service.agentavailable."+_sessionid};
                                </script>
                                <!--  create subscription for device to receive agent
available events -->
                                <gsgNotification:create apiVersion="_data.APIVersion"
content="deviceSubscriptionAgentAvailable" />
                        </onentry>
                        <transition event="gsgNotification.create.done"
target="handleRegisterDeviceExpired">
                                <!-- TODO: no one needs to be notified of subscription -->
                        </transition>
                        <transition event="error.gsgNotification.create" target="error">
                                <!-- TODO: Log error message-->
                        </transition>
                </state>
                <!-- Register on behalf of device for ttl expired with notification service --
>
                <state id="handleRegisterDeviceExpired">
                        <onentry>
                                <!-- Register with notification service -->
                                <script>
                                        var deviceSubscriptionExpired = {"subscriberId":
                                        _sessionid,
                                        "notificationDetails":
                                        {"deviceId":_data._device_notification_id,
"type":_data._device_os}, "expire":_data.service_TTL,
                                        "filter":"a2c.advanced.service.expired."+_sessionid};
                                </script>
                                <!--  create subscription for device to receive when service
expires -->
```

```
                                <!--  mobile app should handle this notification and recreate
the service -->
                                <gsgNotification:create apiVersion="_data.APIVersion"
content="deviceSubscriptionExpired" />
                        </onentry>
                        <transition event="gsgNotification.create.done"
target="handleRegisterSession">
                                <!-- TODO: no one needs to be notified of subscription -->
                        </transition>
                        <transition event="error.gsgNotification.create" target="error">
                                <!-- TODO: Log error message-->
                        </transition>
                </state>
                <!-- Register for session data with notification service -->
                <state id="handleRegisterSession">
                        <onentry>
                                <!-- Register with notification service -->
                                <script>
                                        var waitingForAgentMessage = {"message":
"\{\"_id\":\"" + _sessionid + "\", \"_status\":\"Waiting for Agent\",
\"_dialog\":\"waiting_for_agent.html\"\}","tag":"a2c.advanced.service.statuschanged."+_sessionid};
                                </script>
                                <gsgNotification:publish apiVersion="_data.APIVersion"
content="waitingForAgentMessage" />
                        </onentry>
                        <transition event="gsgNotification.create.done">
                                <gsgNotification:publish apiVersion="_data.APIVersion"
content="waitingForAgentMessage" />
                                <!-- Start service expire timer -->
                                <send target="'_internal'" event="'service.ttl.expired'"
delay="_data.service_TTL  + 's'" />
                        </transition>
                        <!--  TODO: handle subscription error -->
                        <transition event="subscribe.error" target="error">
                                <!-- TODO: Log error message-->
                        </transition>
                        <transition event="gsgNotification.publish.done"
target="waitForAgent"/>
                        <transition event="subscribe.error" target="error"/>
                </state>
                <state id="waitForAgent">
                        <onentry>
                                <!-- TODO: check agent availability but do not route -->
                                <queue:submit route="false"
timeout="_data.queueSubmitTimeout">
                                        <queue:targets type="agentgroup">
                                                <queue:target name="_data.defaultAgentGroup"/>
                                        </queue:targets>
                                </queue:submit>
                        </onentry>
                        <transition event="queue.submit.done"
target="checkAccessInfoViaEvent"/>
                        <transition event="error.queue.submit" target="error">
                                <!-- TODO: Log error message - agent not found - retry? -->
                        </transition>
                        <transition event="service.ttl.expired" target="error">
                                <!-- TODO: Log error message - service expired -->
                        </transition>
                </state>
                <!-- Check _access_info_via_event -->
                <state id="checkAccessInfoViaEvent">
                        <onentry>
                                <script>
```

```
                                        var agentAvailableMessage = {        "message":
"\{\"_id\":\"" + _sessionid + "\", \"_resource_group\":\"" + _data._resource_group + "\",
\"_status\":\"Agent Available\"\}",

"tag":"a2c.advanced.service.agentavailable."+_sessionid};
                                </script>
                        </onentry>
                        <transition target="reserveResource"
cond="_data._access_info_via_event == "true"" >
                        </transition>
                        <transition target="waitForReserveRequest" cond="true">
                                <!-- After agent is available, notify the device -->
                                <gsgNotification:publish apiVersion="_data.APIVersion"
content="agentAvailableMessage" />
                        </transition>
                        <transition event="service.ttl.expired" target="error">
                                <!-- TODO: Log error message - service expired -->
                        </transition>
                </state>
                <!-- Wait for reserve request -->
                <state id="waitForReserveRequest">
                        <transition event="reserve" target="reserveResource">
                                <script>
                                        _data.reserveSendId = _event.sendid;
                                        if ( _event.data.param.hasOwnProperty(
'_provide_code' ) )
                                        _data.reserve_provide_code =
_event.data.param._provide_code;
                                </script>
                        </transition>
                        <transition event="service.ttl.expired" target="error">
                                <!-- TODO: Log error message - service expired -->
                        </transition>
                </state>
                <state id="reserveResource">
                        <onentry>
                                <service:reserve requestid="_data.requestId" _id="_sessionid"
_resource_group="_data._resource_group" _provide_code="reserve_provide_code"
_phone_number="_data._phone_number" />
                        </onentry>
                        <transition event="gsgBasedServices.reserve.done"
target="waitForCall" >
                                <script>
                                        var myResultSetObject = eval('(' +
_event.data.content + ')');
                                        var resultReserve = new Object();
                                        resultReserve._id = _sessionid;
                                        if ( myResultSetObject.hasOwnProperty(
'_access_number' ) )
                                        resultReserve._access_number =
myResultSetObject._access_number;
                                        if ( myResultSetObject.hasOwnProperty( '_access_code'
) )
                                        resultReserve._access_code =
myResultSetObject._access_code;
                                        if ( myResultSetObject.hasOwnProperty(
'_expiration_time' ) )
                                        resultReserve._expiration_time =
myResultSetObject._expiration_time;
                                </script>
                                <!--  send response to the reserve event -->
                                <ws:response requestid="_data.reserveSendId"
resultcode="JSON.stringify( resultReserve )" />
```

```
                        </transition>
                        <transition event="error.gsgBasedServices.reserve" target="error">
                                <!-- TODO: Log error message-->
                        </transition>
                        <transition event="service.ttl.expired" target="error">
                                <!-- TODO: Log error message - service expired -->
                        </transition>
                </state>
                <state id="waitForCall">
                        <!-- wait for call (voice interaction) from inbound.scxml to be
associated to this session -->
                        <transition event="interaction.present" target="ConnectToAgent">
                        </transition>
                        <transition event="timer" target="exit" />
                </state>
                <!--  Call and data are available. Attach data and add custom routing logic.
-->
                <state id="ConnectToAgent">
                        <onentry>
                                <log expr="'Checking for agent availability in agent group: '
+ _data.defaultAgentGroup"/>
                                <queue:submit route="true" timeout="3">
                                        <queue:targets>
                                                <queue:target name="_data.defaultAgentGroup"
type="agentgroup" />
                                        </queue:targets>
                                </queue:submit>
                        </onentry>
                        <!-- Agents are available for the DNIS and call is being routed to an
agent -->
                        <transition event="queue.submit.done" target="AgentConnected">
                        </transition>
                        <!-- Could not get and agent for a period of time so let try again -
NEED TO ADD LOGIC TO NOT RETRY FOREVER -->
                        <transition event="error.queue.submit">
                                <log expr="'Again checking for agent availability in agent
group: ' + _data.defaultAgentGroup"/>
                                <!-- TODO: Implement retry logic here -->
                        </transition>
                </state>
                <state id="AgentConnected">
                        <onentry>
                                <send event="'CallTimeout'" delay="'300s'"/>
                        </onentry>
                        <!-- Interaction is gone so clean up -->
                        <transition event="interaction.deleted" target="exit">
                                <assign location="_data.ixnid" expr="''"/>
                        </transition>
                        <!-- The call has been over 5 minutes -->
                        <transition event="CallTimeout" target="exit">
                        </transition>
                </state>
        </state>
        <final id="exit">
                <onentry>
                        <gsgNotification:deleteSubscriber apiVersion="_data.APIVersion"
subscriberID="_sessionid" />
                </onentry>
        </final>
        <final id="error">
                <onentry>
                        <gsgNotification:deleteSubscriber apiVersion="_data.APIVersion"
subscriberID="_sessionid" />
```

```
                </onentry>
        </final>
</scxml>
```