GENESYS™

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.

Genesys Mobile Services API Reference

Genesys Mobile Engagement 8.1.1

12/29/2021

Table of Contents

Welcome

Overview of Genesys Mobile Services APIs

Genesys Mobile Services contains multiple APIs, each dedicated to performing certain tasks as described below. Select the API name for a more detailed examination of the operations and responses they use.

- Storage API — Storage is general-purpose API that allows users to temporarily store arbitrary data. Data may consist of key/value pairs of strings or binary objects.
- Service API — This API is used by customer-facing applications to manage different type of contact center related services (for example the app-to-connect-basic service provides the necessary contact center access information so the end-user and associated application can initiate an interaction with the contact center).
- Node API — This is a base ping API implementation which will be used by load balancers to determine the health of a given GMS node to determine if it can use this GMS node when it load balancing API requests across the set of GMS nodes.
- Notification API — This set of event-driven APIs is used to manage notifications between applications and Genesys systems. Users subscribe to an event and provide an indication of how the notification should be delivered, then events are published to the system.

  Note: The Notification API is only intended to be used with Orchestration Server-based Services, not from external mobile applications.
- Chat API — This API is used by customer-facing applications to create and manage a chat session associated with a a contact center related services. A single service is associated with a single chat.

Overview of Services Provided by Genesys Mobile Services

- Genesys Mobile Services-based Services — These services are executed in the Genesys Mobile Services server. They provide fixed functionality that can only be tuned through configuration options.
- Orchestration Server-based Services — These services are executed in the ORS server. They include sample services that can be altered and customized by a developer to meet specific customer requirements.
- Genesys Mobile Services-Call Flows — Detailed call flow diagrams for the services included with Genesys Mobile Services.

APIs

Articles in this chapter offer details about the various APIs provided by Genesys Mobile Services. Please select an API from the list below:

- Storage API (with Sample)
- Service API
- Node API
- Notification API
- Push Notification Service

Storage API

Overview

Storage is general purpose API that allows users to temporarily store arbitrary data. Data may consist of key/value pairs of strings or binary objects.

API

Create

Allows for the creation of a new storage area in GMS.

Operation

Method	POST		
URL	/genesys/1/storage/**{ttl}**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{ttl}	number	yes	The time to live for this data, specified in seconds. The data is automatically deleted after is has been stored for **{ttl}** seconds.
Body: A MultiPart form or a URL encoded form consisting of different items representing the key/value pairs to store.			

Response

A JSON object with the property id, identifying the assigned id for this storage request.

HTTP code	200
HTTP message	OK

Example

The following example stores:

- Key1, Key2, Key3 and FileKey

The time-to-live of the data is 1 hour.

Operation

```
Request URL:http://localhost:8080/genesys/1/storage/3600
Request Method:POST
Status Code:200 OK
Request Headersview source
Accept:*/*
Accept-Charset:ISO-8859-1,utf-8;q=0.7,*;q=0.3
Accept-Encoding:gzip,deflate,sdch
Accept-Language:en-US,en;q=0.8
Connection:keep-alive
Content-Length:13028
Content-Type:multipart/form-data; boundary=----WebKitFormBoundaryy16qocbN6tmPORZL
Host:localhost:8080
Origin:http://localhost:8080
Referer:http://localhost:8080/genesys/resources/storagetest.html
User-Agent:Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.7 (KHTML, like Gecko) Chrome/
16.0.912.77 Safari/535.7
Request Payload
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="Key1"
Value1
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="Key2"
Value2
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="Key3"
Value3
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="FileKey"; filename="MyPic.png"
Content-Type: image/png
------WebKitFormBoundaryy16qocbN6tmPORZL--
```

Result

The above data is now stored for up to 1 hour with an id of 39a98e24-b03b-4191-b756-1efe8f3b16b8.

```
HTTP 200 OK
{ "id":"39a98e24-b03b-4191-b756-1efe8f3b16b8" }
```

Update

Updates a storage area that has already been created in GMS.

Operation

Method	POST		
URL	/genesys/1/storage/**{id}**/**{ttl}**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{id}	string	yes	The id of the allocated storage to be updated.
{ttl}	number	yes	The time to live for this

Method	POST		
			data, specified in seconds. The data is automatically deleted after is has been stored for **{ttl}** seconds.
Body: A MultiPart form consisting of different ites representing the key/value pairs to store. This may be sting values or files.			

Response

HTTP code	200
HTTP message	OK

Example

The following example updates the keys:

- Key1, Key2, Key3 and FileKey

The time-to-live for all of the keys in this update is 1 hour.

Operation

```
Request URL:http://localhost:8080/genesys/1/storage/b8e8eb60-3f14-493d-90da-0034aca34b55/3600
Request Method:POST
Status Code:200 OK
Request Headersview source
Accept:*/*
Accept-Charset:ISO-8859-1,utf-8;q=0.7,*;q=0.3
Accept-Encoding:gzip,deflate,sdch
Accept-Language:en-US,en;q=0.8
Connection:keep-alive
Content-Length:171539
Content-Type:multipart/form-data; boundary=----WebKitFormBoundaryPu8S1YopPtZq8Z54
Host:localhost:8080
Origin:http://localhost:8080
Referer:http://localhost:8080/genesys/resources/storagetest.html
User-Agent:Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.7 (KHTML, like Gecko) Chrome/
16.0.912.77 Safari/535.7
Request Payload
------WebKitFormBoundaryPu8S1YopPtZq8Z54
Content-Disposition: form-data; name="Key1"
Value6
------WebKitFormBoundaryPu8S1YopPtZq8Z54
Content-Disposition: form-data; name="Key2"
Value7
------WebKitFormBoundaryPu8S1YopPtZq8Z54
Content-Disposition: form-data; name="Key3"
Value8
------WebKitFormBoundaryPu8S1YopPtZq8Z54
Content-Disposition: form-data; name="FileKey"; filename="0016_001.pdf"
Content-Type: application/pdf
------WebKitFormBoundaryPu8S1YopPtZq8Z54--
Response Headersview source
Cache-Control:no-cache
no-store
```

```
Content-Length:2
Content-Type:application/json
Date:Sat, 04 Feb 2012 02:06:43 GMT
Pragma:no-cache
Server:Apache-Coyote/1.1
```

Result

The above data is now stored for up to 1 hour with an id of 39a98e24-b03b-4191-b756-1efe8f3b16b8.

```
HTTP 200 OK
```

Query (all keys)

Queries all of the keys in a storage area that has already been created in GMS.

Operation

Method	GET		
URL	/genesys/1/storage/**{id}**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{id}	string	yes	The id of the allocated storage to be updated.
Body: Not used			

Response

HTTP code	200
HTTP message	OK

Example

The following example queries all of the keys associated with id b8e8eb60-3f14-493d-90da-0034aca34b55

Operation

```
Request URL:http://localhost:8080/genesys/1/storage/b8e8eb60-3f14-493d-90da-0034aca34b55
Request Method:GET
```

Result

```
{"Key2":"Value7","Key1":"Value6","Key3":"Value8","FileKey":"http://127.0.0.1:8080/genesys/1/
storage/binary/b8e8eb60-3f14-493d-90da-0034aca34b55/FileKey"
```

Query (one key)

Queries all of the keys in a storage area that has already been created in GMS.

Operation

Method	GET		
URL	/genesys/1/storage/**{id}/{key}**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{id}	string	yes	The id of the allocated storage to be updated.
{key}	string	yes	The key of the specificaly requested value
Body: Not used			

Response

HTTP code	200
HTTP message	OK

Example

The following example queries the value of Key1 from the data associated with id b8e8eb60-3f14-493d-90da-0034aca34b55

Operation

```
Request URL:http://localhost:8080/genesys/1/storage/b8e8eb60-3f14-493d-90da-0034aca34b55/Key1
Request Method:GET
```

Result

```
Value1
```

Query (one binary key)

Queries all of the keys in a storage area that has already been created in GMS.

Operation

Method	GET		
URL	/genesys/1/storage/binary/**{id}/{key}**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{id}	string	yes	The id of the allocated storage to be updated.
{key}	string	yes	The key of the specificaly requested value

Method	GET
Body: Not used	

Response

HTTP code	200
HTTP message	OK
Body	The file that was stored for the specified key.

Example

The following example queries the value of Key1 from the data associated with id b8e8eb60-3f14-493d-90da-0034aca34b55

Operation

```
Request URL:http://localhost:8080/genesys/1/storage/binary/
b8e8eb60-3f14-493d-90da-0034aca34b55/FileKey
Request Method:GET
```

Delete

Deletes all of the keys in a storage area that has already been created in GMS.

Operation

Method	DELETE		
URL	/genesys/1/storage/**{id}**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{id}	string	yes	The id of the allocated storage to be deleted.
Body: Not used			

Response

HTTP code	200
HTTP message	OK

Example

The following example deletes all of the keys associated with id b8e8eb60-3f14-493d-90da-0034aca34b55

Operation

```
Request URL:http://localhost:8080/genesys/1/storage/b8e8eb60-3f14-493d-90da-0034aca34b55
Request Method:DELETE
```

Configuration

This section will describe configuration that is specific to this API.

Samples

Storage API HTML Sample

Notes

The Time-To-Live of the data is expressed as a 32-bit integer in seconds. This gives a max value of around 65 years.

Keys may not begin with an _.

Storage API HTML Sample

```
<meta http-equiv="content-script-type" content="text/javascript">
<script src="http://code.jquery.com/jquery-1.7.1.min.js"
        type="text/javascript"></script>
<script type="text/javascript">
jQuery.support.cors = true;

var storageId = "";
var defaults = {
                Key1: 'Value1',
                Key2: 'Value2',
                Key3: 'Value3',
                FileKey: 'FileKey',
        ttl: 3600,
    CreateData: '{ "a":"valuea", "b":"valueb", "c":"valuec" }',
        UpdateData: '{ "a":"new_valuea", "d":"new_valued" }'
};
function doPost( url, callback )
{
        var data = new Object();
        data[ 'a' ] = $("#Key1").val();
        data[ 'b' ] = $("#Key2").val();
        data[ 'c' ] = $("#Key3").val();

        $.post( url, data, callback );
        return;
}
function query() {
        $.get( '/genesys/1/storage/' + storageId, function(data) {
                        $("#query_result_label").text( JSON.stringify( data ) );
                });
}
function create() {
        doPost('/genesys/1/storage/' + $("#ttl").val(), function( result ) {
                storageId = result.id;
                $("#storage_id_label").text( storageId );
        });
}
function update() {
        if ( storageId == '' ) return;
        doPost('/genesys/1/storage/' + storageId + "/" + $("#ttl").val() );
}
function del() {
        $.ajax({
                  type: 'DELETE',
                  url: '/genesys/1/storage/' + storageId
                });
}
$(function(){
    $("#Control input").each(function () {
        $(this).val(defaults[this.id]);
    });
    $("#create").click(function () {
                create();
    });
    $("#query").click(function () {
                query();
    });
    $("#update").click(function () {
                update();
    });
    $("#delete").click(function () {
                del();
    });
});
</script>
<b>GSG Storage Test Controls</b>
<div id="Control">
        <div>
                <label for="ttl">TTL</label><input id="ttl">
        </div>
        <div>
                <label for="Key1">Key1</label><input id="Key1">
        </div>
        <div>
                <label for="Key2">Key2</label><input id="Key2">
        </div>
        <div>
                <label for="Key3">Key3</label><input id="Key3">
        </div>
</div>
<button id="create">Create</button>
<button id="update">Update</button>
<button id="query">Query</button>
<button id="delete">Delete</button>
<p />
<div>Storage id:</div>
<div id="storage_id_label"></div>
<div>Query results:</div>
<div id="query_result_label"></div>
<div></div>
```

Service API

Overview

This API is used by customer facing applications to manage different type of contact center related services (for example the app-to-connect-basic service provides the necessary contact center access information so the end user and associated application can initiate an interaction with the contact center).

API

Create

This API creates and initiates a service. It will support the creation and initiation of an service that is configured in Genesys Mobile Services. For details on the supported set of services, see the Overview of Services.

Operation

Method	POST		
URL	/genesys/1/service/**{service}**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{service}	string	yes	The name of the service that is to be created and initiated.
Body: The body can be either a MultiPart form or x-www-form-urlencoded form consisting of different items representing the key/value pairs associated with the given service type. In the case of MultiPart, the values can be strings or files but with urlencoded the values can only strings.			

Response

HTTP code	200
HTTP message	OK
Body	A JSON object with the following: {"id": **${service_id}, {service_specific_data'**}'} where: • ${service_id} is the identifier assigned to the created service instance.

HTTP code	200
	• ${service_specific_data} is service specific data that can be returned when the service is created.

If a matching services does not find a match, it will return the following status code.

HTTP code	404
HTTP message	Not Found

Example

The following example starts a request-interaction service:

- with the end user's phone number and application data: current user's location, preferred language, and end user's picture.

Operation

```
Request URL:http://localhost:8080/genesys/1/service/request-interaction
Request Method:POST
Status Code:200 OK
Request Headersview source
Accept:*/*
Accept-Charset:ISO-8859-1,utf-8;q=0.7,*;q=0.3
Accept-Encoding:gzip,deflate,sdch
Accept-Language:en-US,en;q=0.8
Connection:keep-alive
Content-Length:13028
Content-Type:multipart/form-data; boundary=----WebKitFormBoundaryy16qocbN6tmPORZL
Host:localhost:8080
Origin:http://localhost:8080
Referer:http://localhost:8080/GMS-web/resources/servicetest.html
User-Agent:Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.7 (KHTML, like Gecko) Chrome/
16.0.912.77 Safari/535.7
Request Payload
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="_phone_number"
6504669999
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="_provide_code"
true
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="language"
french
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="current_location_latitude"
48.8583
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="current_location_longitude"
2.2944
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="FileKey"; filename="MyPic.png"
Content-Type: image/png
------WebKitFormBoundaryy16qocbN6tmPORZL--
```

Result
The above service will be started with an id of 39a98e24-b03b-4191-b756-1efe8f3b16b8.

```
HTTP 200 OK
{ "_id":"39a98e24-b03b-4191-b756-1efe8f3b16b8","_access_number":"8003449999",
_access_code="7684"  }
```

Service specific Request

This API allows the application to perform a specific request against given service. For details on the specific requests that can be performed for a given type of service, see the Overview of Services.

Important Note: only to be used for `request-inbound-...` or `request-outbound-...` based services otherwise it will be rejected.

Operation

Method	POST		
URL	/genesys/1/service/***{service_id}***/**{request}**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{service_id}	string	yes	The id of the service that is to be deleted.
{request}	string	yes	This is the name of the request that is to be performed.
Body: The body can be either a MultiPart form or x-www-form-urlencoded form consisting of different items representing the key/value pairs associated with the given service request. In the case of MultiPart, the values can be strings or files but with urlencoded the values can only strings.			

Response

HTTP code	200
HTTP message	OK
Body	This will contain the appropriate output data (JSON data) that is defined by the given service request definition.

Example

The following example requests status of a `request-inbound-poll` service:

Operation

```
Request URL:http://localhost:8080/GMS-web/gms/1/service/39a98e24-b03b-4191-b756-1efe8f3b16b8/
status
Request Method:POST
Status Code:200 OK
Request Headersview source
Accept:*/*
Accept-Charset:ISO-8859-1,utf-8;q=0.7,*;q=0.3
```

```
Accept-Encoding:gzip,deflate,sdch
Accept-Language:en-US,en;q=0.8
Connection:keep-alive
Host:localhost:8080
Origin:http://localhost:8080
Referer:http://localhost:8080/GMS-web/resources/servicetest.html
User-Agent:Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.7 (KHTML, like Gecko) Chrome/
16.0.912.77 Safari/535.7
Request Payload
```

Result The status for service 39a98e24-b03b-4191-b756-1efe8f3b16b8 has been retreived.

```
HTTP 200 OK
{ "status":"waiting-for-agent", "dialog":"/waitingforagent.html" }
```

Notes

Parameters that begin with an underscore (_) are passed to ORS. Anything else is considered user data, and is saved in storage. The stored data can be retrieved using _data_id parameter passed in scxml.

Node API

Overview

This is a base ping API implementation which will be used by load balancers to determine the health of a given GMS node to determine if it can use this GMS node when it load balancing API requests across the set of GMS nodes.

API

Query

This API queries the status of a given GMS node. The application (load balancer) querying the nodes must have their explicit host addresses and port.

Operation

Method	GET		
URL	/genesys/1/node		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
None			
Body: None			

Response

HTTP code	**200**
HTTP message	**OK**

Notes

user defined parameters may not begin with an _. This is reserved for service specific parameters.

Notification API

Overview

Note: This API should not be used from external mobile applications. It is designed and intended for use only from Orchestration Server-based Services. In release 8.1.100.28, comet was added as a notification subscription for device_os parameters.

This set of APIs is used to manage notifications between applications and Genesys systems. It is event driven. That is, consumers subscribe to an event and provide an indication of how the notification should be delivered, and events are published to the system. For the GSM, delayed use case, it could work as follows:

1. The mobile application would trigger a subscription for an ORS event; something like ors.contact.12345678; the application would specify the device id and the type (ie. iOS).
2. When ORS determines that an agent is available or will be available soon it will push a message to GMS with the event ors.contact.12345678.
3. GMS would push the message to the mobile device.

Structures

The following are the API data structures. All structures are in JSON format. The servlet expects JSON (consumes = "application/json"), so media type **application/json** is expected. It's absence or incorrect value can result in 415 (Unsupported Media Type) error.

subscription

The subscription data is used to identify the subscriber of the given set of events.

subscription request

```
{ "subscriberId":"${subscriberId}",
  "providerName":"${providerName}",
  "notificationDetails":{
    "deviceId":"${id}",
    "properties":
       {"${key2}":"${val2}",
        "debug":"${debug}",
        "${key1}":"${val1}"},
    "type":"${type}"},
   "expire":30,
   "filter":"${filter}"
}
```

Here:

- **subscriberId**– id of subscriber (mandatory)
- **expire** - this parameter defines the time in seconds after which the subscription expires (not mandatory, default value is configurable)
- **filter** - (mandatory) the filter which is applied to the tags of incoming events. If filter matches the tag - the event will be published to destination, specified by subscription. Note: event is published to ALL subscription which specify the matching filter. The format of filter see further.
- **providerName** – this is the name of the provider which this subscription is for (optional). If not specified, the subscription is for default provider.
- **language** - (optional) describes the language used by this subscription. if not present, GMS will treat localizedstring as a normal message. Please see Push Notification Service for details on language.
- **notificationDetails** - (mandatory) describes all the information needed for delivering the event to concrete subscriber
    - type - (mandatory) this parameter defines what type of notification mechanism that the application wants to use. Valid values are **ios** (to-apple), **android** (to-android-device), gcm (to-android-gcm-device), **comet** (to-cometd-client), **httpcb** (callback POST to provided url) and **orscb** (callback to ORS) (see more information here Push Notification Service).
    - deviceId - (mandatory), id of device to deliver message to (in the case of http or ORS callback - see the details here Push Notification Service
    - properties – (optional) the String-String map of properties – additional properties which can be needed for notification delivering. If the information provided is not enough for corresponding publisher – the error will be returned.
        - **debug** - this indicates if the production or debug provider connection is to be used to send the notifications. The subscription will be sent to debug channel if ${debug} value is **debug** or **true**.

Please note that notificationDetails.properties are not needed for **android, gcm** or **ios** or **httpcb** or **orscb** notifications - for them only the correct **deviceId** is required. Both notificationDetails.properties and deviceId is not needed for **comet** but **gms_user** header is required. For example, the request for android push notification subscription might look like this (please note absence of *properties* entry):

```
{"subscriberId":"$The_subscriber_9774",
 "notificationDetails":{
    "deviceId":"9774d56d682e549c",
    "type":"android"},
 "expire":30,
 "filter":"ors.context.123456"}
```

subscription response

If OK:

```
{"id":"${id}"}
```

- returns the ID of created subscription.

event

The events are published by internal Enterprise components. The Notification service matches the event to subscription using event's tag and subscriptions filters and notifies the subscriptions with matching filters. Event looks like this:

```
{
  "message":"${message}",
  "tag":"${tag}",
  "mediaType":"${mediaType}",
  "notificationDetails":
  {
    "deviceId":"${devideId}",
    "type":"${type}",
    "properties":
    {
     "debug":"{true/false}"
    }
  }
}
```

- tag – (mandatory) the message tag
- message– (optional) some string. It may contain string representation of ANY data: notification service is message-agnostic – it ALWAYS interprets message as String. If no message is specified – the empty string will be sent to subscribers. The only restriction on **message** format is: it must not crash the JSON parser which attempts to parse the request body. If this happens - the BAD_REQUEST response will be returned.
- mediaType - (optional) "**string**" for a simple string, "**localizestring**" for a string with localized parameter. see Genesys_Mobile_Services_Localization_File
- providerName - (optional) this is the name of the provider which this subscription is for (optional). If not specified, the subscription is for default provider.
- notificationDetails - (optional) if not present we send notification to default subscribers. It allows to send notification to a specific device.
- devideId - (mandatory) the id of the device (Android device id or IPad id for instance).
- type - (mandatory) type of the notification (gcm, ios...).
- properties - (optional).
- debug - (optional) allow to display debug log for this notification.

Filters and tags

Tag can not be null or empty string. The format of tag, specified in event, is like the java package name alphanumeric string with '.' delimiters. Underscores are allowed and first symbol may be number. Please note: at the moment only English alphanumeric chars are allowed. The filter can not be null or empty string. The format of filter entry is similar to the tag format, but in addition allowed wildcart ‘*’ after last ‘.’ (or only ‘*’ – denotes subscription to all events), the last char can not be '.'. So, the channels like the following are allowed:

- * - subscription to all channels
- ors.* - subscriptions to all channels starting with ors.
- ors.events.agentavailabilty.context.1234560 – subscription to the only 1 channel specified.

When publishing event - the tag is matched versus the filters of all active subscriptions and all matching subscriptions are notified (the best we can: push delivery is not 100% reliable). For example, consider the Notification Event published with tag **ors.agentavailability.agent123.available**. Such notification will be propagated to the subscriptions with any of following filters:

- *
- ors.*
- ors.agentavailability.*
- ors.agentavailability.agent123.*
- ors.agentavailability.agent123.available

APIs

The standard InternalServerError with code 500 or BAD_REQUEST with code 400 can be returned as response to each request, so it's not mentioned in further descriptions (except some cases when syntax of body is involved). NOTE: this API is intended for internal usage. NOTE: All POST request MUST specify media type **application/json**.

Create subscription

This allows an application to subscribe to a given set of events.

Operation

Method	POST		
URL	/genesys/{api version}/notification/subscription		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
Body: JSON with subscription (see above)			

Response

A JSON object with the property id, identifying the assigned id for this storage request.

HTTP code	200
HTTP message	OK

In the case of incorrect request syntax (see requirements above) the BAD_REQUEST error will be returned.

HTTP code	400
HTTP message	BAD REQUEST

If the subscription is being created for the push type which is not enabled at the moment, the NOT_FOUND error will be returned.

HTTP code	404
HTTP message	NOT FOUND

Delete subscription

This call cancels/terminates a given subscription.

Operation

Method	DELETE		
URL	/genesys/{api version}/notification/subscription/**{subscription-id}**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{subscription-id}	String	yes	the id of the subscription to cancel

Returns

Nothing

HTTP code	200
HTTP message	OK

Delete subscription for given subscriber

This call cancels/terminates all subscription for a given subscriber.

Operation

Method	DELETE		
URL	/genesys/{api version}/notification/subscription/subscriber/**{subscriberId}**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			

Method	DELETE		
{subscriberId}	String	yes	the id of the subscriber whose subscriptions will be cancelled

Returns

Nothing

HTTP code	200
HTTP message	OK

Publish event

This allows an application to publish event (for internal usage only!).

Operation

Method	POST		
URL	/genesys/{api version}/notification/publish		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
Body: JSON with event(see above)			

Response

Nothing

HTTP code	200
HTTP message	OK

In the case of incorrect request body syntax (see requirements above) the BAD_REQUEST error will be returned.

HTTP code	400
HTTP message	BAD REQUEST

In the case if error happened during notification publishing (http post to specified url failed or returned not 200, some network troubles, APNS or C2DM services reported some error (authorization problems, temporary service unavailability for C2DM, e.t.c.) the SERVICE_UNAVAILABLE error will be returned.

HTTP code	503
HTTP message	SERVICE UNAVAILABLE

For more details please see Push Notification Service

Configuration

Configuring Genesys Mobile Services

Links

Push Notification Service

Chat API

Introduced in release: **8.1.100.14**

Overview

Prerequisite: Before using the Chat API, you must configure your Genesys Mobile Services deployment correctly.

The Chat API is used by customer-facing applications to create and manage a chat session associated with contact center-related services. One service is associated with exactly one chat session. There are two types of supported services:

1. Basic Services (Genesys Mobile Services-Based)

- allows an application to pass business context data in the service creation request, using the fixed service name `request-chat`
- no corresponding Orchestration Server (ORS) session will be created
- data is going to be preserved by Genesys Mobile Services using the specified time to live parameter (or the configured default value)
- chat interaction could be initiated by an application at any point
- routing logic associated with specified interaction endpoint (or the configured by default value) would be responsible for finding an appropriate agent
- both polling and async (CometD) modes of message delivery are supported

2. Advance Services (Genesys Mobile Services + ORS-Based)

- allows an application to define custom service processing logic
- an application can create a service session and pass business context data
- session logic can use agent availability information and notify the application when a chat-enabled resource with the appropriate business skills is available
- session logic can also engage in pre- and post-chat interaction dialogs with the customer by pushing dialog information to the client application (for example, in URL/HTML/XML form)
- `advance-chat-pool` session is provided as an example of the advanced service logic and can be customized to fit particular business needs
- both polling and async (CometD) modes of message delivery are supported

Important Note: When using asynchronous messaging with CometD, all HTTP headers must include the `gms_user` header.

Structures

The Chat API uses the data structures described in this section (in JSON format) to exchange data. Requests are accepted in **'application/x-www-form-urlencoded'** or **'multipart/form-data'** formats, and responses are returned in **'application/json'** format. If an expected value is missing or incorrect, then a 415 (Unsupported Media Type) error will occur.

Chat Interaction API Resources

The chat interaction is used to represent the current state of the chat session and transcript. This information is returned in the HTTP response of each API request in poll mode or delivered asynchronously in push mode (CometD). Note: you need to create service session before you can create chat interaction.

Create Chat Interaction: /genesys/1/service/{sessionid}/ixn/chat?notify_by=comet&firstName=Buzz&lastName=Brain⊂ject=French&email=b.b%40gmail.com

```
{
 "chatIxnState" : "CONNECTED",
 "chatSessionId": "000C2a7VVQRB001U",
 "transcriptPosition" : 1,
 "chatServiceMessage" : "Chat service is available"
}
```

Create Chat Interaction: /genesys/1/service/{sessionid}/ixn/chat?_verbose=true¬ify_by=comet&firstName=Buzz&lastName=Brain⊂ject=French&email=b.b%40gm

```
{
   "chatIxnState": "CONNECTED",
   "chatSessionId": "000C2a7VVQRB001U",
   "transcriptPosition": "1",
   "chatServiceMessage": "Chat service is available",
   "userId": "015E4FD3CD890036",
   "secureKey": "b306749dabfa1cf6",
   "checkChatServiceLoadBalancerPath": "/WebAPI812/SimpleSamples812/ChatHA/
ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=350",
  "chatServerLoadBalancerAlias": "350",
  "chatServerHost": "135.225.51.225",
  "chatWebApiPort": "4856",
  "isTLSrequired": "false",
  "clientTimeZoneOffset": "-420",
  "_chatIxnAPI_SEND_URL": "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/chat/
send",
  "_chatIxnAPI_REFRESH_URL": "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/
chat/refresh",
  "_chatIxnAPI_START_TYPING_URL": "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/
ixn/chat/startTyping",
  "_chatIxnAPI_STOP_TYPING_URL": "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/
chat/stopTyping",
  "_chatIxnAPI_DISCONNECT_URL": "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/
chat/disconnect",
  "_chatIxnAPI_REFRESH_FROM_START_URL": "/genesys/1/service/
9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/chat/refresh?transcriptPosition=1"
}
```

Attribute Descriptions:

- chatIxnState – The current state of the chat session.
- chatSessionId – Session ID associated with the chat.
- transcriptPosition – The current position in the chat dialog or transcript for this user.
- chatServiceMessage – A diagnostic message used for debugging.

The following are only returned if the `_verbose` parameter in the API request is true:

- userId – User ID assigned by the Genesys Chat Server.
- secureKey – The security key for this chat session.
- checkChatServiceLoadBalancer – Indicates that we should check the chat load balancer for the appropriate Chat Server to use.
- PathchatServerLoadBalancerAlias – The alias for the Chat Server that is assigned to this chat session by the Chat Server load balancer.
- chatServerHost – Host name for the Chat Server assigned to this chat session from the Chat Server load balancer.
- chatWebApiPort – Port number of the Chat Server load balancer
- isTLSrequired – Indicates whether a TLS connection is required for the Chat Server.
- clientTimeZoneOffset - Time zone offset specified by the user client. Could be used to convert UTC time returned by server into user local time.
- _chatIxnAPI_SEND_URL – URL used to send chat messages for this chat session.
- _chatIxnAPI_REFRESH_URL – URL used to refresh the chat transcript for this chat session.
- _chatIxnAPI_START_TYPING_URL – URL used to indicate that the user started typing a chat message for this chat session.
- _chatIxnAPI_STOP_TYPING_URL – URL used to indicate that the user stopped typing a chat message for this chat session.
- _chatIxnAPI_DISCONNECT_URL – URL used to disconnect the user from the chat session.
- _chatIxnAPI_REFRESH_FROM_START_URL – URL used to refresh the chat transcript from the beginning of the session.

Refresh Chat Transcript: /genesys/1/service/{sessionid}/ixn/chat/refresh?message=hello%20agent

```
{
 "chatIxnState" : "TRANSCRIPT",
 "chatSessionId" :"000BRa84KRFB00BK",
 "transcriptPosition" : 5",
 "chatServiceMessage" : "Chat service is available",
 "transcriptToShow" : [["Notice.Joined","ksippo","has joined the
session","35","AGENT"],["Notice.TypingStarted","ksippo","is
typing","42","AGENT"],["Message.Text","ksippo","hello
customer","48","AGENT"],["Message.Text","VasyaP","hello agent","71","CLIENT"]]",
 "startedAt" : 2012-06-09T06:15:35Z"
}
```

Refresh chat transcript: /genesys/1/service/{sessionid}/ixn/chat/refresh?_verbose=true&message=hello%20agent

```
{
   "chatIxnState": "TRANSCRIPT",
   "chatSessionId":"000BRa84KRFB00BK",
   "transcriptPosition": "5",
   "chatServiceMessage": "Chat service is available",
   "transcriptToShow":    [
            [
         "Notice.Joined",
         "ksippo",
         "has joined the session",
         "15",
         "AGENT"
      ],
            [
         "Message.Text",
         "VasyaP",
         "hello agent",
         "26",
         "CLIENT"
      ],
            [
         "Notice.TypingStarted",
         "ksippo",
         "is typing",
         "57",
         "AGENT"
      ],
            [
         "Message.Text",
         "ksippo",
         "hello customer",
         "61",
         "AGENT"
      ]
   ],
   "startedAt": "2012-06-09T22:26:17Z",
   "userId": "015E4FD3CD890036",
   "secureKey": "b306749dabfa1cf6",
   "checkChatServiceLoadBalancerPath": "/WebAPI812/SimpleSamples812/ChatHA/
ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=350",
   "chatServerLoadBalancerAlias": "350",
   "chatServerHost": "135.225.51.225",
   "chatWebApiPort": "4856",
   "isTLSrequired": "false",
   "clientTimeZoneOffset": "-420",
   "_chatIxnAPI_SEND_URL": "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/chat/
send",
   "_chatIxnAPI_REFRESH_URL": "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/
chat/refresh",
   "_chatIxnAPI_START_TYPING_URL": "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/
ixn/chat/startTyping",
   "_chatIxnAPI_STOP_TYPING_URL": "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/
ixn/chat/stopTyping",
   "_chatIxnAPI_DISCONNECT_URL": "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/
chat/disconnect",
   "_chatIxnAPI_REFRESH_FROM_START_URL": "/genesys/1/service/
9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/chat/refresh?transcriptPosition=1"
}
```

Attributes description:

- startedAt - Chat interaction start time (in UTC).

- transcriptToShow - An ordered array of transcript events. Each event is represented by another array of the following format:
  `[{Event type}, {Agent nickname}, {Chat message}, {Number of seconds from interaction start}, {Type of user}]`
  Where:
  - Event type: `{"Message.Text", "Notice.Joined", "Notice.Left", "Notice.TypingStart", "Notice.TypingStop", "Notice.PushUrl"}`
  - Type of user: `{"AGENT", "CLIENT", "EXTERNAL"}`

Service API Resources

Basic Chat: genesys/1/service/request-chat

```
{
   "_id": "a7e6ed0b-0380-4223-97f8-b3c7d93205e8"
}
```

Basic Chat: genesys/1/service/request-chat?_verbose=true

```
{
   "_chatIxnAPI-CREATE-URL": "/genesys/1/service/a7e6ed0b-0380-4223-97f8-b3c7d93205e8/ixn/
chat",
   "_id": "a7e6ed0b-0380-4223-97f8-b3c7d93205e8"
}
```

Advanced Chat Poll: genesys/1/service/advanced-chat-poll

```
{
   "_id": "EKUJPKAQ197CFA6SJQKTJ03DBG00001C"
}
```

Advanced Chat Poll: genesys/1/service/advanced-chat-poll?_verbose=true

```
{
   "_id": "EKUJPKAQ197CFA6SJQKTJ03DBG00001C",
   "_session_comand_STATUS_URL": "/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001C/status"
}
```

Advanced Chat Poll with poke enabled: genesys/1/service/request-chat-poll?_device_notification_id=b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb

```
{
   "_id": "EKUJPKAQ197CFA6SJQKTJ03DBG00001C",
   "_session_comand_STATUS_URL": "/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001C/status"
}
```

COMETD Based Chat API

CometD handshake request:

```
POST http://localhost:8080/genesys/cometd
Content-Type:        application/json; charset=UTF-8
gms_user:        BuzzBrain

{"version":"1.0","minimumVersion":"0.9","channel":"/meta/handshake","id":"0"}
```

Response:

```
Content-Type:       application/json;charset=UTF-8

[{"id":"0","minimumVersion":"1.0","supportedConnectionTypes":["callback-polling","long-
polling"],"successful":true,"channel":"/meta/
handshake","ext":{"ack":true},"clientId":"d1jw6puyen98c1baidgmlcivxs","version":"1.0"}]
```

CometD connect request:

```
POST http://localhost:8080/genesys/cometd
Content-Type:       application/json; charset=UTF-8
gms_user:       BuzzBrain

{"channel":"/meta/
connect","clientId":"d1jw6puyen98c1baidgmlcivxs","id":"1","connectionType":"long-polling"}
```

Response:

```
Content-Type:       application/json;charset=UTF-8

[{"id":"1","successful":true,"advice":{"interval":0,"reconnect":"retry","timeout":60000},"channel":"/meta/
connect"}]
```

CometD subscribe:

```
POST http://localhost:8080/genesys/cometd
Content-Type:       application/json; charset=UTF-8
gms_user:       BuzzBrain

[{"channel":"/meta/
subscribe","subscription":"/_genesys","clientId":"eurw5v1ezp5yn17z3cdrf3jxsc","id":"2"}]
```

Response:

```
Content-Type:       application/json;charset=UTF-8

[{"id":"3","successful":true,"channel":"/meta/connect"}]
```

CometD polling:

```
POST http://localhost:8080/genesys/cometd
Content-Type:       application/json; charset=UTF-8
gms_user:       BuzzBrain

{"clientId":"eurw5v1ezp5yn17z3cdrf3jxsc","id":"4","channel":"/meta/
connect","connectionType":"long-polling"}
```

Response:

```
Content-Type:       application/json;charset=UTF-8

[{"id":"4","successful":true,"channel":"/meta/connect"}]
```

... continue cometd polling ...

Create service session:

```
POST http://localhost:8080/genesys/1/service/request-chat
Content-Type:        application/x-www-form-urlencoded; charset=UTF-8
gms_user:        BuzzBrain

_verbose=true
```

Response:

```
Content-Type:        application/json;charset=UTF-8

{"_chatIxnAPI-CREATE-URL":"/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/
chat","_id":"0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f"}
```

Create chat interaction for session 0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f:

```
POST http://localhost:8080/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat
Content-Type:        application/x-www-form-urlencoded; charset=UTF-8
gms_user:        BuzzBrain

_verbose=true¬ify_by=comet&firstName=Buzz&lastName=Brain⊂ject=French&email=b.b%40gmail.com
```

Response:

```
Content-Type:        application/json;charset=UTF-8

{"message":"Messages will sent over CometD service channel \/_genesys"}
```

... continute cometd polling ...:

CometD polling:

```
POST http://localhost:8080/genesys/cometd
Content-Type:        application/json; charset=UTF-8
gms_user:        BuzzBrain

{"clientId":"eurw5v1ezp5yn17z3cdrf3jxsc","id":"32","channel":"/meta/
connect","connectionType":"long-polling"}
```

Response:

```
Content-Type:        application/json;charset=UTF-8

[{"data":{"id":"ccd229401d5f11e200000805691661bf","message":{"chatSessionId":"000BWa7RECRK001D","transcriptPosi
service is
available","startedAt":"2012-10-23T22:19:42Z","chatIxnState":"TRANSCRIPT","transcriptToShow":[]},"tag":"service
connect"}]
```

Polling agent 'Joined' message through CometD:

```
POST http://localhost:8080/genesys/cometd
Content-Type:        application/json; charset=UTF-8
gms_user:        BuzzBrain

{"clientId":"eurw5v1ezp5yn17z3cdrf3jxsc","id":"33","channel":"/meta/
connect","connectionType":"long-polling"}
```

Response:

```
Content-Type:        application/json;charset=UTF-8
```

```
[{"data":{"id":"3cc44da01d6011e200000805691661bf","message":{"chatSessionId":"000BWa7RECRK001D","transcriptPosi
service is
available","startedAt":"2012-10-23T22:19:42Z","chatIxnState":"TRANSCRIPT","transcriptToShow":[["Notice.Joined",
joined the
session","21","AGENT"]]},"tag":"service.chat.refresh.0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f"},"channel":"/_genesy
connect"}]]
```

Polling agent 'StartTyping' message through CometD:

```
POST http://localhost:8080/genesys/cometd
Content-Type:        application/json; charset=UTF-8
gms_user:        BuzzBrain

{"clientId":"eurw5v1ezp5yn17z3cdrf3jxsc","id":"34","channel":"/meta/
connect","connectionType":"long-polling"}
```

Response:

```
Content-Type:        application/json;charset=UTF-8

[{"data":{"id":"3cc44da01d6011e200000805691661bf","message":{"chatSessionId":"000BWa7RECRK001D","transcriptPosi
service is
available","startedAt":"2012-10-23T22:19:42Z","chatIxnState":"TRANSCRIPT","transcriptToShow":[["Notice.TypingSt
typing","212","AGENT"]]},"tag":"service.chat.refresh.0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f"},"channel":"/_genesy
connect"}]]
```

Polling agent chat message through CometD:

```
POST http://localhost:8080/genesys/cometd
Content-Type:        application/json; charset=UTF-8
gms_user:        BuzzBrain

{"clientId":"eurw5v1ezp5yn17z3cdrf3jxsc","id":"35","channel":"/meta/
connect","connectionType":"long-polling"}
```

Response:

```
Content-Type:        application/json;charset=UTF-8

[{"data":{"id":"3fc6b8301d6011e200000805691661bf","message":{"chatSessionId":"000BWa7RECRK001D","transcriptPosi
service is
available","startedAt":"2012-10-23T22:19:42Z","chatIxnState":"TRANSCRIPT","transcriptToShow":[["Message.Text","
connect"}]]
```

Send client chat message:

```
POST http://localhost:8080/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
refresh
Content-Type:        application/x-www-form-urlencoded; charset=UTF-8

_verbose=true&message=hello+agent
```

Response:

```
Content-Type:        application/json;charset=UTF-8

{"message":"Messages will sent over CometD service channel \/_genesys"}
```

Client message is being echoed back through CometD channel as a response to "refresh"

or "send" request:

```
POST http://localhost:8080/genesys/cometd
Content-Type:       application/json; charset=UTF-8
gms_user:       BuzzBrain

{"clientId":"eurw5v1ezp5yn17z3cdrf3jxsc","id":"36","channel":"/meta/
connect","connectionType":"long-polling"}
```

Response:

```
Content-Type:       application/json;charset=UTF-8

[{"data":{"id":"472e2a901d6011e200000805691661bf","message":{"chatServerLoadBalancerAlias":"349","clientTimeZon
SimpleSamples812/ChatHA/
ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=349","_chatIxnAPI_REFRESH_FROM_START_URL":"/genesys/
1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
refresh?transcriptPosition=1","_chatIxnAPI_REFRESH_URL":"/genesys/1/service/
0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
refresh","chatSessionId":"000BWa7RECRK001D","isTLSrequired":"false","_chatIxnAPI_DISCONNECT_URL":"/genesys/
1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
disconnect","userId":"015D508717FE0027","chatServiceMessage":"Chat service is
available","_chatIxnAPI_STOP_TYPING_URL":"/genesys/1/service/
0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
stopTyping","_chatIxnAPI_START_TYPING_URL":"/genesys/1/service/
0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
startTyping","transcriptToShow":[["Message.Text","BuzzB","hello
agent","230","CLIENT"]],"_chatIxnAPI_SEND_URL":"/genesys/1/service/
0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
send","chatServerHost":"demosrv.genesyslab.com"},"tag":"service.chat.refresh.0da34a9d-0c5e-46a6-953c-6cc1aa82bf
connect"}]
```

Disconnect chat session:

```
POST http://localhost:8080/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
disconnect
Content-Type:       application/x-www-form-urlencoded; charset=UTF-8

_verbose=true
```

Response:

```
Content-Type:       application/json;charset=UTF-8

{
 "chatIxnState" : "DISCONNECTED",
 "transcriptPosition" : "6",
 "chatServiceMessage" : "Chat was finished",
 "chatSessionId" : "000BWa7RECRK001D",
 "userId" : "015D508717FE0027",
 "secureKey" : "29a6758abacdd33e",
 "checkChatServiceLoadBalancerPath" : "/WebAPI812/SimpleSamples812/ChatHA/
ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=349",
 "chatServerLoadBalancerAlias" : "349",
 "chatServerHost" : "demosrv.genesyslab.com",
 "chatWebApiPort" : "4853",
 "isTLSrequired" : "false",
 "clientTimeZoneOffset" : "-420",
 "_chatIxnAPI_SEND_URL" : "/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
send",
 "_chatIxnAPI_REFRESH_URL" : "/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/
chat/refresh",
 "_chatIxnAPI_START_TYPING_URL" : "/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/
ixn/chat/startTyping",
 "_chatIxnAPI_STOP_TYPING_URL" : "/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/
chat/stopTyping",
 "_chatIxnAPI_DISCONNECT_URL" : "/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/
chat/disconnect",
 "_chatIxnAPI_REFRESH_FROM_START_URL" : "/genesys/1/service/
0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/refresh?transcriptPosition=1"
}
```

Quick Start Examples

The following quick start examples show how you can establish a CometD connection to receive asynchronous notification, and how to create a service.

Using CometD to Receive Event Updates

If you are using CometD to get event updates on the chat session then you need to set up a CometD connection with a subscription for `/_genesys`. You also need to make sure 'gms_user' header in all cometd related requests is set to the value uniquely representing application end user. Typically this value would be setup (or at least verified) by security gateway located between client application and GMS.

CometD handshake request

```
POST http://localhost:8080/genesys/cometd
Accept-Encoding: gzip,deflate
Content-Type: application/json;charset=UTF-8
gms_user: BuzzBrain
{"version":"1.0","minimumVersion":"0.9","channel":"/meta/handshake","id":"0"}

HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:30:10 GMT
Content-Type: application/json
Content-Length: 230
 [{"id":"0","minimumVersion":"1.0","supportedConnectionTypes":["websocket","callback-
polling","long-polling"], "successful":true,"channel":"/meta/handshake","ext":
"ack":true},"clientId":"44xkkazwfabw73jrvjsvoy4ul","version":"1.0"}]
```

CometD /meta/connect subscription request

```
POST  http://localhost:8080/genesys/cometd
Accept-Encoding: gzip,deflate
Content-Type: application/json;charset=UTF-8
gms_user: BuzzBrain
{"channel":"/meta/
connect","clientId":"44xkkazwfabw73jrvjsvoy4ul","id":"1","connectionType":"long-polling"}

HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:30:10 GMT
Content-Type: application/json
 Content-Length: 116
[{"id":"1","successful":true,"advice":{"interval":0,"reconnect":"retry","timeout":60000},"channel":"/meta/
connect"}]
```

CometD /_genesys subscription request

```
POST  http://localhost:8080/genesys/cometd Accept-Encoding: gzip,deflate
Content-Type: application/json;charset=UTF-8
gms_user: BuzzBrain
[{"channel":"/meta/
subscribe","subscription":"/_genesys","clientId":"44xkkazwfabw73jrvjsvoy4ul","id":"2"}]

HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:30:10 GMT
Content-Type: application/json
Content-Length: 85
[{"id":"2","subscription":"/_genesys","successful":true,"channel":"/meta/subscribe"}]
```

CometD long polling request

```
POST  http://localhost:8080/genesys/cometd
Accept-Encoding: gzip,deflate
Content-Type: application/json;charset=UTF-8
gms_user: BuzzBrain
{"clientId":"44xkkazwfabw73jrvjsvoy4ul","id":"3","channel":"/meta/
connect","connectionType":"long-polling"}

HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:30:10 GMT
Content-Type: application/json
Content-Length: 85
[{"id":"4","successful":true,"channel":"/meta/connect"}]
```

Creating a Genesys Mobile Services-Based Service Associated with a Chat Session

The following section illustrates the process of creating and using a service.

Create a Service:

Request:

```
POST http://localhost:8080/genesys/1/service/request-chat-poll HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/x-www-form-urlencoded

_verbose=false
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:23:29 GMT
Content-Type: application/json

{"_id":"EKUJPKAQ197CFA6SJQKTJ03DBG00001M"}
```

Use the `_id` field from the response to check service status until it changes to "available":

Request:

```
POST http://localhost:8080/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001M/status HTTP/1.1
Accept-Encoding: gzip,deflate
```

```
Content-Type: application/x-www-form-urlencoded
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:26:26 GMT
Content-Type: application/json
Content-Length: 185

{
   "message":    {
      "_id": "EKUJPKAQ197CFA6SJQKTJ03DBG00001M",
      "_status": "waiting",
      "_dialog": "waiting_for_agent.html"
   },
   "tag": "a2c.advanced.service.statuschanged.EKUJPKAQ197CFA6SJQKTJ03DBG00001M"
}
```

Repeat request until status changes:

Request:

```
POST http://localhost:8080/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001M/status HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/x-www-form-urlencoded
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:28:25 GMT
Content-Type: application/json
Content-Length: 186

{
   "message":    {
      "_id": "EKUJPKAQ197CFA6SJQKTJ03DBG00001M",
      "_status": "available",
      "_dialog": "agent_available.html"
   },
   "tag": "a2c.advanced.service.agentavailable.EKUJPKAQ197CFA6SJQKTJ03DBG00001M"
}
```

Create chat interaction using same sessionid: To create a chat interaction that is associated with a service, a ixn/chat request is sent with the parameters to initiate the chat session.

Parameter Name	Mandatory	Description
firstName	no	First name of the user. If provided will be attached as "fldnFirstName" to the chat interaction.
lastName	no	Last name of the user. If provided will be attached as "fldnLastName" to the chat interaction.
email	no	e-Mail address of the subject. If provided will be attached as "fldnEmailAddress" to the chat interaction.

Parameter Name	Mandatory	Description
subject	yes	Subject of the service and chat session.
userDisplayName	no	Available since GMS 8.1.100.28. Nickname to be displayed in the chat conversation.
notify_by	false	If using a CometD connection for the asynchronous receiving of chat messages, then supply this parameter with the value "comet".

Request:

```
POST http://localhost:8080/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001M/ixn/chat HTTP/
1.1
Accept-Encoding: gzip,deflate
Content-Type: application/x-www-form-urlencoded

notify_by=comet&firstName=Vasya&lastName=Pupkin&email=Vasya.Pupkin@genesyslab.com⊂ject=test
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:30:10 GMT
Content-Type: application/json
Content-Length: 119

{
 "chatIxnState" : "CONNECTED",
 "transcriptPosition" : "1",
 "chatServiceMessage" : "Chat service is available"
}
```

Refresh chat transcript and show messages to the user:

Request:

```
POST http://localhost:8080/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001M/ixn/chat/
refresh HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/x-www-form-urlencoded
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:33:00 GMT
Content-Type: application/json
Content-Length: 367

{"_id":"B2FS3346K151548QMEAFD89TE8000EBJ","comet_channel":"/_genesys"}
```

Send user's message:

Request:

```
POST http://localhost:8080/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001M/ixn/chat/
```

```
refresh HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/x-www-form-urlencoded

message=hello agent
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:34:38 GMT
Content-Type: application/json
Content-Length: 241

{"_id":"B2FS3346K151548QMEAFD89TE8000EBJ","comet_channel":"/_genesys"}
```

Disconnect user from chat:

Request:

```
POST http://localhost:8080/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001M/ixn/chat/
disconnect HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/x-www-form-urlencoded
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 07:43:07 GMT
Content-Type: application/json
Content-Length: 114

{
 "chatIxnState" : "DISCONNECTED",
 "transcriptPosition" : "9",
 "chatServiceMessage" : "Chat was finished"
}
```

Chat Interaction APIs

Start Chat

This API creates and initiates a Chat Session. It works with the service session created through Genesys Mobile Services.

Note: In release **8.1.100.28**, the ixn/chat parameters are now optional, and added parameter userDisplayName.

Operation

Method	POST		
URL	/genesys/1/service/***{service_id}***/ixn/***chat***		
Parameter	**Type**	**Mandatory**	**Description**

Method	POST		
URI Parameters			
{service_id}	string	yes	The identifier of the service that the chat session is suppose to be associated with.
Body: The body will be `x-www-form-urlencoded` form consisting of different items representing the key/value pairs associated with the request.			
Body Properties: The following are the properties: • _verbose - This will allow the application to get all the detail attributes associated with the chat session in the corresponding response. • notify_by - If specified should be "comet". • firstName - User's first name. Optional. • lastName - User's last name. Optional. • email - User's email address. Optional. • subject - Subject of the chat conversation. • userDisplayName - Nickname displayed in the chat conversation. Optional.			

Response

HTTP code	200
HTTP message	OK
Body	A chat JSON object for details on the properties of the object. See the section on data structures for more details.
Notes	The chat session id will be the service ID. The Genesys Mobile Services code for this API will keep track of the service ID to the chat server session.

HTTP code	503
HTTP message	Service Unavailable
Body	None
Notes	Returned if the service has not sent a notification to the application that an agent is available.

Example

Request:

```
POST http://localhost:8080/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/
chat?_verbose=true HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/<code>x-www-form-urlencoded</code>

firstName=Vasya&lastName=Pupkin&email=Vasya.Pupkin@genesyslab.com⊂ject=test
```

Response:

```
HTTP/1.1 200 OK
Date: Sat, 09 Jun 2012 22:26:16 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Content-Length: 1225
Server: Jetty(7.6.0.v20120127)

{
 "chatIxnState" : "CONNECTED",
 "chatSessionId" : "000C2a7VVQRB001U",
 "transcriptPosition" : "1",
 "chatServiceMessage" : "Chat service is available",
 "userId" : "015E4FD3CD890036",
 "secureKey" : "b306749dabfa1cf6",
 "checkChatServiceLoadBalancerPath" : "/WebAPI812/SimpleSamples812/ChatHA/
ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=350",
 "chatServerLoadBalancerAlias" : "350",
 "chatServerHost" : "135.225.51.225",
 "chatWebApiPort" : "4856",
 "isTLSrequired" : "false",
 "clientTimeZoneOffset" : "-420",
 "_chatIxnAPI_SEND_URL" : "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/chat/
send",
 "_chatIxnAPI_REFRESH_URL" : "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/
chat/refresh",
 "_chatIxnAPI_START_TYPING_URL" : "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/
ixn/chat/startTyping",
 "_chatIxnAPI_STOP_TYPING_URL" : "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/
chat/stopTyping",
 "_chatIxnAPI_DISCONNECT_URL" : "/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/
chat/disconnect",
 "_chatIxnAPI_REFRESH_FROM_START_URL" : "/genesys/1/service/
9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/chat/refresh?transcriptPosition=1"
}
```

Refresh Chat

This API refreshes the users view with the latest updates to the Chat session. It can also be used to simultaneously send a user message to the chat session.

Operation

Method	POST		
URL	/genesys/1/service/***{service_id}***/**ixn**/***chat***/**refresh**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{service_id}	string	yes	The identifier of the service that the chat session is associated with.
Body: The body will be `x-www-form-urlencoded` form consisting of different items representing the key/value pairs associated with the request.			

Method	POST

Body Properties: The following are the properties:

- transcriptPosition – This optional property indicates the current position in the chat session that the current user is in.
- message – This optional property is a chat message that will be added to the chat session/transcript.
- _verbose - This optional property will allow the application to get all the detail attributes associated with the chat session in the corresponding response.

Response

HTTP code	200
HTTP message	OK
Body	A chat JSON object for details on the properties of the object. See the section on data structures for more details.
Notes	The main property is the list of chat message that have been communicated (transcriptToShow).

HTTP code	503
HTTP message	Service Unavailable
Body	None
Notes	This is returned if the service has not sent a notification to the application that an agent is available.

Example

Request:

```
POST http://localhost:8080/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001H/ixn/chat/
refresh?_verbose=true HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/<code>x-www-form-urlencoded</code>
message=aaa
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 06:50:49 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Content-Length: 1495
Server: Jetty(7.6.0.v20120127)
{
   "chatIxnState": "TRANSCRIPT",
   "chatSessionId": "000C2a7VVQRB001W",
   "transcriptPosition": "5",
   "chatServiceMessage": "Chat service is available",
   "transcriptToShow":    [
            [
         "Notice.Joined",
         "ksippo",
         "has joined the session",
         "18",
         "AGENT"
      ],
            [
         "Notice.TypingStarted",
         "ksippo",
         "is typing",
         "21",
         "AGENT"
      ],
            [
         "Message.Text",
         "ksippo",
         "hello customer",
         "28",
         "AGENT"
      ],
            [
         "Message.Text",
         "VasyaP",
         "aaa",
         "172",
         "CLIENT"
      ]
   ],
   "startedAt": "2012-06-10T06:47:58Z",
   "userId": "015E4FD4431D0039",
   "secureKey": "4ba8838b7cf9afc4",
   "checkChatServiceLoadBalancerPath": "/WebAPI812/SimpleSamples812/ChatHA/
ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=350",
   "chatServerLoadBalancerAlias": "350",
   "chatServerHost": "135.225.51.225",
   "chatWebApiPort": "4856",
   "isTLSrequired": "false",
   "clientTimeZoneOffset": "-420",
   "_chatIxnAPI_SEND_URL": "/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001H/ixn/chat/
send",
   "_chatIxnAPI_REFRESH_URL": "/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001H/ixn/chat/
refresh",
   "_chatIxnAPI_START_TYPING_URL": "/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001H/ixn/
chat/startTyping",
   "_chatIxnAPI_STOP_TYPING_URL": "/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001H/ixn/
chat/stopTyping",
   "_chatIxnAPI_DISCONNECT_URL": "/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001H/ixn/
chat/disconnect",
   "_chatIxnAPI_REFRESH_FROM_START_URL": "/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001H/
ixn/chat/refresh?transcriptPosition=1"
}
```

Start Typing

This API allows the application to indicate that the user started typing a chat message for the session.

Operation

Method	POST		
URL	/genesys/1/service/***{service_id}***/**ixn**/***chat***/**startTyping**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{service_id}	string	yes	The identifier of the service that the chat session is suppose to be associated with.
Body: The body will be `x-www-form-urlencoded` form consisting of different items representing the key/ value pairs associated with the request.			
Body Properties: The following are the properties: • _verbose - This will allow the application to get all the detail attributes associated with the chat session in the correspondingresponse.			

Response

HTTP code	200
HTTP message	OK
Body	A chat JSON object for details on the properties of the object. See the section on data structures for more details.
Notes	None

HTTP code	503
HTTP message	Service Unavailable
Body	None
Notes	This is returned if the service has not sent a notification to the application that an agent is available.

Example

Request:

```
POST http://localhost:8080/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001J/ixn/chat/
startTyping HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/<code>x-www-form-urlencoded</code>
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 07:38:38 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Content-Length: 246
Server: Jetty(7.6.0.v20120127)
{
   "chatIxnState": "TRANSCRIPT",
   "transcriptPosition": "8",
   "chatServiceMessage": "Chat service is available",
   "transcriptToShow": [   [
      "Notice.TypingStarted",
      "VasyaP",
      "is typing",
      "57",
      "CLIENT"
   ]],
   "startedAt": "2012-06-10T07:37:42Z"
}
```

Stop Typing

This API allows the application to indicate that the user has stopped typing a chat message for the session.

Operation

Method	POST		
URL	/genesys/1/service/***{service_id}***/**ixn**/***chat***/**stopTyping**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{service_id}	string	yes	The identifier of the service that the chat session is suppose to be associated with.
Body: The body will be `x-www-form-urlencoded` form consisting of different items representing the key/value pairs associated with the request.			
Body Properties: The following are the properties: • _verbose - This will allow the application to get all the detail attributes associated with the chat session in the correspondingresponse.			

Response

HTTP code	200
HTTP message	OK
Body	A chat JSON object for details on the properties of the object. See the section on data structures for more details.
Notes	None

HTTP code	503
HTTP message	Service Unavailable
Body	None
Notes	This is returned if the service has not sent a notification to the application that an agent is available.

Example

Request:

```
POST http://localhost:8080/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001J/ixn/chat/
stopTyping HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/<code>x-www-form-urlencoded</code>
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 07:38:58 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Content-Length: 251
Server: Jetty(7.6.0.v20120127)
{
   "chatIxnState": "TRANSCRIPT",
   "transcriptPosition": "9",
   "chatServiceMessage": "Chat service is available",
   "transcriptToShow": [   [
      "Notice.TypingStopped",
      "VasyaP",
      "stopped typing",
      "77",
      "CLIENT"
   ]],
   "startedAt": "2012-06-10T07:37:42Z"
}
```

Disconnect from chat session

This API allows the application to disconnect user from the chat session.

Operation

Method	POST		
URL	/genesys/1/service/***{service_id}***/**ixn**/***chat***/**disconnect**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{service_id}	string	yes	The identifier of the service that the chat session is suppose to be associated with.
Body: The body will be `x-www-form-urlencoded` form consisting of different items representing the key/value pairs associated with the request.			
Body Properties: The following are the properties: • _verbose - This will allow the application to get all the detail attributes associated with the chat session in the correspondingresponse.			

Response

HTTP code	200
HTTP message	OK
Body	A chat JSON object for details on the properties of the object. See the section on data structures for more details.
Notes	None

HTTP code	503
HTTP message	Service Unavailable
Body	None
Notes	This is returned if the service has not sent a notification to the application that an agent is available.

Example

Request:

```
POST http://localhost:8080/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001J/ixn/chat/
disconnect HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/<code>x-www-form-urlencoded</code>
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 07:43:07 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Content-Length: 114
Server: Jetty(7.6.0.v20120127)
{
 "chatIxnState" : "DISCONNECTED",
 "transcriptPosition" : "9",
 "chatServiceMessage" : "Chat was finished"
}
```

Basic Chat Service API

Create basic chat service

This API allows the application to create basic chat service session and then initiate chat interaction immediately or when user is ready.

Note: If agent availability need to be checked before chat interaction is started - use one of the advanced sessions (for example: request-chat-poll)

Operation

Method	POST		
URL	/genesys/1/service/request-chat		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
'request-chat'	String	yes	Name of the preconfigured basic chat service
Body: The body will be x-www-form-urlencoded form consisting of different items representing the key/value pairs associated with the request.			
Body Properties: The following are the properties: • _verbose - This will allow the application to get all the detail attributes associated with the chat session in the corresponding response. • ... - Any other business data attributes can also be passed.			

Response

HTTP code	200
HTTP message	OK
Body	A chat JSON object for details on the properties of the object. See the section on data structures for more details.
Notes	None

HTTP code	503
HTTP message	Service Unavailable
Body	None
Notes	This is send if the service has not sent a notification to the application that an agent is available.

Example

Request:

```
POST http://localhost:8080/genesys/1/service/request-chat HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/x-www-form-urlencoded

_verbose=true
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 07:49:46 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
```

```
Content-Type: application/json
Transfer-Encoding: chunked
Server: Jetty(7.6.0.v20120127)

{
"_chatIxnAPI-CREATE-URL":"/genesys/1/service/81f0ef4e-99dd-43ea-8366-8d27a2cbd605/ixn/chat",
"_id":"81f0ef4e-99dd-43ea-8366-8d27a2cbd605"
}
```

Advanced Chat Service API

Create advanced chat service

This API allows the application to create advanced chat service session and then initiate chat interaction when agent is available or other enterprise side conditions are met.

Note: ORS 'request-chat-poll' sample session is available.

Operation

Method	POST		
URL	genesys/1/service/request-chat-poll		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
'request-chat-poll'	String	yes	Name of the preconfigured advanced chat service
Body: The body will be x-www-form-urlencoded form consisting of different items representing the key/ value pairs associated with the request.			
Body Properties: The following are the properties: • _verbose - This will allow the application to get all the detail attributes associated with the chat session in the correspondingresponse. • ... - Any other business data attributes can also be passed.			

Response

HTTP code	200
HTTP message	OK
Body	A chat JSON object for details on the properties of the object. See the section on data structures for more details.
Notes	None

HTTP code	503
HTTP message	Service Unavailable

HTTP code	503
Body	None
Notes	This is send if the service has not sent a notification to the application that an agent is available.

Example

Request:

```
POST http://localhost:8080/genesys/1/service/request-chat-poll HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/x-www-form-urlencoded

_verbose=true
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:01:58 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Transfer-Encoding: chunked
Server: Jetty(7.6.0.v20120127)

{
"_id":"EKUJPKAQ197CFA6SJQKTJ03DBG00001K",
"_session_comand_STATUS_URL":"/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001K/status"
}
```

Execute 'status' request

This API allows the application to send data to the advanced service session and receive reply back immediately.

Operation

Method	POST		
URL	genesys/1/service/***{sessionid}***/status		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
sessionid	String	yes	The identifier of the service that the chat session is suppose to be associated with.
Body: The body will be x-www-form-urlencoded form consisting of different items representing the key/value pairs associated with the request.			
Body Properties: The following are the properties: • _verbose - This will allow the application to get all the detail attributes associated with the chat			

Method	POST
session in the correspondingresponse. • ... - any other business data attributes could be passed	

Response

HTTP code	200
HTTP message	OK
Body	A chat JSON object for details on the properties of the object. See the section on data structures for more details.
Notes	None

HTTP code	503
HTTP message	Service Unavailable
Body	None
Notes	This is send if the service has not sent a notification to the application that an agent is available.

Example

Request:

```
POST http://localhost:8080/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001K/status HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/x-www-form-urlencoded
_verbose=true
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:06:44 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Content-Length: 364
Server: Jetty(7.6.0.v20120127)
{
   "message":    {
      "_id": "EKUJPKAQ197CFA6SJQKTJ03DBG00001K",
      "_status": "available",
      "_dialog": "agent_available.html",
      "_chatIxnAPI_CREATE_URL": "/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001K/ixn/
chat",
      "_session_comand_STATUS_URL": "/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001K/
status"
   },
   "tag": "a2c.advanced.service.agentavailable.EKUJPKAQ197CFA6SJQKTJ03DBG00001K"
}
```

Execute 'internal-poke' request

Note: The internal-poke feature is available in GMS 8.1.100.28 or higher.

This API allows the agent desktop to send poke request to the advanced service session. When the session receives the internal-poke request the poke message is sent to the application as a push notification. Additional config options should be configured for the advanced GMS service to enable this API.

This API is available when,

- Poke related config options are configured and
- Service was created with device_os and device_notification_id parameters to enable native push notification are specified and
- Chat interaction is successfully routed to the agent by the advanced service (implies ixn/chat request was already executed)

'poke_url' is available as attached data in the chat interaction and should be used to invoke this request. Firewall can be configured to allow /internal-* requests to GMS from agent desktop.

Operation

Method	POST		
URL	genesys/1/service/***{sessionid}***/internal-poke		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
sessionid	String	yes	The identifier of the service that the chat session is suppose to be associated with.
Body: The body will be x-www-form-urlencoded form consisting of different items representing the key/value pairs associated with the request.			
Body Properties: The following are the properties: • poke_message - This message is sent as a push notification to the application. If not specified, default poke message configure in the service options is used • ... - any other business data attributes could be passed			

Response

HTTP code	200
HTTP message	OK
Body	OK
Notes	None

HTTP code	503
HTTP message	Service Unavailable
Body	None
Notes	This is send if the service has not sent a notification to the application that an agent is available.

Example

Request:

```
POST http://localhost:8080/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001K/internal-poke
HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/x-www-form-urlencoded

_verbose=true
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:06:44 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Content-Length: 364
Server: Jetty(7.6.0.v20120127)

OK
```

Overview of Services

This chapter provides a brief overview of the different services and call flows supported by Genesys Mobile Services. For additional details, please refer to appropriate pages below.

! **Note:** For all types of services, the *gms_user* HTTP Header is stored as part of the service data.

Genesys Mobile Services-based Services

These Services are executed in the Genesys Mobile Services server. They provide fixed functionality that can only be tuned through configuration options.

- Genesys Mobile Services-based Services
  - request-interaction
  - match-interaction
  - request-access
  - request-chat

Orchestration Server-based Services

These Services are executed in the Orchestration server. They include sample services which can be altered and customized by a developer to meet specific customer requirements.

- Orchestration Server-based Services
  - Common Characteristics
  - request-inbound-immediate (Sample)
  - request-inbound-delay (Sample)
  - request-inbound-poll (Sample)
  - request-inbound-delay-location
  - request-outbound-immediate (Sample)
  - request-outbound-delay
  - request-outbound-poll
  - request-chat-delay

Genesys Mobile Services Call Flows

The following link provides detailed call flow diagrams for the Services included with Genesys Mobile Services.

- Call Flows

Genesys Mobile Services-based Services

Business Services

These are services which provide a specific business oriented functionality. These services are used by external applications.

request-interaction

Overview

This is a basic service which helps an application/end user contact the contact center. It has the following characteristics:

- It supports only customer initiated voice contacts.
- It stores and maintains application data with the service.
- It returns access information in the response of the Create API.
- It supports very basic access number allocation (random and locking)
- It supports reserving the access information when allocated for the application for a configurable period of time.
- 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.

Create Request Data

These are the service specific parameters that will be supplied on the Create service API.

Parameters	Mandatory	Description
_phone_number	Yes	The phone number of the device that the application is running on. This data will be used to match the specified data when the device/ application calls the supplied access number.
_provide_code	No	This indicates if the service should return an access code which will add more security and reliability when trying to correlate the incoming call with the service. The value is a boolean. If not present,

Parameters	Mandatory	Description
		then GMS will use the value that was configured for the service. If it is not configured and not supplied on the Create API request then the value will be false. Note: Settings in Configuration Manager override request parameters.
_resource_group	No	This identifies the type of resource group that is need to help this end user. This maps to a configured set of access numbers. If not present, then GMS will use the group that was configured for the service. If it is not configured and not supplied on the Create API request then the request will be rejected.
{appdataname}	No	This is data that is supplied by the application and used to help the contact center resources better service the end user. The application can supply as many application data parameters as they want. These parameters may be string values or files. They should add to the multi-part structure in the body.

Create Response Data

These are the service specific parameters that will be supplied on the Create service API response.

Parameters	Mandatory	Description
_access_number	Yes	This is the access number which was allocated for this application. The application should use this number to contact the contact center.
_access_code	No	This is the access code that should be supplied by the application or end user when the call is established to further authentication the application/user. This will be present when the Create API specifies that it needs a access code (_provide_code = true).
_expiration_time	No	This is the amount time (in seconds) that this access information will be locked/reserved for the service.

Specific Requests

There are no specific requests for this service.

Events

There are no events associated with this service.

Customization

Customers can customize their own services based on the request-interaction service. The way you do this is by defining your custom service in CME. You would create a section under the services section in the GMS application object as the name of your service. You would then specific the configuration options and appropriate value for your service. Most of these options are parameters that would be passed to an request-interaction service but have been given pre-defined values via configuration. For details on the configuration options see the section below. This will allow you to simplify the API signature for your service. Once the new service is defined the application can use it. The following is an example: We would create an iPhoneService section under the services section and set the following configuration options:

Option	Value
type	built-in
name	request-interaction
ttl	7200
provide_code	false
resource_group	iPhoneService

The following is the example Create API invocation for the iPhoneService service:

```
Request URL:http://localhost:8080/gms-web/gms/1/service/iPhoneService
Request Method:POST
Accept:*/*
Accept-Charset:ISO-8859-1,utf-8;q=0.7,*;q=0.3
Accept-Encoding:gzip,deflate,sdch
Accept-Language:en-US,en;q=0.8
Connection:keep-alive
Content-Length:xxxx
Content-Type:multipart/form-data; boundary=----Boundary
Request Payload
------Boundary
Content-Disposition: form-data; name="_phone_number"
6504669999
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="current_location_latitude"
48.8583
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="current_location_longitude"
2.2944
------WebKitFormBoundaryy16qocbN6tmPORZL
```

Configuration

The following are the configuration options that can be defined for a customized service based on the request-interaction service: Note: if one of the request-interaction parameters are passed on Create API for a customized version of the service, the configuration option value will supercede the passed value (i.e. it will be ignored).

Option	Type	Mandatory	Value	Description
type	string	true	builtin	This is the type of service that will be used. The values can be either built-in or ors.
service	string	true	request-interaction	This will always be "request-interaction" for basic related services.
ttl	int	false	user defined - if not provided, the default will be 30	This specifies the default time to live for all stored data.
resource_group	string	true	{none}	Specifies the name of the resource group from which to select a resource.
provide_code	boolean	false	user defined - if not provided, the default will be false	If enabled, the service returns an access code that may be used to identify the inbound call

request-chat

Overview

This is a basic chat service which helps an application/end user contact the contact center. It has the following characteristics:

- It supports only customer initiated chat session
- It stores and maintains application data with the service.
- It is responsible for routing the the chat interaction to a specified (or configured) interaction endpoint
- It support both poll and async (via cometd) mode of message delivery

Note: See the Genesys Mobile Services Basic Chat Service API for more information.

Access Information Utility Services

These are services that provide a utility or composite Access Information related functionality to other services or applications. They are to be used in conjunction with the business services.

match-interaction

Overview

This service will do the following for **ALL** Services:

- It looks through all the services for one that matches the input criteria.
- only the service id (session id) and data id will be returned to the requester.
- The service will always release, the matched service access information will be unreserved (access resources returned).

Note: The user of this service must use the Storage APIs to retrieve any data that was associated with the matched service.

Create Request Data

These are the service specific parameters that will be supplied on the Create service API.

Parameters	Mandatory	Description
_phone_number	No	This is the phone number of the party that is calling and will be matched against the _phone_number property of the services.
_access_number	Yes	This is the number that the party called and will be matched against the _access_number property associated with the services.
_access_code	No	This is the code assigned to the party that is calling will be matched against the _access_code property assigned to the services.

Create Response Data

Parameters	Mandatory	Description
_id	Yes	This is the id of the matching service.
_data_id	Yes	This is the id of the matching service's data that is in GMS Storage.

Specific Requests

There are no specific requests for this service.

Events

There are no events associated with this service.

Configuration

The following are the configuration options that are defined for this service:

Option	Type	Mandatory	Default	Description
type	string	true	builtin	This is the type of service that will be used. THis will always be builtin.
name	string	true	match-interaction	This is the name of the matching service.

request-access

Overview

This service provides a service with access information that has been allocated for it and can then be used to contact the contact center. This service will do the following for any service:

- It will validate that the requesting service is active and running.
- It will acquire the appropriate access information based on the basic allocation algorithm.
  - It can reserve the access information for a configurable period of time
  - Simple random or round robin allocation
- 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.

Create Request Data

These are the service specific parameters that will be supplied on the Create service API.

Parameters	Mandatory	Description
_id	Yes	This is the identifier of the service which the allocated access information should be associated

Parameters	Mandatory	Description
		with.
_provide_code	No	This indicates if the service should return an access code which will add more security and reliability when trying to correlate the incoming call with the service. The value is a boolean. The default is false.
_phone_number	No	This is the phone number that is to be associated with the reserved access information.
_resource_group	Yes	This is the resource group from which an available access number will be taken.
_booking_expiration_timeout	No	This is the resource _booking_expiration_timeout used to book the resource. This parameter will override the value in Resource options and in CME options.

Create Response Data

Parameters	Mandatory	Description
_access_number	Yes	This is the access number which was allocated for this application. The application should use this number to contact the contact center.
_access_code	No	This is the access code that should be supplied by the application or end user when the call is established to further authentication the application/user. This will be present when the Create API specifies that it needs a access code (_provide_code = true).
_expiration_time	Yes	This is the amount time that this access information will be locked/reserved for the service.

Specific Requests

There are no specific requests for this service.

Events

There are no events associated with this service.

Configuration

The following are the configuration options that are defined for this service:

Option	Type	Mandatory	Default	Description
access_code_length	int	false	6	This is the length of the access code which can be allocated.

Orchestration Server-based Services

Common Characteristics

Create Request Data

For all parameters that are also configuration options, if the parameter is present on the Create request and this is a corresponding configuration option, the configuration option value will override the parameter on the request. The following parameters will be sent from GMS to all ORS based services when the service is created/started.

Parameters	Mandatory	Description
_data_id	Yes	This is the identifier associated with the application data was storaged by GMS when the service was created. The ORS based service can access this data via the GMS Storage APIs using this id.

Requests

The following are the service requests that must be supported by all ORS based services.

gms.start (internal only)

This request actually starts the service. It is sent by GMS to ORS immediately affter the session is created.

Request Parameters

There are no specific parameters for this request - all the parameters should have been passed on the actual creation of the session.

Response Parameters

The start service specific response parameters can vary based on the service that is being started. The following are the response parameters that are common across all ORS based services. This data will be returned on the Create API response for the service.

Parameters	Mandatory	Description
_id	Yes	This is the ors session id for this service.

update

This specific request updates the application data associated with the service.

Note: this request is processed by the GMS server and is NOT passed on to the ORS based service. If the services wants the updates to the associated application data, it can subscribe to the changes via the Notification APIs.

Request Parameters

The following are the request parameters.

Parameters	Mandatory	Description
{appdataname}	Yes	This is the set of application data which is to be updated/added to the service. They will be either MultiPart form or x-www-form-urlencoded form encoded. In the case of MultiPart, the values can be strings or files but with urlencoded the values can only strings.

Response Parameters

There are no specific parameters for this response.

request-inbound-immediate

Overview

This is a basic service which helps an application/end user contact the contact center. It is the same functionality as GMS based request-interaction but the service processing is done on ORS. It allows the service developer to customize this logic easily. It has the following characteristics:

- It supports only customer initiated voice contacts.
- It stores and maintains application data with the service.
- It returns access information in the response of the Create API.

- It supports very basic access number allocation (random and locking)
- It supports reserving the access information when allocated for the application for a configurable period of time.
- 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.

Create Request Data

These are the service specific parameters that will be supplied on the Create service API.

Parameters	Mandatory	Description
_phone_number	No	The phone number of the device that the application is running on. This data will be used to match the specified data when the device/application calls the supplied access number.
_provide_code	No	This indicates if the service should return an access code which will add more security and reliability when trying to correlate the incoming call with the service. The value is a boolean. If not present, then GMS will use the value that was configured for the service. If it is not configured and not supplied on the Create API request then the value will be false.
_resource_group	No	This identifies the type of resource group that is need to help this end user. This maps to a configured set of access numbers. If not present, then GMS will use the group that was configured for the service. If it is not configured and not supplied on the Create API request then the request will be rejected.
{appdataname}	No	This is data that is supplied by the application and used to help the contact center resources better service the end user. The application can supply as many application data parameters as they want. These parameters may be string values or files. They should add to the multi-part structure in the body.

Create Response Data

Important Note: These additional parameters are returned in the gms.start request response.

Parameters	Mandatory	Description
_access_number	Yes	This is the access number which was allocated for this application. The application should use this number to contact the contact center.
_access_code	No	This is the access code that should be supplied by the application or end user when the call is established to further authentication the application/user. This will be present when the Create API specifies that it needs a access code (_provide_code = true).
_expiration_time	No	This is the amount time (in seconds) that this access information will be locked/reserved for the service.

Specific Requests

There are no specific requests for this service.

Events

There are no events associated with this service.

Customization

The following are the configuration options that can be defined for a customized service based on this request-inbound-delay service:

Note: if one of the service parameters are passed on Create API for a customized version of the service, the configuration option value will supercede the passed value (i.e. it will be ignored).

Option	Type	Mandatory	Value	Description
type	string	true	ors	This is the type of service that will be used. The values can be either built-in or ors.
name	string	true	http://localhost/scxml-services/request-inbound-immediate.scxml	This is the url of the service's SCXML application.

Option	Type	Mandatory	Value	Description
ttl	int	false	User defined - if not provided, the default will be 3600	This specifies the default time to live for all stored data.
resource_group	string	true	user defined	Specifies the name of the resource group from which to select a resource.
provide_code	boolean	false	User defined - if not provided, the default will be false	If enabled, the service returns an access code that may be used to identify the inbound call

Sample

request-inbound-immediate

request-inbound-delay

Overview

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.

Create Request Data

These are the service specific parameters that will be supplied on the Create service API.

Parameters	Mandatory	Description
_phone_number	No	The phone number of the device that the application is running on. This data will be used to match the specified data when the device/ application calls the supplied access number.
_provide_code	No	This indicates if the service should return an access code which will add more security and reliability when trying to correlate the incoming call with the service. The value is a boolean. If not present, then GMS will use the value that was configured for the service. If it is not configured and not supplied on the Create API request then the value will be false.
_device_os	No	This is the type operating system that will receive notifications. The value can be either iOS or Android. If not present, then ORS will use the value that was configured for the service. If it is not configured and not supplied on the Create API request then the value will be rejected.
_device_notification_id	Yes	This is the id of the device to which the events/notifications are to be pushed to
_access_info_via_event	No	This indicates if the application wants the access information in the event when the agent is available or the application will come back later and get the information. The default is true (the access information will be delivery via the event).
_resource_group	No	This identifies the type of resource group that is need to help this end user. This maps to a configured set of access numbers.If not present, then GMS will use the group that was configured for the service. If it is not configured and not supplied on the Create API request then the request will be rejected.
{appdataname}	No	This is data that is supplied by the application and used to help the

Parameters	Mandatory	Description
		contact center resources better service the end user. The application can supply as many application data parameters as they want. These parameters may be string values or files. They should add to the multi-part structure in the body.

Create Response Data

None

Specific Requests

This is the set of service specific requests that can be done using the service Specific Request API.

reserve

This request provides the application with access information that is to be used with this service.

Note: This service will use the request-access service to reserve the access information for this service using the service specific data (resource_group, etc.).

Request parameters

The specific request parameters are: hese are the service specific parameters that will be supplied on the Create service API.

Parameters	Mandatory	Description
_provide_code	No	This indicates if the service should return an access code which will add more security and reliability when trying to correlate the incoming call with the service. The value is a boolean. If not present, then GMS will use the value that was configured for the service. If it is not configured and not supplied on the Create API request then the value will be false.

Response parameters

The specific response parameters are:

Parameters	Mandatory	Description
_access_number	Yes	This is the access number which was allocated for this application. The application should use this number to contact the contact center.
_access_code	No	This is the access code that should be supplied by the application or end user when the call is established to further authentication the application/user. This will be present when the Create API specifies that it needs a access code (_provide_code = true).
_expiration_time	No	This is the amount time that this access information will be locked/reserved for the service.

Events

The following are the elements that will be generated from this service.

A2C.advanced.service.agentavailable.{session_id}

This event is sent to the application that created this service when the service logic has found an agent that is available to handle the customer's request. The properties returned in the event are mainly passed on if the service is configured to get the necessary access information and return it in this event or if the application will ask for it after getting notified with this event.

Parameters	Mandatory	Description
_id	Yes	This is the identifier of the service.
_access_number	No	This is the access number which was allocated for this application. The application should use this number to contact the contact center. This is present if the _access_info_via_event parameter was true.
_access_code	No	This is the access code that should be supplied by the application or end user when the call is established to further authentication the application/user. This will be present when the Create API specifies that it needs a access code (_provide_code = true). This is present if the _access_info_via_event parameter was true.
_expiration_time	No	This is the amount time that this

Parameters	Mandatory	Description
		access information will be locked/ reserved for the service.This is present if the _access_info_via_event parameter was true.

A2C.advanced.service.statuschanged.{session_id}

This event is sent to the application that created this service whenever the service logic has changed state. This is customizable by the service logic developer.

Parameters	Mandatory	Description
_id	Yes	This is the identifier of the service.
_status	Yes	This is the new status of the service.
_dialog	No	This is the recommended dialog that should be used to communicate with the customer. This can be a string which has a dialog description. This can be a url to an actual application dialog that is recommend to be presented to the end user of the application associated with this service.

A2C.advanced.service.expired.{session_id}

This event is sent to the application that created this service when the service expires and ends.

Parameters	Mandatory	Description
_id	Yes	This is the service identifier.

Customization Configuration

The following are the configuration options that can be defined for a customized service based on this request-inbound-delay service:

Note: if one of the service parameters are passed on Create API for a customized version of the service, the configuration option value will supercede the passed value (i.e. it will be ignored).

Option	Type	Mandatory	Value	Description
type	string	true	ors	This is the type of service that will be used. The values can be either built-in or ors.
name	string	true	http://localhost/	This is the url of the

Option	Type	Mandatory	Value	Description
			scxml-services/ request-inbound-delay.scxml	service's SCXML application.
ttl	int	false	User defined - if not provided, the default will be 3600	This specifies the default time to live for all stored data.
resource_group	string	true	user defined	Specifies the name of the resource group from which to select a resource.
provide_code	boolean	false	User defined - if not provided, the default will be false	If enabled, the service returns an access code that may be used to identify the inbound call
_access_info_via_event	boolean	false	User defined - if not provided, the default will be true	If enabled, the service return the access information in the A2C.advanced.service.agent event.
device_os	string	true	user defined	This identifies the type of device operating system that the application is running on.

Sample

request-inbound-delay

request-inbound-poll

Overview

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.
- Access number allocation/reservation will be done by the application using the reserve request. This request will process the appropriate allocation algorithm in conjunction with the Genesys Mobile Services-based request-access service.

- It only supports polling for the available of an agent.

Create Request Data

These are the service specific parameters that will be supplied on the Create service API.

Parameters	Mandatory	Description
_phone_number	No	The phone number of the device that the application is running on. This data will be used to match the specified data when the device/ application calls the supplied access number.
_resource_group	No	This identifies the type of resource group that is need to help this end user. This maps to a configured set of access numbers. If not present, then GMS will use the group that was configured for the service. If it is not configured and not supplied on the Create API request then the request will be rejected.
{appdataname}	No	This is data that is supplied by the application and used to help the contact center resources better service the end user. The application can supply as many application data parameters as they want. These parameters may be string values or files. They should add to the multi-part structure in the body.

Create Response Data

None

Specific Requests

This is the set of service specific requests that can be done using the service Specific Request API.

status

This request provides the application with the current status of the service.

Request parameters

There are no specific parameters for this request .

Response parameters

The specific response parameters are:

Parameters	Mandatory	Description
_status	Yes	This is the current status of the service.
_dialog	No	This is the recommended dialog that should be used to communicate with the customer.

reserve

This request provides the application with access information that is to be used with this service.

Note: This service will use the request-access service to reserve the access information for this service using the service specific data (resource_group, etc.).

Request parameters

The specific request parameters are: hese are the service specific parameters that will be supplied on the Create service API.

Parameters	Mandatory	Description
_provide_code	No	This indicates if the service should return an access code which will add more security and reliability when trying to correlate the incoming call with the service. The value is a boolean. The default is false. If not present, then GMS will use the value that was configured for the service. If it is not configured and not supplied on the Create API request then the value will be false.
_longitude	No	This is the application's current longitude. It wil be used by service to calcuate the corresponding resource group from which the access number is to be allocated. If not specified the basic reservation functionality will be used.
_latitude	No	This is the application's current latitude. It wil be used by service to calcuate the corresponding resource group from which the access number is to be allocated. If not specified the basic reservation functionality will be used.

Response parameters

The specific response parameters are:

Parameters	Mandatory	Description
_access_number	Yes	This is the access number which was allocated for this application. The application should use this number to contact the contact center.
_access_code	No	This is the access code that should be supplied by the application or end user when the call is established to further authentication the application/user. This will be present when the Create API specifies that it needs a access code (_provide_code = true).
_expiration_time	No	This is the amount time that this access information will be locked/ reserved for the service.

Events

There are no events for this service.

Customization Configuration

The following are the configuration options that can be defined for a customized service based on the request-inbound-poll service:

Note: if one of the service parameters are passed on Create API for a customized version of the service, the configuration option value will supercede the passed value (i.e. it will be ignored).

Option	Type	Mandatory	Value	Description
type	string	true	ors	This is the type of service that will be used. The values can be either built-in or ors.
name	string	true	http://localhost/scxml-services/request-inbound-poll.scxml	This is the url of the service's SCXML application.
ttl	int	false	user defined - if not provided, the default will be 3600	This specifies the default time to live for all stored data.
resource_group	string	true	user defined	Specifies the name of the resource

Option	Type	Mandatory	Value	Description
				group from which to select a resource.
provide_code	boolean	false	user defined - if not provided, the default will be false.	If enabled, the service returns an access code that may be used to identify the inbound call

Sample

request-inbound-poll

request-inbound-delay-location

Overview

This is an advanced service which helps an application/end user contact the contact center. It has the following characteristics:

- It runs on ORS as an SCXML session.
- 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
  - Location based allocation
    - This service will use a 3rd party system to covert the application's longitude and latitude values in a city which they are in.
    - The resource groups **must** be configured based on the name of cities. The city name calculated from the application's location information will be used as the resource group when reserving access information.
- 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.

Create Request Data

These are the service specific parameters that will be supplied on the Create service API.

Parameters	Mandatory	Description
_phone_number	No	The phone number of the device that the application is running on. This data will be used to match the specified data when the device/ application calls the supplied access number.
_provide_code	No	This indicates if the service should return an access code which will add more security and reliability when trying to correlate the incoming call with the service. The value is a boolean. If not present, then GMS will use the value that was configured for the service. If it is not configured and not supplied on the Create API request then the value will be false.
_device_os	No	This is the type operating system that will receive notifications. The value can be either iOS or Android. If not present, then ORS will use the value that was configured for the service. If it is not configured and not supplied on the Create API request then the value will be rejected.
_device_notification_id	Yes	This is the id of the device to which the events/notifications are to be pushed to
_access_info_via_event	No	This indicates if the application wants the access information in the event when the agent is available or the application will come back later and get the information. The default is true (the access information will be delivery via the event).
_latitude	Yes	This is the current latiture of the device
_longitude	Yes	This is the current longitude of the device.
{appdataname}	No	This is data that is supplied by the application and used to help the contact center resources better service the end user. The application can supply as many application data parameters as they want. These parameters may

Parameters	Mandatory	Description
		be string values or files. They should add to the multi-part structure in the body.

Create Response Data

None

Specific Requests

There are no specific requests for this service.

Events

The following are the elements that will be generated from this service.

A2C.advanced.service.agentavailable.{session_id}

This event is sent to the application that created this service when the service logic has found an agent that is available to handle the customer's request. The properties returned in the event are mainly passed on if the service is configured to get the necessary access information and return it in this event or if the application will ask for it after getting notified with this event.

Parameters	Mandatory	Description
_id	Yes	This is the identifier of the service.
_access_number	No	This is the access number which was allocated for this application. The application should use this number to contact the contact center. This is present if the _access_info_via_event parameter was true.
_access_code	No	This is the access code that should be supplied by the application or end user when the call is established to further authentication the application/user. This will be present when the Create API specifies that it needs a access code (_provide_code = true). This is present if the _access_info_via_event parameter was true.
_expiration_time	No	This is the amount time that this access information will be locked/reserved for the service.This is present if the _access_info_via_event parameter was true.

A2C.advanced.service.statuschanged.{session_id}

This event is sent to the application that created this service whenever the service logic has changed state. This is customizable by the service logic developer.

Parameters	Mandatory	Description
_id	Yes	This is the identifier of the service.
_status	Yes	This is the new status of the service.
_dialog	No	This is the recommended dialog that should be used to communicate with the customer. This can be a string which has a dialog description. This can be a url to an actual application dialog that is recommend to be presented to the end user of the application associated with this service.

A2C.advanced.service.expired.{session_id}

This event is sent to the application that created this service when the service expires and ends.

Parameters	Mandatory	Description
_id	Yes	This is the service identifier.

Customization Configuration

The following are the configuration options that can be defined for a customized service based on this request-inbound-delay-location service:

Note: if one of the service parameters are passed on Create API for a customized version of the service, the configuration option value will supercede the passed value (i.e. it will be ignored).

Option	Type	Mandatory	Value	Description
type	string	true	ors	This is the type of service that will be used. The values can be either built-in or ors.
name	string	true	http://localhost/scxml-services/request-inbound-delay-location.scxml	This is the url of the service's SCXML application.
ttl	int	false	user defined - if not provided, the default will be 3600	This specifies the default time to live for all stored data.
provide_code	boolean	false	user defined - if not provided, the	If enabled, the service returns an

Option	Type	Mandatory	Value	Description
			default will be false.	access code that may be used to identify the inbound call
_access_info_via_event	boolean	false	user defined - if not provided, the default will be true.	If enabled, the service return the access information in the A2C.advanced.service.agent event.
device_os	string	true	user defined	This identifies the type of device operating system that the application is running on.

request-outbound-immediate

Overview

This is an advanced service which helps an application/end user contact the contact center. It has the following characteristics:

- It supports contact center initiated voice contacts.
- It supports initiating the call immediately.
- It stores and maintains application data with the service.

Create Request Data

These are the service specific parameters that will be supplied on the Create service API.

Parameters	Mandatory	Description
_phone_number	Yes	The phone number of the device that the application is running on. This data will be used to match the specified data when the device/ application calls the supplied access number.
{appdataname}	No	This is data that is supplied by the application and used to help the contact center resources better service the end user. The application can supply as many application data parameters as they want. These parameters may be string values or files. They

Parameters	Mandatory	Description
		should add to the multi-part structure in the body.

Create Response Data

None

Specific Requests

There are no specific requests for this service.

Events

There are no events for this service.

Customization Configuration

This service can not be customized.

Sample

request-outbound-immediate

request-outbound-delay

Overview

This is an advanced service which helps an application/end user contact the contact center. It has the following characteristics:

- It supports contact center initiated voice contacts.
- It supports waiting for the appropriate available agent before initiating the call.
- When the agent is available, it notifies the application and it can acknowledge that the call can be initiated.
- It stores and maintains application data with the service.

Create Request Data

These are the service specific parameters that will be supplied on the Create service API.

Parameters	Mandatory	Description
_phone_number	Yes	The phone number of the device that the application is running on. This data will be used to match the specified data when the device/ application calls the supplied access number.
_device_os	No	This is the type operating system that will receive notifications. The value can be either iOS or Android. If not present, then ORS will use the value that was configured for the service. If it is not configured and not supplied on the Create API request then the value will be rejected.
_device_notification_id	Yes	This is the id of the device to which the events/notifications are to be pushed to
{appdataname}	No	This is data that is supplied by the application and used to help the contact center resources better service the end user. The application can supply as many application data parameters as they want. These parameters may be string values or files. They should add to the multi-part structure in the body.

Create Response Data

None

Specific Requests

accept

This request provides the application with the ability to accept the initation of the call.

Request parameters

There are no specific request parameters.

Response parameters

There are no specific response parameters.

Events

The following are the elements that will be generated from this service.

A2C.advanced.service.agentavailable.{session_id}

This event is sent to the application that created this service when the service logic has found an agent that is available to handle the customer's request. The properties returned in the event are mainly passed on if the service is configured to get the necessary access information and return it in this event or if the application will ask for it after getting notified with this event.

Parameters	Mandatory	Description
_id	Yes	This is the identifier of the service.
_access_number	No	This is the access number which was allocated for this application. The application should use this number to contact the contact center. This is present if the _access_info_via_event parameter was true.
_access_code	No	This is the access code that should be supplied by the application or end user when the call is established to further authentication the application/user. This will be present when the Create API specifies that it needs a access code (_provide_code = true). This is present if the _access_info_via_event parameter was true.
_expiration_time	No	This is the amount time that this access information will be locked/reserved for the service.This is present if the _access_info_via_event parameter was true.

A2C.advanced.service.statuschanged.{session_id}

This event is sent to the application that created this service whenever the service logic has changed state. This is customizable by the service logic developer.

Parameters	Mandatory	Description
_id	Yes	This is the identifier of the service.
_status	Yes	This is the new status of the service.
_dialog	No	This is the recommended dialog that should be used to communicate with the customer. This can be a string which has a

Parameters	Mandatory	Description
		dialog description. This can be a url to an actual application dialog that is recommend to be presented to the end user of the application associated with this service.

A2C.advanced.service.expired.{session_id}

This event is sent to the application that created this service when the service expires and ends.

Parameters	Mandatory	Description
_id	Yes	This is the service identifier.

Customization Configuration

The following are the configuration options that can be defined for a customized service based on this request-outbound-delay service:

Note: if one of the service parameters are passed on Create API for a customized version of the service, the configuration option value will supercede the passed value (i.e. it will be ignored).

Option	Type	Mandatory	Value	Description
type	string	true	ors	This is the type of service that will be used. The values can be either built-in or ors.
name	string	true	http://localhost/scxml-services/request-outbound-delay.scxml	This is the url of the service's SCXML application.
ttl	int	false	user defined - if not provided, the default will be 3600	This specifies the default time to live for all stored data.
device_os	string	true	user defined	This identifies the type of device operating system that the application is running on.

request-outbound-poll

Overview

This is an advanced service which helps an application/end user contact the contact center. It has the following characteristics:

- It supports contact center initiated voice contacts.
- It supports waiting for the appropriate available agent before initiating the call.
- The application must poll the service to determine when an agent is available. At that point, the application can acknowledge that the call can be initiated.
- It stores and maintains application data with the service.

Create Request Data

These are the service specific parameters that will be supplied on the Create service API.

Parameters	Mandatory	Description
_phone_number	Yes	The phone number of the device that the application is running on. This data will be used to match the specified data when the device/ application calls the supplied access number.
{appdataname}	No	This is data that is supplied by the application and used to help the contact center resources better service the end user. The application can supply as many application data parameters as they want. These parameters may be string values or files. They should add to the multi-part structure in the body.

Create Response Data

None

Specific Requests

status

This request provides the application with the current status of the service.

Request parameters

There are no specific parameters for this request .

Response parameters

The specific response parameters are:

Parameters	Mandatory	Description
_status	Yes	This is the current status of the service.
_dialog	No	This is the recommended dialog that should be used to communicate with the customer.

accept

This request provides the application with the ability to accept the initation of the call.

Request parameters

There are no specific request parameters.

Response parameters

There are no specific response parameters.

Events

There are no events for this service.

Customization Configuration

This service can not be customized.

request-chat-delay

Overview

This is a basic chat service which helps an application/end user contact the contact center. It has the following characteristics:

- It supports only customer initiated chat session
- It stores and maintains application data with the service.
- It is responsible for routing the the chat interaction to a specified (or configured) interaction endpoint
- It support both poll and async (via cometd) mode of message delivery

Create Request Data

These are the service specific parameters that will be supplied on the Create service API.

Parameters	Mandatory	Description
firstName	Yes	This is the first name of the customer.
lastName	Yes	This is the last name of the customer.
email	Yes	This is the email address of the customer.
subject	Yes	This is the reason for contacting the contact center.
device_os	No	"comet" for chat messages to be deliverd via cometd. For poll, this parameter should not be specified.

Create Response Data

Parameters	Mandatory	Description
_id	Yes	This is the id of the created service.

Specific Requests

This is the set of service specific requests that can be done using the service Specific Request API.

chat

This request creates a chat interaction and routes it to the configured interaction end-point.

Request parameters

There are no specific request parameters.

Response parameters

The specific response parameters are:

Parameters	Mandatory	Description
chatIxnState	Yes	This is the state of the chat interaction. It will be "connected" when the chat session is connected

Parameters	Mandatory	Description
		to an agent.
transcriptPosition	Yes	This is the current position in the chat dialog or transcript for this user.
chatServiceMessage	Yes	This is a diagnostic message used for debugging.

Events

There are no events associated with this service.

Customization Configuration

The following are the configuration options that are defined for this service:

Option	Type	Mandatory	Default	Description
type	string	true	builtin	This is the type of service that will be used. The values is always built-in for this service.
service	string	true	request-chat-delay	This can be customized.
ttl	int	false	user defined - if not provided, the default will be 30	This specifies the default time to live for all stored data.

request-inbound-immediate

```
<scxml version="1.0" xmlns="http://www.w3.org/2005/07/scxml"
        xmlns:queue="www.genesyslab.com/modules/queue"
        xmlns:ixn="http://www.genesyslab.com/modules/interaction"
        xmlns:service="http://www.genesyslab.com/modules/dfm/gsgBasedServices/v1"
        xmlns:ws="http://www.genesyslab.com/modules/ws"
        xmlns:session="www.genesyslab.com/modules/session"
        initial="initial">
        <!--
                This is a basic service which helps an application/end user contact the
contact center.  It allows the service developer to customize this logic easily.
                It has the following characteristics:
                        It supports only customer initiated voice contacts.
                        It stores and maintains application data with the service.
                        It returns access information in the response of the Create API.
                        It supports very basic access number allocation (random and locking)
                        It supports reserving the access information when allocated for the
application for a configurable period of time.
                        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>
                <data id="reqid"/>
                <data id="startSendId" />
        </datamodel>
        <state id="initial">
                <transition event="gms.start" target="handleStart">
                        <script>
                                <!-- Save this event id so we can respond to it -->
                                _data.startSendId = _event.sendid;
                        </script>
                </transition>
        </state>
        <state id="handleStart">
                <onentry>
                        <script>
                                __Log("ResourceGroup: " + _data._resource_group);
                                __Log("ProvideCode: " + _data._provide_code);
                                __Log("ANI: " + _data._phone_number);
                        </script>
                        <!-- Request a resource for this user; that is a phone number and
perhaps an access code -->
                        <service:reserve requestid="_data.requestId" _id="_sessionid"
                                _resource_group="_data._resource_group"
_provide_code="_data._provide_code" _phone_number="_data._phone_number" />
                </onentry>
                <transition event="gsgBasedServices.reserve.done" target="waitForCall" >
                        <!--
                                Build a response to the request; it may include
access_number, access_code and/or expiration time
                        -->
                        <script>
                                var myResultSetObject = eval('(' + _event.data.content + ')');
                                var result = { _id: _sessionid };
                                if ( myResultSetObject.hasOwnProperty( '_access_number' ) )
                                result._access_number = myResultSetObject._access_number;
                                if ( myResultSetObject.hasOwnProperty( '_access_code' ) )
                                result._access_code = myResultSetObject._access_code;
                                if ( myResultSetObject.hasOwnProperty( '_expiration_time' ) )
                                result._expiration_time = myResultSetObject._expiration_time;
                        </script>
                        <ws:response requestid="_data.startSendId"
resultcode="JSON.stringify( result )" />

                        <!-- Start an internal timer in case the user doesn't call within the
specified time -->
                        <send target="'_internal'" event="'timer'"
delay="(myResultSetObject.hasOwnProperty( '_expiration_time' ) ? result._expiration_time :
_data._ttl) + 's'" />
                </transition>
                <transition event="error.*" target="error" >
                        <script>
                                var result = { error: uneval( _event ) };
                        </script>
                        <!-- Send a response to the client indicating that we had some kind
of error -->
                        <ws:response requestid="_data.startSendId"
resultcode="JSON.stringify( result )" />
                </transition>
        </state>
        <state id="waitForCall">
                <transition event="interaction.present" target="ConnectToAgent">
                        <!--
                                When this event is received, it means that the call has been
'attached' to this session by the inboudn scxml;
                                It is now under the control of this session and can be
handled in any way desired.
                         -->
                        <!--
                                This would disconnect the caller, but that wouldn't make any
sense!
                                <ixn:terminate requestid="_data.reqid"
interactionid="_genesys.ixn.interactions[0].g_uid" reason="'finished service'"
resource="_genesys.ixn.interactions[0].voice.dnis"/>
                        -->
                </transition>

                <!-- If this event is received it means the caller didn't match into the
system withing the expiration time -->
                <transition event="timer" target="exit" />
        </state>
        <!-- The below code is where you would put your routing logic        -->
        <state id="ConnectToAgent">
                <onentry>
                        <log expr="'Checking for agent availability in agent group:
Billing'"/>
                        <!-- Ask Router to find an agent -->
                        <queue:submit route="true" timeout="300">
                                <queue:targets>
                                        <queue:target name="'Billing'" type="agentgroup"
statserver="'Stat_Server'" />
                                </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"/>
                <!-- Could not get and agent for a period of time so let's try again - NEED
TO ADD LOGIC TO NOT RETRY FOREVER -->
```

```
                <transition event="error.queue.submit">
                        <session:fetch srcexpr="_data.callback_url" method="'post'"
type="'application/json'" enctype="'application/x-www-form-urlencoded'">
                                <param name="event" expr="'AgentNotAvailable'"/>
                                <param name="sessionid" expr="_sessionid"/>
                                <param name="end_state" expr="'waiting for voice agent'"/>
                                <param name="dialog" expr="'wait_for_agent.html'"/>
                        </session:fetch>
                        <log expr="'Again checking for agent availability in agent group:
Billing'"/>
                        <queue:submit route="true" timeout="30">
                                <queue:targets>
                                        <queue:target name="'Billing'" type="agentgroup"
statserver="'Stat_Server'" />
                                </queue:targets>
                        </queue:submit>
                </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"/>
        </state>
        <final id="exit"/>
        <final id="error"/>
</scxml>
```

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>
```

request-inbound-poll

```
<scxml version="1.0" xmlns="http://www.w3.org/2005/07/scxml"
        xmlns:ixn="http://www.genesyslab.com/modules/interaction"
        xmlns:service="http://www.genesyslab.com/modules/dfm/gsgBasedServices/v1"
        xmlns:ws="http://www.genesyslab.com/modules/ws"
        xmlns:session="www.genesyslab.com/modules/session"
        xmlns:yahoo_placefinder="http://www.genesyslab.com/modules/dfm/yahoo_placefinder/v1"
        initial="outside">
        <!--
        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.
        Access number allocation/reservation will be done by the application using the
reserve request.
        This request will process the appropriate allocation algorithm in conjunction with
the GMS based A2C-Reserve-AccessInfo service.
        It only supports polling for the availability of an agent.
        -->
        <script>
                var statusId = null;
                var reserveId = null;
                var provideCode = false;
                var attemptCounter = 0;
                var callerCity = null;
                var callerState = null;
        </script>
        <datamodel>
                <data id="reqid"/>
        </datamodel>
        <state id="outside" initial="initial">
                <state id="initial">
                        <onentry>
                                <!-- Create an overall timer to ensure that the session does
not go on forever -->
                                <send target="'_internal'" event="'timer'" delay="_data._ttl
+ 's'" />
                        </onentry>
                        <transition event="gms.start" target="run">
                                <script>
                                        <!-- Save the session id in the result object as it
must be returned to the client in the response to gms.start -->
                                        var result = new Object();
                                        result._id = _sessionid;
                                </script>
                                <!-- Do the appropriate logic to wait to get an agent; ie. do
the <queue:submit> or wait for a period of time, etc.) -->
                                <queue:submit route="true" timeout="300">
                                        <queue:targets>
                                                <queue:target name="'Billing'"
type="agentgroup" statserver="'Stat_Server'" />
                                        </queue:targets>
                                </queue:submit>
                                <!-- Send a response to the client indicating the result -->
                                <ws:response requestid="_event.sendid"
```

```
resultcode="JSON.stringify( result )" />
                        </transition>
                </state>
                <state id="run">
                        <!-- Session specific request to get the status -->
                        <transition event="status" target="status" >
                                <script>
                                        statusId = _event.sendid;
                                </script>
                        </transition>
                        <!-- Session specific request to reserve an access number using
location information -->
                        <transition event="reserve" cond="_event.data.param.hasOwnProperty(
'_latitude' ) && _event.data.param.hasOwnProperty( '_longitude' )" target="getLocation" >
                                <script>
                                        reserveId = _event.sendid;
                                        if ( _event.data.param.hasOwnProperty(
'_provide_code' ) ) provideCode = _event.data.param._provide_code;
                                        if ( _data.hasOwnProperty( '_provide_code' ) )
provideCode = _data._provide_code;
                                </script>
                                <!-- Request the location using yahoo placefinder given the
latitude and longitude -->
                                <yahoo_placefinder:query_location appid="'yourappid'"
lat="_event.data.param._latitude" long="_event.data.param._longitude" />
                        </transition>
                        <!-- Session specific request to reserve an access number -->
                        <transition event="reserve" target="default_reserve" >
                                <script>
                                        reserveId = _event.sendid;
                                        if ( _event.data.param.hasOwnProperty(
'_provide_code' ) ) provideCode = _event.data.param._provide_code;
                                        if ( _data.hasOwnProperty( '_provide_code' ) )
provideCode = _data._provide_code;
                                </script>
                        </transition>
                        <!--
                                Handler for Router's response indication an agent is
reserved.  Practically speaking, for this scenario there is
                                nothing we can do as most likely the agent reservation will
be lost before the next status request.
                         -->
                        <transition event="queue.submit.done">
                                <!-- Change the state of the service based on finding an
agent. -->
                        </transition>
                </state>
                <!-- This request provides the application with the current status of the
service; no request specific parameters -->
                <state id="status">
                        <transition target="run" >
                                <!--
                                  These are specific response parameters from the status
request; this sample doesn't actually track a status for the
                                  agent (it just keeps a counter of the number of requests).
It should really give some indication of the current
                                  queue time.  Perhaps if the remaining queue time is low,
let the user get into the queue.
                                 -->
                                <script>
                                        var result = new Object();
                                        result._status = "" + ++attemptCounter;
                                        result._dialog = "Whatever text we want";
                                </script>
                                <!-- Send a response to the client indicating the result -->
                                <ws:response requestid="statusId" resultcode="JSON.stringify(
result )" />
                        </transition>
                </state>
                <state id="getLocation">
                        <!--
                        Build a response to the yahoo placefinder request; includes city and
state of the caller
                        -->
                        <transition  event="yahoo_placefinder.query_location.done"
target="reserve_by_location" >
                                <script>
                                        var myResultSetObject = eval('(' +
_event.data.content + ')');
                                        callerCity =
myResultSetObject.ResultSet.Results[0].city;
                                        callerState =
myResultSetObject.ResultSet.Results[0].state;
                                </script>
                        </transition>
                        <transition event="error.yahoo_placefinder.query_location"
target="error"/>
                </state>
                <!--
                This request provides the application with access information that is to be
used with this service; request specific parameters are provide code, longitude, and/or
latitude.
                This service will use the A2C-Reserve-AccessInfo service to reserve the
access information for this service using the service specific data (resource_group, etc.).
                -->
                <state id="reserve_by_location">
                        <onentry>
                                <script>
                                        var resource_group = ( callerCity + ":" + callerState
).toLowerCase();resource_group;
                                </script>
                                <!-- Request a resource for this user; that is a phone number
and perhaps an access code -->
                                <service:reserve requestid="_data.requestId" _id="_sessionid"
_resource_group="resource_group" _provide_code="provideCode"
_phone_number="_data._phone_number" />
                        </onentry>
                        <transition event="error.gsgBasedServices.reserve"
target="default_reserve" />
                        <!--
                                Build a response to the request; it may include
access_number, access_code and/or expiration time
                        -->
                        <transition event="gsgBasedServices.reserve.done"
target="waitForCall" >
                                <script>
                                        var myResultSetObject = eval('(' +
_event.data.content + ')');
                                        var result = new Object();
                                        result._id = _sessionid;
                                        if ( myResultSetObject.hasOwnProperty(
'_access_number' ) )
                                        result._access_number =
myResultSetObject._access_number;
                                        if ( myResultSetObject.hasOwnProperty( '_access_code'
) )
```

```
                                        result._access_code = myResultSetObject._access_code;
                                        if ( myResultSetObject.hasOwnProperty(
'_expiration_time' ) )
                                        result._expiration_time =
myResultSetObject._expiration_time;
                                </script>
                                <!-- Send a response to the client indicating the result -->
                                <ws:response requestid="reserveId"
resultcode="JSON.stringify( result )" />
                        </transition>
                </state>
                <state id="default_reserve">
                        <onentry>
                                <!-- Request a resource for this user; that is a phone number
and perhaps an access code -->
                                <service:reserve requestid="_data.requestId" _id="_sessionid"
_resource_group="_data._resource_group" _provide_code="provideCode"
_phone_number="_data._phone_number" />
                        </onentry>
                        <!--
                        Build a response to the request; it may include access_number,
access_code and/or expiration time
                        -->
                        <transition event="gsgBasedServices.reserve.done"
target="waitForCall" >
                                <script>
                                        var myResultSetObject = eval('(' +
_event.data.content + ')');
                                        var result = new Object();
                                        result._id = _sessionid;
                                        if ( myResultSetObject.hasOwnProperty(
'_access_number' ) )
                                        result._access_number =
myResultSetObject._access_number;
                                        if ( myResultSetObject.hasOwnProperty( '_access_code'
) )
                                        result._access_code = myResultSetObject._access_code;
                                        if ( myResultSetObject.hasOwnProperty(
'_expiration_time' ) )
                                        result._expiration_time =
myResultSetObject._expiration_time;
                                </script>
                                <!-- Send a response to the client indicating the result -->
                                <ws:response requestid="reserveId"
resultcode="JSON.stringify( result )" />
                        </transition>
                </state>
                <!-- Waits for the call to be associated to this session by the inbound
.scxml -->
                <state id="waitForCall">
                        <transition event="interaction.present" target="exit">
                                <!--
                                        Once here, this script has full control over the
interaction and may route the call as needed;
                                        For this sample, the call is simply disconnected.
                                -->
                                <ixn:terminate requestid="_data.reqid"
interactionid="_genesys.ixn.interactions[0].g_uid" reason="'finished service'"
resource="_genesys.ixn.interactions[0].voice.dnis"/>
                        </transition>
                </state>
                <!-- This event means that the user did not match within the allowed time -->
                <transition event="timer" target="exit" />
```

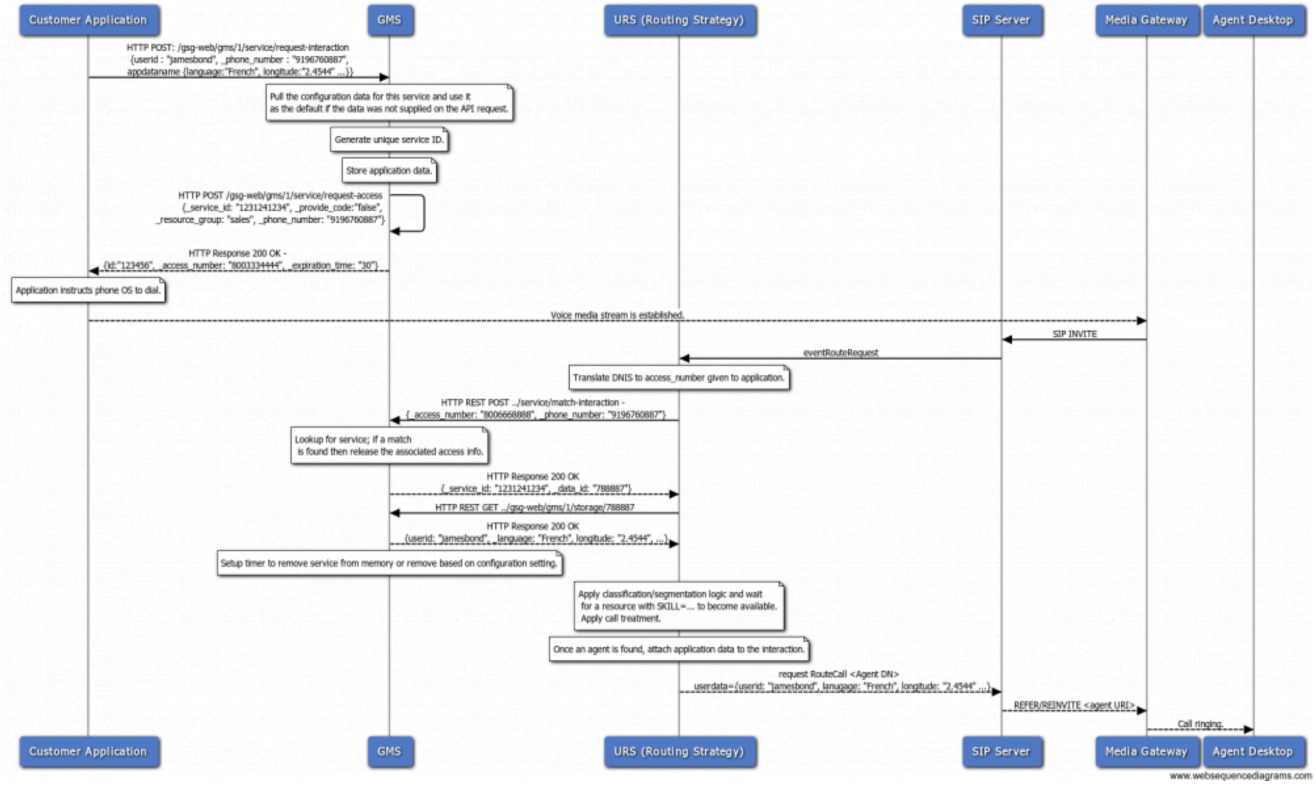
```
        </state>
        <final id="exit"/>
        <final id="error"/>
</scxml>
```

request-outbound-immediate

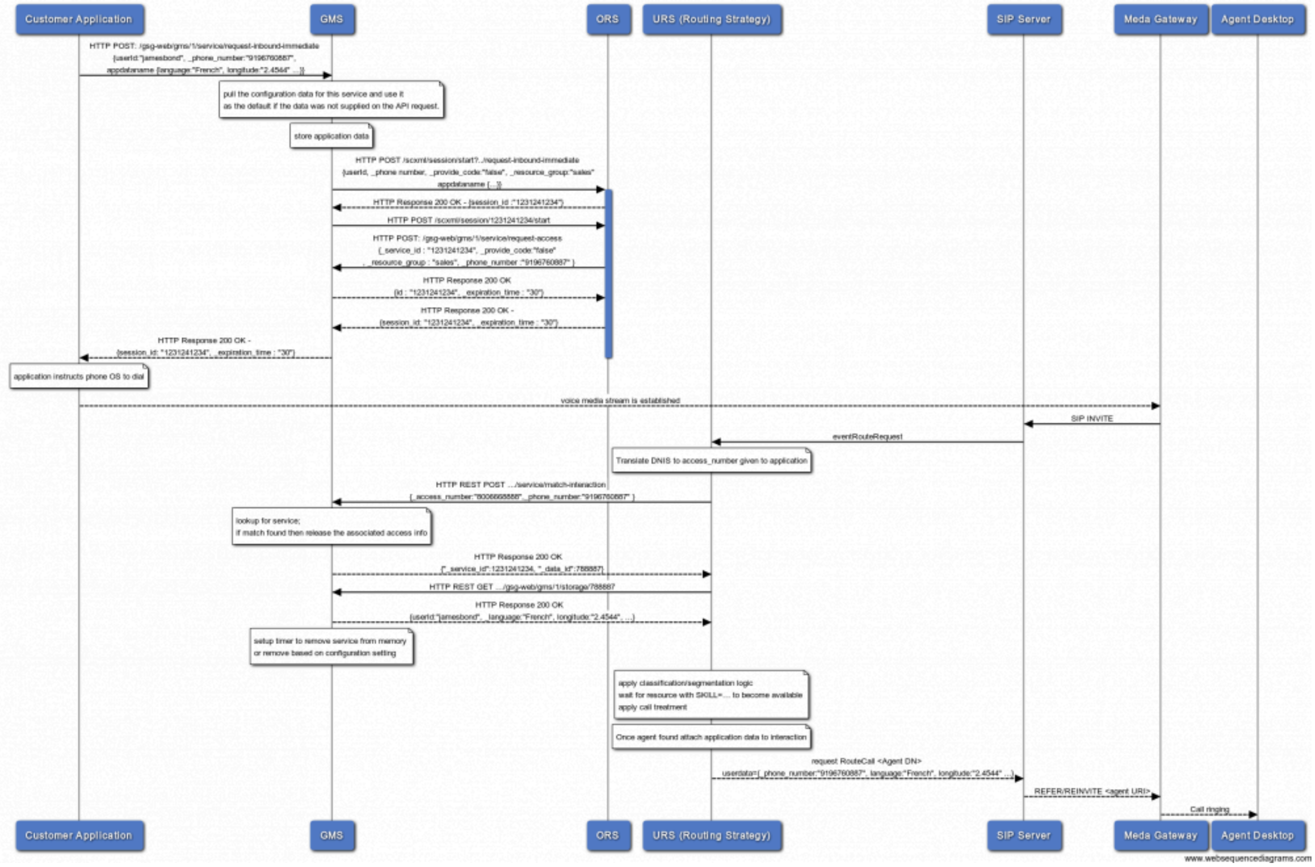
```
<scxml version="1.0" xmlns="http://www.w3.org/2005/07/scxml"
        xmlns:queue="www.genesyslab.com/modules/queue"
        xmlns:yahoo_placefinder="http://www.genesyslab.com/modules/dfm/yahoo_placefinder/v1"
        xmlns:ws="http://www.genesyslab.com/modules/ws" xmlns:session="www.genesyslab.com/
modules/session"
        xmlns:ixn="http://www.genesyslab.com/modules/interaction" initial="initial">
        <datamodel>
                <!-- parameters -->
                <data id="_phone_number" _type="parameter" />
                <data id="_data_id" _type="parameter" />
                <data id="_ttl" _type="parameter" />
                <!-- globals -->
                <data id="reqid" />
                <data id="startSendId" />
        </datamodel>
        <state id="initial">
                <transition event="gms.start" target="serviceEntry">
                        <script>
                                _data.startSendId = _event.sendid;
            </script>
                </transition>
        </state>
        <state id="serviceEntry" initial="handleStart">
                <onentry>
                        <send event="'serviceTimeout'" delay="_data._ttl"/>
                </onentry>
                <transition event="serviceTimeout" target="error" />

                <state id="handleStart">
                        <onentry>
                                <script>
                                        // TODO: Update routepoint below. Add a routepoint to
config
                                        to be used as the source below
                                        var src = {"type":"rp",
                                        "dn":"gms_preconfig_callorig_rp"};

                                        // TODO: Attach data as required.
                                        Can also fetch data using _data_id from storage
                                        var userInfo = new
                                        Object();
                                        data['name1'] = 'value1';
                                        data['name2'] = 'value2';
                                </script>
                                <ixn:createcall requestid="_data.reqid" from="src"
                                        to="_data._phone_number" udata="userInfo" />
                        </onentry>

                        <transition event="voice.createcall.done"
target="waitForOutboundCall" />
                        <transition event="error.voice.createcall" target="error" />

                        <onexit>
                                <ws:response requestid="_data.startSendId"
resultcode="_sessionid" />
                        </onexit>
                </state>
                <state id="waitForOutboundCall">
                        <transition event="interaction.present" target="connectToAgent" />
                </state>

                <state id="connectToAgent">
                        <onentry>
                                <!--
                                        TODO: route interaction using queue:submit or
ixn:redirect as
                                        appropriate
                                -->
                                <!--
                                        queue:submit …
                                                OR
                                        ixn:redirect to configured agent DN …
                                -->
                        </onentry>
                        <!--
                                Have the appropriate set of transitions to handle the results
of the
                                routing action.
                        -->
                        <transition event="*.done" target="exit" />
                </state>
        </state>
        <final id="exit" />
        <final id="error" />
</scxml>
</div>
```

Call Flows

request-interaction - No Delay

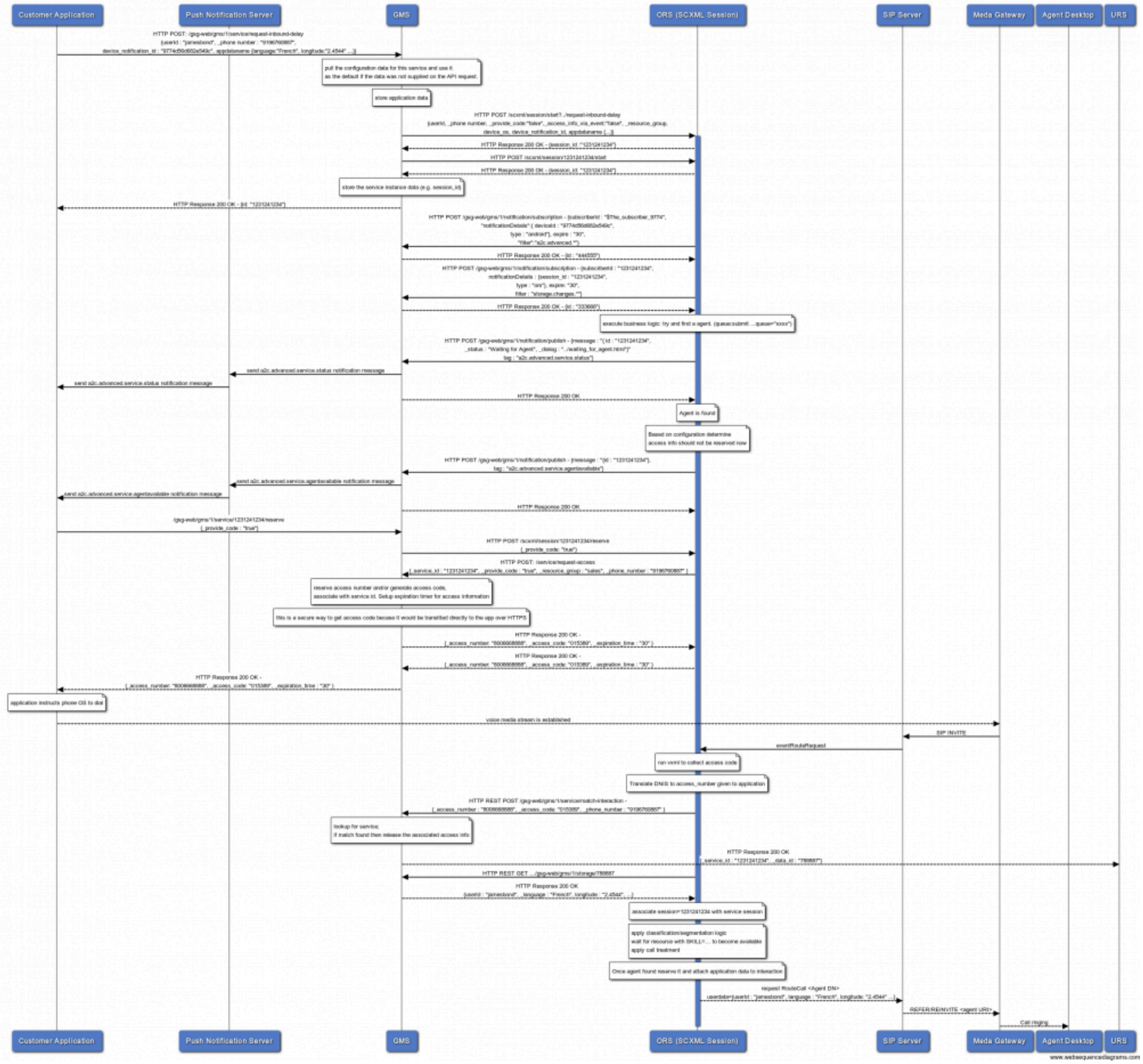

Customer Application
GMS
URS (Routing Strategy)
SIP Server
Media Gateway
Agent Desktop
HTTP POST: /gsg-web/gms/1/service/request-interaction
{userid : "jamesbond", _phone_number : "9196760887",
appdataname {language:"French", longitude:"2.4544" ...}}
Pull the configuration data for this service and use it
as the default if the data was not supplied on the API request.
Generate unique service ID.
Store application data.
HTTP POST /gsg-web/gms/1/service/request-access
{_service_id: "1231241234", _provide_code:"false",
_resource_group: "sales", _phone_number: "9196760887"}
HTTP Response 200 OK -
{id:"123456", _access_number: "8003334444", _expiration_time: "30"}
Application instructs phone OS to dial.
Voice media stream is established.
SIP INVITE
eventRouteRequest
Translate DNIS to access_number given to application.
HTTP REST POST .../service/match-interaction -
{_access_number: "8006668888", _phone_number: "9196760887"}
Lookup for service; if a match
is found then release the associated access info.
HTTP Response 200 OK
{_service_id: "1231241234", _data_id: "788887"}
HTTP REST GET ../gsg-web/gms/1/storage/788887
HTTP Response 200 OK
{userid: "jamesbond", _language: "French", longitude: "2.4544", ...}
Setup timer to remove service from memory or remove based on configuration setting.
Apply classification/segmentation logic and wait
for a resource with SKILL=.... to become available.
Apply call treatment.
Once an agent is found, attach application data to the interaction.
request RouteCall <Agent DN>
userdata={userid: "jamesbond", lanugage: "French", longitude: "2.4544" ...}
REFER/REINVITE <agent URI>
Call ringing.
Customer Application
GMS
URS (Routing Strategy)
SIP Server
Media Gateway
Agent Desktop
www.websequencediagrams.com

request-inbound-immediate - No Delay

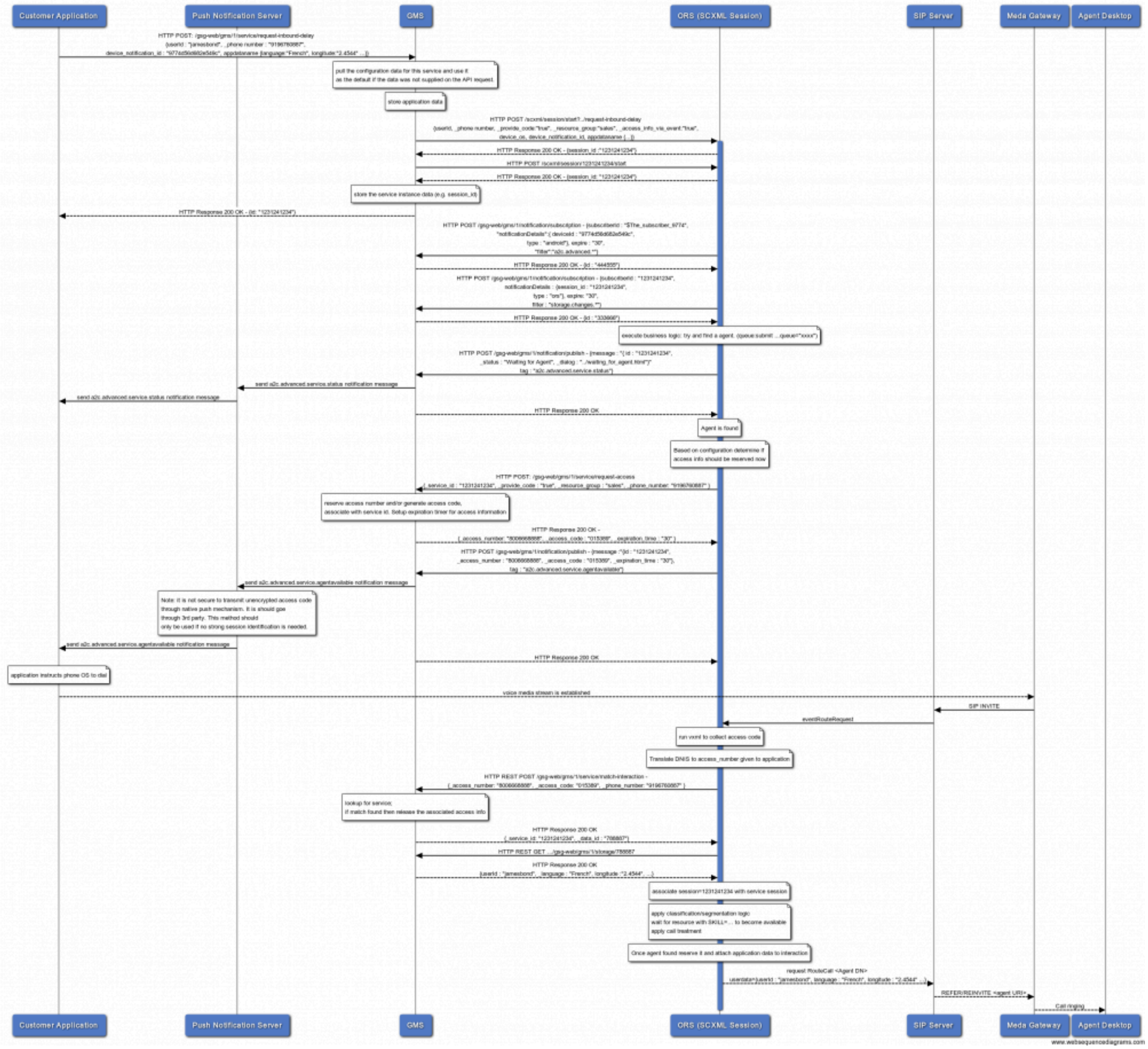

Customer Application
GMS
ORS
URS (Routing Strategy)
SIP Server
Meda Gateway
Agent Desktop
HTTP POST: /gsg-web/gms/1/service/request-inbound-immediate
{userId:"jamesbond", _phone_number:"9196760887",
appdataname {language:"French", longitude:"2.4544" ...}}
pull the configuration data for this service and use it
as the default if the data was not supplied on the API request.
store application data
HTTP POST /scxml/session/start?../request-inbound-immediate
{userId, _phone number, _provide_code:"false", _resource_group:"sales"
appdataname {...}}
HTTP Response 200 OK - {session_id :"1231241234"}
HTTP POST /scxml/session/1231241234/start
HTTP POST: /gsg-web/gms/1/service/request-access
{_service_id : "1231241234", _provide_code:"false"
, _resource_group : "sales", _phone_number :"9196760887" }
HTTP Response 200 OK
{id : "1231241234", _expiration_time : "30"}
HTTP Response 200 OK -
{session_id: "1231241234", _expiration_time : "30"}
HTTP Response 200 OK -
{session_id: "1231241234", _expiration_time : "30"}
application instructs phone OS to dial
voice media stream is established
SIP INVITE
eventRouteRequest
Translate DNIS to access_number given to application
HTTP REST POST .../service/match-interaction
{_access_number:"8006668888",_phone_number:"9196760887" }
lookup for service;
if match found then release the associated access info
HTTP Response 200 OK
{"_service_id":1231241234, "_data_id":788887}
HTTP REST GET .../gsg-web/gms/1/storage/788887
HTTP Response 200 OK
{userId:"jamesbond", _language:"French", longitude:"2.4544", ...}
setup timer to remove service from memory
or remove based on configuration setting
apply classification/segmentation logic
wait for resource with SKILL=... to become available
apply call treatment
Once agent found attach application data to interaction
request RouteCall <Agent DN>
userdata={_phone_number:"9196760887", language:"French", longitude:"2.4544" ...}
REFER/REINVITE <agent URI>
Call ringing
Customer Application
GMS
ORS
URS (Routing Strategy)
SIP Server
Meda Gateway
Agent Desktop
www.websequencediagrams.com

request-inbound-delay - Delay

Customer originated, push notification, separate call for access code

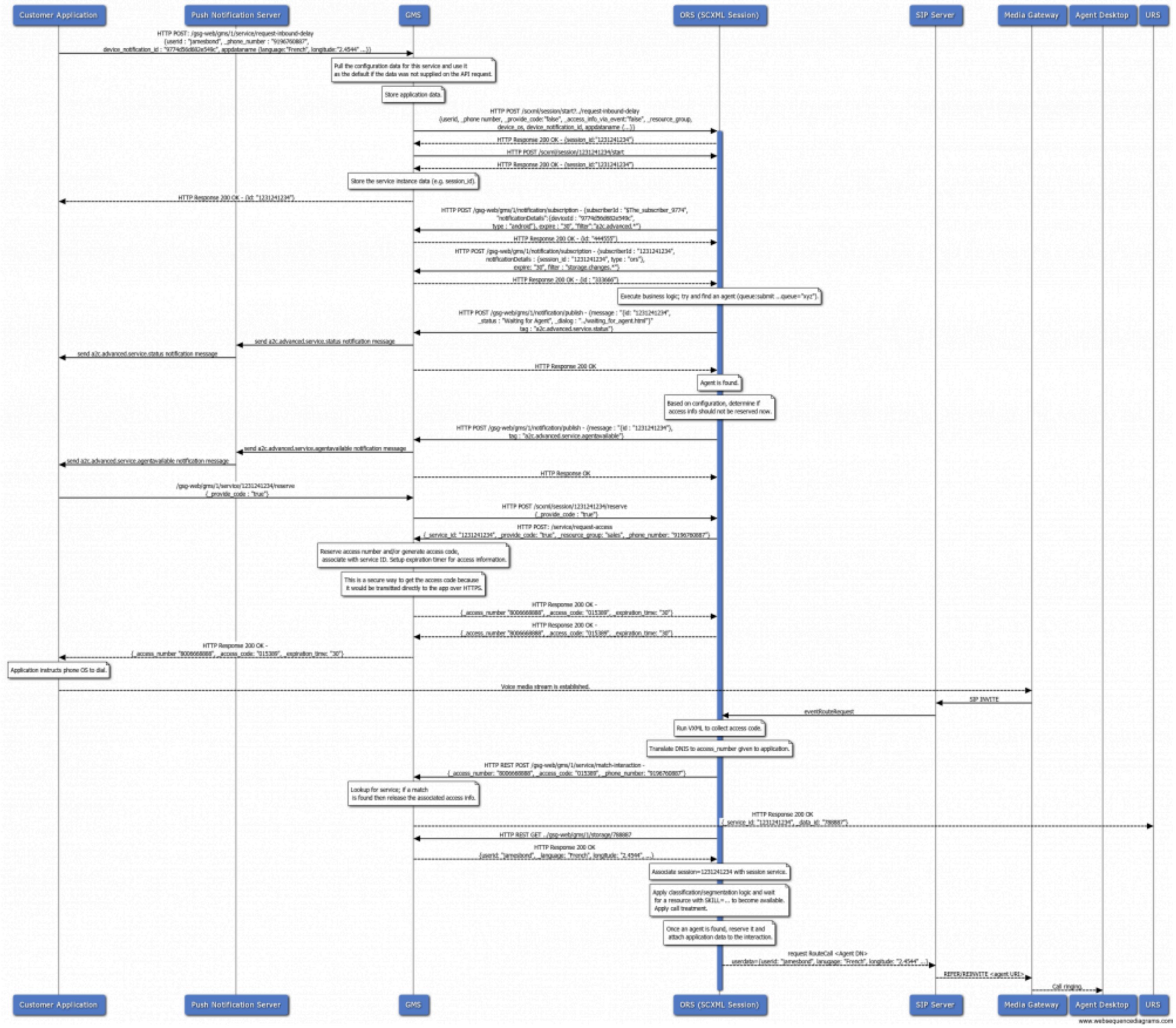

Customer Application
Push Notification Server
GMS
ORS (SCXML Session)
SIP Server
Meda Gateway
Agent Desktop
URS
HTTP POST: /gsg-web/gms/1/service/request-inbound-delay
{userId : "jamesbond", _phone number : "9196760887",
device_notification_id : "9774d56d682e549c", appdatename {language:"French", longitude:"2.4544" ...}}
pull the configuration data for this service and use it
as the default if the data was not supplied on the API request.
store application data
HTTP POST /scxml/session/start?../request-inbound-delay
{userId, _phone number, _provide_code:"false", _access_info_via_event:"false", _resource_group,
device_os, device_notification_id, appdatename {...}}
HTTP Response 200 OK - {session_id :"1231241234"}
HTTP POST /scxml/session/1231241234/start
HTTP Response 200 OK - {session_id: "1231241234"}
store the service instance data (e.g. session_id)
HTTP Response 200 OK - {id: "1231241234"}
HTTP POST /gsg-web/gms/1/notification/subscription - {subscriberId : "$The_subscriber_9774",
"notificationDetails":{ deviceId : "9774d56d682e549c",
type : "android"}, expire : "30",
"filter":"a2c.advanced.*"}
HTTP Response 200 OK - {id : "444555"}
HTTP POST /gsg-web/gms/1/notification/subscription - {subscriberId : "1231241234",
notificationDetails : {session_id : "1231241234",
type : "ors"}, expire: "30",
filter : "storage.changes.*"}
HTTP Response 200 OK - {id : "333666"}
execute business logic: try and find a agent. (queue:submit ... queue="xxxx")
HTTP POST /gsg-web/gms/1/notification/publish - {message : "{ id : "1231241234",
_status : "Waiting for Agent", _dialog : "../waiting_for_agent.html"}"
tag : "a2c.advanced.service.status"}
send a2c.advanced.service.status notification message
send a2c.advanced.service.status notification message
HTTP Response 200 OK
Agent is found
Based on configuration determine
access info should not be reserved now
HTTP POST /gsg-web/gms/1/notification/publish - {message : "{id : "1231241234"},
tag : "a2c.advanced.service.agentavailable"}
send a2c.advanced.service.agentavailable notification message
send a2c.advanced.service.agentavailable notification message
HTTP Response 200 OK
/gsg-web/gms/1/service/1231241234/reserve
{_provide_code : "true"}
HTTP POST /scxml/session/1231241234/reserve
{_provide_code: "true"}
HTTP POST: /service/request-access
{_service_id : "1231241234", _provide_code : "true", _resource_group : "sales", _phone_number : "9196760887" }
reserve access number and/or generate access code,
associate with service id. Setup expiration timer for access information
this is a secure way to get access code becase it would be transfiled directly to the app over HTTPS
HTTP Response 200 OK -
{_access_number: "8008668888", _access_code: "015389", _expiration_time : "30" }
HTTP Response 200 OK -
{_access_number: "8008668888", _access_code: "015389", _expiration_time : "30" }
HTTP Response 200 OK -
{_access_number: "8008668888", _access_code: "015389", _expiration_time : "30" }
application instructs phone OS to dial
voice media stream is established
SIP INVITE
eventRouteRequest
run vxml to collect access code
Translate DNIS to access_number given to application
HTTP REST POST /gsg-web/gms/1/service/match-interaction -
{_access_number : "8008668888", _access_code: "015389", _phone_number : "9196760887" }
lookup for service;
if match found then release the associated access info
HTTP Response 200 OK
{_service_id : "1231241234", _data_id : "786887"}
HTTP REST GET .../gsg-web/gms/1/storage/786887
HTTP Response 200 OK
{userId : "jamesbond", _language : "French", longitude : "2.4544", ...}
associate session='1231241234 with service session
apply classification/segmentation logic
wait for resource with SKILL= ... to become available
apply call treatment
Once agent found reserve it and attach application data to interaction
request RouteCall <Agent DN>
userdata={userId : "jamesbond", language : "French", longitude: "2.4544" ...}
REFER/REINVITE <agent URI>
Call ringing
www.websequencediagrams.com

Customer originated, push notification with Access Number/Access Code

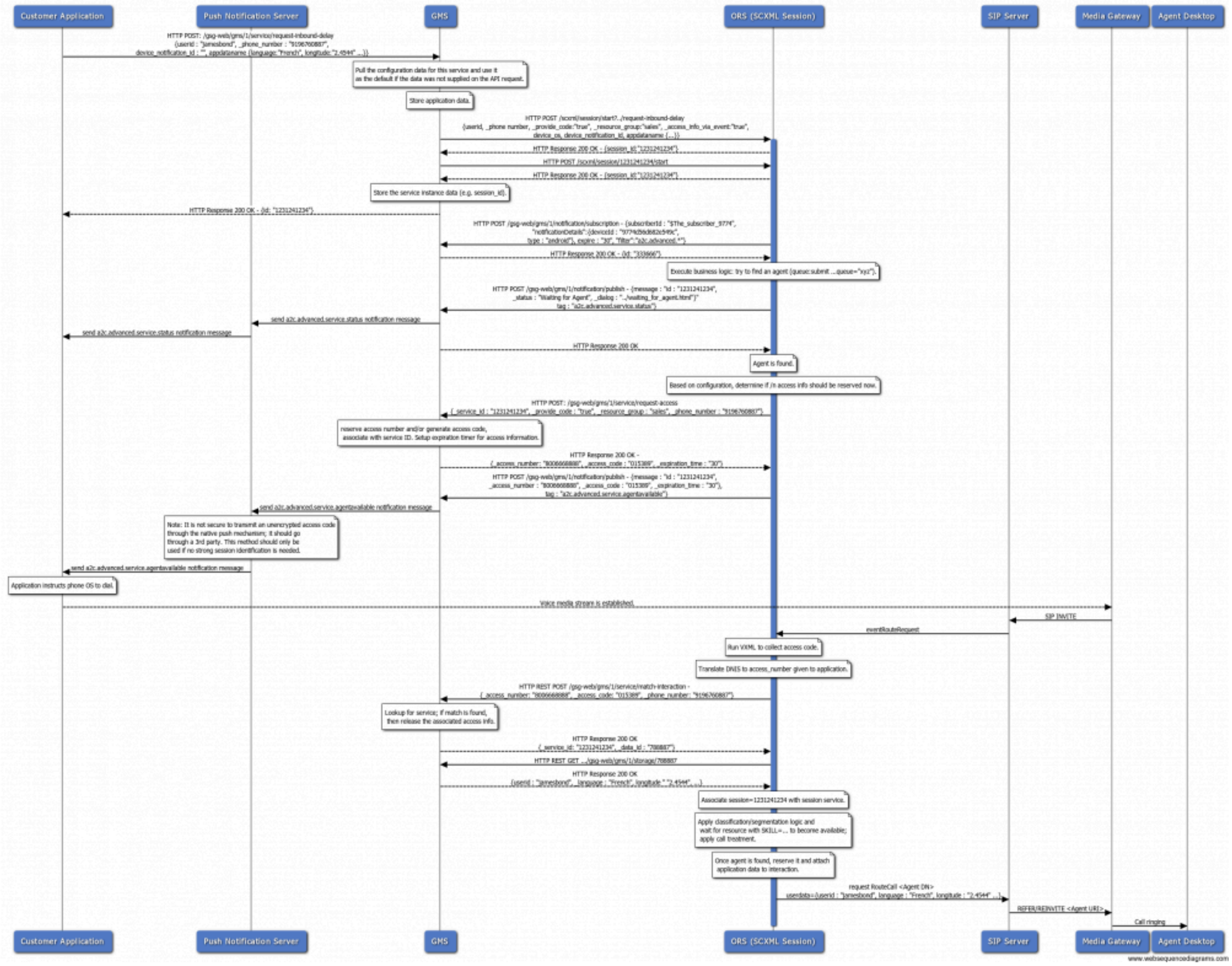

Customer Application
Push Notification Server
GMS
ORS (SCXML Session)
SIP Server
Meda Gateway
Agent Desktop
HTTP POST: /gsg-web/gms/1/service/request-inbound-delay
{userId : "jamesbond", _phone number : "9196780887",
device_notification_id : "9774d56d682e549c", appdataname {language:"French", longitude:"2.4544" ...}}
pull the configuration data for this service and use it
as the default if the data was not supplied on the API request.
store application data
HTTP POST /scxml/session/start?../request-inbound-delay
{userId, _phone number, _provide_code:"true", _resource_group:"sales", _access_info_via_event:"true",
device_os, device_notification_id, appdataname {...}}
HTTP Response 200 OK - {session_id :"1231241234"}
HTTP POST /scxml/session/1231241234/start
HTTP Response 200 OK - {session_id: "1231241234"}
store the service instance data (e.g. session_id)
HTTP Response 200 OK - {id: "1231241234"}
HTTP POST /gsg-web/gms/1/notification/subscription - {subscriberId : "$The_subscriber_9774",
"notificationDetails":{ deviceId : "9774d56d682e549c",
type : "android"}, expire : "30",
"filter":"a2c.advanced.*"}
HTTP Response 200 OK - {id : "444555"}
HTTP POST /gsg-web/gms/1/notification/subscription - {subscriberId : "1231241234",
notificationDetails : {session_id : "1231241234",
type : "ors"}, expire: "30",
filter : "storage.changes.*"}
HTTP Response 200 OK - {id : "333666"}
execute business logic: try and find a agent. (queue:submit ...queue="xxxx")
HTTP POST /gsg-web/gms/1/notification/publish - {message : "{ id : "1231241234",
_status : "Waiting for Agent", _dialog : "../waiting_for_agent.html"}"
tag : "a2c.advanced.service.status"}
send a2c.advanced.service.status notification message
send a2c.advanced.service.status notification message
HTTP Response 200 OK
Agent is found
Based on configuration determine if
access info should be reserved now
HTTP POST: /gsg-web/gms/1/service/request-access
{_service_id : "1231241234", _provide_code : "true", _resource_group : "sales", _phone_number: "9196780887" }
reserve access number and/or generate access code,
associate with service id. Setup expiration timer for access information
HTTP Response 200 OK -
{_access_number: "8008668888", _access_code : "015389", _expiration_time : "30" }
HTTP POST /gsg-web/gms/1/notification/publish - {message :"{id : "1231241234",
_access_number : "8008668888", _access_code : "015389", _expiration_time : "30"},
tag : "a2c.advanced.service.agentavailable"}
send a2c.advanced.service.agentavailable notification message
Note: It is not secure to transmit unencrypted access code
through native push mechanism. It is should goe
through 3rd party. This method should
only be used if no strong session identification is needed.
send a2c.advanced.service.agentavailable notification message
HTTP Response 200 OK
application instructs phone OS to dial
voice media stream is established
SIP INVITE
eventRouteRequest
run vxml to collect access code
Translate DNIS to access_number given to application
HTTP REST POST /gsg-web/gms/1/service/match-interaction -
{_access_number: "8008668888", _access_code: "015389", _phone_number: "9196780887" }
lookup for service;
if match found then release the associated access info
HTTP Response 200 OK
{_service_id: "1231241234", _data_id : "788887"}
HTTP REST GET .../gsg-web/gms/1/storage/788887
HTTP Response 200 OK
{userId : "jamesbond", _language : "French", longitude :"2.4544", ...}
associate session=1231241234 with service session
apply classification/segmentation logic
wait for resource with SKILL= ... to become available
apply call treatment
Once agent found reserve it and attach application data to interaction
request RouteCall <Agent DN>
userdata={userId : "jamesbond", language : "French", longitude : "2.4544" ...}
REFER/REINVITE <agent URI>
Call ringing
www.websequencediagrams.com

request-inbound-delay-location

Customer originated, push notification, separate call for access code

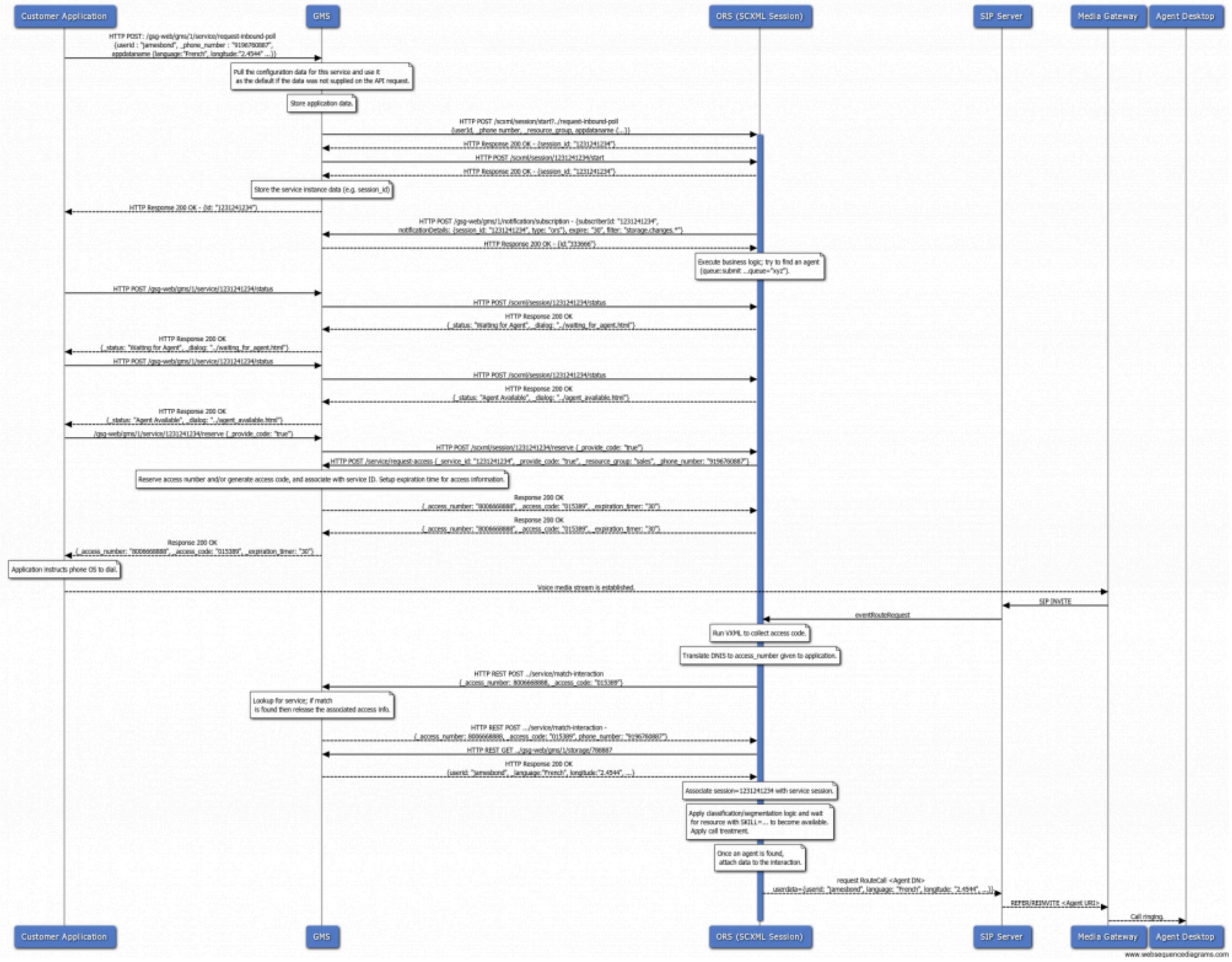

Customer Application
Push Notification Server
GMS
ORS (SCXML Session)
SIP Server
Media Gateway
Agent Desktop
URS
HTTP POST: /gsg-web/gms/1/service/request-inbound-delay
{userid : "jamesbond", _phone_number : "9196760887",
device_notification_id : "9774d56d682e549c", appdataname {language:"French", longitude:"2.4544" ...}}
Pull the configuration data for this service and use it
as the default if the data was not supplied on the API request.
Store application data.
HTTP POST /scxml/session/start?../request-inbound-delay
{userid, _phone number, _provide_code:"false", _access_info_via_event:"false", _resource_group,
device_os, device_notification_id, appdataname {...}}
HTTP Response 200 OK - {session_id:"1231241234"}
HTTP POST /scxml/session/1231241234/start
HTTP Response 200 OK - {session_id:"1231241234"}
Store the service instance data (e.g. session_id).
HTTP Response 200 OK - {id: "1231241234"}
HTTP POST /gsg-web/gms/1/notification/subscription - {subscriberId : "$The_subscriber_9774",
"notificationDetails":{deviceId : "9774d56d682e549c",
type : "android"}, expire : "30", "filter":"a2c.advanced.*"}
HTTP Response 200 OK - {id: "444555"}
HTTP POST /gsg-web/gms/1/notification/subscription - {subscriberId : "1231241234",
notificationDetails : {session_id : "1231241234", type : "ors"},
expire: "30", filter : "storage.changes.*"}
HTTP Response 200 OK - {id : "333666"}
Execute business logic; try and find an agent (queue:submit ...queue="xyz").
HTTP POST /gsg-web/gms/1/notification/publish - {message : "{id: "1231241234",
_status : "Waiting for Agent", _dialog : "../waiting_for_agent.html"}"
tag : "a2c.advanced.service.status"}
send a2c.advanced.service.status notification message
send a2c.advanced.service.status notification message
HTTP Response 200 OK
Agent is found.
Based on configuration, determine if
access info should not be reserved now.
HTTP POST /gsg-web/gms/1/notification/publish - {message : "{id : "1231241234"},
tag : "a2c.advanced.service.agentavailable"}
send a2c.advanced.service.agentavailable notification message
send a2c.advanced.service.agentavailable notification message
HTTP Response OK
/gsg-web/gms/1/service/1231241234/reserve
{_provide_code : "true"}
HTTP POST /scxml/session/1231241234/reserve
{_provide_code : "true"}
HTTP POST: /service/request-access
{_service_id: "1231241234", _provide_code: "true", _resource_group: "sales", _phone_number: "9196760887"}
Reserve access number and/or generate access code,
associate with service ID. Setup expiration timer for access information.
This is a secure way to get the access code because
it would be transitted directly to the app over HTTPS.
HTTP Response 200 OK -
{_access_number "8006668888", _access_code: "015389", _expiration_time: "30"}
HTTP Response 200 OK -
{_access_number "8006668888", _access_code: "015389", _expiration_time: "30"}
HTTP Response 200 OK -
{_access_number "8006668888", _access_code: "015389", _expiration_time: "30"}
Application instructs phone OS to dial.
Voice media stream is established.
SIP INVITE
eventRouteRequest
Run VXML to collect access code.
Translate DNIS to access_number given to application.
HTTP REST POST /gsg-web/gms/1/service/match-interaction -
{_access_number: "8006668888", _access_code: "015389", _phone_number: "9196760887"}
Lookup for service; if a match
is found then release the associated access info.
HTTP Response 200 OK
{_service_id: "1231241234", _data_id: "788887"}
HTTP REST GET ../gsg-web/gms/1/storage/788887
HTTP Response 200 OK
{userid: "jamesbond", _language: "French", longitude: "2.4544", ...}
Associate session=1231241234 with session service.
Apply classification/segmentation logic and wait
for a resource with SKILL=... to become available.
Apply call treatment.
Once an agent is found, reserve it and
attach application data to the interaction.
request RouteCall <Agent DN>
userdata={userid: "jamesbond", lanugage: "French", longitude: "2.4544" ...}
REFER/REINVITE <agent URI>
Call ringing.
www.websequencediagrams.com

Customer originated, push notification with Access Number/Access Code

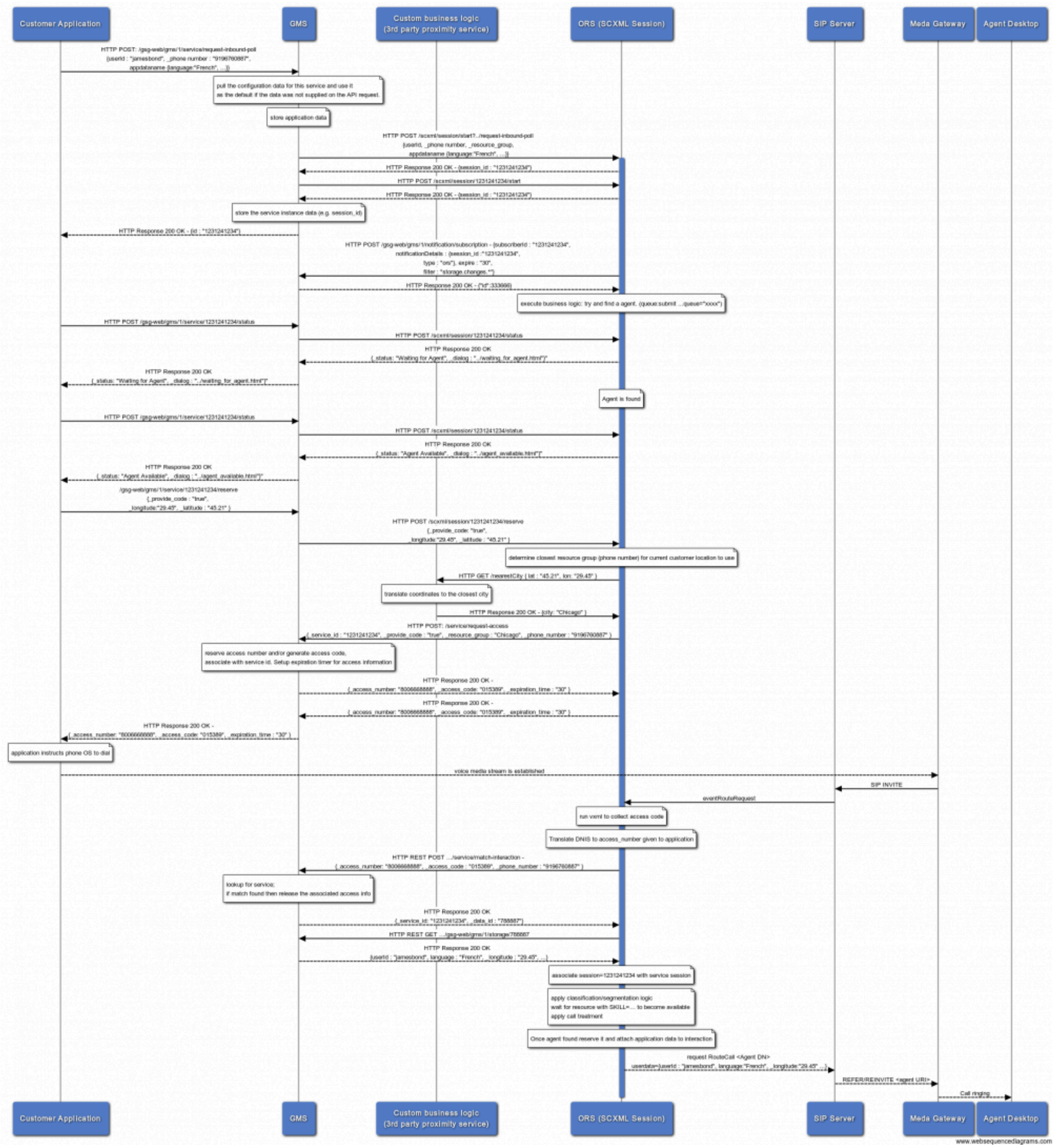

Customer Application
Push Notification Server
GMS
ORS (SCXML Session)
SIP Server
Media Gateway
Agent Desktop
HTTP POST: /gsg-web/gms/1/service/request-inbound-delay {userid : "jamesbond", _phone_number : "9196760887", device_notification_id : "", appdataname {language:"French", longtitude:"2.4544" ...}}
Pull the configuration data for this service and use it as the default if the data was not supplied on the API request.
Store application data.
HTTP POST /scxml/session/start?../request-inbound-delay {userid, _phone number, _provide_code:"true", _resource_group:"sales", _access_info_via_event:"true", device_os, device_notification_id, appdataname {...}}
HTTP Response 200 OK - {session_id:"1231241234"}
HTTP POST /scxml/session/1231241234/start
HTTP Response 200 OK - {session_id:"1231241234"}
Store the service instance data (e.g. session_id).
HTTP Response 200 OK - {id: "1231241234"}
HTTP POST /gsg-web/gms/1/notification/subscription - {subscriberId : "$The_subscriber_9774", "notificationDetails":{deviceId : "9774d56d682e549c", type : "android"}, expire : "30", "filter":"a2c.advanced.*"}
HTTP Response 200 OK - {id: "333666"}
Execute business logic: try to find an agent (queue:submit ...queue="xyz").
HTTP POST /gsg-web/gms/1/notification/publish - {message : "id : "1231241234", _status : "Waiting for Agent", _dialog : "../waiting_for_agent.html"}" tag : "a2c.advanced.service.status"}
send a2c.advanced.service.status notification message
send a2c.advanced.service.status notification message
HTTP Response 200 OK
Agent is found.
Based on configuration, determine if /n access info should be reserved now.
HTTP POST: /gsg-web/gms/1/service/request-access {_service_id : "1231241234", _provide_code : "true", _resource_group : "sales", _phone_number : "9196760887"}
reserve access number and/or generate access code, associate with service ID. Setup expiration timer for access information.
HTTP Response 200 OK - {_access_number: "8006668888", _access_code : "015389", _expiration_time : "30"}
HTTP POST /gsg-web/gms/1/notification/publish - {message : "id : "1231241234", _access_number : "8006668888", _access_code : "015389", _expiration_time : "30"}, tag : "a2c.advanced.service.agentavailable"}
send a2c.advanced.service.agentavailable notification message
Note: It is not secure to transmit an unencrypted access code through the native push mechanism; it should go through a 3rd party. This method should only be used if no strong session identification is needed.
send a2c.advanced.service.agentavailable notification message
Application instructs phone OS to dial.
Voice media stream is established.
SIP INVITE
eventRouteRequest
Run VXML to collect access code.
Translate DNIS to access_number given to application.
HTTP REST POST /gsg-web/gms/1/service/match-interaction - {_access_number: "8006668888", _access_code: "015389", _phone_number: "9196760887"}
Lookup for service; if match is found, then release the associated access info.
HTTP Response 200 OK {_service_id: "1231241234", _data_id : "708887"}
HTTP REST GET .../gsg-web/gms/1/storage/708887
HTTP Response 200 OK {userid : "jamesbond", _language : "French", longtitude " "2.4544", ...}
Associate session=1231241234 with session service.
Apply classification/segmentation logic and wait for resource with SKILL=... to become available; apply call treatment.
Once agent is found, reserve it and attach application data to interaction.
request RouteCall <Agent DN> userdata={userid : "jamesbond", language : "French", longitude : "2.4544" ...}
REFER/REINVITE <Agent URI>
Call ringing
www.websequencediagrams.com

request-inbound-poll

Basic Allocation


Customer Application
GMS
ORS (SCXML Session)
SIP Server
Media Gateway
Agent Desktop
HTTP POST: /gsg-web/gms/1/service/request-inbound-poll
{userid : "jamesbond", _phone_number : "9196760887",
appdataname {language:"French", longitude:"2.4544" ...}}
Pull the configuration data for this service and use it
as the default if the data was not supplied on the API request.
Store application data.
HTTP POST /scxml/session/start?../request-inbound-poll
{userId, _phone number, _resource_group, appdataname {...}}
HTTP Response 200 OK - {session_id: "1231241234"}
HTTP POST /scxml/session/1231241234/start
HTTP Response 200 OK - {session_id: "1231241234"}
Store the service instance data (e.g. session_id)
HTTP Response 200 OK - {id: "1231241234"}
HTTP POST /gsg-web/gms/1/notification/subscription - {subscriberId: "1231241234",
notificationDetails: {session_id: "1231241234", type: "ors"}, expire: "30", filter: "storage.changes.*"}
HTTP Response 200 OK - {id:"333666"}
Execute business logic; try to find an agent
(queue:submit ...queue="xyz").
HTTP POST /gsg-web/gms/1/service/1231241234/status
HTTP POST /scxml/session/1231241234/status
HTTP Response 200 OK
{_status: "Waiting for Agent", _dialog: "../waiting_for_agent.html"}
HTTP Response 200 OK
{_status: "Waiting for Agent", _dialog: "../waiting_for_agent.html"}
HTTP POST /gsg-web/gms/1/service/1231241234/status
HTTP POST /scxml/session/1231241234/status
HTTP Response 200 OK
{_status: "Agent Available", _dialog: "../agent_available.html"}
HTTP Response 200 OK
{_status: "Agent Available", _dialog: "../agent_available.html"}
/gsg-web/gms/1/service/1231241234/reserve {_provide_code: "true"}
HTTP POST /scxml/session/1231241234/reserve {_provide_code: "true"}
HTTP POST /service/request-access {_service_id: "1231241234", _provide_code: "true", _resource_group: "sales", _phone_number: "9196760887"}
Reserve access number and/or generate access code, and associate with service ID. Setup expiration time for access information.
Response 200 OK
{_access_number: "8006668888", _access_code: "015389", _expiration_timer: "30"}
Response 200 OK
{_access_number: "8006668888", _access_code: "015389", _expiration_timer: "30"}
Response 200 OK
{_access_number: "8006668888", _access_code: "015389", _expiration_timer: "30"}
Application instructs phone OS to dial.
Voice media stream is established.
SIP INVITE
eventRouteRequest
Run VXML to collect access code.
Translate DNIS to access_number given to application.
HTTP REST POST ../service/match-interaction
{_access_number: 8006668888, _access_code: "015389"}
Lookup for service; if match
is found then release the associated access info.
HTTP REST POST .../service/match-interaction -
{_access_number: 8006668888, _access_code: "015389", phone_number: "9196760887"}
HTTP REST GET ../gsg-web/gms/1/storage/788887
HTTP Response 200 OK
{userid: "jamesbond", _language:"French", longitude:"2.4544", ...}
Associate session=1231241234 with service session.
Apply classification/segmentation logic and wait
for resource with SKILL=... to become available.
Apply call treatment.
Once an agent is found,
attach data to the interaction.
request RouteCall <Agent DN>
userdata={userid: "jamesbond", language: "French", longitude: "2.4544", ...}}
REFER/REINVITE <Agent URI>
Call ringing.
www.websequencediagrams.com

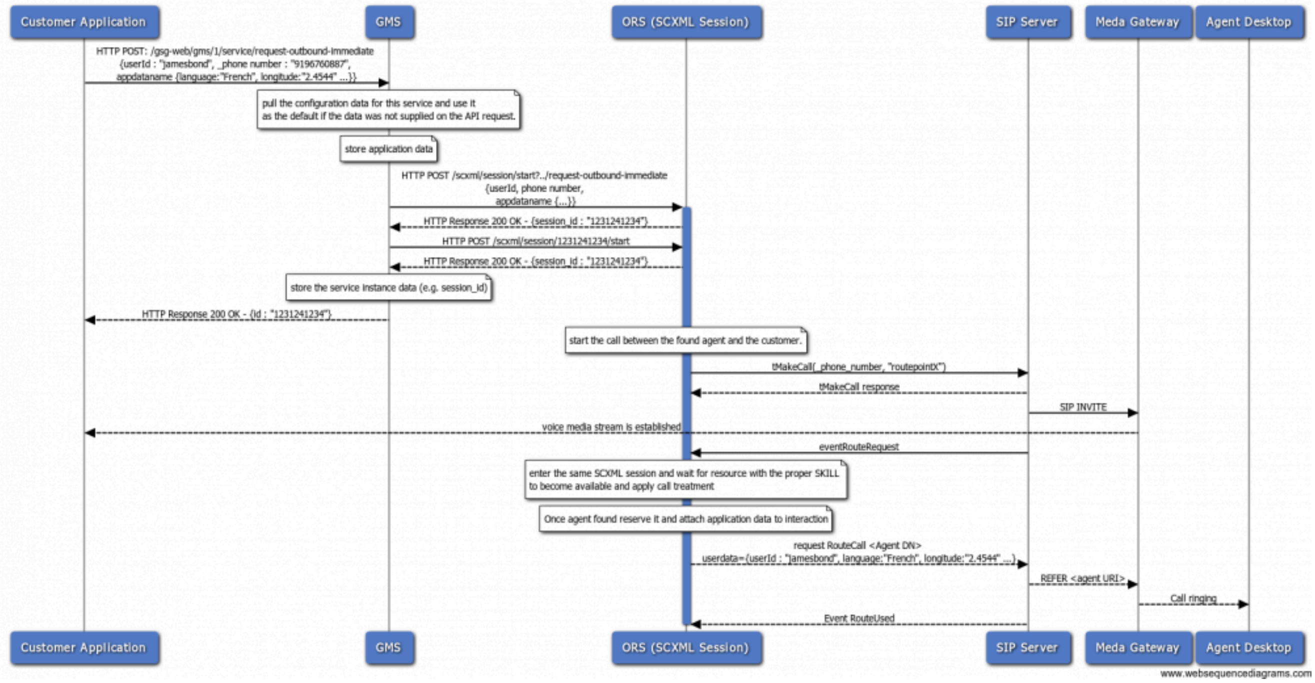
Location based Allocation


Customer Application
GMS
Custom business logic (3rd party proximity service)
ORS (SCXML Session)
SIP Server
Meda Gateway
Agent Desktop
HTTP POST: /gsg-web/gms/1/service/request-inbound-poll
{userId : "jamesbond", _phone number : "9196760887",
appdataname {language:"French", ...}}
pull the configuration data for this service and use it
as the default if the data was not supplied on the API request.
store application data
HTTP POST /scxml/session/start?../request-inbound-poll
{userId, _phone number, _resource_group,
appdataname {language:"French", ...}}
HTTP Response 200 OK - {session_id : "1231241234"}
HTTP POST /scxml/session/1231241234/start
HTTP Response 200 OK - {session_id : "1231241234"}
store the service instance data (e.g. session_id)
HTTP Response 200 OK - {id : "1231241234"}
HTTP POST /gsg-web/gms/1/notification/subscription - {subscriberId : "1231241234",
notificationDetails : {session_id :"1231241234",
type : "ors"}, expire : "30",
filter : "storage.changes.*"}
HTTP Response 200 OK - {"id":333666}
execute business logic: try and find a agent. (queue:submit ...queue="xxxx")
HTTP POST /gsg-web/gms/1/service/1231241234/status
HTTP POST /scxml/session/1231241234/status
HTTP Response 200 OK
{_status: "Waiting for Agent", _dialog : "../waiting_for_agent.html"}
HTTP Response 200 OK
{_status: "Waiting for Agent", _dialog : "../waiting_for_agent.html"}
Agent is found
HTTP POST /gsg-web/gms/1/service/1231241234/status
HTTP POST /scxml/session/1231241234/status
HTTP Response 200 OK
{_status: "Agent Available", _dialog : "../agent_available.html"}
HTTP Response 200 OK
{_status: "Agent Available", _dialog : "../agent_available.html"}
/gsg-web/gms/1/service/1231241234/reserve
{_provide_code : "true",
_longitude:"29.45", _latitude : "45.21" }
HTTP POST /scxml/session/1231241234/reserve
{_provide_code: "true",
_longitude:"29.45", _latitude : "45.21" }
determine closest resource group (phone number) for current customer location to use
HTTP GET /nearestCity { lat : "45.21", lon: "29.45" }
translate coordinates to the closest city
HTTP Response 200 OK - {city: "Chicago" }
HTTP POST: /service/request-access
{_service_id : "1231241234", _provide_code : "true", _resource_group : "Chicago", _phone_number : "9196760887" }
reserve access number and/or generate access code,
associate with service id. Setup expiration timer for access information
HTTP Response 200 OK -
{_access_number: "8006666888", _access_code: "015389", _expiration_time : "30" }
HTTP Response 200 OK -
{_access_number: "8006666888", _access_code: "015389", _expiration_time : "30" }
HTTP Response 200 OK -
{_access_number: "8006666888", _access_code: "015389", _expiration_time : "30" }
application instructs phone OS to dial
voice media stream is established
SIP INVITE
eventRouteRequest
run vxml to collect access code
Translate DNIS to access_number given to application
HTTP REST POST .../service/match-interaction -
{_access_number: "8006666888", _access_code : "015389", _phone_number : "9196760887" }
lookup for service;
if match found then release the associated access info
HTTP Response 200 OK
{_service_id: "1231241234", _data_id : "788887"}
HTTP REST GET .../gsg-web/gms/1/storage/788887
HTTP Response 200 OK
{userId : "jamesbond", language : "French", _longitude : "29.45", ...}
associate session=1231241234 with service session
apply classification/segmentation logic
wait for resource with SKILL=.... to become available
apply call treatment
Once agent found reserve it and attach application data to interaction
request RouteCall <Agent DN>
userdata={userId : "jamesbond", language:"French", _longitude:"29.45" ...}
REFER/REINVITE <agent URI>
Call ringing
www.websequencediagrams.com

a2c-advanced-sync-enterprise-initiated request-outbound-immediate a2c-advanced-async-enterprise-initiated request-outbound-delay a2c-advanced-poll-enterprise-initiated request-outbound-poll

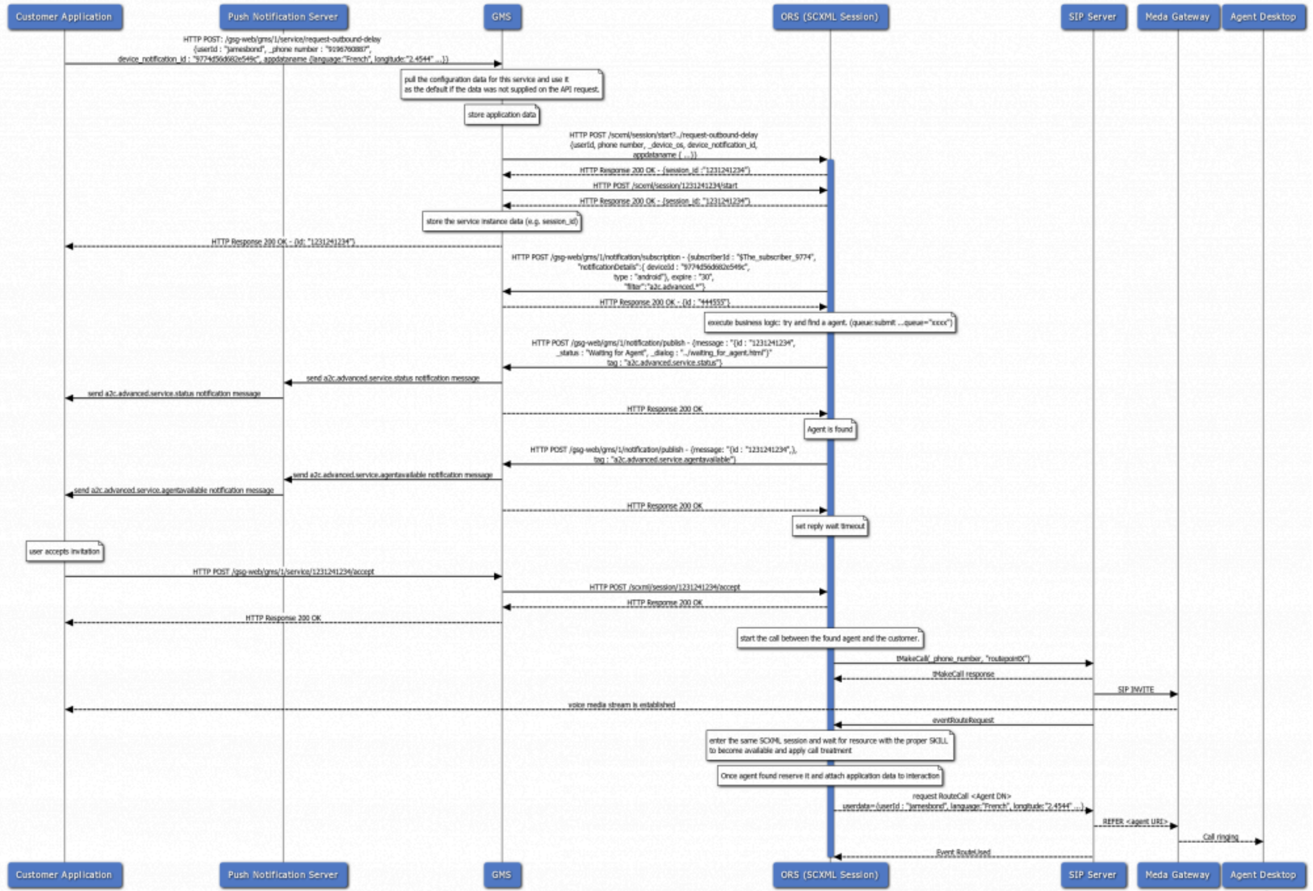
Advanced Outbound Requests

No Delay – request-outbound-immediate


Customer Application
GMS
ORS (SCXML Session)
SIP Server
Meda Gateway
Agent Desktop
HTTP POST: /gsg-web/gms/1/service/request-outbound-immediate
{userId : "jamesbond", _phone number : "9196760887",
appdataname {language:"French", longitude:"2.4544" ...}}
pull the configuration data for this service and use it
as the default if the data was not supplied on the API request.
store application data
HTTP POST /scxml/session/start?../request-outbound-immediate
{userId, phone number,
appdataname {...}}
HTTP Response 200 OK - {session_id : "1231241234"}
HTTP POST /scxml/session/1231241234/start
HTTP Response 200 OK - {session_id : "1231241234"}
store the service instance data (e.g. session_id)
HTTP Response 200 OK - {id : "1231241234"}
start the call between the found agent and the customer.
tMakeCall(_phone_number, "routepointX")
tMakeCall response
SIP INVITE
voice media stream is established
eventRouteRequest
enter the same SCXML session and wait for resource with the proper SKILL
to become available and apply call treatment
Once agent found reserve it and attach application data to interaction
request RouteCall <Agent DN>
userdata={userId : "jamesbond", language:"French", longitude:"2.4544" ...}
REFER <agent URI>
Call ringing
Event RouteUsed
Customer Application
GMS
ORS (SCXML Session)
SIP Server
Meda Gateway
Agent Desktop

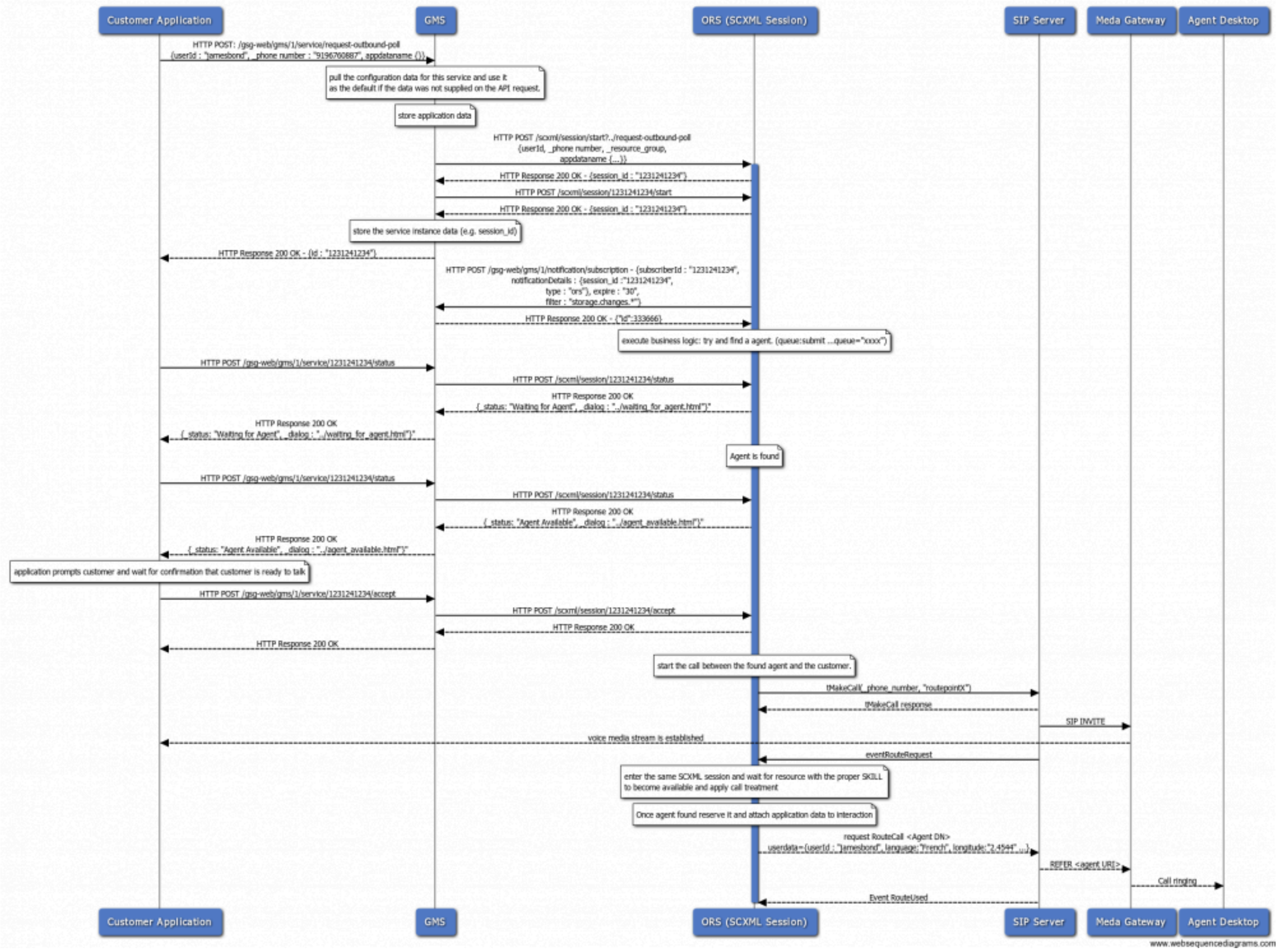
www.websequencediagrams.com

Delay with Notification – request-outbound-delay


Customer Application
Push Notification Server
GMS
ORS (SCXML Session)
SIP Server
Meda Gateway
Agent Desktop
HTTP POST: /gsg-web/gms/1/service/request-outbound-delay
{userId : "jamesbond", _phone number : "9196760887",
device_notification_id : "9774d56d682e549c", appdataname {language:"French", longitude:"2.4544" ...}}
pull the configuration data for this service and use it
as the default if the data was not supplied on the API request.
store application data
HTTP POST /scxml/session/start?../request-outbound-delay
{userId, phone number, _device_os, device_notification_id,
appdataname { ...}}
HTTP Response 200 OK - {session_id :"1231241234"}
HTTP POST /scxml/session/1231241234/start
HTTP Response 200 OK - {session_id: "1231241234"}
store the service instance data (e.g. session_id)
HTTP Response 200 OK - {id: "1231241234"}
HTTP POST /gsg-web/gms/1/notification/subscription - {subscriberId : "$The_subscriber_9774",
"notificationDetails":{ deviceId : "9774d56d682e549c",
type : "android"}, expire : "30",
"filter":"a2c.advanced.*"}
HTTP Response 200 OK - {id : "444555"}
execute business logic: try and find a agent. (queue:submit ...queue="xxxx")
HTTP POST /gsg-web/gms/1/notification/publish - {message : "{id : "1231241234",
_status : "Waiting for Agent", _dialog : "../waiting_for_agent.html"}"
tag : "a2c.advanced.service.status"}
send a2c.advanced.service.status notification message
send a2c.advanced.service.status notification message
HTTP Response 200 OK
Agent is found
HTTP POST /gsg-web/gms/1/notification/publish - {message: "{id : "1231241234",},
tag : "a2c.advanced.service.agentavailable"}
send a2c.advanced.service.agentavailable notification message
send a2c.advanced.service.agentavailable notification message
HTTP Response 200 OK
set reply wait timeout
user accepts invitation
HTTP POST /gsg-web/gms/1/service/1231241234/accept
HTTP POST /scxml/session/1231241234/accept
HTTP Response 200 OK
HTTP Response 200 OK
start the call between the found agent and the customer.
tMakeCall(_phone_number, "routepointX")
tMakeCall response
SIP INVITE
voice media stream is established
eventRouteRequest
enter the same SCXML session and wait for resource with the proper SKILL
to become available and apply call treatment
Once agent found reserve it and attach application data to interaction
request RouteCall <Agent DN>
userdata={userId : "jamesbond", language:"French", longitude:"2.4544" ...}
REFER <agent URI>
Call ringing
Event RouteUsed
Customer Application
Push Notification Server
GMS
ORS (SCXML Session)
SIP Server
Meda Gateway
Agent Desktop

www.websequencediagrams.com

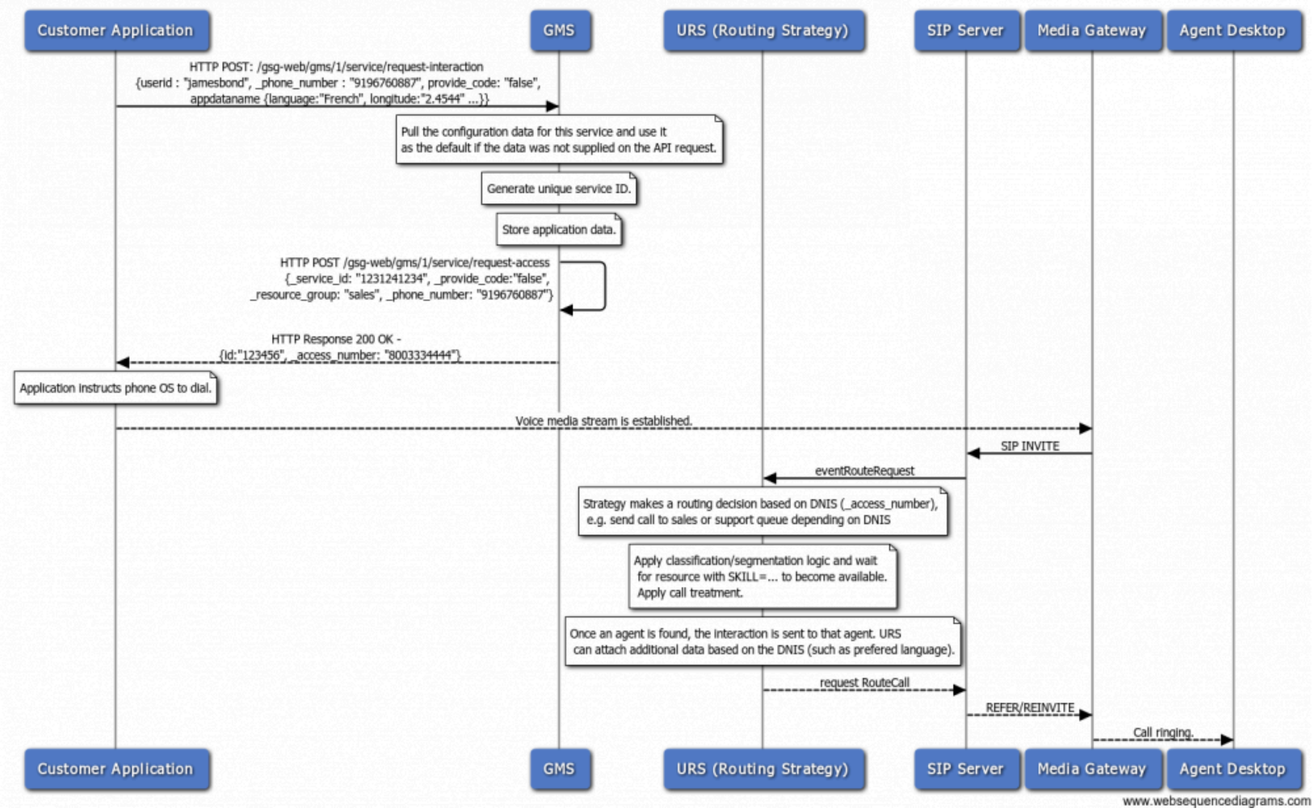
Delay-Polling – request-outbound-poll


Customer Application
GMS
ORS (SCXML Session)
SIP Server
Meda Gateway
Agent Desktop
HTTP POST: /gsg-web/gms/1/service/request-outbound-poll
{userId : "jamesbond", _phone number : "9196760887", appdataname {}}
pull the configuration data for this service and use it
as the default if the data was not supplied on the API request.
store application data
HTTP POST /scxml/session/start?../request-outbound-poll
{userId, _phone number, _resource_group,
appdataname {...}}
HTTP Response 200 OK - {session_id : "1231241234"}
HTTP POST /scxml/session/1231241234/start
HTTP Response 200 OK - {session_id : "1231241234"}
store the service instance data (e.g. session_id)
HTTP Response 200 OK - {id : "1231241234"}
HTTP POST /gsg-web/gms/1/notification/subscription - {subscriberId : "1231241234",
notificationDetails : {session_id :"1231241234",
type : "ors"}, expire : "30",
filter : "storage.changes.*"}
HTTP Response 200 OK - {"id":333666}
execute business logic: try and find a agent. (queue:submit ...queue="xxxx")
HTTP POST /gsg-web/gms/1/service/1231241234/status
HTTP POST /scxml/session/1231241234/status
HTTP Response 200 OK
{_status: "Waiting for Agent", _dialog : "../waiting_for_agent.html"}"
HTTP Response 200 OK
{_status: "Waiting for Agent", _dialog : "../waiting_for_agent.html"}"
Agent is found
HTTP POST /gsg-web/gms/1/service/1231241234/status
HTTP POST /scxml/session/1231241234/status
HTTP Response 200 OK
{_status: "Agent Available", _dialog : "../agent_available.html"}"
HTTP Response 200 OK
{_status: "Agent Available", _dialog : "../agent_available.html"}"
application prompts customer and wait for confirmation that customer is ready to talk
HTTP POST /gsg-web/gms/1/service/1231241234/accept
HTTP POST /scxml/session/1231241234/accept
HTTP Response 200 OK
HTTP Response 200 OK
start the call between the found agent and the customer.
tMakeCall(_phone_number, "routepointX")
tMakeCall response
SIP INVITE
voice media stream is established
eventRouteRequest
enter the same SCXML session and wait for resource with the proper SKILL
to become available and apply call treatment
Once agent found reserve it and attach application data to interaction
request RouteCall <Agent DN>
userdata={userId : "jamesbond", language:"French", longitude:"2.4544", ...}
REFER <agent URI>
Call ringing
Event RouteUsed
Customer Application
GMS
ORS (SCXML Session)
SIP Server
Meda Gateway
Agent Desktop
www.websequencediagrams.com

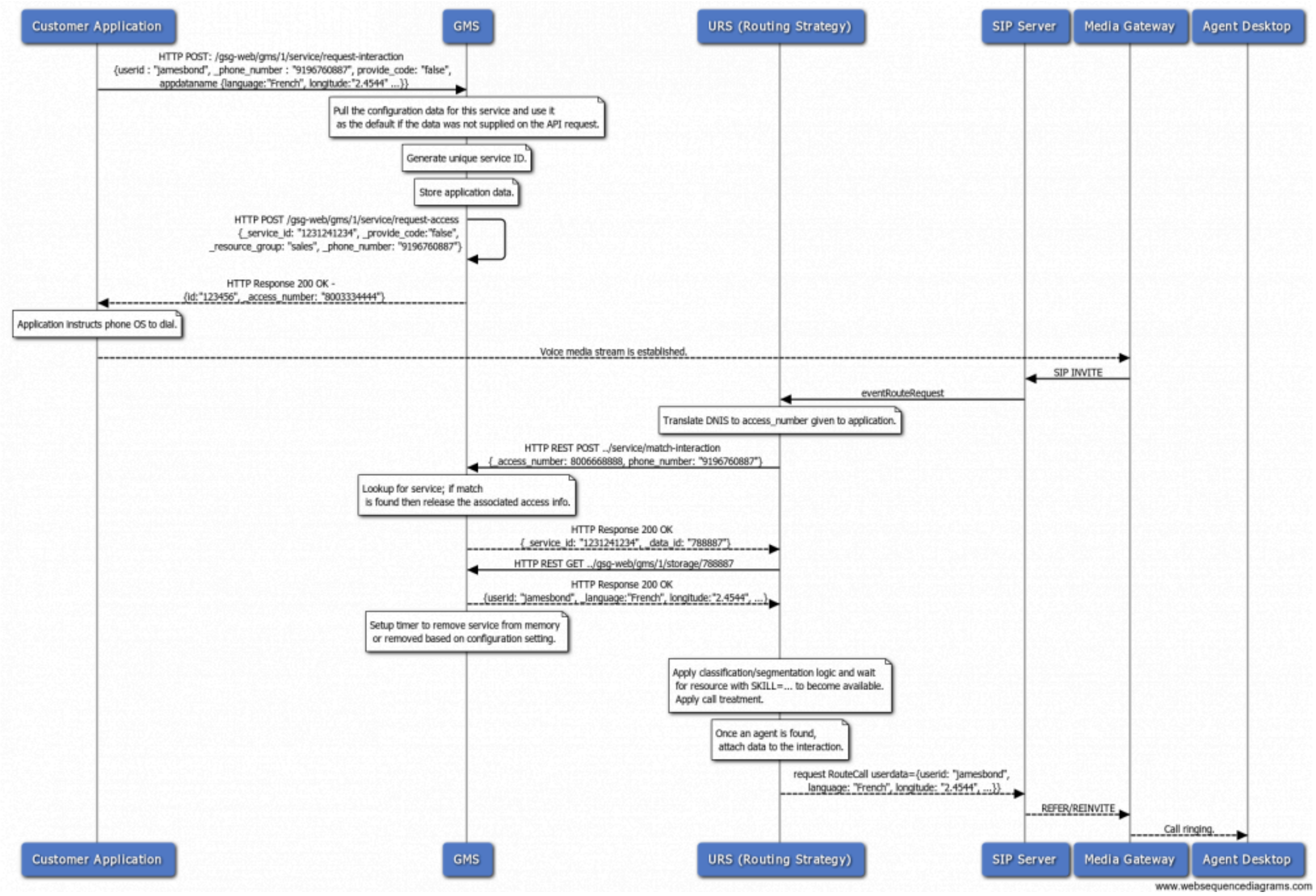
Access Number Allocation

No Locking

DNIS only - no match is done here

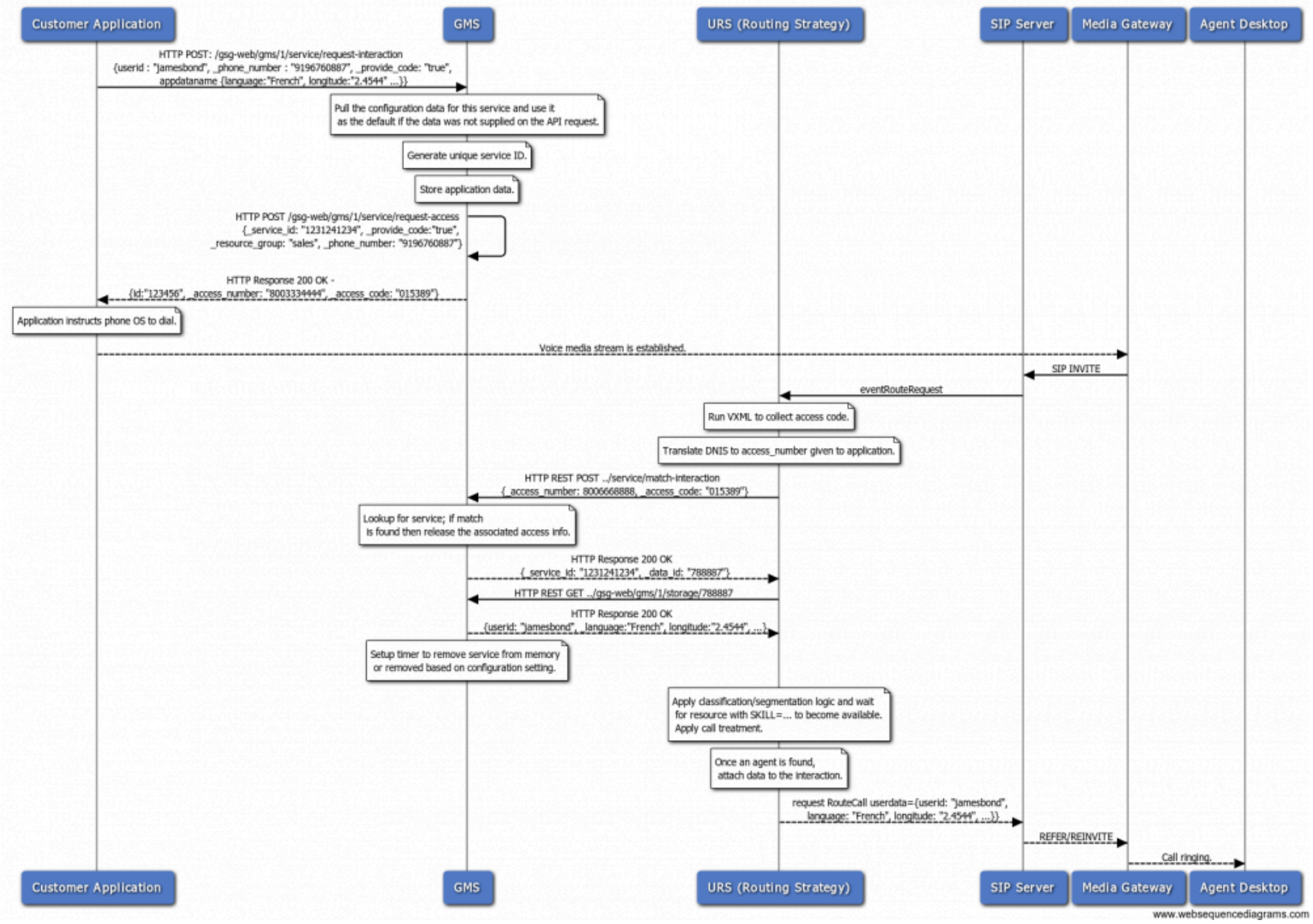

Customer Application
GMS
URS (Routing Strategy)
SIP Server
Media Gateway
Agent Desktop
HTTP POST: /gsg-web/gms/1/service/request-interaction
{userid : "jamesbond", _phone_number : "9196760887", provide_code: "false",
appdataname {language:"French", longitude:"2.4544" ...}}
Pull the configuration data for this service and use it
as the default if the data was not supplied on the API request.
Generate unique service ID.
Store application data.
HTTP POST /gsg-web/gms/1/service/request-access
{_service_id: "1231241234", _provide_code:"false",
_resource_group: "sales", _phone_number: "9196760887"}
HTTP Response 200 OK -
{id:"123456", _access_number: "8003334444"}
Application instructs phone OS to dial.
Voice media stream is established.
SIP INVITE
eventRouteRequest
Strategy makes a routing decision based on DNIS (_access_number),
e.g. send call to sales or support queue depending on DNIS
Apply classification/segmentation logic and wait
for resource with SKILL=... to become available.
Apply call treatment.
Once an agent is found, the interaction is sent to that agent. URS
can attach additional data based on the DNIS (such as prefered language).
request RouteCall
REFER/REINVITE
Call ringing.
Customer Application
GMS
URS (Routing Strategy)
SIP Server
Media Gateway
Agent Desktop
www.websequencediagrams.com

Match DNIS + ANI


Customer Application
GMS
URS (Routing Strategy)
SIP Server
Media Gateway
Agent Desktop
HTTP POST: /gsg-web/gms/1/service/request-interaction
{userid : "jamesbond", _phone_number : "9196760887", provide_code: "false",
appdataname {language:"French", longitude:"2.4544" ...}}
Pull the configuration data for this service and use it
as the default if the data was not supplied on the API request.
Generate unique service ID.
Store application data.
HTTP POST /gsg-web/gms/1/service/request-access
{_service_id: "1231241234", _provide_code:"false",
_resource_group: "sales", _phone_number: "9196760887"}
HTTP Response 200 OK -
{id:"123456", _access_number: "8003334444"}
Application instructs phone OS to dial.
Voice media stream is established.
SIP INVITE
eventRouteRequest
Translate DNIS to access_number given to application.
HTTP REST POST ../service/match-interaction
{_access_number: 8006668888, phone_number: "9196760887"}
Lookup for service; if match
is found then release the associated access info.
HTTP Response 200 OK
{_service_id: "1231241234", _data_id: "788887"}
HTTP REST GET ../gsg-web/gms/1/storage/788887
HTTP Response 200 OK
{userid: "jamesbond", _language:"French", longitude:"2.4544", ...}
Setup timer to remove service from memory
or removed based on configuration setting.
Apply classification/segmentation logic and wait
for resource with SKILL=... to become available.
Apply call treatment.
Once an agent is found,
attach data to the interaction.
request RouteCall userdata={userid: "jamesbond",
language: "French", longitude: "2.4544", ...}}
REFER/REINVITE
Call ringing.
Customer Application
GMS
URS (Routing Strategy)
SIP Server
Media Gateway
Agent Desktop

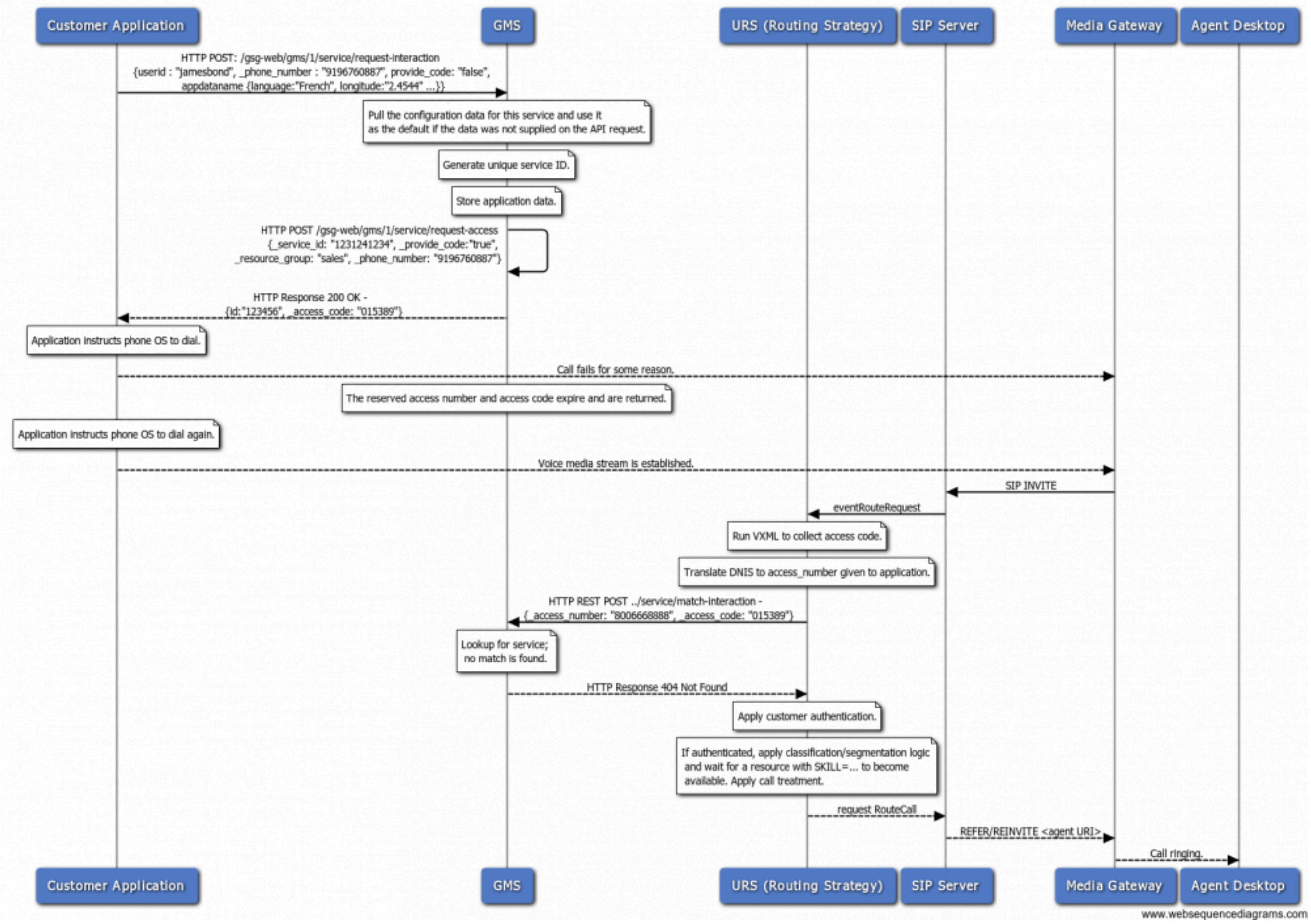
www.websequencediagrams.com

Match DNIS + Access Code


Customer Application
GMS
URS (Routing Strategy)
SIP Server
Media Gateway
Agent Desktop
HTTP POST: /gsg-web/gms/1/service/request-interaction
{userid : "jamesbond", _phone_number : "9196760887", _provide_code: "true",
appdataname {language:"French", longitude:"2.4544" ...}}
Pull the configuration data for this service and use it
as the default if the data was not supplied on the API request.
Generate unique service ID.
Store application data.
HTTP POST /gsg-web/gms/1/service/request-access
{_service_id: "1231241234", _provide_code:"true",
_resource_group: "sales", _phone_number: "9196760887"}
HTTP Response 200 OK -
{id:"123456", _access_number: "8003334444", _access_code: "015389"}
Application instructs phone OS to dial.
Voice media stream is established.
SIP INVITE
eventRouteRequest
Run VXML to collect access code.
Translate DNIS to access_number given to application.
HTTP REST POST ../service/match-interaction
{_access_number: 8006668888, _access_code: "015389"}
Lookup for service; if match
is found then release the associated access info.
HTTP Response 200 OK
{_service_id: "1231241234", _data_id: "788887"}
HTTP REST GET ../gsg-web/gms/1/storage/788887
HTTP Response 200 OK
{userid: "jamesbond", _language:"French", longitude:"2.4544", ...}
Setup timer to remove service from memory
or removed based on configuration setting.
Apply classification/segmentation logic and wait
for resource with SKILL=... to become available.
Apply call treatment.
Once an agent is found,
attach data to the interaction.
request RouteCall userdata={userid: "jamesbond",
language: "French", longitude: "2.4544", ...}}
REFER/REINVITE
Call ringing.
Customer Application
GMS
URS (Routing Strategy)
SIP Server
Media Gateway
Agent Desktop

www.websequencediagrams.com

No Match


Customer Application
GMS
URS (Routing Strategy)
SIP Server
Media Gateway
Agent Desktop
HTTP POST: /gsg-web/gms/1/service/request-interaction
{userid : "jamesbond", _phone_number : "9196760887", provide_code: "false",
appdataname {language:"French", longitude:"2.4544" ...}}
Pull the configuration data for this service and use it
as the default if the data was not supplied on the API request.
Generate unique service ID.
Store application data.
HTTP POST /gsg-web/gms/1/service/request-access
{_service_id: "1231241234", _provide_code:"true",
_resource_group: "sales", _phone_number: "9196760887"}
HTTP Response 200 OK -
{id:"123456", _access_code: "015389"}
Application instructs phone OS to dial.
Call fails for some reason.
The reserved access number and access code expire and are returned.
Application instructs phone OS to dial again.
Voice media stream is established.
SIP INVITE
eventRouteRequest
Run VXML to collect access code.
Translate DNIS to access_number given to application.
HTTP REST POST ../service/match-interaction -
{_access_number: "8006668888", _access_code: "015389"}
Lookup for service;
no match is found.
HTTP Response 404 Not Found
Apply customer authentication.
If authenticated, apply classification/segmentation logic
and wait for a resource with SKILL=... to become
available. Apply call treatment.
request RouteCall
REFER/REINVITE <agent URI>
Call ringing.
Customer Application
GMS
URS (Routing Strategy)
SIP Server
Media Gateway
Agent Desktop
www.websequencediagrams.com

Locking

The only difference between locking and non-locking call flows is the algorithm that determines how access numbers are handed out. In this case, a given access number is only assigned to a single service which allows for a more reliable match.

Match DNIS + ANI

Same as the equivalent no lock case.

Match DNIS + Access Code

Same as the equivalent no lock case.

No Match

Same as the equivalent no lock case.

Push Notification Service

Overview

This page contains useful information about Push Notification service. There are four different types of push notification supported in Genesys Mobile Services:

- HttpCallback Notification
- Android Notification
- Apple Notification
- Orchestration Server Callback Notification

In addition to discussing these different types of notification, this page also describes Details on Notification Propagation. For details about the configuration options available for various types of notification, see push Section.

HttpCallback Notification

This channel is used for pushing notification as POST requests to a provided URL. The notification server expects a response status of 200 (HTTP_OK). The body is ignored. If the response status is not 200 then the notification is considered to fail (see Details on Notification Propagation for more details).

Subscription Request

The URL to POST the message is specified by deviceId in the subscription request. When an event comes to the NotificationService and its tag matches the corresponding subscription, the POST request will be sent to the URL, specified by *notificationDetails.deviceId*.

Usage

The HTTP callback notification channel will send the HTTP request to specified URL as a reaction to notification publishing. The format of callback HTTP described above. The connection will be plain HTTP without TLS/SSL. The HTTP request will be done with POST method (hardcoded, not configurable), where body will be the plain string, passed as "message" in notification (see Notification API). Sample Request body:

```
{"subscriberId":"A1",
 "notificationDetails":{
    "deviceId":" http://localhost:8080/gms-web/gms/httpcb_notification/value/suffix",
    "type":"httpcb"},
 "filter":"*"}
```

Android Notification

C2DM Service

Android notification relies on the Android Cloud to Device Messaging (C2DM) service, described here: http://code.google.com/android/c2dm/. C2DM notifications are made on behalf of an account that is registered in Google services and described by the configuration options for the Genesys Mobile Services Application object. Some key points about C2DM to take into consideration when creating your applications:

- Each account has a limited capacity (quota). For more information about quotas, see: http://code.google.com/android/c2dm/quotas.html
- Message size limit is 1024 bytes.
- The push-to-android functionality requires an HTTPS connection to Google Services, so your environment must be configured to allows HTTPS connections to the following addresses to use this functionality:
  - https://www.google.com/accounts/ClientLogin
  - https://android.apis.google.com/c2dm/send

Keystore/Truststore Configuration Hints

The default Java keystore/trustore on Windows Server 2003 allows connections to required endpoints without any additional configuration. However, if you are using a different environment (OS, security policies, Servlet container, and JVM settings) there may be additional configuration steps to permit the necessary connections. This section contains the instructions for configuring your system when the default JVM keystore is replaced with the *-Djavax.net.ssl.keyStore* and *-Djavax.net.ssl.trustStore* JVM startup options on Windows systems. For other operating systems or keystore/truststore configurations, refer to the documentation for your environment. To configure the keystore:

1. Use your web browser or another tool to retrieve the certificates required for the following addresses:
   - https://www.google.com/accounts/ClientLogin
   - android.apis.google.com
2. Import those certificates into the keystore you plan to use.

> **Note:** If the keystore password is null or an empty string and the keystore contains a key, then Java may fail to establish the HTTPS connection. In this case user can:
>
> - update the keystore password to provide the correct value (recommended)
> - disable certificate validation by setting the *push.android.ssl_trust_all* option to *true* (highly unadvised)

Client Application Implementation

For an application to receive messages, it must meet the following requirements:

- When the application starts, it must register itself in the C2DM service by specifying the Google

Services account that it will receive notifications from. The account name must be configurable because it will be unique for each customer.

- The push service uses *data.message=<message_body>* in the service-to-C2DM POST request body. When the Android client application receives a notification, it should use "message" as the key to extract the passed information. Sample code for message extraction is provided below:

```
@Override
    public void onMessage(Context context, Intent intent) {
        Bundle extras = intent.getExtras();
        if (extras != null) {
            String payloadValue = (String) extras.get("message");
            //...
        }else {
         //...
        }
    }
```

GCM Service

Android gcm notification relies on the new Google Cloud Messaging (GCM) service, described here: http://developer.android.com/guide/google/gcm/. GCM notifications are made on behalf of an apiKey that is created in Google services (see http://developer.android.com/guide/google/gcm/gs.html) and described by the configuration options for the Genesys Mobile Services Application object. Some key points about GCM to take into consideration when creating your applications:

- No quota.
- Message size limit is 4096 bytes.
- The push-to-android functionality requires an HTTPS connection to Google Services, so your environment must be configured to allows HTTPS connections to the following addresses to use this functionality:
    - https://android.googleapis.com/gcm/send.

Keystore/Truststore Configuration Hints

The default Java keystore/trustore on Windows Server 2003 allows connections to required endpoints without any additional configuration. However, if you are using a different environment (OS, security policies, Servlet container, and JVM settings) there may be additional configuration steps to permit the necessary connections. This section contains the instructions for configuring your system when the default JVM keystore is replaced with the *-Djavax.net.ssl.keyStore* and *-Djavax.net.ssl.trustStore* JVM startup options on Windows systems. For other operating systems or keystore/truststore configurations, refer to the documentation for your environment. To configure the keystore:

1. Use your web browser or another tool to retrieve the certificates required for the following addresses:
    - https://android.googleapis.com/gcm/send.
2. Import those certificates into the keystore you plan to use.

Note: If the keystore password is null or an empty string and the keystore contains a key, then Java may fail to establish the HTTPS connection. In this case user can:

- update the keystore password to provide the correct value (recommended)
- disable certificate validation by setting the *push.android.ssl_trust_all* option to *true* (highly unadvised)

Client Application Implementation

For an application to receive messages, you can follow the recommandations from Google : http://developer.android.com/guide/google/gcm/gs.html#android-app Check the "**Writing the Android Application**" section for more information.

Apple Notification

As a provider, Genesys Mobile Services communicates with the Apple Push Notification service over an asynchronous binary interface. This interface is a high-speed, high-capacity interface for providers; it uses a streaming TCP socket design in conjunction with binary content. The binary interface of the production environment is available through gateway.push.apple.com, port 2195; the binary interface of the sandbox (development) environment is available through gateway.sandbox.push.apple.com, port 2195. You may establish multiple, parallel connections to the same gateway or to multiple gateway instances. See more details here: Provider Communication with Apple Push Notification Service

Client Application Implementation

Incoming notifications are the string representation of a JSON object. To receive the message itself, please extract the node with *key=message*.

CometD Notification

Note: Available in 8.1.100.28.

This channel is used for pushing notifications on the CometD channel. When using CometD to get notifications, the CometD connection should be set up with a subscription for /_genesys.

You also need to make sure that the 'gms_user' header in all CometD related requests is set to the value uniquely representing the application end user. Typically, this value would be set up (or at least verified) by the security gateway located between the client application and GMS.

CometD handshake request

```
POST http://localhost:8080/genesys/cometd
Accept-Encoding: gzip,deflate
Content-Type: application/json;charset=UTF-8
gms_user: BuzzBrain
{"version":"1.0","minimumVersion":"0.9","channel":"/meta/handshake","id":"0"}

HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:30:10 GMT
```

```
Content-Type: application/json
Content-Length: 230
 [{"id":"0","minimumVersion":"1.0","supportedConnectionTypes":["websocket","callback-
polling","long-polling"], "successful":true,"channel":"/meta/handshake","ext":
"ack":true},"clientId":"44xkkazwfabw73jrvjsvoy4ul","version":"1.0"}]
```

CometD /meta/connect subscription request

```
POST  http://localhost:8080/genesys/cometd
Accept-Encoding: gzip,deflate
Content-Type: application/json;charset=UTF-8
gms_user: BuzzBrain
{"channel":"/meta/
connect","clientId":"44xkkazwfabw73jrvjsvoy4ul","id":"1","connectionType":"long-polling"}
```

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:30:10 GMT
Content-Type: application/json
 Content-Length: 116
[{"id":"1","successful":true,"advice":{"interval":0,"reconnect":"retry","timeout":60000},"channel":"/meta/
connect"}]
```

CometD /_genesys subscription request

```
POST  http://localhost:8080/genesys/cometd Accept-Encoding: gzip,deflate
Content-Type: application/json;charset=UTF-8
gms_user: BuzzBrain
[{"channel":"/meta/
subscribe","subscription":"/_genesys","clientId":"44xkkazwfabw73jrvjsvoy4ul","id":"2"}]
```

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:30:10 GMT
Content-Type: application/json
Content-Length: 85
[{"id":"2","subscription":"/_genesys","successful":true,"channel":"/meta/subscribe"}]
```

CometD long polling request

```
POST  http://localhost:8080/genesys/cometd
Accept-Encoding: gzip,deflate
Content-Type: application/json;charset=UTF-8
gms_user: BuzzBrain
{"clientId":"44xkkazwfabw73jrvjsvoy4ul","id":"3","channel":"/meta/
connect","connectionType":"long-polling"}
```

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:30:10 GMT
Content-Type: application/json
Content-Length: 85
[{"id":"4","successful":true,"channel":"/meta/connect"}]
```

Localization of push messages

GMS support localized message. To allow this features device must supply a language at subscription time, corresponding to the application language. For example language can be:

Country	Language
English (United States)	en_US
English	en
Estonian	et
French	fr
...	...

Localization file format is described here.

```
{"subscriberId":"A1",
 "notificationDetails":{
    "deviceId":" http://localhost:8080/gms-web/gms/httpcb_notification/value/suffix",
    "type":"httpcb"},
 "language":"de",
 "filter":"*"}
```

See more details on configuring the push section.

Orchestration Server Callback Notification

Subscription

When subscribing to Orchestration Server callback, the user provides the Orchestration Server sessionId. This parameter is specified by *notificationDetails.deviceId*, with the type to be used specified as *orscb*.

Notification Propagation

The notification event contains 2 parameters: tag and message. The tag parameter is used for matching the subscription. If the subscription is for Orchestration Server callback, the following mappings have place:

- notificationDetails.deviceId - mapped to Orchestration Server sessionId
- notificationevent.tag - mapped to Orchestration Server eventName
- message - mapped to the message

Configuration

At the moment no specific configuration options exist for Orchestration Server callback - it relies on the corresponding OrsService.

Providers

You will need to add the certificate-related configuration options in the current push configuration section to a NEW type section that defines the credentials for the set of customer-specific notification providers. The provider can be specified as part of the notification subscription request.

For each notification provider, create a section with the following name format: push.provider.providername. For example, push.provider.SalesAppl. This will allow you to define a different push notification provider (connection) for each group of notification messages that are sent to applications.

You can define a provider for a group of events that are to be sent to a specific application or to be sent as part of a given service. This ensures that a given application does not get messages that they were not intended to receive. This provider definition can be associated with a given service’s CME definition or can be passed on the Create Service API for a given application.

If there is no provider defined for a subscription, then the default configuration options defined as part of the Push configuration section will be used.

The provider-related configuration options can be found here: Configuration Options There will also be a set of these credential configuration options for debugging purposes. So, there will be two provider connections for a provider. The application will be able to specify which provider (production or debug) connection.

Support of OS specific capabilities associated with the notification message

Each Push Notification System has a set of attributes that is sent to the application along with the base notification message. These attributes are usually related to the message definition itself and not to a given instance of the message being sent. So these additional OS attributes will be configured as part of the provider configuration definition. For each event you will create a section with the following name format – push.provider.**providername**.event.**eventname**. For example, push.provider.SalesAppl.event.mobile.statuschanged. This is done so that the Notification APIs do not have to have these OS specific attributes provided on the API calls. This can defined for each notification message associated with each provider or defined at the general provider level for each event. In addition, you can provide these OS specific attributes for various event groups. For example, you can do it at the individual event level (mobile.statuschanged) or at an event sub-grouping (mobile.). These attributes are all independent of the level they are defined at so you could end up picking up values for the different attributes from different levels in the hierarchy. This is in the order in which they will be selected. (first to last):

- Use the event definition values associated with a specific provider definition
- Use the event definition values associated with a general provider definition
- Use the OS specific attribute values associated with push section

In addition, the event definition can contain multiple different OS specific attributes so you can have iOS and Android attributes defined under the same event definition. So the notification framework high level logic for processing published events would be:

- Find the subscriptions that have registered to receive this event
- Get the subscriptions associated provider’s event configuration options for this event
- If available use them, otherwise, check the general event configuration options under the provider configuration section. If available use them otherwise get the general configuration options under the Push configuration section. If available use them otherwise this event message does not have an OS specific attributes to apply.
- Form the PNS specific message with the input from the Publish API and the event configuration options if available
- Send the message over the appropriate provider connection to the PNS.

Consider the example to illustrate the rules. Let's say that we have the subscription associated with provider **SalesApp** and with filter **A2C.*** (match all events starting with A2C). Consider that we have the following set of sections with OS-specific message formatting options:

- (0) push
- (1) push.provider.event
- (2) push.provider.event.internal
- (3) push.provider.event.internal.advanced
- (4) push.provider.event.A2C
- (5) push.provider.event.A2C.service
- (6) push.provider.event.A2C.service.statuschanged
- (7) push.provider.event.A2C.service.internal
- (8) push.provider.event.A2C.service.statuschanged.agentavailable
- (9) push.provider.SalesApp.event
- (10) push.provider.SalesApp.event.A2C.service.internal
- (11) push.provider.SalesApp.event.A2C.service.statuschanged

Consider that we have the incoming event with tag A2C.service.statuschanged.agentavailable. This event's tag will match the filter of our subscription associated with provider **SalesApp** and with filter **A2C.***. So, we will go through the chain of sections in the following order (from most default to most concrete): **0->1->4->5->6->8->9->11** We'll traverse this chain replacing and overwriting the options from more default sections with the corresponding options from more concrete sections (this is equivalent to seeking for all options in more concrete sections first, and accessing more default only if not found in more concrete). The result set of options will be used for OS-specific message formatting.

Localization File

Overview

The localization file allows you to customize the way you send a message to subscribers. You can define several messages based on the language of the customer.

Localization file

Format

```
<?xml version="1.0" encoding="UTF-8" ?>
<messages>
  <message id="welcome">
    <locale language="en_US">
      <entry key="text">Welcome</entry>
    </locale>
    <locale language="de">
      <entry key="text">Willkommen</entry>
    </locale>
    <locale language="fr">
      <entry key="text">Bonjour</entry>
    </locale>
    <locale language="es">
      <entry key="text">\u00A1Hola</entry>
    </locale>
<locale language="ja">
      <entry key="text">\u3053\u3093\u306B\u3061\u306F</entry>
    </locale>
  </message>
   <message id="welcomeArgs">
    <locale language="en_US">
      <entry key="text">Dear customer $customer.lastname, how can you be reminded</entry>
    </locale>
    <locale language="de">
      <entry key="text">Sehr geehrter Kunde $customer.lastname, wie können Sie daran erinnert
werden</entry>
    </locale>
    <locale language="fr">
      <entry key="text">Cher client $customer.lastname, comment pouvez-vous être
rappelé</entry>
    </locale>
    <locale language="es">
      <entry key="text">$customer.lastname Estimado cliente, \u00BFc\u00F3mo puede ser
recordado</entry>
    </locale>
  </message>
</messages>
```

Arguments

Arguments can be added to the message, GMS will replace the arguments in the message with correct value provided when you publish the message.

<table>
<tr><th>Simple style</th><th>Expanded style</th></tr>
<tr><td>

```
{
   "message":"welcome",
   "tag":"yourtag",
   "mediaType":"localizestring",
   "locArgs":{
       "customer.lastname":"Doe"
   }
}
```

</td><td>

```
{
   "message":"welcome",
   "tag":"yourtag",
   "mediaType":"localizestring",
   "locArgs":{
       "customer":{
            "lastname":"Doe"
       }
   }
}
```

</td></tr>
</table>

For example for language option (provided at subscription time) equals to "de" (German), the customer will receive the following message: `Sehr geehrter Kunde Doe, wie können Sie daran erinnert werden`

Genesys Mobile Services Configuration

Please refer to the push section documentation on Genesys Mobile Services Configuration Options.