



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 8.5.1

12/29/2021

# Table of Contents

<b>Genesys Mobile Services API Reference</b>	<b>3</b>
New in This Document	6
Storage API	8
Storage API HTML Sample	15
Node API	19
Notification API	21
Chat APIs	28
Chat API Version 1	29
Chat API Version 1 Push Notification and Background State	70
Chat API Version 2	75
Pattern Matcher API	76
Service API	81
Calendar Service API	96
Capacity API	99
Callback Services API	101
Digital Channels API	172
Chat API Version 2	173
Chat API Version 2 with CometD	191
Email API Version 2	218
Open Media API	221
API Responses	229
Phone Number Validation API	237
Stat Service API	243
Push Notification Service	272
Callback Push Notifications for Android	282
Callback Push Notifications for iOS	288
Chat API Version 1 Push Notification and Background State	70
Localization File	299

# Genesys Mobile Services API Reference

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

## Important

This API is only intended to be used with Orchestration Server-based Services, not from external mobile applications.

- **Chat APIs** — This API is used by customer-facing applications to create and manage a chat session associated with a contact center-related service. A single service is associated with a single chat.
- **Email API** — This API is used by customer-facing applications to create and manage an email message associated with a contact center-related service. A single service is associated with a single email.
- **Service API** — This API is used by customer-facing applications to manage different types of contact center-related services.
- **Callback Services API** — This API handles call back services, such as initiating, canceling, rescheduling, and queries. Other Callback-related APIs are the following:
  - **Calendar Service API** — The Calendar Service API queries for all office hours so that your mobile app can make an intelligent first offer to the user based on what's available in the next couple days, or with an explanatory hint such as "Our offices are open from 8:00 am to 5:00 pm, please enter the desired time."
  - **Capacity API** — The Capacity Service enables you to define the number of scheduled callbacks that are allowed for Callback for a given time slot in the week. Then, your Callback service refers to your Capacity service and to your Office Hours service to adjust the agent availability and the number of scheduled callbacks.
- **Digital Channels API** — These APIs allow Genesys Mobile Engagement to work with Genesys digital channels such as chat, email, and open media.
  - **Chat API Version 2**
  - **Chat API Version 2 for CometD**

- [Email API](#)
- [Open Media API](#)
- [Stat Service API](#) — This API is used to interact with the Genesys Statistics Server (Stat Server). The API provides the request so an application can get statistics related to the Contact Center.
- [Pattern Matcher API](#) — This API allows you to create and manage pattern lists that you can use to check parameter values and define exceptions in your GMS service.
- [Phone Number Validation API](#) — This API wraps Google's common Java, C++, and JavaScript library for the purposes of parsing, formatting, and validating international phone numbers.

### Additional Information

- [Push Notification Service](#) — Contains useful information about the Push Notification service.
- [Localization File](#) — The localization file enables you to customize the way you send a message to subscribers.
- [New in This Document](#) — Provides a document change history.

## Authentication Header

The following services require Authentication Header.

Service	Requires Authentication Header?
Statistic Service	Yes.
Node Service	Only for queries to /genesys/admin/1/node/status.
Callback Service	Depends on the configuration.
Storage Service	Depends on the configuration.
Notification Service	Depends on the configuration.

## Learn about Templates, Scenarios, and APIs

Available templates, scenarios, and APIs

Template Name	Detailed Built-in	Related API(s)
Get Service (get.zip)	<a href="#">Get and Basic Get Services</a>	<a href="#">Node API</a> to check GMS nodes health and manage your nodes: start, suspend, stop.
Match Interaction (match-interaction.zip)	<a href="#">Match interaction</a>	<ul style="list-style-type: none"><li>• <a href="#">Service API</a> to check that a voice call with an existing GMS service is associated with the access number.</li></ul>

Template Name	Detailed Built-in	Related API(s)
		<ul style="list-style-type: none"> <li><b>Storage API</b> to allow users to temporarily store arbitrary data. Data may consist of key/value pairs of strings or binary objects.</li> </ul>
Office Hours	Office-hours	<b>Calendar Service API</b> to create and manage office hours, special events, and more.
Request Access	Request-access	<b>Service API</b> to request resources.
Request Chat		Create a chat session in the Chat Server using the <b>Chat API v1</b> .
Request Interaction	Simple Voice Inbound-Immediate Call	See the <b>scenario</b> page.
URS Statistic (urs-stat.zip)		<b>Stat Service API</b> to query URS Stat.
Callback (callback.zip)	User Originated Immediate	Query to create an inbound immediate service.
	User Originated Delayed	Query to create an inbound delay service.
	Chat Immediate	Chat APIs
	Chat Delayed	Chat APIs
	User Terminated Immediate	Callback Services API
	User Terminated Delayed	Callback Services API
	User Terminated Scheduled	Callback Services API.
	User Terminated Delayed Agent Preview	Callback Services API
Capacity (capacity.zip)	Capacity	<b>Capacity API</b> to manage Agent availability.

# New in This Document

## Important

- [Scenarios](#) have been moved to the [Service Management Help](#).
- [Digital Channel APIs Configuration](#) has been moved to the [Deployment Guide](#).

### The following topics have been added or changed in the **GMS 8.5.114** release:

- [Chat V2 API over CometD](#) now supports push notifications through Firebase Cloud Messaging and Apple Push Notification Service.

### The following topics have been added or changed in the **GMS 8.5.112** release:

- The [Chat Session File Management](#) feature was introduced.
- New [EWT APIs](#) were introduced.
- The `enable_notification_hybrid_mode` option was deprecated.

### The following topics have been added or changed in the **GMS 8.5.111** release:

- The `enable_notification_hybrid_mode` and `max_message_size` options were added to the [Chat Service options](#) used to configure the Digital Channels API.

### The following topics have been added or changed in the **GMS 8.5.110** release:

- The new page [Open Media API](#) has been added.
- The new page [Open Media Service Options](#) has been added.
- The section [Export Cancelled Callback Records](#) was added to the Callback Services API.
- The mailbox option was added to the [Email Service options](#) used to configure the Digital Channels API.

### The following topics have been added or changed in the **GMS 8.5.109** release:

- The new page [Chat API Version 2 with CometD](#) has been added.
- The section [Query-EWT for Virtual Queues](#) was added to the Callback Services API.
- The `enable_notification_mode` option was added to the [Chat Service options](#) used to configure the Digital Channels API.

### The following topics have been added or changed in the **GMS 8.5.108** release:

- The [Phone Number Validation API](#) page was added.

- The [Chat Service Options](#) page was updated.
- The [Email Service Options](#) page was updated.
- The [Change Node Status](#) section was added to the Node API page.

**The following topics have been added or changed in the GMS 8.5.107 release:**

- The [Callback Services API](#) page was updated.

**The following topics have been added or changed in the GMS 8.5.106 release:**

- The [Callback Services API](#) page was updated.

**The following topics have been added or changed in the GMS 8.5.104 release:**

- The [Chat API Version2](#) and the [Email API](#) are now grouped under the [Digital Channels API](#).
- The [Digital Channels API](#) also includes information about:
  - [API Responses](#)
  - [Configuring the Digital Channels API](#)

**The following topics have been added or changed in the GMS 8.5.102 release:**

- The [Chat API](#) was deleted; you should now refer to [Chat API Version1](#) (former version available) and [Chat API Version2](#) which will replace gradually the first one.
- The [Email API](#) page was added.

**The following topics have been added or changed in the GMS 8.5.101.05 release:**

- The [Stat Service Extended API](#) now supports filters.

**The following topics have been added or changed in the GMS 8.5.100.05 release:**

- The [Chat API](#) page changed, to add new parameters related to chat push notifications and update query samples.
- The [Chat Push Notification and Background State](#) page was added, to reflect push notification enhancements for managing applications' background state during chat sessions.

# Storage API

## Overview

The Storage API is a 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 the creation of a new storage area in Genesys Mobile Services (GMS).

Starting 8.5.103.xx, this operation requires the `gms_user` header to be passed to associate the `custom_id` to the customer. To retrieve the stored value, you also need to pass the `gms_user` header.

### Operation

Method	POST		
URL	/genesys/1/storage/custom/{customId}/{ttl}		
Parameter	Type	Mandatory	Description
URI Parameters			
{customId}	string	yes	Custom ID of the storage. Available starting 8.5.103.xx
{ttl}	number	yes	The time to live for this data, specified in seconds. The data is automatically deleted after it has been stored for {ttl} seconds. The ttl must be greater than zero (0). If an incorrect value is specified, a default of 30 seconds is defined.
<b>Body:</b> 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.



<b>HTTP code</b>	200
<b>HTTP message</b>	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 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 data, specified in seconds. The data is automatically deleted after it has been stored for {ttl} seconds. The ttl must be greater than zero (0). If an incorrect value is specified, a default of 30 seconds is defined.
<b>Body:</b> A MultiPart form consisting of different items representing the key/value pairs to store. This may be string 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.
<b>Body:</b> 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 one 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 specifically requested value
<b>Body:</b> 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 one 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 specifically requested value
<b>Body:</b> 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.
<b>Body:</b> 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

## Samples

[Storage API HTML Sample](#)

## Notes

Keys may not begin with an underscore (\_).

## 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>
```

# Node API

This is a base ping API implementation, which load balancers will use to check the health of a given GMS node to determine if it can use this particular GMS node when it load balances API requests across the set of GMS nodes.

## Query Status

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

### Important

You need an Authentication header for this query.

GET /genesys/1/admin/node/status			
Parameter	Type	Mandatory	Description
URI Parameters			
None			

## Response

HTTP code	200
HTTP message	OK
Body	The response is a string representing the status of GMS ONLINE/OFFLINE (OFFLINE means that GMS will stop in the next few seconds).

## Example

```
$ curl -v -u default:password
http://localhost:8080/genesys/1/admin/node/status
< HTTP/1.1 200 OK
< Expires: Thu, 11 Aug 2016 10:59:27 GMT
< Set-Cookie: JSESSIONID=inglubvlfu69bzw6lvksz6jo2;
Path=/genesys;HttpOnly
< Cache-Control: max-age=300
< Content-Type: text/plain; charset=ISO-8859-1
< Content-Length: 6
```

<  
ONLINE

## Change Node Status

This API allows you to change the status of a GMS Node to offline or online. If a load balancer queries the node, it will get the new GMS status and will load balance the request accordingly.

If you set the GMS Node status to OFFLINE, the node's status will be Suspended in Solution Control Server.

### Query

#### Important

You need an Authentication header for this query.

POST /genesys/1/admin/node/changestatus/{status}			
Parameter	Type	Mandatory	Description
URI Parameters			
Status	String	yes	ONLINE/OFFLINE

### Response

HTTP code	200
HTTP message	OK

### Example

```
POST /genesys/1/admin/node/changestatus/OFFLINE HTTP/1.1
Host: 127.0.0.1:8080
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded

200 OK
```

# Notification API

## Overview

### Important

Do not use the Publish part of this API from a mobile device. The API 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 GMS delayed use case, it can work as follows:

1. The mobile application triggers a subscription for an ORS event; something like ors.contact.12345678; the application specifies the device id and the type (for example, iOS).
2. When ORS determines that an agent is available, or will soon be available, it will push a message to GMS with the event ors.contact.12345678.
3. GMS pushes 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. Its absence or incorrect value can result in a 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}"
  }
```

```

    "authorization": "ZGVtbzo=",
    "expire":30,
    "filter":"${filter}"
}

```

Where:

- **subscriberId** - The id of subscriber (mandatory).
- **authorization**- (Optional) If basic authorization is needed on the custom HTTP channel. The value of the **authorization** parameter will be added to the HTTP Headers request sent to the custom http channel.
- **expire** - This parameter defines the time, in seconds, after which the subscription expires (optional; 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. See [Genesys Mobile Services PushNotificationService](#) 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), **wns** (to-wns-client) (see more information [here](#)).
  - **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 that can be needed for notification delivering. If the information provided is not enough for corresponding publisher, an 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**.

Note: The notificationDetails.properties are not needed for **android**, **gcm** or **ios** or **httpcb** or **orscb** notifications - 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 (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": "${mediaType}",
  "notificationDetails":
  {
    "deviceId": "${deviceId}",
    "type": "${type}",
    "properties":
    {
      "debug": "{true/false}"
    }
  }
}
```

Where:

- 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 - a BAD\_REQUEST response will be returned.
- mediaType - (optional) "**string**" for a simple string, "**localizestring**" for a string with localized parameter. See [Localization File](#).
- providerName - (optional) This is the name of the provider that this subscription is for. If not specified, the subscription is for the default provider.
- notificationDetails - (optional) If not present, notification is sent to default subscribers. It allows sending the notification to a specific device.
- deviceId - (mandatory) The id of the device (for example, Android device id or iPad id).
- type - (mandatory) Type of the notification (gcm, ios...).
- properties - (optional).
- debug - (optional) Allows the display of the debug log for this notification.

## Filters and Tags

The tag cannot be null or an 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 wildcard '\*' 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.agentavailability.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 is not mentioned in further descriptions (except some cases when syntax of body is involved). **Notes:** this API is intended for internal usage. All POST requests must specify media type **application/json**.

### Create Subscription

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

#### Operation

**POST** /genesys/{api version}/notification/subscription

**Body:** JSON with subscription (see above)



## Response

### Success

<b>HTTP code</b>	200
<b>HTTP message</b>	OK
<b>Body</b>	A JSON object with the property id, identifying the assigned id for this storage request.

### Errors

- In the case of incorrect request syntax (see requirements above) the BAD\_REQUEST error will be returned.

<b>HTTP code</b>	400
<b>HTTP message</b>	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.

<b>HTTP code</b>	404
<b>HTTP message</b>	NOT FOUND

## Delete Subscription

This call cancels/terminates a given subscription.

### Operation

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

## Response

### Success

<b>HTTP code</b>	200
<b>HTTP message</b>	OK

**Error** If a problem occurs during subscription removal, the following status code is returned:

<b>HTTP code</b>	404
<b>HTTP message</b>	Not Found
<b>Body</b>	<pre>{"message": "Subscription ID not found", "exception": "com.genesyslab.gsg .services.notification.SubscriptionNotFoundException"}</pre>

## Delete subscription for given subscriber

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

### Operation

<b>DELETE /genesys/{api version}/notification/subscription/subscriber/{subscriberId}</b>			
<b>URI Parameters</b>			
<b>Parameter</b>	<b>Type</b>	<b>Mandatory</b>	<b>Description</b>
<b>{subscriberId}</b>	String	yes	the id of the subscriber whose subscriptions will be cancelled

### Returns

#### Success

<b>HTTP code</b>	200
<b>HTTP message</b>	OK

#### Error

If a problem occurs during removing subscriptions of the subscriberId, the following status code is returned:

<b>HTTP code</b>	404
<b>HTTP message</b>	Not Found
<b>Body</b>	<pre>{"message": "Subscriber ID not found", "exception": "com.genesyslab.gsg.services.notification .SubscriberNotFoundException"}</pre>

## Publish Event

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

### Operation

<b>POST /genesys/{api version}/notification/publish</b>
<b>Body:</b> JSON event (see the example below.)

## Example

The following example sends a message to iOS, with a different `alertMessage.body` parameter:

```
POST /genesys/1/notification/publish HTTP/1.1
Host: 172.25.157.93:8080
gms_user: 4f295ba0c0f0a7f1e5ef068bf1d0732e0e70fda7c443081bb3cc5698fa9a276c
Content-Type: application/json
Cache-Control: no-cache

{
  "message": "Agent availability",
  "tag": "gms.notification.agentstatus",
  "mediaType": "STRING",
  "notificationDetails": {
    "deviceId": "XXXXXXXXXXXXXXXXXXXX",
    "type": "ios",
    "properties": {
      "apple.alertMessage.body": "Agent is available.",
      "apple.badge": 9,
      "apple.sound": "bingbong.aiff",
    }
  }
}
```

## Response

### Success

<b>HTTP code</b>	200
<b>HTTP message</b>	OK

### Errors

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

<b>HTTP code</b>	400
<b>HTTP message</b>	BAD REQUEST

If the error occurs during notification publishing (http post to specified url failed or did not return 200), or if the error occurs during network issues, or APNS or C2DM services report an error (authorization issues, temporary service unavailability for C2DM, and so on) the `SERVICE_UNAVAILABLE` error will be returned.

<b>HTTP code</b>	503
<b>HTTP message</b>	SERVICE UNAVAILABLE

# Chat APIs

Starting in 8.5.114, Chat API version 2 is available for both Web and Mobile App development.

- **Chat API version 2:** Use this API for Chat initiated by Web and Mobile App clients. Genesys recommends that you use this API for new deployments. New features will be added to Chat API vVersion 2 only.
- **Chat API version 1:** This API is documented to support older deployments. Genesys recommends that you upgrade to Chat API version 2 if you are still using Chat API version 1.

# Chat API Version 1

## Important

This API is documented to support older deployments. Genesys recommends that you upgrade to **Chat API version 2** if you are still using Chat API version 1.

## Prerequisites

Before using the Chat API described on this page, you must **configure your Genesys Mobile Services deployment** correctly. Then, before you can create Chat sessions, you must **create** your chat service (of type **ors** or **builtin**) with the Service Management UI. Then, you will be able to use this chat service to request chat session IDs.

## Getting Started

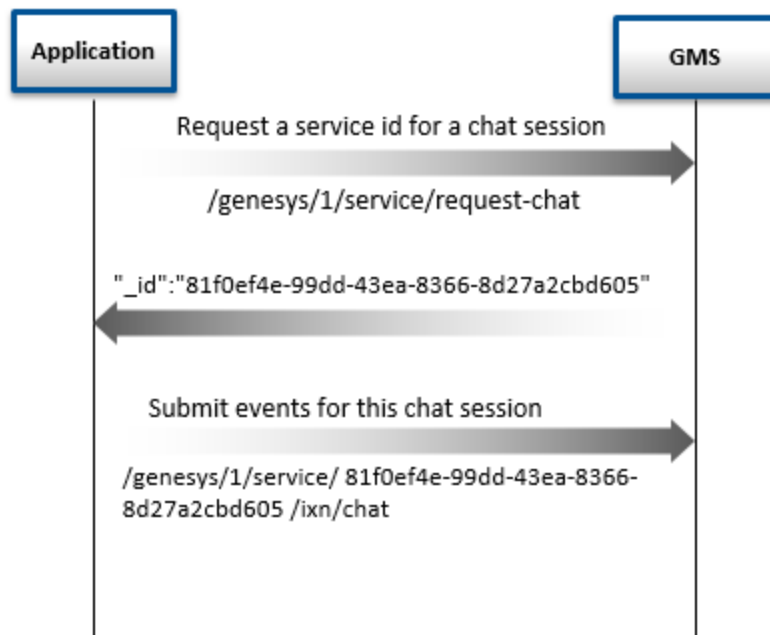
The Chat API is used by customer-facing applications to create and manage a chat session associated with contact center-related services.

### Basic Chat

To start chat session:

- **Create** the built-in service of type request - chat that provides the Basic Chat Service API.
- Use the Basic Chat Service API to create a Chat Service ID for each chat session. The built-in request - chat service created in the Service Management UI provides the Basic Chat Service API.

Then, you can use the Chat Service ID to perform queries for your chat session.



Chat API Version 1 (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 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.
- The routing logic associated with the specified interaction endpoint (or the configured default value) would be responsible for finding an appropriate agent.
- Both polling and async (CometD) modes of message delivery are supported.
- Applications can handle background state through [chat push notifications](#)

### Tip

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

## ORS-Chat Services

Instead of creating a basic chat service to request chat session IDs, you can configure one of the following chat service of type `ors` in the Service Management UI:

- [Chat Immediate](#)
- [Chat Delayed](#)

Then, use this `ors` service to retrieve your new service ID associated with your chat session. For instance if you configured the `chat-immediate` service, you will post to the `chat-immediate` service:

```
POST http://localhost:8080/genesys/1/service/callback/chat-immediate
```

You can now use this ID in the [Chat Interaction APIs](#).

## Structures

The Chat API uses the data structures described in this section (in JSON format) to exchange data. Requests are accepted in **`application/json`**, **`'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 your application receives a 415 (Unsupported Media Type) error.

## JSON-Encoding

For each method detailed below, if you wish to use JSON in your body, you must use JSON-encoding for body capabilities by setting the Content-Type to `application/json; charset=UTF-8`. For example, if you use the query for the chat session creation:

### Request

```
POST http://localhost:8080/genesys/1/service/fb80ed7b-a164-46ef-a5b8-f14f99236042/ixn/chat
HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/json; charset=UTF-8

{
  "verbose": "true",
  "first_name": "John",
  "last_name": "Doe",
  "subject": "GMSDemo",
  "userDisplayName": "JohnDoe"
}
```

### 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

{
  "chatIxnState" : "CONNECTED",
  "transcriptPosition" : "1",
}
```

```
{
  "chatSessionId" : "000F3aBC3NXB001F",
  "userId" : "017356D8446D002A",
  "secureKey" : "ffe843f41bfleec2",
  "checkChatServiceLoadBalancerPath" : "null",
  "chatServerLoadBalancerAlias" : "371",
  "chatServerHost" : "bsdemo2012jac",
  "chatWebApiPort" : "9002",
  "isTLSrequired" : "false",
  "clientTimeZoneOffset" : "60",
  "_chatIxnAPI_SEND_URL" : "/genesys/1/service/fb80ed7b-a164-46ef-a5b8-f14f99236042/ixn/chat/send",
  "_chatIxnAPI_REFRESH_URL" : "/genesys/1/service/fb80ed7b-a164-46ef-a5b8-f14f99236042/ixn/chat/refresh",
  "_chatIxnAPI_START_TYPING_URL" : "/genesys/1/service/fb80ed7b-a164-46ef-a5b8-f14f99236042/ixn/chat/startTyping",
  "_chatIxnAPI_STOP_TYPING_URL" : "/genesys/1/service/fb80ed7b-a164-46ef-a5b8-f14f99236042/ixn/chat/stopTyping",
  "_chatIxnAPI_DISCONNECT_URL" : "/genesys/1/service/fb80ed7b-a164-46ef-a5b8-f14f99236042/ixn/chat/disconnect",
  "_chatIxnAPI_REFRESH_FROM_START_URL" : "/genesys/1/service/fb80ed7b-a164-46ef-a5b8-f14f99236042/ixn/chat/refresh?transcriptPosition=1",
  "_chatIxnAPI_SEND_CUSTOM_URL" : "/genesys/1/service/fb80ed7b-a164-46ef-a5b8-f14f99236042/ixn/chat/customNotice"
}
```

### Warning

The current implementation supports simple key/value parameters. You cannot manage complex JSON-structured values such as maps and arrays. See the above code example above.

## 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).

### Important

You need to create the service session with the basic chat service before you can create a chat interaction. Since 8.5.1, you can specify push notification parameters to manage your application's background state (see sections below).

The Chat Interaction Attributes are the following:

- `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.
- \_chatIxnAPI\_SEND\_CUSTOM\_URL – URL used to send a custom notice to the chat session.

### Example of Chat Interaction Resources

```
/genesys/1/service/{sessionid}/ixn/chat
?notify_by=comet&firstName=Buzz&lastName=Brain
cject=French&email=b.b%40gmail.com
&push_notification_deviceid=deviceid
&push_notification_type=ios

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

```
/genesys/1/service/{sessionid}/ixn/chat?_verbose=true
-ify_by=comet&firstName=Buzz&lastName=Braincject=French
&email=b.b%40gmail.com&push_notification_deviceid=deviceid
&push_notification_type=ios
```

```
{
  "chatIxnState": "CONNECTED",
  "chatSessionId": "000C2a7VVQRB001U",
  "transcriptPosition": "1",
  "chatServiceMessage": "Chat service is available",
  "userId": "015E4FD3CD890036",
  "secureKey": "b306749dabfalcf6",
  "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"
}
```

## Chat Transcript Resource

The Chat Transcript Attributes are the following:

- **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}, {message index}] where:
  - Event type: {"Message.Text", "Notice.Joined", "Notice.Left", "Notice.TypingStart", "Notice.TypingStop", "Notice.PushUrl"}
  - Type of user: {"AGENT", "CLIENT", "EXTERNAL"}

Starting in 8.5.105.xx, the message index field has been added to the chat message to indicate the position of each message in the transcript. Your chat client application can now manage the comet and the API response channel and merge the messages according to their index in the transcript. This will avoid duplicate messages on screen.

This example shows how to send messages with their transcript position:

```
$ curl --data "transcriptPosition=1&message=good"
http://localhost:8080/genesys/1/service
/9f704400-7967-4586-821e-2e2ad5c1585f/ixn/chat/send
{
  "chatIxnState" : "TRANSCRIPT",
  "transcriptPosition" : "3",
  "chatSessionId" : "000F7aBK86UC001F",
  "transcriptToShow" :
  [ ["Notice.Joined", "Kristi Sippola", "has joined the session",
    "21", "AGENT", "2"],
    ["Message.Text", "127.0.0.1", "good", "35", "CLIENT", "3"] ],
  "startedAt" : "2016-06-01T14:51:37Z"
}

$ curl --data "transcriptPosition=4&message=right"
http://localhost:8080/genesys/1/service
/9f704400-7967-4586-821e-2e2ad5c1585f/ixn/chat/send
{
  "chatIxnState" : "TRANSCRIPT",
  "transcriptPosition" : "15",
  "chatSessionId" : "000F7aBK86UC001F",
  "transcriptToShow" :
  [ ["Notice.TypingStarted", "Kristi Sippola",
    "user is typing", "55", "AGENT", "5"],
    ["Message.Text", "Kristi Sippola", "position 2", "57", "AGENT", "6"],
    ["Notice.TypingStarted", "Kristi Sippola",
    "user is typing", "57", "AGENT", "7"],
    ["Message.Text", "Kristi Sippola", "hello", "57", "AGENT", "8"],
    ["Notice.TypingStarted", "Kristi Sippola",
    "user is typing", "58", "AGENT", "9"],
    ["Message.Text", "Kristi Sippola", "allo", "58", "AGENT", "10"],
    ["Notice.TypingStarted", "Kristi Sippola",
    "user is typing", "58", "AGENT", "11"],
    ["Message.Text", "Kristi Sippola", "bonjour", "58", "AGENT", "12"],
    ["Notice.TypingStarted", "Kristi Sippola",
    "user is typing", "58", "AGENT", "13"],
    ["Message.Text", "Kristi Sippola", "Hallo", "59", "AGENT", "14"],
    ["Message.Text", "127.0.0.1", "right", "65", "CLIENT", "15"] ],
  "startedAt" : "2016-06-01T14:51:37Z"
}
```

## Important

The index data set of the transcript allows missing numbers due to configurable filtering events system. For example, if you have index series like "1, 3, 4, 5, 7", the unavailable "2, 6" index numbers can be filtered events).

### Refresh Chat Transcript Examples

`/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"
}
```

`/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"
}

```

## Basic Chat Service 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"
}

```

## 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

<b>Method</b>	POST		
<b>URL</b>	/genesys/1/service/request-chat		
<b>Parameter</b>	<b>Type</b>	<b>Mandatory</b>	<b>Description</b>
<b>URI Parameters</b>			
'request-chat'	String	yes	Name of the preconfigured basic chat service
<b>Body:</b> The body will be x-www-form-urlencoded form consisting of different items representing the key/value pairs associated with the request.			
<b>Body Properties:</b> The following are the properties:			
<ul style="list-style-type: none"><li>• <code>_verbose</code> - This will allow the application to get all the detail attributes associated with the chat session in the corresponding response.</li><li>• ... - Any other business data attributes can also be passed.</li></ul>			

## Response

<b>HTTP code</b>	200
<b>HTTP message</b>	OK
<b>Body</b>	A chat JSON object for details on the properties of the object. See the section on <a href="#">data structures</a> for more details.
<b>Notes</b>	None
<b>HTTP code</b>	503
<b>HTTP message</b>	Service Unavailable
<b>Body</b>	None
<b>Notes</b>	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
{
  "_chatIxnAPI-CREATE-URL":
  "/genesys/1/service/81f0ef4e-99dd-43ea-8366-8d27a2cbd605/ixn/chat",
  "_id": "81f0ef4e-99dd-43ea-8366-8d27a2cbd605"
}

```

## Chat Interaction APIs

### Start Chat

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

#### Operation

POST /genesys/1/service/{service_id}/ixn/chat			
Parameter	Type	Mandatory	Description
<b>URI Parameters</b>			
<b>{service_id}</b>	string	yes	The identifier of the service that the chat session is suppose to be associated with.
<b>Body:</b> The body can be either a MultiPart form, or x-www-form-urlencoded form, or JSON key/value pairs associated with the request.			
_verbose	boolean	yes if JSON	Allows the application to get all the detail attributes associated with the chat session in the corresponding response.
message	string	yes if JSON	Contains a custom notice message that replaces the standard system notice in the chat session/transcript.
notify_by	string	no	If specified, should be "comet".
firstName	string	no	User's first name. Optional.
lastName	string	yes	User's last name. Optional.
email	string	yes	User's email address. Optional.
subject	string	yes	Subject of the chat conversation.

POST /genesys/1/service/{service_id}/ixn/chat			
subscriptionID	string	yes	ID of the subscription created to receive specific events on Comet channel disconnection.
userDisplayName	string	yes	Nickname displayed in the chat conversation. Optional.
push_notification_deviceid	string	no	Device ID to use for chat push notifications (used to manage <b>background state</b> ). If not specified, push notifications are disabled.
push_notification_type	string	no	Device Type to use for chat push notification. Possible values are ios, gcm, android, comet, httpcb, orscb.
push_notification_debug	boolean	no	Set to true to enable debug push notifications in the GMS server. Default value is false.
push_notification_language	string	no	Set the language to use for chat push notifications. (See <a href="#">localization instructions here</a> ).
push_notification_maxsize	integer	no	Limit the payload size of push notifications messages. If you do not add the <code>push_notification_maxsize</code> parameter to your query, the the payload size is set to 4096 bytes by default.

**Tip**  
 For iOS version 7 and earlier, your iOS device limits the push notification payload size to 256 bytes. Set this parameter to 256 to ensure correct push notifications.



## Response

<b>HTTP code</b>	200
<b>HTTP message</b>	OK
<b>Body</b>	A chat JSON object for details on the properties of the object. See the section on <a href="#">data structures</a> for more details.
<b>Notes</b>	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.

## Error

<b>HTTP code</b>	503
<b>HTTP message</b>	Service Unavailable
<b>Body</b>	None
<b>Notes</b>	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/x-www-form-urlencoded
firstName=Vasya&lastName=Pupkin&email=Vasya.Pupkin@genesyslab.com
cject=test
```

### Response (if transcriptPosition input parameter is null):

```
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
{
  "chatIxnState": "TRANSCRIPT",
  "chatSessionId": "000BRa84KRFB00BK",
  "transcriptPosition": "5",
  "chatServiceMessage": "Chat service is available",
  "startedAt": "2012-06-09T22:26:17Z",
  "userId": "015E4FD3CD890036",
  "secureKey": "b306749dabfalcf6",
  "checkChatServiceLoadBalancerPath":
"/WebAPI812/SimpleSamples812/ChatHA/ChatLBServerInfo.jsp
?chatServerLoadBalancerAlias=350",
  "chatServerLoadBalancerAlias": "350",
  "chatServerHost": "localhost",
  "chatWebApiPort": "4856",
  "isTLSrequired": "false",
```

```

    "clientTimeZoneOffset": "-420",
    "_chatIxnAPI_SEND_URL":
"/genesys/1/service/{service_id}/ixn/chat/send",
    "_chatIxnAPI_REFRESH_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh",
    "_chatIxnAPI_START_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/startTyping",
    "_chatIxnAPI_STOP_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/stopTyping",
    "_chatIxnAPI_DISCONNECT_URL":
"/genesys/1/service/{service_id}/ixn/chat/disconnect",
    "_chatIxnAPI_REFRESH_FROM_START_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh?transcriptPosition=1"
}

```

**Response (if the transcript input parameter is set [transcriptToShow output parameter is set]):**

```

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
{
  "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": "localhost",
    "chatWebApiPort": "4856",
    "isTLSrequired": "false",
    "clientTimeZoneOffset": "-420",
    "_chatIxnAPI_SEND_URL":
"/genesys/1/service/{service_id}/ixn/chat/send",
    "_chatIxnAPI_REFRESH_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh",
    "_chatIxnAPI_START_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/startTyping",
    "_chatIxnAPI_STOP_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/stopTyping",
    "_chatIxnAPI_DISCONNECT_URL":
"/genesys/1/service/{service_id}/ixn/chat/disconnect",
    "_chatIxnAPI_REFRESH_FROM_START_URL":
"/genesys/1/service/{service_id}/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

POST /genesys/1/service/{service_id}/ixn/chat/refresh			
Parameter	Type	Mandatory	Description
<b>URI Parameters</b>			
<b>{service_id}</b>	string	yes	The identifier of the service that the chat session is associated with.
<b>Body:</b> The body can be either a MultiPart form, or x-www-form-urlencoded form, or JSON key/value pairs associated with the request.			
<b>Body Properties:</b> The following are the properties:			
<ul style="list-style-type: none"> <li>transcriptPosition - This optional property indicates the current position in the chat session that the current user is in. This property is ignored when notify_by = comet when starting the chat session (ixn/chat)</li> <li>message - This optional property is a chat message that will be added to the chat session/transcript.</li> <li>_verbose - This optional property will allow the application to get all the detail attributes associated with the chat session in the corresponding response.</li> </ul>			

### Response

<b>HTTP code</b>	200
<b>HTTP message</b>	OK

<b>HTTP code</b>	200
<b>Body</b>	A chat JSON object for details on the properties of the object. See the section on <a href="#">data structures</a> for more details.
<b>Notes</b>	The main property is the list of chat message that have been communicated (transcriptToShow).

## Error

<b>HTTP code</b>	503
<b>HTTP message</b>	Service Unavailable
<b>Body</b>	None
<b>Notes</b>	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/x-www-form-urlencoded
message=aaa
```

### Response (if transcriptPosition input parameter is null):

```
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
{
  "chatIxnState": "TRANSCRIPT",
  "chatSessionId": "000BRa84KRFB00BK",
  "transcriptPosition": "5",
  "chatServiceMessage": "Chat service is available",
  "startedAt": "2012-06-09T22:26:17Z",
  "userId": "015E4FD3CD890036",
  "secureKey": "b306749dabfalcf6",
  "checkChatServiceLoadBalancerPath":
"/WebAPI812/SimpleSamples812/ChatHA/ChatLBServerInfo.jsp
?chatServerLoadBalancerAlias=350",
  "chatServerLoadBalancerAlias": "350",
  "chatServerHost": "localhost",
  "chatWebApiPort": "4856",
  "isTLSrequired": "false",
  "clientTimeZoneOffset": "-420",
  "_chatIxnAPI_SEND_URL":
"/genesys/1/service/{service_id}/ixn/chat/send",
  "_chatIxnAPI_REFRESH_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh",
  "_chatIxnAPI_START_TYPING_URL":
```

```

"/genesys/1/service/{service_id}/ixn/chat/startTyping",
  "_chatIxnAPI_STOP_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/stopTyping",
  "_chatIxnAPI_DISCONNECT_URL":
"/genesys/1/service/{service_id}/ixn/chat/disconnect",
  "_chatIxnAPI_REFRESH_FROM_START_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh?transcriptPosition=1"
}

```

**Response (if transcript input parameter is set [transcriptToShow output parameter is set]):**

```

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
{
  "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": "localhost",
  "chatWebApiPort": "4856",

```

```

    "isTLSrequired": "false",
    "clientTimeZoneOffset": "-420",
    "_chatIxnAPI_SEND_URL":
"/genesys/1/service/{service_id}/ixn/chat/send",
    "_chatIxnAPI_REFRESH_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh",
    "_chatIxnAPI_START_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/startTyping",
    "_chatIxnAPI_STOP_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/stopTyping",
    "_chatIxnAPI_DISCONNECT_URL":
"/genesys/1/service/{service_id}/ixn/chat/disconnect",
    "_chatIxnAPI_REFRESH_FROM_START_URL":
"/genesys/1/service/{service_id}/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

POST /genesys/1/service/{service_id}/ixn/chat/startTyping			
Parameter	Type	Mandatory	Description
<b>URI Parameters</b>			
<b>{service_id}</b>	string	yes	The identifier of the service that the chat session is suppose to be associated with.
<b>Body:</b> The body can be either a MultiPart form, or x-www-form-urlencoded form, or JSON key/value pairs associated with the request.			
_verbose	boolean	yes if JSON	This will allow the application to get all the detail attributes associated with the chat session in the corresponding response.
message	string	yes if JSON	This mandatory property must contain a custom notice message that will be added to the chat session/transcript.

### Response

<b>HTTP code</b>	200
<b>HTTP message</b>	OK
<b>Body</b>	A chat JSON object for details on the properties of the object. See the section on <a href="#">data structures</a> for more details.
<b>Notes</b>	None

## Error

<b>HTTP code</b>	503
<b>HTTP message</b>	Service Unavailable
<b>Body</b>	None
<b>Notes</b>	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/EKUJPKAQ197CFA6SJQKTJ03DBG
00001J/ixn/chat/startTyping 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 07:38:38 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Content-Length: 246
{
  "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

POST /genesys/1/service/{service_id}/ixn/chat/stopTyping			
Parameter	Type	Mandatory	Description
<b>URI Parameters</b>			
<b>{service_id}</b>	string	yes	The identifier of the

POST /genesys/1/service/{service_id}/ixn/chat/stopTyping			
			service that the chat session is suppose to be associated with.
<b>Body:</b> The body can be either a MultiPart form, or x-www-form-urlencoded form, or JSON key/value pairs associated with the request.			
_verbose	boolean	yes if JSON	This will allow the application to get all the detail attributes associated with the chat session in the corresponding response.
message	string	yes if JSON	This mandatory property must contain a custom notice message that will be added to the chat session/transcript.

## Response

<b>HTTP code</b>	200
<b>HTTP message</b>	OK
<b>Body</b>	A chat JSON object for details on the properties of the object. See the section on <a href="#">data structures</a> for more details.
<b>Notes</b>	None

## Error

<b>HTTP code</b>	503
<b>HTTP message</b>	Service Unavailable
<b>Body</b>	None
<b>Notes</b>	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/EKUJPKAQ197CFA6SJQKTJ03DBG
00001J/ixn/chat/stopTyping 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 07:38:58 GMT
Pragma: no-cache
Cache-Control: no-cache
```



```

Cache-Control: no-store
Content-Type: application/json
Content-Length: 251
{
  "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

POST /genesys/1/service/{service_id}/ixn/chat/disconnect			
Parameter	Type	Mandatory	Description
<b>URI Parameters</b>			
<b>{service_id}</b>	string	yes	The identifier of the service that the chat session is suppose to be associated with.
<b>Body:</b> The body can be either a MultiPart form, or x-www-form-urlencoded form, or JSON key/value pairs associated with the request.			
_verbose	boolean	yes if JSON	This will allow the application to get all the detail attributes associated with the chat session in the corresponding response.
message	string	yes if JSON	This mandatory property must contain a custom notice message that will be added to the chat session/transcript.

### Response

<b>HTTP code</b>	200
<b>HTTP message</b>	OK
<b>Body</b>	A chat JSON object for details on the properties of the object. See the section on <a href="#">data structures</a> for more details.
<b>Notes</b>	None

## Error

<b>HTTP code</b>	503
<b>HTTP message</b>	Service Unavailable
<b>Body</b>	None
<b>Notes</b>	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/EKUJPKAQ197CFA6SJQKTJ03D
BG00001J/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
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Content-Length: 114
{
  "chatIxnState" : "DISCONNECTED",
  "transcriptPosition" : "9",
  "chatServiceMessage" : "Chat was finished"
}
```

## Send Custom Notice

This API query allows you to send a user notice that contains a custom message.

## Operation

Method	POST		
URL	/genesys/1/service/{service_id}/ixn/chat/customNotice		
Parameter	Type	Mandatory	Description
URI Parameters			
{service_id}	string	Yes	The identifier of the service associated with the chat session.
Body	The body can be either a MultiPart form, or x-www-form-urlencoded form, or JSON key/value pairs associated with the request.		
Body Properties			
_verbose	boolean	No	Set to true to allow the application to get all the detail attributes

Method	POST		
			associated with the chat session in the corresponding response.
message	string	Yes	Custom notice message that will replace the standard system notice in the chat session transcript.

## Response

HTTP code	200
HTTP message	OK
Body	A chat JSON object for the details of the object properties. See the <a href="#">data structures</a> for more details.

## Error

If a problem occurs during subscription, the following status codes are returned:

HTTP code	503
HTTP Message	Service Unavailable
Body	None

### Important

This error is returned if the service could not send a notification to the agent application.

## Example

### Request

```
POST http://localhost:8080/genesys/1/service
/EKUJPKAQ197CFA6SJQKTJ03DBG00001H/ixn/chat
/customNotice?_verbose=true HTTP/1.1
```

```
Accept-Encoding: gzip,deflate
Content-Type: application/x-www-form-urlencoded
message=photo.png
```

### 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
{
  "chatIxnState": "TRANSCRIPT",
  "chatSessionId": "000BRa84KRFB00BK",
  "transcriptPosition": "5",
  "chatServiceMessage": "Chat service is available",
  "startedAt": "2012-06-09T22:26:17Z",
  "userId": "015E4FD3CD890036",
  "secureKey": "b306749dabfalcf6",
  "checkChatServiceLoadBalancerPath":
"/WebAPI812/SimpleSamples812/ChatHA/ChatLBServerInfo.jsp
?chatServerLoadBalancerAlias=350",
  "chatServerLoadBalancerAlias": "350",
  "chatServerHost": "localhost",
  "chatWebApiPort": "4856",
  "isTLSrequired": "false",
  "clientTimeZoneOffset": "-420",
  "_chatIxnAPI_SEND_URL":
"/genesys/1/service/{service_id}/ixn/chat/send",
  "_chatIxnAPI_REFRESH_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh",
  "_chatIxnAPI_START_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/startTyping",
  "_chatIxnAPI_STOP_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/stopTyping",
  "_chatIxnAPI_DISCONNECT_URL":
"/genesys/1/service/{service_id}/ixn/chat/disconnect",
  "_chatIxnAPI_REFRESH_FROM_START_URL":
"/genesys/1/service/{service_id}/ixn/chat
/refresh?transcriptPosition=1",
  "_chatIxnAPI_SEND_CUSTOM_URL" :
"/genesys/1/service/{service_id}/ixn/chat/customNotice"
}

```

## 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 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"}]
```

## 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.
subject	yes	Subject of the service and chat session.
userDisplayName	no	Available since GMS 8.1.100.27. Nickname to be displayed in the chat conversation.
notify_by	no	If using a CometD connection for the asynchronous receiving of chat messages, then supply this parameter with the value "comet".
push_notification_deviceid	no	Device ID to use for chat push notifications (used to manage <b>background state</b> ). If not specified, push notifications are disabled.
push_notification_type	no	Device Type to use for chat push notification. Possible values are: ios, gcm, android, comet, httpcb, orscb.
push_notification_debug	no	Set to true to enable debug push notifications in the GMS server. Default value is false.

Parameter Name	Mandatory	Description
push_notification_language	no	Set the language to use for chat push notifications. (See localisation instructions <a href="#">here</a> )
push_notification_maxsize	no	Limit the payload size of push notifications messages. If you do not add the push_notification_maxsize parameter to your query, the payload size is set to 4096 bytes by default.  <b>Tip</b> For iOS version 7 and earlier, your iOS device limits the push notification payload size to 256 bytes. Set this parameter to 256 to ensure correct push notifications.

**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
&push_notification_deviceid=deviceid&push_notification_type=ios
&push_notification_debug=true&push_notification_language=en
```

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

## CometD-Based Chat API

### CometD Handshake

**Request**

```
POST http://localhost:8080/genesys/cometd/handshake
gms_user:
b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673
Content-Type: application/json;charset=UTF-8
[{"version":"1.0","minimumVersion":"0.9",
"channel":"/meta/handshake",
```

```
"supportedConnectionType":["long-polling","callback-polling"],
"advice":{"timeout":60000,"interval":0},"id":"1"}}
```

## Response

```
Content-Type: application/json;charset=UTF-8
[{"id":"1","minimumVersion":"1.0","supportedConnectionTypes":
[{"callback-polling","long-polling"},
"successful":true,"channel":"/meta/handshake","ext":{"ack":true},
"clientId":"3vym301sjdtc218qabm5w0z8yb","version":"1.0"}]
```

## CometD Subscribe

### Request

```
POST http://localhost:8080/genesys/cometd/
gms_user:
b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673
```

```
Content-Type: application/json;charset=UTF-8
[{"channel":"/meta/subscribe","subscription":"/_genesys","id":"2",
"clientId":"3vym301sjdtc218qabm5w0z8yb"}]
```

### Response

```
Content-Type: application/json;charset=UTF-8
[{"id":"2","subscription":"/_genesys","successful":true,
"channel":"/meta/subscribe"}]
```

## CometD Connect

### Request

Note, ext component containing transcriptPosition is optional, see section on network connection loss below.

```
POST http://localhost:8080/genesys/cometd/connect
gms_user:
b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673
```

```
Content-Type: application/json;charset=UTF-8
[{"channel":"/meta/connect","connectionType":"long-polling",
"advice":{"timeout":0},
"id":"3","clientId":"3vym301sjdtc218qabm5w0z8yb"},
"ext":{"transcriptPosition": "4"}]
```

### Response

```
Content-Type: application/json;charset=UTF-8
[{"id":"3","successful":true,"advice":
{"interval":0,"reconnect":"retry","timeout":60000},
"channel":"/meta/connect"}]
```

## Create Service Session

### Request

```
POST http://localhost:8080/genesys/1/service/request-chat
gms_user:
b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673
```

```
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
```

```
_verbose=true
```

## Response

```
Content-Type: application/json; charset=UTF-8
{
  "_chatIxnAPI-CREATE-URL":
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/chat",
  "_id": "4d1697a9-dda5-4742-8a6f-fbc01c25c640",
  "_data_id": "429-791eaae8-571e-4633-9a73-cc936336f8e2"
}
```

## Create Chat Interaction for Session 4d1697a9-dda5-4742-8a6f-fbc01c25c640

### Request

```
POST
http://localhost:8080/genesys/1/service
/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/chat
gms_user:
b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
```

```
_verbose=true&ify_by=comet&FirstName=John&LastName=Harry
cject=French&EmailAddress=j.h%40gmail.com
```

### Response

```
Content-Type: application/json; charset=UTF-8
{
  "chatServerLoadBalancerAlias": "371",
  "_chatIxnAPI_SEND_CUSTOM_URL":
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/chat/customNotice",
  "clientTimeZoneOffset": "120",
  "transcriptPosition": "1",
  "chatWebApiPort": "9002",
  "chatIxnState": "CONNECTED",
  "comet_channel": "/_genesys",
  "secureKey": "1b21478a91a7d1dc",
  "checkChatServiceLoadBalancerPath":
"/WebAPI812/SimpleSamples812/ChatHA/ChatLBServerInfo.jsp
?chatServerLoadBalancerAlias=371",
  "_chatIxnAPI_REFRESH_FROM_START_URL":
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640
/ixn/chat/refresh?transcriptPosition=1",
  "_chatIxnAPI_REFRESH_URL":
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640
/ixn/chat/refresh",
  "chatSessionId": "000E5aA2A40P000Q",
  "isTLSrequired": "false",
  "_chatIxnAPI_DISCONNECT_URL":
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640
/ixn/chat/disconnect",
  "chatServiceMessage": "Chat service is available",
  "userId": "0173542518870006",
}
```

```
"_chatIxnAPI_STOP_TYPING_URL":  
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640  
/ixn/chat/stopTyping",  
"_chatIxnAPI_START_TYPING_URL":  
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640  
/ixn/chat/startTyping",  
"_chatIxnAPI_SEND_URL":  
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640  
/ixn/chat/send",  
"chatServerHost": "demosrv.genesyslab.com"  
}
```

## Polling Agent 'Joined' Message Through CometD

### Request

```
POST http://localhost:8080/genesys/cometd/connect  
gms_user:  
b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673  
Content-Type: application/json;charset=UTF-8
```

```
[{"channel":"/meta/connect","connectionType":"long-polling",  
"id":"4","clientId":"3vym301sjdtc218qabm5w0z8yb"}]
```

### Response

```
Content-Type: application/json;charset=UTF-8  
[  
{  
  "data": {  
    "id": "7b239ff0455011e4b31cbb293fafa316",  
    "message": {  
      "chatSessionId": "000E5aA2A40P000Q",  
      "transcriptPosition": "2",  
      "chatServiceMessage": "Chat service is available",  
      "startedAt": "2014-09-26T07:40:55Z",  
      "chatIxnState": "TRANSCRIPT",  
      "transcriptToShow": [ [ "Notice.Joined", "Kristi Sippola",  
        "has joined the session", "14", "AGENT" ] ] },  
      "tag": "service.chat.refresh.4d1697a9-dda5-4742-8a6f-fbc01c25c640",  
      "channel": "/_genesys",  
      "id": "4",  
      "successful": true,  
      "channel": "/meta/connect"  
    }  
  }  
]
```

## Polling Agent 'StartTyping' Message Through CometD

### Request

```
POST http://localhost:8080/genesys/cometd/connect  
gms_user:  
b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673  
Content-Type: application/json;charset=UTF-8
```

```
[{"channel":"/meta/connect","connectionType":"long-polling",  
"id":"5","clientId":"3vym301sjdtc218qabm5w0z8yb"}]
```

### Response

```
Content-Type: application/json;charset=UTF-8  
[  
{  
  "data": {  
    "id": "802c88e0455011e4b31cbb293fafa316",  
    "message": {  
      "chatSessionId": "000E5aA2A40P000Q",  

```

```

"transcriptPosition": "3",
"chatServiceMessage": "Chat service is available",
"startedAt": "2014-09-26T07:40:55Z",
"chatIxnState": "TRANSCRIPT",
"transcriptToShow":
[["Notice.TypingStarted", "Kristi Sippola",
"is typing", "22", "AGENT"]]],
"tag": "service.chat.refresh.4d1697a9-dda5-4742-8a6f-fbc01c25c640",
"channel": "/_genesys",
{"id": "5", "successful": true, "channel": "/meta/connect"}
]

```

## Polling Agent Chat Message Through CometD

### Request

```

POST http://localhost:8080/genesys/cometd/connect
gms_user:
b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673
Content-Type: application/json; charset=UTF-8

```

```

[{"channel": "/meta/connect", "connectionType": "long-polling",
"id": "6", "clientId": "3vym301sjdtc218qabm5w0z8yb"}]

```

### Response

```

Content-Type: application/json; charset=UTF-8
[
{"data": {"id": "816f9030455011e4b31cbb293fafa316",
"message": {"chatSessionId": "000E5aA2A40P000Q",
"transcriptPosition": "4",
"chatServiceMessage": "Chat service is available",
"startedAt": "2014-09-26T07:40:55Z",
"chatIxnState": "TRANSCRIPT",
"transcriptToShow": [["Message.Text", "Kristi
Sippola", "Hello", "23", "AGENT"]]],
"tag": "service.chat.refresh.4d1697a9-dda5-4742-8a6f-fbc01c25c640",
"channel": "/_genesys",
{"id": "6", "successful": true, "channel": "/meta/connect"}
}
]

```

## Send Client Chat Message

### Request

```

POST
http://localhost:8080/genesys/1/service
/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/chat/refresh
gms_user:
"b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673
Content-Type: application/x-www-form-urlencoded; charset=UTF-8

_verbose=true&message=Hi%20Verbose

```

### Response

```

Content-Type: application/json; charset=UTF-8
{
"chatServerLoadBalancerAlias": "371",

```

```

"_chatIxnAPI_SEND_CUSTOM_URL":
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640
/ixn/chat/customNotice",
"clientTimeZoneOffset": "120",
"transcriptPosition": "5",
"chatWebApiPort": "9002",
"startedAt": "2014-09-26T07:40:55Z",
"chatIxnState": "TRANSCRIPT",
"comet_channel": "/_genesys",
"secureKey": "1b21478a91a7d1dc",
"checkChatServiceLoadBalancerPath":
"/WebAPI812/SimpleSamples812/ChatHA/ChatLBServerInfo.jsp
?chatServerLoadBalancerAlias=371",
"_chatIxnAPI_REFRESH_FROM_START_URL":
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn
/chat/refresh?transcriptPosition=1",
"_chatIxnAPI_REFRESH_URL":
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640
/ixn/chat/refresh",
"isTLSrequired": "false",
"chatSessionId": "000E5aA2A40P000Q",
"_chatIxnAPI_DISCONNECT_URL":
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640
/ixn/chat/disconnect",
"chatServiceMessage": "Chat service is available",
"userId": "0173542518870006",
"_chatIxnAPI_STOP_TYPING_URL":
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640
/ixn/chat/stopTyping",
"_chatIxnAPI_START_TYPING_URL":
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640
/ixn/chat/startTyping",
"_chatIxnAPI_SEND_URL":
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640
/ixn/chat/send",
"chatServerHost": "demosrv.genesyslab.com"
}

```

## Client Message is Being Echoed Back Through CometD Channel as a Response to "refresh" or "send" Request

### Request

```

POST http://localhost:8080/genesys/cometd/connect
gms_user:
b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673
Content-Type: application/json;charset=UTF-8

[{"channel": "/meta/connect", "connectionType": "long-polling", "id": "7",
"clientId": "3vym301sjdtc218qabm5w0z8yb"}]

```

### Response

```

Content-Type: application/json;charset=UTF-8
[
{"data": {"id": "867b3840455011e4b31cbb293fafa316",
"message": {"chatSessionId": "000E5aA2A40P000Q",
"transcriptPosition": "5",
"chatServiceMessage": "Chat service is available",
"startedAt": "2014-09-26T07:40:55Z",

```

```

        "chatIxnState": "TRANSCRIPT",
        "transcriptToShow": [
            [
                "Message.Text",
                "127.0.0.1",
                "Hi Verbose",
                "32",
                "CLIENT"
            ]
        ],
        "tag": "service.chat.refresh.4d1697a9-dda5-4742-8a6f-fbc01c25c640",
        "channel": "/_genesys",
        {
            "id": "7",
            "successful": true,
            "channel": "/meta/connect"
        }
    ]
}

```

## CometD Polling

### Request

POST http://localhost:8080/genesys/cometd/connect  
gms\_user:  
b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673  
Content-Type: application/json; charset=UTF-8

```

[
    {
        "channel": "/meta/connect",
        "connectionType": "long-polling",
        "id": "8",
        "clientId": "3vym301sjdtc218qabm5w0z8yb"
    }
]

```

### Response

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

```

{
    "id": "8",
    "successful": true,
    "advice": {
        "reconnect": "none"
    },
    "channel": "/meta/connect"
}

```

## Disconnect Chat Session

### Request

POST  
http://localhost:8080/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/chat/disconnect  
gms\_user:  
b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673  
Content-Type: application/x-www-form-urlencoded; charset=UTF-8  
\_verbose=true

### Response

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

```

{
    "chatIxnState" : "DISCONNECTED",
    "transcriptPosition" : "5",
    "chatServiceMessage" : "Chat was finished",
    "chatSessionId" : "000E5aA2A40P000Q",
    "userId" : "0173542518870006",
    "secureKey" : "1b21478a91a7d1dc",
    "checkChatServiceLoadBalancerPath" :
        "/WebAPI812/SimpleSamples812/ChatHA/ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=371",
    "chatServerLoadBalancerAlias" : "371",
    "chatServerHost" : "demosrv.genesyslab.com",
    "chatWebApiPort" : "9002",
    "isTLSrequired" : "false",
    "clientTimeZoneOffset" : "120",
    "_chatIxnAPI_SEND_URL" :
        "/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn

```

```
/chat/send",
"_chatIxnAPI_REFRESH_URL" :
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn
/chat/refresh",
"_chatIxnAPI_START_TYPING_URL" :
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn
/chat/startTyping",

"_chatIxnAPI_STOP_TYPING_URL" :
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn
/chat/stopTyping",

"_chatIxnAPI_DISCONNECT_URL" :
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn
/chat/disconnect",
"_chatIxnAPI_REFRESH_FROM_START_URL":
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn
/chat/refresh?transcriptPosition=1",
"_chatIxnAPI_SEND_CUSTOM_URL" :
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn
/chat/customNotice"
}
```

## CometD Unsubscribe

### Request

```
POST http://localhost:8080/genesys/cometd/
gms_user:
  b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673

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

[{"channel":"/meta/unsubscribe","subscription":"/_genesys",
"id":"9","clientId":"3vym301sjdtc218qabm5w0z8yb"}]
```

### Response

```
Content-Type: application/json;charset=UTF-8
[{"id":"9","subscription":"/_genesys","successful":true,
"channel":"/meta/unsubscribe"}]
```

## CometD Disconnect

### Request

```
POST http://localhost:8080/genesys/cometd/disconnect
gms_user:
b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673
Content-Type: application/json;charset=UTF-8

[{"channel":"/meta/disconnect","id":"10","clientId":
"3vym301sjdtc218qabm5w0z8yb"}]
```

### Response

```
Content-Type: application/json;charset=UTF-8
[{"id":"10","successful":true,"channel":"/meta/disconnect"}]
```



---

## Chat Push Notification and Background State

### Mobile Application Background State Issues

If an iOS or Android mobile application is moved to the background state, the operating system does not close active TCP connections. So, if your app does not handle the foreground and background events, the server-side of your CometD connection isn't notified.

More specifically, if your app handles chat with GMS, the agent may send a new message which would result in sending the long poll response. Here is a list of the various scenarios which can happen according to your implementation.

- The Cometd server attempts to send long poll response but fails and closes socket of current long-poll request.
- The Cometd server sets a 10 second timer; if the app does not reconnect within this period, the app is considered to be gone and its context is destroyed.

Later, if the app returns to the foreground, it does not receive chat notifications from the Cometd server and the chat session appears to be broken.

To avoid these type of issues, your app should listen for events that both iOS and Android submit when your app enters the background (or foreground) state. See the Official documentation.

- Apple provides background instructions for iOS, [here](#).
- Android provides best background practices [here](#).

Then, your app should implement the Cometd Meta Disconnect message detailed below to handle background State issues.

### Cometd Meta Disconnect Messages

To handle background state, your app must send Cometd /meta/disconnect messages to GMS and later, if your app returns to foreground, it should establish a new Cometd session as detailed below.

1. When your app enters the background state, it sends a Comet/meta/disconnect message to GMS which includes the highest transcript position received by your app, as for example:

```
{
  "channel": "/meta/disconnect",
  "clientId": "71k6evjfbffq9eir3jhn6udxz",
  "ext": {
    "transcriptPosition": "4"
  },
  "id": "4"
}
```

2. If GMS receives a `/meta/disconnect` message, it flags your app as disconnected and sends a chat Notice. Custom event to the agent. The default value is `customer is not online`. You can configure this text.
  1. Edit your GMS application (with Configuration Manager for example).
  2. Navigate to **Options > Service.<service\_name>**.
  3. Edit the `_agent_timeout_notification_message` value.
3. If the agent sends a new message while your app is disconnected, GMS starts a configurable timer. You can set this timer value to zero
  1. Edit your GMS application (with Configuration Manager for example).
  2. Navigate to **Options > Service.<service\_name>**.
  3. Edit the `_client_timeout_notification` value.

When the timer expires, GMS sends a push notification (iOS apns, Android gcm, or http). Typically, the notification results in a dialog which allows the user to optionally return the app to the foreground.

4. The chat session may have ended when the app returns to the foreground, either because the agent left the session or because the session timed out as a result of user inactivity. In that case, the app no longer receives Cometd notifications. To detect if the session has ended, call the Chat Refresh Chat API – it will return a 4xx reply to indicate that a session has ended.
5. When the app returns to the foreground and if the session has not ended, it must start a new Cometd session with GMS. GMS flags the app as connected and sends a Cometd notification that includes all chat transcript events that occurred after the app entered the background state, as for example:

```
[
  {
    "data":{
      "id":"fdf93730c1e411e4a5e8f1be76b28efd",
      "message":
        { "chatSessionId":"0001BaAFC4J8000N",
          "transcriptPosition":"6",
          "chatServiceMessage":"Chat service is available",
          "startedAt":"2015-03-03T20:36:12Z",
          "chatIxnState":"TRANSCRIPT", "transcriptToShow":
            [ ["Notice.TypingStarted","agentName","is typing","28","AGENT"],
              ["Message.Text","agentName","Hello","29","AGENT"] ] },
          "tag":"service.chat.refresh.180-655e104c-a6cf-447d-b94b-f132d49a35fe"
        },
      "channel":"/_genesys"
    },
    { "successful":true, "channel":"/meta/connect" }
  ]
```

## Important

- If your app returns to foreground before the timer expires, no push notification is sent, but your app must still start a new Cometd session and it will receive a Cometd notification (which includes all new transcript data) from GMS.
- If your app does not return to foreground after the first push notification is sent, GMS sends additional push notification messages for each new agent event. Note that these

events are sent immediately.

## Content of Push Notification Messages

GMS triggers push notification messages if the agent submits new chat events to the app flagged as disconnected. The following events can result in a notification:

- Notice.TypingStarted
- Notice.TypingStopped
- Notice.Joined
- Notice.Left
- Notice.PushUrl
- Notice.Custom
- Message.Text

Your application's configuration includes a filter to exclude events from triggering a push notification. The `filtering_chat_events` option in the push section sets the default value for this filter to `Notice.TypingStarted,Notice.TypingStopped`. You can still set a different value for your service.

1. Edit your GMS application (with Configuration Manager for example).
2. Navigate to **Options > Service.<service\_name>**.
3. Edit the `_filtering_chat_events` option to add new event types to exclude.

For the Android and http push notification types, notification messages include the content of the filtered agent events.

The following example of the push notification message includes the content of the filtered agent events.

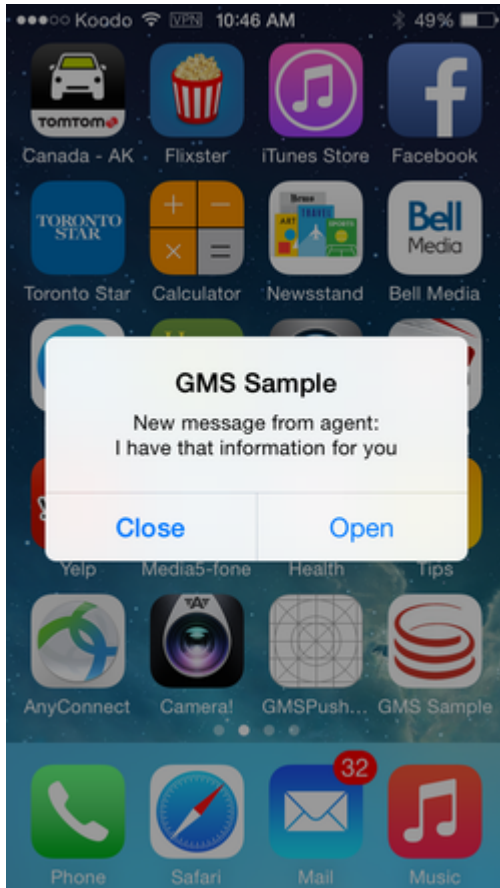
```
"message": {
  "message": "New message from Agent",
  "serviceId": "180-e941460d-1eb0-4f0b-9022-b99ca9ee41f7",
  "lastTranscript": [
    {"Message.Text": "A line of text."}
    {"Message.Text": "Another line of text."}
  ]
}
```

By default, the message attribute (or notification message) is set to `New message from Agent`, but you can change this text in your service options.

1. Edit your GMS application (with Configuration Manager for example).
2. Navigate to **Options > Service.<service\_name>**.
3. Edit the value of the `_client_timeout_notification_message` option.

The notification is tagged as `chat.newagentmessage`. In addition to the notification message, the notification content provides the `lastTranscript` text that can be displayed to the user.

- You can display this the `lastTranscript` text in a popup a or banner notification to the user.



## Specific Configuration for Chat Push Notification

To customize your GMS configuration for push notification messages (for both iOS and Android), you can create a `push.provider.event.chat.newagentmessage` section in the **Options** tab (with Configuration Manager for instance). You can add there any of your Android or iOS options.

- This additional configuration is not mandatory.

### Tip

- Read more about options of the GMS push section [here](#).
- Read more about the OS specific capabilities associated with the notification message [here](#).

## CometD and Loss of Network Connection

Mobile devices are subject to temporary loss of carrier or wifi network connectivity. If this occurs when a chat session is active, the mobile device may not receive CometD transcription notifications that occurred during or around the time of the loss.

The mobile device CometD client should detect when network connectivity has been lost; for example, the current long-poll request will return an error. When this occurs, the CometD client should establish a new session starting with a handshake sequence. To support the mobile client informing GMS on the transcription notifications already received, the first /meta/connect message that is sent after the handshake can optionally include the client's transcriptPosition in the ext component. An example connect message is shown below.

```
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",
  "connectionType": "long-polling"
  "advice": {
    "timeout": 0
  },
  "ext": {
    "transcriptPosition": "14"
  },
  "id": "1"
}
```

If the transcriptPosition is provided this way the next CometD notification sent by GMS will include transcript content starting at the next position, so the client will only receive the content that was missed due to the network connection loss.

Note, If the transcriptPosition is not provided in the /meta/connect after a network loss, GMS will estimate the client transcriptPosition based on the timing of the loss of network connection. Typically this will result in the next CometD notification including some transcript content that was previously received, but could also result in the transcript content starting at an offset greater than what was received. If the client does not provide the transcriptPosition upon reconnecting after a network connection loss, it is recommended to call the refresh API function to reset the local transcript display.

---

# Chat API Version 1 Push Notification and Background State

## Mobile Application Background State Issues

If an iOS or Android mobile application is moved to the background state, the operating system does not close active TCP connections. So, if your app does not handle the foreground and background events, the server-side of your CometD connection isn't notified.

More specifically, if your app handles chat with GMS, the agent may send a new message which would result in sending the long poll response. Here is a list of the various scenarios which can happen according to your implementation.

- The Cometd server attempts to send long poll response but fails and closes socket of current long-poll request.
- The Cometd server sets a 10-second timer; if the app does not reconnect within this period, the app is considered to be gone and its context is destroyed.

Later, if the app returns to the foreground, it does not receive chat notifications from the Cometd server and the chat session appears to be broken.

To avoid these types of issues, your app should listen for events that both iOS and Android submit when your app enters the background (or foreground) state. See the Official documentation:

- Apple provides background instructions for iOS, [here](#)
- Android provides best background practices [here](#).

Then, your app should implement the Cometd Meta Disconnect message detailed below to handle background State issues.

## Cometd Meta Disconnect Messages

To handle background state, your app must send Cometd `/meta/disconnect` messages to GMS and later, if your app returns to foreground, it should establish a new Cometd session as detailed below.

1. When your app enters the background state, it sends a Comet/meta/disconnect message to GMS which includes the highest transcript position received by your app, as for example:

```
{
  "channel": "/meta/disconnect",
  "clientId": "71k6evjfbffqq9eir3jhn6udxz",
  "ext":
```

```

    {
      "transcriptPosition": "4"
    },
    "id": "4"
  }

```

2. If GMS receives a `/meta/disconnect` message, it flags your app as disconnected and sends a chat Notice.Custom event to the agent. The default value is `customer is not online`. You can configure this text.
  1. Edit your GMS application (with Configuration Manager for example).
  2. Navigate to **Options > Service.<service\_name>**.
  3. Edit the `_agent_timeout_notification_message` value.
3. If the agent sends a new message while your app is disconnected, GMS starts a configurable timer. You can set this timer value to zero:
  1. Edit your GMS application (with Configuration Manager for example).
  2. Navigate to **Options > Service.<service\_name>**.
  3. Edit the `_client_timeout_notification` value.

`</toggledisplay>`. When the timer expires, GMS sends a push notification (iOS apns, Android gcm, or http). Typically, the notification results in a dialog which allows the user to optionally return the app to the foreground.
4. If the app returns to the foreground, it must start a new Cometd session with GMS. GMS flags the app as connected and sends a Cometd notification that includes all chat transcript events which occurred after the app entered the background state, as for example:

```

[
  {
    "data":{
      "id":"fdf93730c1e411e4a5e8f1be76b28efd",
      "message":
        { "chatSessionId":"0001BaAFC4J8000N", "transcriptPosition":
          "6", "chatServiceMessage":"Chat service is available", "startedAt":
            "2015-03-03T20:36:12Z", "chatIxnState":"TRANSCRIPT",
            "transcriptToShow":[ ["Notice.TypingStarted","agentName","is typing","28","AGENT"],
              ["Message.Text","agentName","Hello","29","AGENT"] ] },
      "tag":
        "service.chat.refresh.180-655e104c-a6cf-447d-b94b-f132d49a35fe"
    },
    "channel":"/_genesys"
  },
  { "successful":true, "channel":"/meta/connect" }
]

```

## Important

- If your app returns to foreground before the timer expires, no push notification is sent, but your app must still start a new Cometd session and it will receive a Cometd notification (which includes all new transcript data) from GMS.
- If your app does not return to foreground after the first push notification is sent, GMS

sends additional push notification messages for each new agent event. Note that these events are sent immediately.

## Content of Push Notification Messages

GMS triggers push notification messages if the agent submits new chat events to the app flagged as disconnected. The following events can result in a notification:

- Notice.TypingStarted
- Notice.TypingStopped
- Notice.Joined
- Notice.Left
- Notice.PushUrl
- Notice.Custom
- Message.Text

Your application's configuration includes a filter to exclude events from triggering a push notification. The `filtering_chat_events` option in the push section sets the default value for this filter to `Notice.TypingStarted,Notice.TypingStopped`. You can still set a different value for your service.

1. Edit your GMS application (with Configuration Manager for example).
2. Navigate to **Options > Service.<service\_name>**.
3. Edit the `_filtering_chat_events` option to add new event types to exclude.

For the Android and HTTP push notification types, notification messages include the content of the filtered agent events. Currently, this is not supported for iOS push notification messages due to the 256-byte payload size limit for this notification type.

The following example of the push notification message includes the content of the filtered agent events.

```
"message": {
  "message": "New message from Agent",
  "serviceId": "180-e941460d-1eb0-4f0b-9022-b99ca9ee41f7",
  "lastTranscript": [
    {"Message.Text": "A line of text."}
    {"Message.Text": "Another line of text."}
  ]
}
```

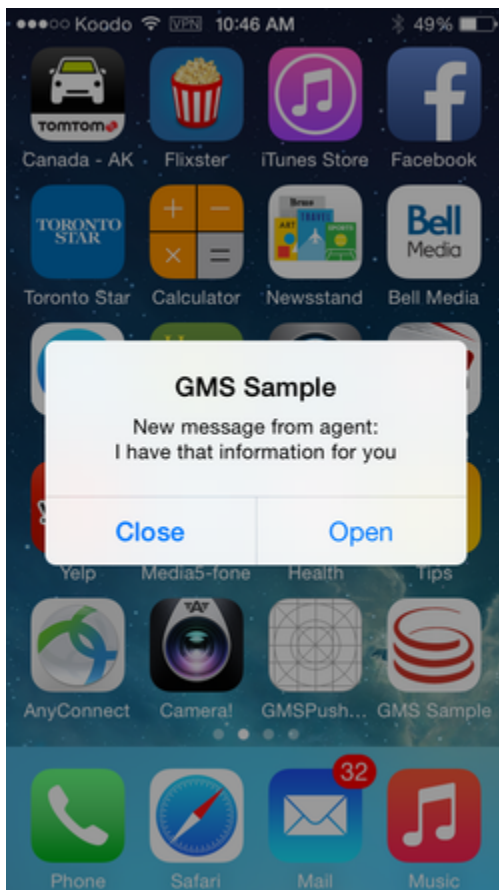
By default, the message attribute (or notification message) is set to `New message from Agent`, but you can change this text in your service options.



1. Edit your GMS application (with Configuration Manager for example).
2. Navigate to **Options > Service.<service\_name>**.
3. Edit the value of the `_client_timeout_notification_message` option.

The notification is tagged as `chat.newagentmessage`. In addition to the notification message, the notification content provides the `lastTranscript` text that can be displayed to the user.

You can display this the `lastTranscript` text in a popup a or banner notification to the user.



## Specific Configuration for Chat Push Notification

To customize your GMS configuration for push notification messages (for both iOS and Android), you can create a `push.provider.event.chat.newagentmessage` section in the **Options** tab (with Configuration Manager for instance). You can add there any of your Android or iOS options.

- This additional configuration is not mandatory.

### Tip

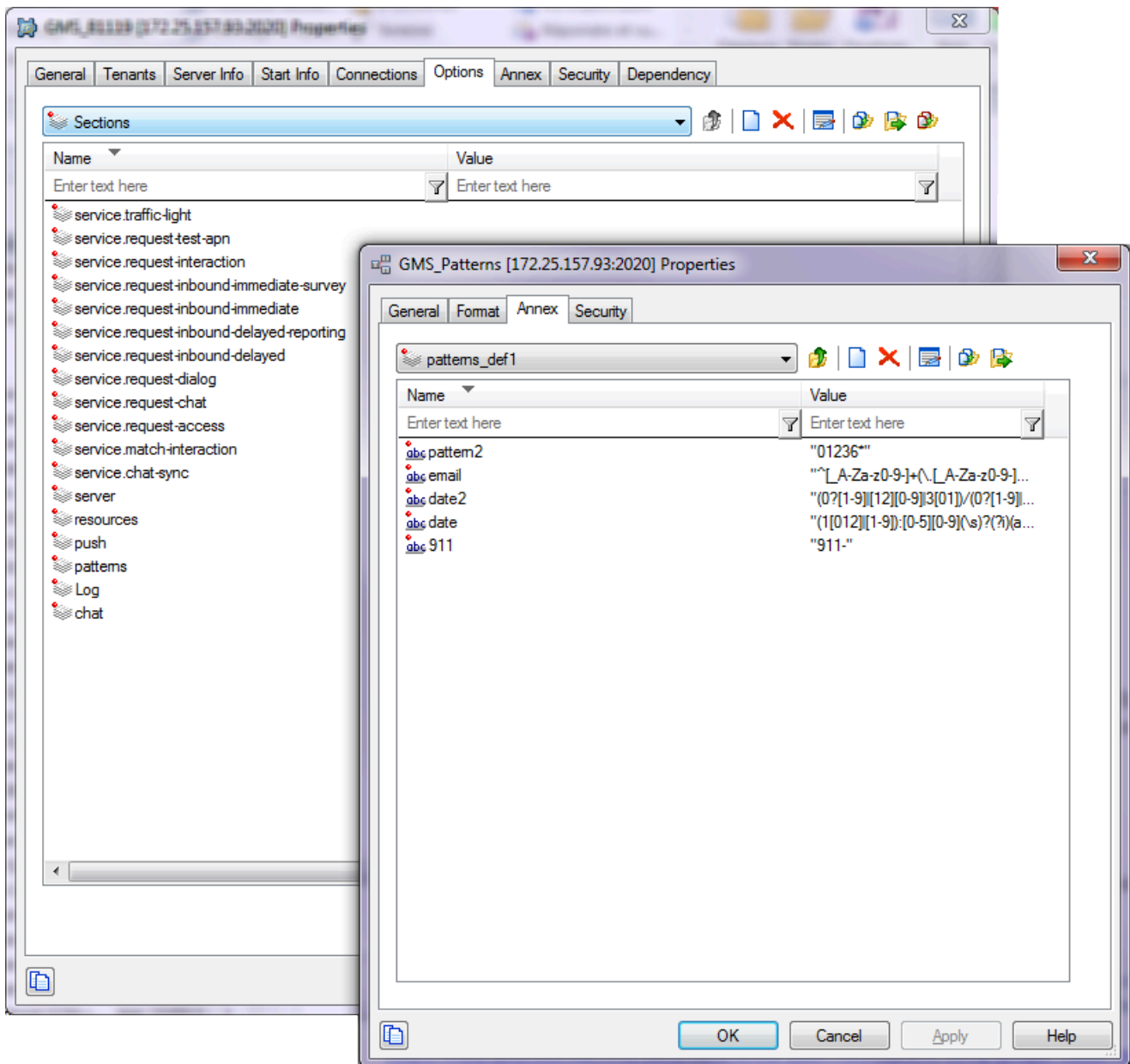
- Read more about options of the GMS push section [here](#).
- Read more about the OS specific capabilities associated with the notification message [here](#).

# Chat API Version 2

Information for using this API has been moved to the [Digital Channels Chat API](#).

# Pattern Matcher API

## Getting Started



The Pattern Matcher API allows you to create and manage pattern lists that you can use to check parameter values and define exceptions in your GMS service.

First, configure a list of patterns or groups of list of patterns in your [GMS Configuration object](#) service, in the GMS Service Management UI as detailed in the [Callback Solution Guide](#).

For example, the left-side screenshot shows the GMS configuration of the patterns\_def1 group.

Once you have defined some patterns, use the Pattern Matcher API queries to check the validity of your parameters. Each query will return the name of the matching pattern (if any) for each parameter. As a result, if none of the parameters match the patterns, the response will be an empty array.

For example, if you submit the following parameters:

```
param1 = 1234562
param2 = john.doe@gmail.com
param3 = 911
```

The API result would be:

```
"param3": "911", "param2": "email"
```

---

## Pattern Format

The exception patterns are regular expressions as defined in the [Java Pattern Class](#).

## List of API Queries

- **POST genesys/1/patterns**: Verify parameters against general patterns list.
- **POST genesys/1/patterns/group/{groupName}**: Verify parameters against a specific group in the patterns list.

## Verify Parameters Against General Patterns List

Use this query to submit a list of parameters to verify. The method returns a JSON array of the parameters that match one of the patterns, where: the keys are the parameters and the values are the name of the matching pattern in the general pattern list. Only strings that match one of the patterns are returned; others are ignored. As a result, if none of the parameters match the patterns, the response will be an empty array.

---

## Operation

### POST genesys/1/patterns

**Body:** The body can be either a MultiPart form or x-www-form-urlencoded form that consists of key/value pairs representing the strings to test.

For example: param1=<string to check>&param2=<string to check>...&param-n=<string to check>

## Response

HTTP code	200
HTTP Message	OK
Body	A JSON array of key-value pairs where the key is the parameter name and the value is the name of the matching pattern. For example: {"param1": "pattern1", "param2": "pattern2", ..., "param-n": "pattern-n"} where pattern-i is the name of a pattern.

## Example

The following example verifies param1, param2, and param3.

```
POST http://127.0.0.1:8080/genesys/1/patterns HTTP/1.1
Host: 127.0.0.1:8080
Connection: keep-alive
Content-Length: 347
Cache-Control: no-cache
Origin: chrome-extension://fdmmgilgnpjigdojojpjoooidkmcomcm
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.22
(KHTML, like Gecko) Chrome/25.0.1364.152 Safari/537.22
Content-Type: multipart/form-data;
boundary=---WebKitFormBoundaryTkdw0u7LG1bBGbnj
Accept: */*
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8,fr;q=0.6,fr-FR;q=0.4
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
-----WebKitFormBoundaryTkdw0u7LG1bBGbnj
Content-Disposition: form-data; name="param1"
12
-----WebKitFormBoundaryTkdw0u7LG1bBGbnj
Content-Disposition: form-data; name="param2"
john.doe@gmail.com
-----WebKitFormBoundaryTkdw0u7LG1bBGbnj
Content-Disposition: form-data; name="param3"
911-
-----WebKitFormBoundaryTkdw0u7LG1bBGbnj--
```

The following response indicates that **param2** and **param3** match some patterns defined in Configuration Manager.

```
HTTP/1.1 200 OK
Date: Thu, 14 Mar 2013 16:13:06 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json; charset=UTF-8
```

Content-Length: 44

```
{"param3": "911", "param2": "email"}
```

## Verify Parameters Against a Specific Group in the Patterns List

Use this query to submit a list of parameters to verify against the patterns defined in a specific group that is part of the general pattern list. The method returns a JSON array of the parameters that match one of the patterns of this group, where the keys are the parameters and the values are the name of the matching pattern in the group pattern list. Only strings that match one of the patterns are returned; others are ignored. As a result, if none of the parameters match the patterns, the response will be an empty array.

### Operation

POST genesys/1/patterns/group/{groupName}			
Parameter	Type	Mandatory	Description
<b>URI Parameters</b>			
<b>{groupName}</b>	String	Yes	The group to which the patterns belong.
<b>Body:</b> The body can be either a MultiPart form or x-www-form-urlencoded form consisting of key/value pairs, representing the strings to test.  For example:  param1=<string to check>&param2=<string to check>...&param-n=<string to check>			

### Response

HTTP code	200
HTTP Message	OK
Body	A JSON array of key-value pairs where the key is the parameter name and the value is the pattern found for this parameter. {"param1": "pattern1", "param2": "pattern2", ..., "param-n": "pattern-n"} where pattern-i is the name of a pattern.

## Example

The following example verifies if the param1 and param2 match the group of patterns "patterns\_def1".

### Operation

```
POST http://127.0.0.1:8080/genesys/1/patterns/group/patterns_def1
HTTP/1.1
Host: 127.0.0.1:8080
Connection: keep-alive
Content-Length: 41
```

```
Cache-Control: no-cache
Origin: chrome-extension://fdmmgilgnpjigdojojpjoooidkmcomcm
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64)
  AppleWebKit/537.22 (KHTML, like Gecko)
Chrome/25.0.1364.152 Safari/537.22
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,fr;q=0.6,fr-FR;q=0.4
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
param1=john.doe%40gmail.com¶m2=blabla
```

## Result

```
HTTP/1.1 200 OK
Date: Thu, 14 Mar 2013 16:22:30 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json; charset=UTF-8
Content-Length: 30
{"param1": "email"}
```

The following response indicates that param1 matches the email pattern of the **patterns\_def1** group defined in Configuration Manager.

## Sample Errors

Your application can receive an HTTP error 403 Forbidden if your service and your GMS configuration do not include the required patterns or group of patterns.

```
HTTP/1.1 403 Forbidden
Date: Thu, 14 Mar 2013 16:23:35 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Length: 107
{"message": "Group (patterns_def1s) unknown.",
 "exception": "com.genesyslab.gsg.services.pattern.MatchPattern"}
```

```
HTTP/1.1 403 Forbidden
Date: Thu, 14 Mar 2013 16:24:32 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Length: 142
{"message": "Service (match-interaction) not configured
to support exceptions.",
 "exception": "com.genesyslab.gsg.services.pattern.MatchPattern"}
```



# Service API

## Overview

This API is used by customer-facing applications to manage different types 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.

### Important

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

## Create Service

This API query creates and initiates a service that you previously configured in the Admin UI in the Configured Services interface.

### Important

Before you start using a service by making REST-queries to its execution name, you must create it with this API query.

## Multipart or Urlencoded

### Multipart or Urlencoded Operation

Method	POST		
URL	/genesys/1/service/{service}		
Parameter	Type	Mandatory	Description
URI Parameters			
{service}	string	yes	The name of the service

Method	POST		
			that is to be created and initiated.
<b>Body:</b> 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 be only strings.			

## Response

HTTP code	200
HTTP message	OK
Body	<p>A JSON object with the following: <code>{"id": "\${service_id}, {service_specific_data}"}</code></p> <p>where:</p> <ul style="list-style-type: none"><li>• <code>\${service_id}</code> is the identifier assigned to the created service instance.</li><li>• <code>\${service_specific_data}</code> is service specific data that can be returned when the service is created.</li></ul>

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"
}
```

## JSON

### JSON Body - Operation

Since 8.5.104, the Create Service query supports JSON inputs. You can now specify key/value pairs as a JSON object. Because ORS allows simple key/value parameters, GMS cannot submit complex JSON-structured values (such as maps and arrays) as ORS request parameters and convert complex values to strings.

For example:

```
{
  "address": "{street=mystreet, city=mycity}", // for map
  "phones": "[9002@SIP_Switch, 5555666]" // for array
}
```

### Important

The JSON API does not support form-encoded binaries.

<b>Method</b>	POST		
<b>URL</b>	/genesys/1/service/{service}		
<b>Parameter</b>	<b>Type</b>	<b>Mandatory</b>	<b>Description</b>
<b>URI Parameters</b>			
{service}	string	yes	The name of the service to create and initiate.
<b>Header</b>			
Content-type: application/json;charset=UTF-8			
<b>Body:</b> A simple JSON structure representing key/value pairs.			
<ul style="list-style-type: none"><li>• Keys that begin with "_" are ignored for storage.</li><li>• No binary values.</li><li>• Complex JSON-structured values (map and arrays) are converted to strings.</li></ul>			
For example:			
<pre>{   "lastname" : "mylastname",   "address": {     "street": "mystreet",     "city": "mycity"   },   "phones": [     "9002@SIP_Switch",     "5555666",     "7775666"   ],   "_key": "ignored" }</pre>			

## Response

<b>HTTP code</b>	200
<b>HTTP message</b>	OK

If the JSON format is incorrect, the query returns a 500 error message.

<b>HTTP code</b>	500
<b>HTTP message</b>	Server Error
Body	{"message": "Could not read JSON: Unexpected character ('\"' (code 34)): [...]"}.

## Example

```
POST http://localhost:8080/genesys/1/service/request-interaction
Content-Type: application/json
gms_user: default
```

```
{
  "_phone_number": "6504669999",
  "_provide_code": "true",
  "language": "french",
  "current_location_latitude": "48.8583",
  "current_location_longitude": "2.2944"
}
```

Status Code: 200 OK

## Update Conversation Expiration

## Service Specific Request

This API allows the application to perform a specific request against a given service.

### Important

This is only to be used for services that support such a request, 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 requested.
{request}	string	yes	This is the name of the request that is to be performed.
<b>Body:</b> 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 be only strings.			

## Response

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

## Example

See the [Chat Interaction APIs](#) for an example.

## Query All Keys

This API queries all of the keys in the storage area that has already been created for the service.

Note: Introduced in 8.5.000.12.

## Operation

<b>Method</b>	GET		
<b>URL</b>	/genesys/1/service/{id}/storage		
<b>Parameter</b>	<b>Type</b>	<b>Mandatory</b>	<b>Description</b>
<b>URI Parameters</b>			
{id}	string	yes	The id of the service.
<b>Body:</b> None			

## Response

<b>HTTP code</b>	200
<b>HTTP message</b>	OK

## Example

The following example queries all of the keys associated with service efef8eb61-1f24-593d-90da-0034aca34b55.

### Operation

Request URL (method GET)  
`http://localhost:8080/genesys/1/service/efef8eb61-1f24-593d-90da-0034aca34b55/storage`

### Result

```
{"Key2": "Value7", "Key1": "Value6", "Key3": "Value8"}
```

## Query One Key

This API queries one of the keys in a storage area that has already been created for the service.

Note: Introduced in 8.5.000.12.

### Operation

Method	GET		
URL	/genesys/1/service/{id}/storage/{key}		
Parameter	Type	Mandatory	Description
URI Parameters			
{id}	string	yes	The id of the service.
{key}	string	yes	The key of the specifically requested value.
<b>Body:</b> None			

### Response

If the key exists, returns 200 OK and the following JSON format value: **{"key4": "value4"}** (not the key value itself).

HTTP code	200
HTTP message	OK

Returns 404: Not Found if the key is not found in the user data.

HTTP code	404
HTTP message	Not Found

### Example

The following example queries the value of Key1 from the data associated with id efef8eb61-1f24-593d-90da-0034aca34b55.

### Operation

Request URL (method GET)  
http://localhost:8080/genesys/1/service/efef8eb61-1f24-593d-90da-0034aca34b55/storage/Key1

Request Method: GET

### Result

Value1

## Create or Update Service Storage

This API allows the creation of a new storage area or an update of the existing storage for a specific service. The TTL of the user data storage is the same TTL as the service.

## Multipart or Urlencoded

### Multipart or Urlencoded

#### Operation

Method	POST		
URL	/genesys/1/service/{id}/storage		
Parameter	Type	Mandatory	Description
URI Parameters			
{id}	string	yes	The id of the service.
<b>Body:</b> A MultiPart form or a URL encoded form consisting of different items representing the key/value pairs to store.			

#### Response

HTTP code	200
HTTP message	OK

### Example

The following example stores:

Key1, Key2, Key3, and FileKey

#### Operation

Request URL:  
http://localhost:8080/genesys/1/service  
/efef8eb61-1f24-593d-90da-0034aca34b55/storage

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
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.

HTTP 200 OK

## JSON

### JSON Body

Starting in 8.5.104, you can update storage data with JSON arrays.

Because ORS allows simple key/value parameters, GMS cannot submit complex JSON-structured values (such as maps and arrays) as ORS request parameters and convert complex values to strings.

For example:

```
{
  "address":{"street=mystreet, city=mycity}", // for map
  "phones":["9002@SIP_Switch, 5555666]" // for array
}
```

### Important

This query cannot handle binary values. To perform a request with an image, use the multipart/form API query.

## Operation

Method	POST		
URL	/genesys/1/service/{id}/storage		
Parameter	Type	Mandatory	Description
URI Parameters			
{id}	string	yes	The id of the service.
Header Parameters	'Content-type: application/json;charset=UTF-8		
<b>Body:</b> The body is a JSON structure representing the key/value pairs.			
<ul style="list-style-type: none"><li>• Keys that begin with "_" are ignored for storage.</li><li>• No binary values.</li><li>• Complex JSON-structured values (map and arrays) are converted to strings.</li></ul>			
For example:			
<pre>{   "lastname" : "mylastname",   "address": {     "street": "mystreet",     "city": "mycity"   },   "phones": [     "9002@SIP_Switch",     "5555666",     "7775666"   ],   "_key": "ignored" }</pre>			

## Response

<b>HTTP code</b>	200
<b>HTTP message</b>	OK
<b>HTTP code</b>	500
<b>HTTP message</b>	Server Error
Body	{"message": "Could not read JSON: Unexpected character ('\"' (code 34)): [...]"} 

## Query Binary (One Binary Key)

This API queries one of the binary keys in a storage area that has already been created for the service.

Note: Introduced in 8.5.002.02

## Operation

<b>Method</b>	GET		
<b>URL</b>	/genesys/1/service/{id}/storage/binary/{key}		
<b>Parameter</b>	<b>Type</b>	<b>Mandatory</b>	<b>Description</b>
<b>URI Parameters</b>			
<b>{id}</b>	string	yes	The id of the service.
<b>{key}</b>	string	yes	The key of the specifically requested value.
<b>Body:</b> None.			

## Response

<b>HTTP code</b>	200
<b>HTTP message</b>	OK
<b>HTTP code</b>	404
<b>HTTP message</b>	Not Found, if the binary key does not exists in user storage.

## Example

The following example queries the value of myBinaryKey from the data associated with id efef8eb61-1f24-593d-90da-0034aca34b55.

## Operation

Request URL (GET Method)  
http://localhost:8080/genesys/1/service  
/efef8eb61-1f24-593d-90da-0034aca34b55/storage/binary/myBinaryKey

## Result

Binary stream content and its content type.

## Delete One Key Storage

This API deletes one of the keys (either binary or non-binary) in a storage area that has already been created for the service.

Note: Introduced in 8.5.002.02

### Operation

Method	DELETE		
URL	/genesys/1/service/{id}/storage/{key}		
Parameter	Type	Mandatory	Description
URI Parameters			
{id}	string	yes	The id of the service.
{key}	string	yes	The key to be deleted.
<b>Body:</b> None.			

### Response

HTTP code	200
HTTP message	OK

### Example

The following example deletes the value of Key1 from the data associated with id efef8eb61-1f24-593d-90da-0034aca34b55.

### Operation

Request URL:  
`http://localhost:8080/genesys/1/service/efef8eb61-1f24-593d-90da-0034aca34b55/storage/Key1`

Request Method: DELETE

### Result

OK

## Delete All Keys in Storage

This API deletes all keys (binary or non-binary) in a storage area that has already been created for the service.

Note: Introduced in 8.5.002.02

## Operation

<b>Method</b>	DELETE		
<b>URL</b>	/genesys/1/service/{id}/storage		
<b>Parameter</b>	<b>Type</b>	<b>Mandatory</b>	<b>Description</b>
<b>URI Parameters</b>			
<b>{id}</b>	string	yes	The id of the service.
<b>Body:</b> None.			

## Response

<b>HTTP code</b>	200
<b>HTTP message</b>	OK

## Example

The following example deletes all of the keys from the data associated with id efef8eb61-1f24-593d-90da-0034aca34b55.

## Operation

Request URL:  
http://localhost:8080/genesys/1/service  
/efef8eb61-1f24-593d-90da-0034aca34b55/storage

Request Method: DELETE

## Result

OK

## Delete Service

This API deletes the service that was created and the storage area associated with it.

## Operation

<b>Method</b>	DELETE		
<b>URL</b>	/genesys/1/service/{id}		
<b>Parameter</b>	<b>Type</b>	<b>Mandatory</b>	<b>Description</b>
<b>URI Parameters</b>			

---

Method	DELETE		
{id}	string	yes	The id of the service.
<b>Body:</b> None.			

## Response

HTTP code	200
HTTP message	OK

## Example

The following example deletes the service id efef8eb61-1f24-593d-90da-0034aca34b55 and all the keys from the data associated with it.

## Operation

Request URL:  
`http://localhost:8080/genesys/1/service/efef8eb61-1f24-593d-90da-0034aca34b55`

Request Method: DELETE

## Result

OK

# Calendar Service API

## Overview

The Calendar Service API queries for all office hours so that your mobile app can make an intelligent first offer to the user based on what's available in the next couple days, or with an explanatory hint such as "Our offices are open from 8:00 am to 5:00 pm, please enter the desired time."

The calendar service API is intended to be tight to your business hours. It lets you handle the following use cases.

- Find whether the office is open at the time of the query.
  - If you call the API without any parameters, the start and end parameters are assumed to be now.
  - Check in the response that the field error is null. If the list of periods is empty, it means that the office is closed, otherwise the office is open.
- Find the office schedule for a given period.
  - If you call the API and set the number-of-days-ahead\_days parameter to 5, the start date is assumed to be now. The response should contain any periods when the office is open.

## Configuration

Make sure to update the existing calendar configuration to set the correct timezone for your business-hours service. For instance, if you configured "EST", or "PST" timezones with the Genesys Administrator Extension, your parameters must use the timezones defined for Java such as "America/Toronto", or "Europe/Paris". See [Wikipedia](#) to get the list of correct timezones.

1. Start Genesys Administrator or Configuration Manager.
2. Edit your GMS application's properties and select the **Options** tab.
3. Select your business hours service; for instance, service.business-hours (that is **service.{service-execution-name}**).
4. Update the \_timezone property.

```
[service.business-hours]
_bh_add1=07-14 12:00-14:30
_bh_regular1=Mon-Fri 01:00-23:00
_service=office-hours
_timezone=Europe/Paris
_ttl=30
_type=builtin
```



## REST API

The Calendar Service REST query retrieves the office hours per day for a provided period.

### Calendar Service REST Request

GET /genesys/1/service/{service-execution-name}			
Parameter	Type	Mandatory	Description
Body			
start	Date	no	Start date is specified in "ISO 8601" format, using UTC as timezone: "YYYY-MM-ddTHH:mm:ssZ". If not specified then it is assumed to be now.
number-of-days	integer	no	Specifies how many days ahead to include into interval beginning from the "start" time. Used as an alternative to the end date. If neither "end", nor "number-of-days" is specified, then end date is assumed to be the same as start date.
end	Date	no	End date is specified in "ISO 8601" format, using UTC as timezone: "YYYY-MM-ddTHH:mm:ssZ". If neither "end", nor "number-of-days" is specified, then end date is assumed to be the same as start date.

#### Important

Other options from configuration section can be specified in the request.

Request example: GET `http://127.0.0.1:8080/genesys/1/service/business-hours?start=2014-12-03T15:00:00.000Z#ber-of-days=2`

### Calendar Service REST Response

The Calendar service returns an array of periods ordered in ascending chronological order. If the array

is empty then there are no calendar events for the specified duration. If an error occurs while processing the request then the error is specified in the "error" data field, the contents of periods array, in that case, are undefined.

- Additional calendar information such as agent availability is returned in tags data field as a key-value map.
- The open\_for property reports how long the office will stay open in hours and minutes. Its format is "hh:mm". Note that the value of the open\_for property does not depend on any of the input parameters. It returns the duration between the time at which the request was issued and the office closing time. If the office is closed, then open\_for = "00:00".

```
> GET /genesys/1/service/business-hours?start=2016-10-05T15:00:00.000Z#ber-of-days=2 HTTP/1.1
> Host: 127.0.0.1:8080
> User-Agent: curl/7.50.3
> Accept: */*
>
< HTTP/1.1 200 OK
< Set-Cookie: JSESSIONID=1wftsntzmfydfybt0v75gbprp;Path=/genesys;HttpOnly
< Expires: Thu, 01 Jan 1970 00:00:00 GMT
< Content-Type: application/json; charset=UTF-8
< Transfer-Encoding: chunked
<
{
  "error": null,
  "open_for": "11:20",
  "periods": [
    {
      "start": "2016-10-05T15:00:00.000Z",
      "end": "2016-10-05T21:00:00.000Z"
    },
    {
      "start": "2016-10-05T23:00:00.000Z",
      "end": "2016-10-06T21:00:00.000Z"
    },
    {
      "start": "2016-10-06T23:00:00.000Z",
      "end": "2016-10-07T15:00:00.000Z"
    }
  ]
}
```

# Capacity API

The Capacity Service enables you to define the number of scheduled callbacks that are allowed for Callback for a given time slot in the week. Then, your Callback service refers to your Capacity service and to your Office Hours service to adjust the agent availability and the number of scheduled callbacks.

You can define exceptions for dates where less or more agents are available, and you can define as many Capacity services that you need to match your Callback services. This way, you can address the agent workload according to the real resources that are available.

## Configuration

Here is an example of GMS configuration:

```
[service.Capacity]
_capacity=[Mon, Tue, Wed, Thu, Fri, Sat, Sun]
_capacity_1={"1":{"0000":1,"0100":3}}
_capacity_2={"2":{"0100":3}}
_capacity_3={"3":{"1000":6,"1100":6,"1200":6,"1300":6,"1400":6,"1500":6,"1600":6,"1700":6,"1800":6,"1900":6,"2000":6}}
_capacity_4={"4":{"1100":3,"0100":3}}
_capacity_5={"5":{"0100":3}}
_capacity_6={"6":{"0100":3}}
_capacity_7={"7":{"0000":1,"0100":3}}
_capacity_add={}
_service=capacity
_timezone=UTC
_type=builtin
```

### Important

You can also configure a **capacity service** in the Service Management UI.

## API

When you add a **capacity service** through the **Configured Services** interface, you can access the capacity service by its execution name and retrieve content through the API.

In order to find the available capacity, the query provides a date and time range. The response object provides a list of time periods and available capacity for each period. Here is a list of input parameters:

<b>GET /genesys/1/service/{capacity-execution-name}?start=&lt;date and time&gt;&amp;end=&lt;date and time&gt;&amp;timezone=&lt;timezone&gt;</b>		
<b>Name</b>	<b>Description</b>	<b>Required</b>
start	yes	Start date and time in ISO-8601 format without a timezone component; for example: 2015-03-17T09:15:00.000
end	yes	End date and time in ISO-8601 format without a timezone component; for example: 2015-03-17T12:45:00.000
timezone	yes	Timezone used for "start" and "end" parameters; for example: America/Los_Angeles

For example, the following request will provide the following results:

```
GET /genesys/1/service/Capacity?start=2016-10-05T15:00:00.000Z#ber-of-days=2 HTTP/1.1
> Host: 127.0.0.1:8080
> User-Agent: curl/7.50.3
> Accept: */*
>
< HTTP/1.1 200 OK
< Set-Cookie: JSESSIONID=1opza084c3vszo631xnrtqw38;Path=/genesys;HttpOnly
< Expires: Thu, 01 Jan 1970 00:00:00 GMT
< Content-Type: application/json; charset=UTF-8
< Transfer-Encoding: chunked
<
{
  "error": null,
  "slots": [
    {
      "utcTime": "2016-10-05T15:00:00.000Z",
      "localTime": "2016-10-05T15:00:00.000Z",
      "durationMin": 540,
      "capacity": 6
    },
    {
      "utcTime": "2016-10-06T01:00:00.000Z",
      "localTime": "2016-10-06T01:00:00.000Z",
      "durationMin": 60,
      "capacity": 3
    },
    {
      "utcTime": "2016-10-06T11:00:00.000Z",
      "localTime": "2016-10-06T11:00:00.000Z",
      "durationMin": 60,
      "capacity": 3
    }
  ],
  "timezone": "UTC"
}
```

# Callback Services API

**Modified in:** 8.5.2

## Getting Started

When you add a **callback service**, you define a **Service Name**, which is referred to as {callback-execution-name} in this API documentation. Each time that you perform a callback query, you must specify the {callback-execution-name} in the URI parameters.

## Accessing your Callback Service

The URLs used by the Callback API are dependent on the execution name of the Callback service that you have just created. Callback services are available at the following URL:

```
http://<host>:<port>/genesys/1/service/callback/{callback-execution-name}
```

For instance, if you create a callback service named callback-for-mobile, then {callback-execution-name} is callback-for-mobile, its configuration in GMS is located in the service.callback-for-mobile section, and you can access the callback service at the following URL:

```
http://<host>:<port>/genesys/1/service/callback/callback-for-mobile
```

## Overwriting Configuration in Queries

To overwrite service configuration parameters in your POST REST queries (Start-Callback), use the `_overwritable_options` option. This option lets you define a list of overwritable parameters that you will be able to pass in the Body of your REST request.

### Important

This list can include the `_ors` and `_target` options only.

For example, if you set:

```
_overwritable_options = _ors,_target
```

Then, you can pass `_ors` and `_target` in your REST query:

```
POST /1/service/callback/callback-for-mobile
{
  "_ors": "http://myors:4421",
```

```
  "_target": Billing@Stat_Server1.GA
}
```

## Passing Configuration Tokens in Queries

### **Added in: 8.5.104**

In your service configuration, you can create token variables that can be used in other configuration parameters. Then, at runtime, you can pass values for these tokens in POST REST queries (Start-Callback) and these values will be used to modify your configuration.

CBCK terminated preview

Search Table Select All

+ Add New Delete

Name	Value	Description
_customer_number		Request match USEROR customer request pa
_service	callback	
<input type="checkbox"/> _type	ors	
<input type="checkbox"/> my_token_name	\$my_token\$	

1 Create a token variable

2 Use the token in your configuration:

URS Queueing (1)

☒ \_target Billing@\$my\_token\$.GA

To create a token variable, create a new service parameter and configure its value with a string matching the following format: `$<any-token-name>$`

For instance, create:

```
my_token_name = $my_token$
```

Then, you can use the body parameter `my_token=<anyvalue>` in your REST queries. As a result, the occurrences of `$my_token$` in this service configuration will be replaced with the query's provided value.

For example, if you wish to create a callback request for the CLBCK-terminated-preview service using the Stat\_Server1 server target, use the following query:

```
POST /genesys/1/service/callback/CLBCK-terminated-preview
HTTP/1.1
Host: 127.0.0.1:8080
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
_customer_number=01822256&my_token=Stat_Server1
```

When GMS receives `my_token=Stat_Server1` in the query information, it replaces the `$my_token$` placeholder with `Stat_Server1` everywhere that it is used in the configuration of CLBCK-terminated-preview. Using our example, the result would be:

```
_target = Billing@Stat_Server1.GA
```

### Tip

Use this feature to avoid duplicating configuration for multiple services that handle the same functionality, but use different queues or servers.

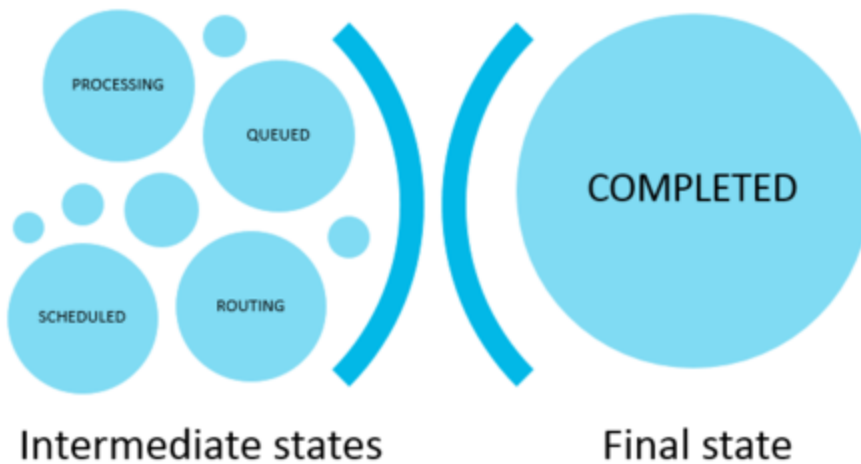
## Understanding Callback States





When the Callback request is submitted, it gets through several callback states and ORS handles some of these callback states while processing the associated callback interaction. You can access the callback status in the `_callback_state` parameter of the callback's JSON representation.

### Important

The `_callback_state` parameter is incompatible with the `_new_desired_time` property.





Callback states	While in ORS	Description
PROCESSING		The customer is connected to an agent and talking with this agent.
QUEUED		The callback is actively waiting for an agent in ORS/URS; the agent is not assigned yet.
SCHEDULED		The Callback service handles the callback (there are no sessions started in ORS). While in this state, the request is handled by the callback service running in GMS until the specified <code>desired_time</code> is approaching.
ROUTING		Customer phone is reached and waiting for an agent.
COMPLETED		The call has ended and the Callback is completed with the reason specified in <code>_callback_reason</code> .
PAUSED		The call is paused. See <a href="#">Pausing Callback</a> for details.

### Callback reasons in COMPLETED State

You can get the following reasons in the `_callback_reason` parameter when receiving the COMPLETED state.

**ABANDONED\_IN\_QUEUE**

The Callback interaction was deleted prior to routing the interaction to the agent because the customer abandoned.

**AGENT\_CONNECTED**

Callback Service successfully routed the interaction to the agent.

**AGENT\_PREVIEW\_CANCEL**

The agent canceled the callback preview request. To get this state reason, create an Agent First Preview service and configure the following options with the following values, for example: `_agent_preview=true`, `_agent_preview_allow_reject=3`, `_agent_preview_set_notready_reason='Coffee Break'`, `_agent_preview_set_notready_reason_attribute=false`, `_agent_preview_set_notready_reason_key='ReasonCode'`, `_agent_preview_timeout_set_notready=true`, `_agent_preview_via_rp=false`

**AGENT\_PREVIEW\_CANCEL\_AFTER\_<n>REJECTS**

The agent rejected the request '<n>' times.

**AM\_CONNECTED**

Callback Service successfully routed the interaction to the answering machine.

**CANCELLED**

Callback Service received a cancel request for this callback.

**CANCELLED\_BY\_ADMIN**

Callback Service received a cancel request from the Service Management UI for this callback.

**FAIL\_AGENT\_CONNECT**

The Callback interaction could not be connected to the agent. This error may happen when the value of `_max_time_to_wait_for_agent_on_the_call` is too short.

**FAIL\_CALL\_TO\_CUSTOMER**

Replaces `FAIL_USER_UNREACHABLE` since GMS 8.5.102.14. Callback Service could not connect the customer.

**FAIL\_ERROR**

Callback Service failed due to an unknown error.

**FAIL\_FAX\_REACHED**

Callback Service could not connect the customer. The provided number was answered by a fax machine.

**FAIL\_INBOUND\_TIMEOUT**

The customer did not make the call within the expected `_booking_expiration_timeout` period defined for User-Originated scenarios.

**FAIL\_INCORRECT\_CONFIG\_MEDIA\_TYPE**

The `_media_type` option is set to an incorrect value. Callback Service only processes voice and chat interactions.

**FAIL\_INTERACTION\_DELETED**

The callback interaction was deleted prior to routing the interaction to the agent. This error may happen when `_wait_for_agent=true` and the agent hung up the call.

**FAIL\_I\_XN\_UNKNOWN\_MEDIA\_TYPE**

The media type of the interaction is not supported by Callback Service. Callback Service only processes voice and chat interactions.

**FAIL\_LOAD\_MESSAGE\_FILE**

Callback Service cannot load the strings resource file specified in the `_notification_message_file` option.

**FAIL\_NO\_CUSTOMER\_NUMBER**

Customer number is missing.

**FAIL\_QUEUEING**

The Callback request could not be queued. This error may happen when an error occurs while requesting the route delay to URS.

**FAIL\_TARGET\_NOT\_FOUND**

Callback Service cannot reserve the requested target to handle the request. This error may happen when the value of `_urs_queued_ttl` is too short.

**FAIL\_TIMEOUT\_TTL**

Callback Service did not manage to handle the request in the specified time (`_ttl`).

**FAIL\_USER\_NO\_CONFIRM**

The user confirmation was not received although it was required; this issue can occur if `_on_user_confirm_timeout` is not set to `CONNECT-ANYWAY`.

**FAIL\_USER\_UNREACHABLE**

Reported as `FAIL_CALL_T0_CUSTOMER` prior to GMS 8.5.102.14.

**NOT\_AVAILABLE**

Callback Service exited with no specified reason.

**SUBMIT\_ERROR**

GMS did not manage to submit the Callback service request to Orchestration Server for processing.

## List of API Queries

The Callback Services API provides the following REST queries:

- **Start or Schedule Callback**—Initiate a Callback request.
- **Cancel-Callback**—Cancel a Callback request.
- **Delete-Callback**—Delete a Callback request.
- **Reschedule-Callback**—Reschedule a Callback request.
- **Query-Availability**—Get the availability for a new callback request.

- [Query-Callback-By-Id](#)—Query a callback by its ID.
- [Query-Callback by Lookup Properties](#)—Query outstanding callbacks by lookup properties.
- [Query-Callback by Queue\(s\)](#)—Query outstanding callbacks by queue(s).
- [Query Counter Watermarks](#)—Query the current set of executed callback instances in queues.
- [Export Callback Records](#)—Export Callback records.

### Important

The documentation for "Query-EWT for Virtual Queues" was moved to the [Stat Service API](#) page.

## Start or Schedule Callback

Initiates a callback request. It validates the request by doing the following:

- Checks parameters, in general (in particular, if the target queue is valid).
- Checks the customer number against exceptions.
- Checks the time criteria of the request against the business.
  - If invalid:
    - Returns the appropriate error.
    - Sends a reporting event to the GMS data manager indicating that the callback request has been rejected.
  - If valid:
    - Creates a unique ID for the request.
    - Sends a reporting event to the GMS data manager indicating that the callback request has been accepted and started.  
This event also indicates the state of the request (immediate or scheduled).
    - If the request needs to be scheduled for a later date/time, the request and its associated data will be stored in the module persistent data storage.
    - If the request can be started now, an ORS session is initiated using the associated SCXML-based service with this particular callback request.  
Note: the provisioned data for the execution service to be started will be used as input along with the input parameters from the request itself.
    - Returns the ID generated for this request.

Starting in 8.5.2, you can redial a COMPLETED callback by submitting the callback ID to create a copy of this callback. The properties and user data of the copied callback are merged with the parameters of the new callback submitted in the POST query.

- The parameters specified in the POST query override the copied properties.

- Internal retry flags and properties such as `_callback_state`, `_ors_session_id`, `_desired_time` will be ignored when creating the callback copy.

### Tip

You can include any of the `_xxx` callback option parameters in your start query if they are not configured in the service; for example `_target`, `_wait_for_agent`, `_paused_services_list`, `_paused_services_id`, or any other `_xxx` parameter listed in the [Callback Service Options Reference Guide](#). If the option is already configured in the service, the query parameter's value is ignored and the service option value is used. See [Overwriting Configuration in queries](#) to learn about overwriting configuration in queries.

## POST /genesys/1/service/callback/{callback-execution-name}

Initiates a callback request.

### Header

Content-type	application/json multipart/form-data application/x-www-form-urlencoded
--------------	--

### URI Parameters

Name	Type	Description
<b>callback-execution-name</b> *required	string <i>path</i>	Name of the callback execution service provisioned in GMS.

### Body (JSON content)

<code>_customer_number</code> *required	string	Number to call back.  This parameter can also be replaced by any parameter specified in the option <code>_mandatory_customer_lookup_keys</code> (comma-separated list of attributes) that can identify a unique customer.
<code>_copy_from_id</code> Introduced in 8.5.2	string	ID of a Callback in COMPLETED state. The properties and user data of this completed callback are copied in the new callback and use for redial.  <ul style="list-style-type: none"> <li>• Properties specified in the POST request will override copied properties.</li> </ul>

		<ul style="list-style-type: none"> <li>The following properties and internal retry flags will be excluded from the copy:</li> </ul>
<code>_desired_time</code>	string	<p>Desired time to have the callback. By default, the desired time is the current time.</p> <p>This option format is ISO 8601 "yyyy-MM-ddTHH:mm:ss.SSSZ" such as "2013-05-28T15:30:00.000Z"</p> <p><b>Note:</b> The Callback is an <i>immediate</i> Callback based on the following rule:  <code>immediate = _desired_time &lt; {current_time} + option(_request_execution_time_buffer) + computed(option(_request_queue_time_stat))</code></p> <p>Note that the desired time is a time, not a duration.</p> <p>Additional examples:</p> <ul style="list-style-type: none"> <li><code>_desired_time</code> is in 1h, <code>_request_execution_time_buffer=300</code> (5min), statistic set is "EstimatedWaitTime" returning, for example, 10min  then the Callback is not immediate and will be submitted later for execution because  <code>immediate = today 16h &gt; today 15h + 5 min + 10 min = false</code></li> <li><code>_desired_time</code> is in 5min, <code>_request_execution_time_buffer=120</code> (2min), statistic set is "EstimatedWaitTime" returning, for example, 5min  then the Callback is immediate and is submitted for execution because  <code>immediate = today 16h05 &gt; today 16h + 2 min + 5 min = true</code></li> </ul>
<code>&lt;property&gt;</code>	string	<p>Any properties key/values to be attached. Key/Values may be used in Orchestration execution service. Keys without an</p>

		underscore prefix are User Attached Data.
_callback_state	string	<p>Forces creation of Callback in a specified state.</p> <p><b>Important:</b> This is for advanced users that handle</p> <p>Callback life-cycle externally to GMS. By default, the _callback_state value is either QUEUED or SCHEDULED depending if the Callback is processed as immediate or scheduled (respectively).</p>
_urs_virtual_queue	string	Queue to use for this callback if several virtual queues are used for callback with identical configuration.
_request_queue_time_stat	string	<p>Queue statistics. For example, "ExpectedWaitTime;Queue;8999@SIP_Server;En</p> <p>Note: If the _request_queue_time_stat option is configured in the Callback service, the request parameter is ignored.</p>

## Responses

Name	Description
<b>200 OK</b>	
<b>Response Body</b> (JSON content)	
_id <i>required</i>	The service id for which a successful callback request was registered.
ID <i>only for immediate callback</i>	Dialog Event ID
Action <i>only for immediate callback</i>	Dialog Action.
Text <i>only for immediate callback</i>	Text to display
OkTitle <i>only for immediate callback</i>	Label for the OK button.

Name	Value
<b>500 Internal Server Error</b>	

Name	Value
<b>Response body (JSON Content)</b>	
code	50006
phrase	ORS_MAX_SUBMIT_RETRIES
message	"Callback {id} reached maximum attempts to submit to ORS reached ({max-attempts})"
exception	com.genesyslab.gsg.services.callback.CallbackExceptionMaxORSSubmitAttempts
properties	{ "id": "callback id", "max-attempts": <value for _max_ors_submit_attempts> }

Name	Value
<b>429 Too Many Requests</b>	
<b>Response body (JSON Content)</b>	
code	40001
phrase	NUMBER_ALREADY_BOOKED
message	"There is already {max_queued} or more Callbacks QUEUED for this number, please refer to _enable_in_queue_checking for detail."
exception	com.genesyslab.gsg.services.callback.CallbackExceptionAlreadyBooked
properties	{ "max_queued": <1 if _enable_in_queue_checking=strict or 2 if _enable_in_queue_checking=true> }

Name	Value
<b>429 Too Many Requests</b>	
<b>Response body (JSON Content)</b>	



Name	Value
code	40002
phrase	THROTTLE_SERVICE_LIMIT
message	"Limit of queued callbacks for {service} is reached."
exception	com.genesyslab.gsg.services.callback.CallbackExceptionThrottled
properties	{ "service": <service name> }

Name	Value
<b>429 Too Many Requests</b>	
<b>Response body (JSON Content)</b>	
code	40003
phrase	THROTTLE_SERVICE_INTERVAL_LIMIT
message	"Limit of queued callbacks for {service} is reached for interval {interval}s."
exception	com.genesyslab.gsg.services.callback.CallbackExceptionThrottled
properties	{ "service": <service name>, "interval": <interval throttling limit> }

Name	Value
<b>429 Too Many Requests</b>	

Name	Value
<b>Response body (JSON Content)</b>	
code	40004
phrase	THROTTLE_SERVICE_PARAMETER_LIMIT
message	"Limit of queued callbacks for {service} is reached for parameter {parameter}. Reached {attempts} times today."
exception	com.genesyslab.gsg.services.callback.CallbackExceptionThrottled
properties	{ "service": <service name>, "parameter": <parameter triggering the throttling>, "attempts": <number of attempts reached> }

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40020
phrase	INVALID_OPERATION
message	<ul style="list-style-type: none"><li>"Request cannot be processed because callback {id} to copy is not COMPLETED. Check parameter _copy_from_id"</li></ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionInvalidOperation
properties	{ "id": <callback id>, "service": <service name>, "time": <ISO UTC time>, "state": <callback state>, "message": <ORS server's message>, "filter": <filtering expression>

---

Name	Value
	}

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40030
phrase	CALLBACK_NOT_FOUND
message	"Callback {id} to copy from cannot be found"
exception	com.genesyslab.gsg.services.callback.CallbackExceptionNotFound
properties	<pre>{   "id": &lt;callback service id&gt;,   "service": &lt;service name&gt;,   "time": &lt;ISO UTC time&gt; }</pre>

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40050, 40051
phrase	<ul style="list-style-type: none"><li>SLOT_UNAVAILABLE (40050)</li><li>SLOT_UNAVAILABLE_PROPOSAL(40051)</li></ul>
message	<ul style="list-style-type: none"><li>"No time slots available."</li></ul>

---

Name	Value
	<ul style="list-style-type: none"> <li>"Too many requests at desired time slot {slot}. Proposing time slots."</li> <li>"Office is closed at desired time slot {slot}. Proposing time slots."</li> </ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionAvailability
properties	<pre>{   "slot": &lt;ISO UTC time range&gt;,   "service": &lt;service name&gt; }</pre>

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50020
phrase	BAD_CONFIGURATION
message	<ul style="list-style-type: none"> <li>"Service option {service} / _default_country is not configured. But option _disallow_impossible_phone_numbers is set. We cannot validate phone numbers without knowing the country."</li> <li>"Service option {service} / _default_country is not configured. But option _disallow_premium_phone_numbers is set. We cannot validate phone numbers without knowing the country."</li> <li>"Unable to parse option: _request_queue_time_stat={statistic}"</li> <li>"Missing default_chat_endpoint option in chat section because this service has parameter _media_type=chat"</li> <li>"Missing default_client_timeout option in chat section because this service has parameter _media_type=chat"</li> </ul>

Name	Value
	<ul style="list-style-type: none"><li>"Option service.{service} / _business_hours_service not configured."</li><li>"Option _business_hours_service is invalid: {message}"</li><li>"Service undefined: {service}"</li><li>"Service {service} has unknown value for option _type"</li><li>"Service {service} has option _type != ors"</li><li>"Service {service} has option _service != callback"</li></ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionConfiguration
properties	<pre>{   "service": &lt;service name&gt; }</pre>

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50005
phrase	CALENDAR_ERROR
message	message returned by Calendar service
exception	com.genesyslab.gsg.services.callback.CallbackExceptionCalendarError
properties	

Name	Value
<b>500 Internal Server Error</b>	

---

Name	Value
<b>Response body (JSON Content)</b>	
code	50004
phrase	CAPACITY_ERROR
message	message returned by Capacity service
exception	com.genesyslab.gsg.services.callback.CallbackExceptionCapacityError
properties	

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50030
phrase	ORS_ERROR
message	<ul style="list-style-type: none"><li>"Invalid ORS response"</li><li>message returned by ORS strategy</li></ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionFromORS
properties	

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50040
phrase	SERVICE_REDIRECT_FAILED

---

Name	Value
message	message from redirected service
exception	com.genesyslab.gsg.services.callback.CallbackExceptionServiceRedirect
properties	

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40040
phrase	NUMBER_REJECTED
message	<ul style="list-style-type: none"><li>"Customer Number is not allowed, because it is invalid. Check option "Customer Number is not allowed, because it is invalid. Check option _disallow_impossible_phone_numbers"</li><li>"Customer Number is not allowed, because it's a premium number. Check option _disallow_premium_phone_numbers"</li><li>"Customer Number is not allowed, because it failed validating. Check option _disallow_impossible_phone_numbers"</li><li>"Customer Number is not allowed. Check option _exceptions"</li></ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionNumber
properties	

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50050

---

Name	Value
phrase	UNKNOWN_ERROR
message	"Error processing callback {message} "
exception	com.genesyslab.gsg.services.callback.CallbackExceptionUnknown
properties	{ "message": <message caught> }

## Example

```
POST http://localhost:8080/genesys/1/service/callback/request-callback
{
  "_customer_number": "5115",
  "usr_customer_name": "Bob Markel",
  "usr_reason": "billing question",
  "_device_notification_id":
  "b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673",
  "_device_os": "comet",
  "_desired_time": "2013-06-17T10:25:00.000Z"
}
```

## Result

```
200 OK
{
  "_id": "a550a12e-ca77-4146-98d0-58960e0939f7"
}
```

The result of this operation is different if the callback is immediate or schedule. If immediate, some information may be returned in response along with `service_id`.

```
200 OK
{
  "ID": "0",
  "Action": "ConfirmationDialog",
  "Text": "You will receive the call shortly",
  "OkTitle": "Ok",
  "_id": "361-58ce803e-362c-477f-8ac8-5bbc93f9acc7"
}
```

## Cancel-Callback

The Cancel-Callback API cancels a Callback request, by doing the following:

- Validates that the request is still in the queue.
  - If not, returns the appropriate error.



- If valid, removes the request from the scheduling queue.
- Checks the state of the Callback request:
  - If `_callback_state=QUEUED`, a callback cancel event is submitted to the execution service.
- Callback request is marked `_callback_state=COMPLETED` with `_callback_reason=CANCELLED`.

## DELETE /genesys/1/service/callback/{callback-execution-name}/{service\_id}

Cancels a Callback request

### URI Parameters

Name	Type	Description
<b>callback-execution-name</b> *required	string <i>path</i>	Name of the callback execution service of 'ors' type provisioned in GMS.
<b>service_id</b> *required	string <i>path</i>	This is the service id returned from the initial start callback response.
discard_ors_failure	boolean	False by default. If true, GMS can bypass ORS failures and marks the cancellation of the callback.  Set this option to true to manage troubleshoot cases that happen if the callback session is exited in ORS while the record is not marked as COMPLETED in GMS.

### Responses

#### 200 OK

No JSON Body

Name	Value
400 Bad Request	
Response body (JSON Content)	
code	40010
phrase	BAD_PARAMETER

Name	Value
message	<ul style="list-style-type: none"> <li>Generic parser exception message: Typically, a bad date parsing may fall there as a bad parameter error with the appropriate statement.</li> <li>Generic missing parameter exception message (case of controller level detection).</li> </ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionBadParameter
properties	<pre>{   "id": &lt;callback id&gt;,   "keys": &lt;missing lookup key&gt;,   "day": &lt;specified day value&gt;,   "properties": &lt;lookup properties&gt;,   "option": &lt;invalid option key&gt; }</pre>

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40020
phrase	INVALID_OPERATION
message	<ul style="list-style-type: none"> <li>"Callback {id} does not contain _desired_time property."</li> <li>"Callback {id} cannot be cancelled or completed - _callback_state={_callback_state}"</li> <li>"Callback {id} cannot be cancelled - unable to process ORS cancel request : {message} "</li> <li>"Callback {id} cannot be cancelled - No ORS session found. ( _callback_state=QUEUED while _ors_session_id=null?)"</li> <li>"Rejecting update : {service}=[{id} @ {time}] -</li> </ul>

Name	Value
	reached state COMPLETED"
exception	com.genesyslab.gsg.services.callback.CallbackExceptionInvalidOperation
properties	<pre>{   "id": &lt;callback id&gt;,   "service": &lt;service name&gt;,   "time": &lt;ISO UTC time&gt;,   "state": &lt;callback state&gt;,   "message": &lt;ORS server's message&gt;,   "filter": &lt;filtering expression&gt; }</pre>

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40030
phrase	CALLBACK_NOT_FOUND
message	<ul style="list-style-type: none"><li>"Callback {id} cannot be found"</li><li>"Callback {id} cannot be found - {service}= [{id} @ {time}]"</li></ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionNotFound
properties	<pre>{   "id": &lt;callback service id&gt;,   "service": &lt;service name&gt;,   "time": &lt;ISO UTC time&gt; }</pre>

Name	Value
500 Internal Server Error	
<b>Response body (JSON Content)</b>	
code	50030
phrase	ORS_ERROR
message	<ul style="list-style-type: none"><li>"Invalid ORS response"</li><li>message returned by ORS strategy</li></ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionFromORS
properties	

Name	Value
500 Internal Server Error	
<b>Response body (JSON Content)</b>	
code	50050
phrase	UNKNOWN_ERROR
message	"Error processing callback {message} "
exception	com.genesyslab.gsg.services.callback.CallbackExceptionUnknown
properties	{ "message": <message caught> }

## Examples

```
DELETE http://localhost:8080/genesys/1/service/callback/BasicCallback/a550a12e-  
ca77-4146-98d0-58960e0939f7  
Result 200 OK
```

```
DELETE http://localhost:8080/genesys/1/service/callback/BasicCallback/a550a12e-  
ca77-4146-98d0-58960e0939f7
```

Result 400 Bad Request

```
{
  "message": "No such request to cancel : [a550a12e-ca77-4146-98d0-58960e0939f7]",
  "exception": "com.genesyslab.gsg.services.callback.CallbackException"
}
```

DELETE http://localhost:8080/genesys/1/service/callback/callback-test/361-cf088d4e-88ab-452c-ac1f-39086cc96cbe

Result 400 Bad Request

```
{
  "message": "Request already cancelled or completed : [361-cf088d4e-88ab-452c-ac1f-39086cc96cbe]",
  "exception":
"com.genesyslab.gsg.services.callback.exceptions.CallbackExceptionInvalidOperation"
}
```

If you set `discard_ors_failure=true`, the previous query will get a 200 OK response, though the error will be logged as an error in ORS.

DELETE http://localhost:8080/genesys/1/service/callback/callback-test/61-cf088d4e-88ab-452c-ac1f-39086cc96cbe?discard\_ors\_failure=true

Result 200 OK

## Reschedule-Callback

The Reschedule-Callback API changes various input parameters associated with a given callback service. This request will have the Callback request id that is to be updated. This API does the following:

- Validates that the request is still in the scheduling queue.
  - If not, returns the appropriate error.
  - If valid, updates the request in the scheduling queue.

Note: The Reschedule operation is available only for requests where `_callback_state=SCHEDULED`.

**PUT /genesys/1/service/callback/{callback-execution-name}/{service\_id}**

Reschedules a Callback request

### Header

Content-type	application/json
	multipart/form-data
	application/x-www-form-urlencoded

### URI Parameters

Name	Type	Description
<b>callback-execution-name</b> *required	string <i>path</i>	Name of the callback execution service of 'ors' type provisioned in GMS.

<b>service_id</b> *required	string <i>path</i>	This is the service id returned from the initial start callback response.
<b>Body (JSON content)</b>		
_new_desired_time	string	The new time for which to reschedule the callback.  If provided and validated through office-hours, _callback_state will be automatically switched to "scheduled" or "immediate", discarding _callback_state property.
_callback_state	string	Possible values are SCHEDULED, QUEUED, ROUTING, PROCESSING, COMPLETED.  Note: The _new_desired_time parameter triggers the re-schedule operation, discarding the _callback_state parameter.
<other properties>	any	Properties to be updated in request.

## Responses

### 200 OK

No JSON Body

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40010
phrase	BAD_PARAMETER
message	<ul style="list-style-type: none"> <li>"Callback {id} does not contain the mandatory customer lookup keys {keys}"</li> <li>"Callback {id} does not contain _desired_time property."</li> <li>"Callback {id} contains _desired_time property in the past (-%ds &lt; %ds &lt; %ds) - epoch %ds"</li> <li>"Callback request contains _desired_time property too far in future (-%ds &lt; %ds &lt; %ds) -</li> </ul>

Name	Value
	<p>epoch %ds"</p> <ul style="list-style-type: none"><li>"Cannot create service, missing mandatory callback option {option}"</li><li>"Cannot create service, empty mandatory callback option {option}"</li><li>Generic parser exception message: Typically, a bad date parsing may fall there as a bad parameter error with the appropriate statement.</li><li>Generic missing parameter exception message (case of controller level detection).</li></ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionBadParameter
properties	<pre>{   "id": &lt;callback id&gt;,   "keys": &lt;missing lookup key&gt;,   "day": &lt;specified day value&gt;,   "properties": &lt;lookup properties&gt;,   "option": &lt;invalid option key&gt; }</pre>

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40020
phrase	INVALID_OPERATION
message	<ul style="list-style-type: none"><li>"Invalid service stored for callback {id}."</li><li>"Request cannot be processed because callback {id} to copy is not COMPLETED. Check parameter _copy_from_id"</li><li>"Callback {id} is no longer scheduled. State={state}"</li><li>"Callback {id} has invalid desired time stored."</li></ul>

Name	Value
	<ul style="list-style-type: none"> <li>"Rejecting update : {service}=[{id} @ {time}] - reached state COMPLETED"</li> </ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionInvalidOperation
properties	<pre>{   "id": &lt;callback id&gt;,   "service": &lt;service name&gt;,   "time": &lt;ISO UTC time&gt;,   "state": &lt;callback state&gt;,   "message": &lt;ORS server's message&gt;,   "filter": &lt;filtering expression&gt; }</pre>

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40030
phrase	CALLBACK_NOT_FOUND
message	<ul style="list-style-type: none"> <li>"Callback {id} cannot be found"</li> <li>"Callback {id} cannot be found - {service}=[{id} @ {time}]"</li> </ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionNotFound
properties	<pre>{   "id": &lt;callback service id&gt;,   "service": &lt;service name&gt;,   "time": &lt;ISO UTC time&gt; }</pre>



Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40050, 40051
phrase	<ul style="list-style-type: none"><li>SLOT_UNAVAILABLE (40050)</li><li>SLOT_UNAVAILABLE_PROPOSAL(40051)</li></ul>
message	<ul style="list-style-type: none"><li>"No time slots available."</li><li>"Too many requests at desired time slot {slot}. Proposing time slots."</li><li>"Office is closed at desired time slot {slot}. Proposing time slots."</li></ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionAvailability
properties	<pre>{   "slot": &lt;ISO UTC time range&gt;,   "service": &lt;service name&gt; }</pre>

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50020
phrase	BAD_CONFIGURATION
message	<ul style="list-style-type: none"><li>"Service option {service} / _default_country is not configured. But option _disallow_impossible_phone_numbers is set. We</li></ul>

Name	Value
	<p>cannot validate phone numbers without knowing the country."</p> <ul style="list-style-type: none"> <li>"Service option {service} / _default_country is not configured. But option _disallow_premium_phone_numbers is set. We cannot validate phone numbers without knowing the country."</li> <li>"Unable to parse option: _request_queue_time_stat={statistic}"</li> <li>"Missing default_chat_endpoint option in chat section because this service has parameter _media_type=chat"</li> <li>"Missing default_client_timeout option in chat section because this service has parameter _media_type=chat"</li> <li>"Option service.{service} / _business_hours_service not configured."</li> <li>"Option _business_hours_service is invalid: {message}"</li> <li>"Service undefined: {service}"</li> <li>"Service {service} has unknown value for option _type"</li> <li>"Service {service} has option _type != ors"</li> <li>"Service {service} has option _service != callback"</li> </ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionConfiguration
properties	<pre>{   "service": &lt;service name&gt; }</pre>

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50005

---

Name	Value
phrase	CALENDAR_ERROR
message	message returned by Calendar service
exception	com.genesyslab.gsg.services.callback.CallbackExceptionCalendarError
properties	

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50004
phrase	CAPACITY_ERROR
message	message returned by Capacity service
exception	com.genesyslab.gsg.services.callback.CallbackExceptionCapacityError
properties	

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50030
phrase	ORS_ERROR
message	<ul style="list-style-type: none"><li>"Invalid ORS response"</li><li>message returned by ORS strategy</li></ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionFromORS

---

---

Name	Value
properties	

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40040
phrase	NUMBER_REJECTED
message	<ul style="list-style-type: none"><li>"Customer Number is not allowed, because it is invalid. Check option "Customer Number is not allowed, because it is invalid. Check option _disallow_impossible_phone_numbers"</li><li>"Customer Number is not allowed, because it's a premium number. Check option _disallow_premium_phone_numbers"</li><li>"Customer Number is not allowed, because it failed validating. Check option _disallow_impossible_phone_numbers"</li><li>"Customer Number is not allowed. Check option _exceptions"</li></ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionNumber
properties	

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50050
phrase	UNKNOWN_ERROR
message	"Error processing callback {message} "

---

Name	Value
exception	com.genesyslab.gsg.services.callback.CallbackExceptionUnknown
properties	{ "message": <message caught> }

## Examples

### Successful Rescheduling

```
PUT http://localhost:8080/genesys/1/service/callback/
BasicCallback/a550a12e-ca77-4146-98d0-58960e0939f7
{
  "_new_desired_time": "2013-05-27T15:05:00.000Z"
}
Result
200 OK
```

### Failed Rescheduling

```
PUT http://localhost:8080/genesys/1/service/callback
/callback-test/361-d61e636da-3109-436c-877e-8d7174277bb9
{
  "_new_desired_time": "2014-07-22T10:00:00.000Z"
}
Result
400 Bad Request
{
  "message": "Callback '361-738dadcb-9d20-4557-8e24-fddb82f9c1b8'
is no longer scheduled. State=PROCESSING",
  "exception": "com.genesyslab.gsg.services.callback.exceptions
.CallbackExceptionInvalidOperation"
}
```

### No availability

```
PUT http://localhost:8080/genesys/1/service
/callback/BasicCallback/a550a12e-ca77-4146-98d0-58960e0939f7
{
  "_new_desired_time": "2013-05-27T16:45:00.000Z"
}
Result
400 Bad Request
{
  "message": "Too many requests at desired time
[2013-05-27T16:45:00.000Z, 2013-05-27T16:50:00.000Z].
Proposing time slots.",
  "exception": "com.genesyslab.gsg.services.callback
.CallbackExceptionAvailability",
  "availability":
  {
    "2013-05-27T16:50:00.000Z": 5,
    "2013-05-27T16:35:00.000Z": 5,
    "2013-05-27T16:40:00.000Z": 5,
    "2013-05-27T16:55:00.000Z": 3,
```

```
    "2013-05-27T16:25:00.000Z": 5,  
    "2013-05-27T16:30:00.000Z": 5  
  }  
}
```

### Sample operation typically performed by ORS execution

```
PUT http://localhost:8080/genesys/1/service/callback  
/callback-test/361-738dadcb-9d20-4557-8e24-fddb82f9c1b8  
{  
  "_callback_state": "PROCESSING",  
  "_reason": ""  
}  
Result  
200 OK  
{}
```

## Delete Callback (Forget Me)

Introduced in **8.5.201**

Deletes one or more Callback Service instance(s) by passing service IDs or Customer Numbers. You can delete a Callback only if it is in SCHEDULED or COMPLETED status. This API enables you to support **General Data Protection Regulation** and enables you to "forget" customers.

To use this query, you need Basic Authentication. Therefore, you must provide the authentication credentials in the auth parameter of the operation. There are two ways to provide credentials in an auth object:

- In an open form containing the username and password fields of a user defined in the Configuration Server.
- In an encoded form using encoded fields, similar to the Basic Authentication header, which is a Base64-encoded composite string of "username:password".

### POST /genesys/1/admin/callback/ops/delete

Deletes one or more callback request(s).

#### Header

Content-type	application/json
--------------	------------------

#### Body (JSON content)

_customer_number	String array	List of Customer Numbers or Service IDs that identify the callback service instances that must be deleted.
_id	String array	List of service IDs that identify the callback service instances that must be deleted.

## Responses

Name	Description	
200 OK		
<b>Response Body</b> (JSON content)		
success <i>required</i>	Array	Array of service IDs and Customer Numbers that were deleted or were considered as successful with a reason. <pre>[{ "_id": "68542134" }, { "reason": "no callback(s) to delete", "customer_number": "132456" } ]</pre>
errors <i>required</i>	Array	Array of service IDs and Customer Numbers that were <b>not</b> deleted with the associated error codes. <pre>[{ "non-existing-lookup-key": "132456", "code": 40010, "phrase": "BAD_PARAMETER", "message": "No such lookup possible for {properties}" }, { "code": 40020, "phrase": "INVALID_OPERATION", "id": "118-576b21b4-a235-4ba5-92d4-102cbbb54bca", "message": "Callback 118-576b21b4-a235-4ba5-92d4-102cbbb54bca cannot be deleted - _callback_state=PROCESSING" } ]</pre>

Name	Value
500 Internal Server Error	
<b>Response body (JSON Content)</b>	
code	50020
phrase	BAD_CONFIGURATION
message	<ul style="list-style-type: none"> <li>"Service option {service} / _default_country is not configured. But option _disallow_impossible_phone_numbers is set. We cannot validate phone numbers without knowing the country."</li> </ul>

Name	Value
	<ul style="list-style-type: none"> <li>"Service option {service} / _default_country is not configured. But option _disallow_premium_phone_numbers is set. We cannot validate phone numbers without knowing the country."</li> <li>"Unable to parse option: _request_queue_time_stat={statistic}"</li> <li>"Missing default_chat_endpoint option in chat section because this service has parameter _media_type=chat"</li> <li>"Missing default_client_timeout option in chat section because this service has parameter _media_type=chat"</li> <li>"Option service.{service} / _business_hours_service not configured."</li> <li>"Option _business_hours_service is invalid: {message}"</li> <li>"Service undefined: {service}"</li> <li>"Service {service} has unknown value for option _type"</li> <li>"Service {service} has option _type != ors"</li> <li>"Service {service} has option _service != callback"</li> </ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionConfiguration
properties	<pre>{   "service": &lt;service name&gt; }</pre>

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40010
phrase	BAD_PARAMETER



Name	Value
message	<ul style="list-style-type: none"> <li>"No such lookup possible for {properties}"</li> <li>"No lookup possible. No properties to look for."</li> <li>Generic parser exception message: Typically, a bad date parsing may result in a bad parameter error with the appropriate statement.</li> <li>Generic missing parameter exception message (case of controller level detection).</li> </ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionBadParameter
properties	<pre>{   "id": &lt;callback id&gt;,   "keys": &lt;missing lookup key&gt;,   "day": &lt;specified day value&gt;,   "properties": &lt;lookup properties&gt;,   "option": &lt;invalid option key&gt; }</pre>

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40020
phrase	INVALID_OPERATION
message	"Cannot process 'filter' parameter correctly : {filter}"
exception	com.genesyslab.gsg.services.callback.CallbackExceptionInvalidOperation
properties	<pre>{   "id": &lt;callback id&gt;,   "service": &lt;service name&gt;,   "time": &lt;ISO UTC time&gt;, }</pre>

Name	Value
	<pre>"state": &lt;callback state&gt;, "message": &lt;ORS server's message&gt;, "filter": &lt;filtering expression&gt; }</pre>

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50050
phrase	UNKNOWN_ERROR
message	"Error processing callback {message} "
exception	com.genesyslab.gsg.services.callback.CallbackExceptionUnknown
properties	{ "message": <message caught> }

## Example

POST http://localhost:8080/genesys/1/admin/callback/ops/delete

```
{
  "_id": ["118-576b21b4-a235-4ba5-92d4-102cbbb54bca"],
  "_customer_number": [
    "132456",
    "1111",
    "3333"
  ]
}
```

## Result

Response: 200 OK

```
{
  "success": [
    {
      "reason": "no callback(s) to delete",
      "_customer_number": "132456"
    },
  ],
}
```

```

    {
      "_id": "118-27f3bed5-6e3a-4c89-903f-dae562b30481"
    },
    {
      "_id": "118-c2ce7a84-d33a-4d8d-88a0-b76a563f2324"
    }
  ],
  "errors": [
    {
      "code": 40020,
      "phrase": "INVALID_OPERATION",
      "_id": "118-576b21b4-a235-4ba5-92d4-102cbbb54bca",
      "message": "Callback 118-576b21b4-a235-4ba5-92d4-102cbbb54bca cannot
be deleted - _callback_state=PROCESSING"
    }
  ]
}
```

Query Callback By ID

Introduced in 8.5.207

Retrieves a callback request by its ID.

GET /genesys/2/service/callback/{callback-execution-name}/{id}

Queries the outstanding callback associated with a given ID.

URI Parameters		
Name	Type	Description
<b>callback-execution-name</b> *required	string <i>path</i>	Name of the callback execution service of 'ors' type provisioned in GMS.
<b>id</b> *required	string <i>path</i>	Callback ID.

Responses

Name	Description
200 OK	
Response Body (JSON content)	
<none>	<ul style="list-style-type: none"><li>If accepted, the currently outstanding callback request.</li></ul> <pre>[   {     "_id": &lt;callback id&gt;,     "desired_time": &lt;ISO UTC time&gt;,     "url": &lt;service URL&gt;,</pre>

Name	Description
	<pre>       "_expiration_time": &lt;ISO UTC time&gt;,       "_service_name": &lt;service-name&gt;,       "_customer_number": &lt;customer number&gt;,       "_callback_state": &lt;callback state&gt;,       "_time_scheduled": &lt;ISO UTC time&gt;     }   ] </pre> <ul style="list-style-type: none"> <li>If not, an error code indicating the reason.</li> </ul>

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40010
phrase	BAD_PARAMETER
message	<ul style="list-style-type: none"> <li>"No such lookup possible for {properties}"</li> <li>"No lookup possible. No properties to look for."</li> <li>Generic parser exception message: Typically, a bad date parsing may result in a bad parameter error with the appropriate statement.</li> <li>Generic missing parameter exception message (case of controller level detection).</li> </ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionBadParameter
properties	<pre> {   "id": &lt;callback id&gt;,   "keys": &lt;missing lookup key&gt;,   "day": &lt;specified day value&gt;,   "properties": &lt;lookup properties&gt;,   "option": &lt;invalid option key&gt; } </pre>

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40020
phrase	INVALID_OPERATION
message	"Cannot process 'filter' parameter correctly : {filter}"
exception	com.genesyslab.gsg.services.callback.CallbackExceptionInvalidOperation
properties	<pre>{   "id": &lt;callback id&gt;,   "service": &lt;service name&gt;,   "time": &lt;ISO UTC time&gt;,   "state": &lt;callback state&gt;,   "message": &lt;ORS server's message&gt;,   "filter": &lt;filtering expression&gt; }</pre>

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50050
phrase	UNKNOWN_ERROR
message	"Error processing callback {message} "
exception	com.genesyslab.gsg.services.callback.CallbackExceptionUnknown
properties	<pre>{   "message": &lt;message caught&gt; }</pre>

## Example

GET http://localhost:8080/genesys/1/service/callback/BasicCallback/120-07f85068-650d-4cce-a5e7-396dfa22455b

## Result

200 OK

```
{
  "_callback_state": "SCHEDULED",
  "_expiration_time": "2020-05-11T11:59:59.000Z",
  "_service_name": "BasicCallback",
  "_id": "124-07f85068-650d-4cce-a5e7-396dfa22455f",
  "_customer_number": "12345",
  "_url": "/genesys/1/service/callback/BasicCallback/120-07f85068-650d-4cce-a5e7-396dfa22455b",
  "_time_scheduled": "2020-04-16T12:52:31.521Z",
  "_desired_time": "2020-04-27T12:00:00.000Z"
}
```

## Query-Callback by Lookup Properties

**Modified in 8.5.111**

The Query-Callback API queries the current set of outstanding Callback services associated with a given property.

**Notes:**

- Outstanding Callback services are requests where `_callback_state` is one of the following values: SCHEDULED, QUEUED, ROUTING, PROCESSING, COMPLETED.
- Properties allowing the Callback request trackback are defined as comma-separated keys with the service option `_customer_lookup_keys`.
- The API returns each callback for which the looked-up property is or was equal to the value specified in the requested property.
- Starting in 8.5.111, you can configure the list of values to be retrieved when calling this query by setting the returned-keys option at the GMS application level.
- To use the `_customer_number` lookup property regardless of whether you specify a callback service name or not in the API URL, the `_fix_plus_on_int_phone_numbers` option must be identical in the callback section and in each service-specific section.
  - This is the expected behavior if you stick to defaults.
  - If a callback service has a distinct value for `_fix_plus_on_int_phone_numbers`, you can only use the `_customer_number` lookup property by specifying the service name in the API URL.

GET /genesys/1/service/callback/{callback-execution-name}?{property=value}

GET /genesys/1/service/callback?{property=value}

Queries the current set of outstanding Callback services associated with a given property.

**URI Parameters**

Name	Type	Description
<b>callback-execution-name</b>	string <i>path</i>	Name of the callback execution service of 'ors' type provisioned in GMS.
<b>property=value</b> *required	string <i>path</i>	This is a property name used to query the callback. Properties allowing the Callback request trackback are defined as comma-separated keys with the service option <code>_customer_lookup_keys</code> .

		If you specify several properties, you may need to use the operand property.
operand	string	<p>Operand to use for the properties defined in the service option <code>_customer_lookup_keys</code>. Possible values are AND or OR. Default is AND.</p> <p>When multiple <b>property=value</b> are provided in the query, the operand specifies which operation to perform on matched Callback requests:</p> <ul style="list-style-type: none"><li>• AND means that all <b>property=value</b> must match;</li><li>• OR means any <b>property=value</b> can match.</li></ul>
<code>_callback_state</code> Since 8.5.101.03	string	<p>Specifies a unique state to filter onto. For example:</p> <ul style="list-style-type: none"><li>• <code>_callback_state='COMPLETED'</code> filters callbacks and returns only callbacks in COMPLETED state.</li><li>• <code>_callback_state='!COMPLETED'</code> filter callbacks and only return the ones that are not COMPLETED.</li></ul> <div><b>Important</b> The character "!" is used to negate a case.</div> <p>You can query the following callback states: SCHEDULED, QUEUED, ROUTING, PROCESSING, COMPLETED.</p>
<code>_desired_time_from</code> Since 8.5.101.03	string	<p>Specifies ISO timestamps. All callback matching lookup properties that were scheduled before this time will be filtered out.</p>



<div><div>_desired_time_to</div><div>Since 8.5.101.03</div></div>	string	Specifies ISO timestamps. All callback matching lookup properties that were scheduled after this time will be filtered out.
---	--------	---

Responses

Name	Type	Description
200 OK		
Response Body (JSON content)		
<none>	<div><ul style="list-style-type: none"><li>If accepted, JSON array of service IDs of the currently outstanding callback requests.</li></ul><pre>[   {     "_id": &lt;callback id&gt;,     "desired_time": &lt;ISO UTC time&gt;,     "_callback_state": &lt;callback state&gt;,     "_expiration_time":&lt;ISO UTC time&gt;,     "_customer_number": &lt;customer number&gt;,     "url": &lt;service URL&gt;   },   ... ]</pre><ul style="list-style-type: none"><li>If not, an error code indicating the reason.</li></ul></div>	

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40010
phrase	BAD_PARAMETER
message	<ul style="list-style-type: none"><li>• "No such lookup possible for {properties}"</li><li>• "No lookup possible. No properties to look for."</li><li>• Generic parser exception message: Typically, a bad date parsing may result in a bad parameter error with the appropriate statement.</li><li>• Generic missing parameter exception message (case of controller level detection).</li></ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionBadParameter
properties	<pre>{   "id": &lt;callback id&gt;,   "keys": &lt;missing lookup key&gt;,   "day": &lt;specified day value&gt;,   "properties": &lt;lookup properties&gt;,   "option": &lt;invalid option key&gt; }</pre>

Name	Value
400 Bad Request	
<b>Response body (JSON Content)</b>	
code	40020
phrase	INVALID_OPERATION
message	"Cannot process 'filter' parameter correctly : {filter}"
exception	com.genesyslab.gsg.services.callback.CallbackExceptionInvalidOperation
properties	<pre>{   "id": &lt;callback id&gt;,   "service": &lt;service name&gt;,   "time": &lt;ISO UTC time&gt;,   "state": &lt;callback state&gt;,   "message": &lt;ORS server's message&gt;,   "filter": &lt;filtering expression&gt; }</pre>

Name	Value
500 Internal Server Error	
<b>Response body (JSON Content)</b>	
code	50050
phrase	UNKNOWN_ERROR
message	"Error processing callback {message} "
exception	com.genesyslab.gsg.services.callback.CallbackExceptionUnknown
properties	{ "message": <message caught> }

## Example

GET http://localhost:8080/genesys/1/service/callback  
/BasicCallback?\_customer\_number=555-5461206

## Result

```
200 OK
[
  {
    "_id": "a550a12e-ca77-4146-98d0-58960e0939f7",
    "desired_time": "2013-05-27T15:05:00.000Z",
    "_callback_state": "QUEUED",
    "_expiration_time": "2014-11-03T18:36:45.000Z",
    "_customer_number": "555-5461206",
    "url": "/1/service/callback/BasicCallback/a550a12e-ca77-4146-98d0-58960e0939f7"
```

---

```
    },
    {
      "_id": "4alea889-1ef7-432d-a543-cff96b4a2daf",
      "desired_time": "2013-05-27T15:10:00.000Z",
      "_callback_state": "SCHEDULED",
      "_expiration_time": "2014-11-03T18:36:45.000Z",
      "_customer_number": "555-5461206",
      "url": "/1/service/callback/BasicCallback/4alea889-1ef7-432d-a543-cff96b4a2daf"
    }
  ]
```

## Query-Availability

v1

### Query-Availability v1

This query returns a simple map of slots in which the office capacity is not full.

#### GET /genesys/1/service/callback/{callback-execution-name}/availability

Returns a simple map of slots in which the office capacity is not full.

##### URI Parameters

Name	Type	Description
<b>callback-execution-name</b> *required	string <i>path</i>	Name of the callback execution service of 'ors' type provisioned in GMS.
JSON Body		
start	date	Start date is specified in ISO 8601 format, using UTC as the timezone: yyyy-MM-ddTHH:mm:ss.SSSZ. If not specified, it is assumed to be now.
timestamp	date	Alias to start parameter; kept for compatibility reasons.
number-of-days	integer	Used as an alternative to the end date. If neither end nor number-of-days is specified, the end date is assumed to be the same as the start date.
end	date	End date is specified in ISO 8601 format, using UTC as timezone: yyyy-MM-ddTHH:mm:ss.SSSZ. If neither end nor number-of-days is specified, the end date is assumed to be the same as the start date.
max-time-slots	integer	Maximum number of time slots to be included in the response when the office is open and capacity is above zero. It can be used to improve the performance of the query over a long period of time.

## Important

If neither of the parameters `number-of-days` and `end` parameters are specified, the default time range matches 1 bucket only (as configured in the `_request_time_bucket` service option).

### Request example:

```
GET http://localhost:8080/genesys/1/service/callback/Callback_VQ/availability?start=2014-12-03T15:00:00.000Z
```

### Response

The Callback controller provides a facet to the availability service, which uses the calendar service underneath. Just as the calendar service takes three non-mandatory input parameters (start, number-of-days, end), the availability service should accept the same parameters and pass them on to the calendar service.

- The response contains a map of time slots and capacity counters.
- The slots are ordered in ascending order.
- Any time slots where the capacity is full (for example, zero) are not provided in the response. Similarly, if the office is closed, those time slots are not provided in the response.

```
200 OK
{
  // All periods are ordered in ascending time order
  "2014-10-17T13:00:00.000Z": "5",
  "2014-10-17T13:10:00.000Z": "4",
  // there were no agents available between 13:20 and 13:30 UTC
  //hence the time slot is not reported
  "2014-10-17T13:30:00.000Z": "5"
}
```

## v2

### Query-Availability v2

This query includes more query options than v1 and returns an array of ordered slots that include detailed capacity information and timezone information.

**GET /genesys/2/service/callback/{callback-execution-name}/availability**

Returns an array of ordered slots that include detailed capacity information and timezone information.

#### URI Parameters

Name	Type	Description
<b>callback-execution-name</b> *required	string <i>path</i>	Name of the callback execution service of 'ors' type provisioned in GMS.
start	date	<p>Start date in the "ISO 8601" format, using the UTC timezone: "yyyy-MM-ddTHH:mm:ss.SSSZ". If not specified, the default start date is the date on which the query was submitted.</p> <ul style="list-style-type: none"> <li>If you set the start parameter, do not set the start-ms or timestamp parameters.</li> <li>You must also set the end or number-of-days parameter; otherwise, the end date is assumed to be the start date.</li> </ul>
start-ms	long	<p>Start date in epoch time, that is, the number of milliseconds since 00:00:00, Thursday, 1 January 1970 (UTC).</p> <ul style="list-style-type: none"> <li>You must also set the end-ms or number-of-days parameter; otherwise, the end date is assumed to be the start-ms date.</li> <li>If you set the start-ms parameter, do not set the start or timestamp parameters.</li> </ul>
number-of-days	integer	Number of days used to define the availability period starting at the start or start-ms date. You can use this parameter instead of the end or of the end-ms parameter.
end	date	End date, in "ISO 8601" format, using the UTC timezone: yyyy-MM-ddTHH:mm:ss.SSSZ. By default, if neither the "end" nor the "number-of-days" parameter is specified, then the end date is assumed to be the start date.
end-ms	long	End date in epoch time, that is the number of milliseconds since 00:00:00, Thursday, 1 January 1970 (UTC).



		Set only one of the end, end-ms, or number-of-days parameters.
max-time-slots	integer	Maximum number of time slots to include in the response if the office is open and the capacity greater than zero. You can use this parameter to improve query performance over a lengthy period of time.
timezone	string	Timezone for the start and end date parameters. Additionally, the response object will return the localTime fields formatted in this timezone.
report-busy	boolean	If true, the response includes the slots where the office is open and where callbacks are booked to full capacity. By default, report-busy is false.
JSON body: <b>None</b> .		

### Important

If neither of the parameters number-of-days, end, and end-ms parameters are specified, the default time range matches 1 bucket only (as configured in the \_request\_time\_bucket service option).

## Responses

If successful, the response returns multiple values that describe the slots, availability, and capacity for a given slot.

Name	Type	Description
200 OK		
<b>Response Body</b> (JSON content)		
slots <i>required</i>	String array of slots	<p>Array of ordered slots and each slot includes the minute duration (durMinutes), and the timezone.</p> <ul style="list-style-type: none"><li>• The array of slots includes detailed information about each slot.</li><li>• Slots are sorted in ascending</li></ul>

Name	Type	Description
		<p>order by their time.</p> <ul style="list-style-type: none"><li>• Slots are all the same duration, specified in the <code>durMinutes</code> value.</li><li>• The <code>"timezone"</code> value specifies the timezone used for the <code>"localTime"</code> fields in slots' information.</li></ul> <pre>{  "slots": [    {      "utcTime": &lt;UTC time&gt;,      "localTime": &lt;UTC time&gt;,      "capacity": &lt;capacity&gt;,      "total": &lt;total&gt;    },    (...)  ]  "durationMin": &lt;duration in minutes&gt;,  "timezone": &lt;timezone&gt;}</pre> <p>Each slot includes:</p> <ul style="list-style-type: none"><li>• <code>"utcTime"</code> specifies when this slot begins in UTC time.</li><li>• <code>"localTime"</code> reports the same time as <code>"utctime"</code>, but formatted using the <code>"timezone"</code> set in the request.</li><li>• <code>"capacity"</code> value is the number of available callbacks that can be scheduled within this timeslot.</li><li>• <code>"total"</code> is the total capacity that is configured for this timeslot.</li></ul>

Name	Value
400 Bad Request	
<b>Response body (JSON Content)</b>	
code	40010

Name	Value
phrase	BAD_PARAMETER
message	<ul style="list-style-type: none"> <li>"day parameter must be between 1 and 7, inclusively. Actual value is: {day}"</li> <li>"No time slots available. The requested time period is in the past."</li> <li>Generic parser exception message: Typically, a bad date parsing may fall there as a bad parameter error with the appropriate statement.</li> <li>Generic missing parameter exception message (case of controller level detection).</li> </ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionBadParameter
properties	<pre>{   "id": &lt;callback id&gt;,   "keys": &lt;missing lookup key&gt;,   "day": &lt;specified day value&gt;,   "properties": &lt;lookup properties&gt;,   "option": &lt;invalid option key&gt; }</pre>

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40050, 40051
phrase	<ul style="list-style-type: none"> <li>SLOT_UNAVAILABLE (40050)</li> <li>SLOT_UNAVAILABLE_PROPOSAL(40051)</li> </ul>
message	<ul style="list-style-type: none"> <li>"No time slots available."</li> <li>"Too many requests at desired time slot {slot}."</li> </ul>

Name	Value
	Proposing time slots." <ul style="list-style-type: none"> <li>"Office is closed at desired time slot {slot}. Proposing time slots."</li> </ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionAvailability
properties	<pre>{   "slot": &lt;ISO UTC time range&gt;,   "service": &lt;service name&gt; }</pre>

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50020
phrase	BAD_CONFIGURATION
message	<ul style="list-style-type: none"> <li>"Option service.{service} / _business_hours_service not configured."</li> <li>"Option _business_hours_service is invalid: {message}"</li> <li>"Service undefined: {service}"</li> <li>"Service {service} has unknown value for option _type"</li> <li>"Service {service} has option _type != ors"</li> <li>"Service {service} has option _service != callback"</li> </ul>
exception	com.genesyslab.gsg.services.callback.CallbackExceptionConfiguration
properties	<pre>{   "service": &lt;service name&gt; }</pre>

---

Name	Value
	}

Name	Value
500 Internal Server Error	
<b>Response body (JSON Content)</b>	
code	50005
phrase	CALENDAR_ERROR
message	message returned by Calendar service
exception	com.genesyslab.gsg.services.callback.CallbackExceptionCalendarError
properties	

Name	Value
500 Internal Server Error	
<b>Response body (JSON Content)</b>	
code	50004
phrase	CAPACITY_ERROR
message	message returned by Capacity service
exception	com.genesyslab.gsg.services.callback.CallbackExceptionCapacityError
properties	

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50050
phrase	UNKNOWN_ERROR
message	"Error processing callback {message} "
exception	com.genesyslab.gsg.services.callback.CallbackExceptionUnknown
properties	{ "message": <message caught> }

## Examples

### Request example:

http://localhost:8010/genesys/2/service/callback/callback-PST  
 /availability?start=2016-04-13T09:00:00.000&end=2016-04-13T16:00:00.000  
 &timezone=America/Toronto

```
{
  "slots": [
    {
      "utcTime": "2016-04-13T13:00:00.000Z",
      "localTime": "2016-04-13T09:00:00.000",
      "capacity": 42,
      "total": 100
    },
    {
      "utcTime": "2016-04-13T13:05:00.000Z",
      "localTime": "2016-04-13T09:05:00.000",
      "capacity": 67,
      "total": 100
    },
    {
      "utcTime": "2016-04-13T13:10:00.000Z",
      "localTime": "2016-04-13T09:10:00.000",
      "capacity": 91,
      "total": 100
    }
  ],
  "durationMin": 5,
  "timezone": "Eastern Standard Time"
}
```

### Important

Existing calendar configurations must be updated for the time zone definition. Instead of EST or PST time zones that were configured using Configuration Manager, you must use time zones as allowed in Java ([http://en.wikipedia.org/wiki/List\\_of\\_tz\\_database\\_time\\_zones](http://en.wikipedia.org/wiki/List_of_tz_database_time_zones)), such as America/Toronto, or Europe/Paris. You must also change the service option `_type` from `ors` to `builtin`.

## Query-Callback by Queue(s)

### Modified in 8.5.111

The Query-Callback API queries the current set of outstanding Callback services in the given queue(s).

Starting in 8.5.111, you can filter and configure the list of values to be passed and retrieved when calling this query through the following options at the GMS application level: `returned-keys` and `filter-keys`.

### Important

Outstanding Callback services are requested if their `_callback_state` is one of the following values: `SCHEDULED`, `QUEUED`, `ROUTING`, `PROCESSING`, `COMPLETED`.

To use this query, you need Basic Authentication. Therefore, you must provide the authentication credentials in the `auth` parameter of the operation. There are two ways to provide credentials in an `auth` object:

- In an open form containing the username and password fields of a user defined in the Configuration Server.
- In an encoded form using encoded fields, similar to the Basic Authentication header, which is a Base64-encoded composite string of "username:password".

```
GET /genesys/1/admin/callback/queues?target={callback-execution-name}&t_time={iso_start_time}&end_time={iso_end_time}
```

Queries the current set of outstanding Callback services in given queue(s).

#### URI Parameters

Name	Type	Description
{iso_start_time}	string	This is the minimum time for

		<p>which to query callback requests.</p> <p>The format is ISO 8601 "yyyy-MM-ddTHH:mm:ss.SSSZ".</p> <p>For example: "2013-05-27T15:30:00.000Z"</p>
{iso_end_time}	string	<p>This is the maximum time for which to query callback requests.</p> <p>If not specified, requests that are due in the next 24 hours are returned.</p> <p>The format is ISO 8601 "yyyy-MM-ddTHH:mm:ss.SSSZ".</p> <p>For example: "2013-05-28T15:30:00.000Z"</p>
{states}	string	<p>Comma-separated list of callback states used to filter the returned results. For example, if states=SCHEDULED,QUEUED, only scheduled and queued callbacks are returned.</p> <p>If not specified, all the callbacks of the given queue are returned.</p>
{max}	integer	<p>This is the maximum number of requests to return for each queue.</p> <p>If not specified, 500 maximum requests per queue are returned.</p>
callback-execution-name	string	<p>Name of the callback execution service provisioned in GMS. For example, BasicCallback.</p> <p>If not specified, the queues for all services are returned.</p>
{max}	integer	<p>This is the maximum number of requests to return for each queue.</p> <p>If not specified, 500 maximum requests per queue are returned.</p>

## Responses

Name	Mandatory	Description
200 OK		
Response Body (JSON content)		



Name	Mandatory	Description
List of target queues <i>required</i>	string	<p>If accepted, a tree list of target queues and the following properties:</p> <pre> &lt;Queue name&gt;: {   "_customer_number":   &lt;customer number&gt;,   "_callback_state": &lt;callback state&gt;,   "_desired_time": &lt;callback UTC desired time&gt;,   "_id": &lt;callback service id&gt;,   "url": &lt;request&gt; }</pre>

Name	Value
<b>400 Bad Request</b>	
<b>Response body (JSON Content)</b>	
code	40020
phrase	INVALID_OPERATION
message	"Query range spans too wide time range (%d / %d). Adjust query parameters for time range."
exception	com.genesyslab.gsg.services.callback.CallbackExceptionInvalidOperation
properties	<pre> {"id": &lt;callback id&gt;,  "service": &lt;service name&gt;,  "time": &lt;ISO UTC time&gt;,  "state": &lt;callback state&gt;,  "message": &lt;ORS server's message&gt;,  "filter": &lt;filtering expression&gt; }</pre>

Name	Value
500 Internal Server Error	
<b>Response body (JSON Content)</b>	
code	50050
phrase	UNKNOWN_ERROR
message	"Error processing callback {message} "
exception	com.genesyslab.gsg.services.callback.CallbackExceptionUnknown
properties	{ "message": <message caught> }

## Example

GET http://localhost:8080/genesys/1/admin/callback/queues

## Result

200 OK

```
{
  "BasicCallback":
  [
    {
      "_customer_number": "654321",
      "_callback_state": "PROCESSING",
      "_desired_time": "2013-06-07T16:25:00.000Z",
      "_id": "fd30abb97bd04885b544893276fb534b",
      "url": "/1/service/callback/BasicCallback/fd30abb97bd04885b544893276fb534b"
    }
  ],
  "AdvancedCallback":
  [
    {
      "_customer_number": "654321",
      "_callback_state": "QUEUED",
      "_desired_time": "2013-06-07T16:35:00.000Z",
      "_id": "07d2ddd506f04b4ba91aba59c4fa8871",
      "url": "/1/service/callback/AdvancedCallback/07d2ddd506f04b4ba91aba59c4fa8871"
    },
    {
      "_customer_number": "654321",
      "_callback_state": "SCHEDULED",
      "_desired_time": "2013-06-07T16:45:00.000Z",
      "_id": "8f68d4969d904d039ccf0101fac39283",
      "url": "/1/service/callback/AdvancedCallback/8f68d4969d904d039ccf0101fac39283"
    }
  ]
}
```

```
}
  ]
}
```

## Query Counter Watermarks

This query counts the current set of executed callback instances per queues or for a given queue. Executed callback instances are:

- Callbacks that are in execution within ORS
- Callbacks do not have their `_callback_state` property set to `SCHEDULED`
- Callbacks do not have their `_callback_state` property set to `COMPLETED` in GMS storage. Callbacks in such a state for more than 3 hours are discarded.

To use this query, you need Basic Authentication. Therefore, you must provide the authentication credentials in the `auth` parameter of the operation. There are two ways to provide credentials in an `auth` object:

- In an open form containing the username and password fields of a user defined in the Configuration Server.
- In an encoded form using encoded fields, similar to the Basic Authentication header, which is a Base64-encoded composite string of "username:password".

### Important

You can use this API to ensure that you do not book more Callbacks than you have licenses for.

GET /genesys/1/admin/callback/watermarks?service\_name={callback-execution-name}

GET /genesys/1/admin/callback/watermarks

Counts the current set of executed callback instances per queues or for a given queue.

#### URI Parameters

Name	Type	Description
{callback-execution-name}	string	Name of a callback execution service. If you set this parameter, the response will return the watermarks for the specified service only. If the service name is not set, the response returns

		<p>the total count of executed callback instances in queues and the count per service.</p> <p>You can query watermarks for several services in a single query. To do so, add as many <code>service_name</code> values as you need to your query:</p> <p>GET <code>/genesys/1/admin/callback/watermarks?service_name=service1&amp;service_name=...</code></p>
--	--	--

Responses

HTTP code	200
HTTP message	OK
Response Body (JSON content)	<p>If accepted, a list of target queues and the count of callbacks that are in execution within ORS or that do not have their <code>_callback_state</code> property set to SCHEDULED or COMPLETED) in GMS storage.</p> <pre>{   "total": &lt;total of callbacks in progress&gt;,   "services": {     &lt;service-1&gt;: &lt;number of callbacks in progress for service-1&gt;,     ...     &lt;service-n&gt;: &lt;number of callbacks in progress for service-n&gt;,   } }</pre>

Name	Value
500 Internal Server Error	
Response body (JSON Content)	
code	50050
phrase	UNKNOWN_ERROR
message	"Error processing callback {message} "
exception	com.genesyslab.gsg.services.callback.CallbackExceptionUnknown
properties	{ "message": <message caught> }

Example

Operation

GET `http://localhost:8080/genesys/1/admin/callback/watermarks`

Result

200 OK

```
{
  "total": 1,
  "services": {
    "callback-immediate": 0,
    "callback-test": 1
  }
}
```

GET `http://localhost:8080/genesys/1/admin/callback/watermarks?service_name=callback-immediate`

Result

200 OK

```
{
  "total": 0,
  "services": {
    "callback-immediate": 0
  }
}
```

Check in Queue Position

This query enables your application to query for the position and Estimated Wait Time while the GMS Service request is in QUEUED status. This query is used to provide additional details in the [Callback UI](#).

Name	Type	Description
POST <code>/genesys/1/service/{callback-service-id}/check-queue-position</code>		
BODY Parameters		
<code>{callback-service-id}</code> <b>required</b>	string	ID of the callback execution service. For example, 445-f4fa53ec-8e93-4836-ba35-f0bd74a025a8.

Important

The GET method is also supported for this feature.

## Response

<b>HTTP code</b>	200
<b>HTTP message</b>	OK
Response Body (JSON content)	<p>JSON-formatted information for the given service ID:</p> <ul style="list-style-type: none"><li>• <b>app_version</b>: Callback strategy version.</li><li>• <b>wt</b>: The time that the call has waited in queue.</li><li>• <b>connid</b>: Interaction ID in the Virtual Queue.</li><li>• <b>ewt</b>: The estimated time that customer will wait for the callback.</li><li>• <b>positioninqueue</b>: The callback's current position in the queue.</li><li>• <b>_position</b>: position of the interaction in the virtual queue (top position = 1).</li><li>• <b>_eta</b>: estimated wait time to the agent availability.</li><li>• <b>_total_waiting</b>: total number of requests waiting in queue.</li><li>• <b>priority</b>: The callback priority in the Virtual Queue.</li><li>• <b>agents_logged_in</b>: The number of agents that have logged in.</li><li>• <b>ors_session_id</b>: ORS session ID of the callback.</li><li>• <b>ewt_at_offer</b>: The estimated wait time when the callback is offered.</li><li>• <b>pos_at_offer</b>: The callback's position in the queue when the callback is offered.</li><li>• <b>callback_type</b>: The type of callback.</li><li>• <b>time_callback_accepted</b>: The time when the callback is accepted.</li><li>• <b>channel</b>: The callback channel.</li><li>• <b>skill_expression</b>: The callback's target or skill expression.</li><li>• <b>ewt_at_first_dial</b>: The estimated wait time when the first outbound call happened.</li><li>• <b>pos_at_first_dial</b>: The callback's position in the queue when the first outbound call happened.</li><li>• <b>time_at_first_dial</b>: The time when the first outbound call happened.</li></ul>

- **dial\_attempt**: The number of dials that agent has attempted.
- **is\_snoozed**: If true, shows that the callback is snoozed.
- **dial\_result**: The result of callback dial.
- **time\_customer\_connected**: The time when the customer connected.

## Errors

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50050
phrase	UNKNOWN_ERROR
message	"Error processing callback {message} "
exception	com.genesyslab.gsg.services.callback.CallbackExceptionUnknown
properties	{ "message": <message caught> }

## Example

### Operation

POST /genesys/1/service/445-f4fa53ec-8e93-4836-ba35-f0bd74a025a8/check-queue-position HTTP/1.1  
Connection: close  
Content-Length: 0  
Content-Type: application/x-www-form-urlencoded; charset=UTF-8

### Response:

```
200 OK
{
  "app_version": "v2.41",
  "wt": 26,
  "connid": "006e02aea54bc008",
  "ewt": 0,
  "positioninqueue": 0,
  "_position": 1,
  "_eta": 0,
  "_total_waiting": 1,
  "priority": 500,
```

```
{
  "agents_logged_in":3,
  "ors_session_id":"00ACLU5N00CV19601K015B5AES000003",
  "ewt_at_offer":0,
  "pos_at_offer":0,
  "callback_type":"WAIT_FOR_AGENT",
  "time_callback_accepted":1508959666,
  "channel":"WEB",
  "skill_expression":"GMSCallbackAgents@stat.GA",
  "ewt_at_first_dial":"100.0",
  "pos_at_first_dial":"1",
  "time_at_first_dial":1508959684,
  "dial_attempt":1,
  "is_snoozed":false,
  "dial_result":"PERSON",
  "time_customer_connected":1508959690
}
```

## Export Cancelled Callback Records

### Added in: 8.5.110

This query exports the callbacks that were cancelled by the Service Management UI only (**Bulk Cancel**).

- The data will be exported in CSV format.
- The request will export the records cancelled from the last 30 days to the next 15 days.
- You can export additional fields with the retrieved callback records.

By default, the CSV report includes the following default properties: `_desired_time`, `_service_name`, `_customer_number`, `urs_virtual_queue`, `vq_for_outbound_calls`, and `target`.

To use this query, you need Basic Authentication. Therefore, you must provide the authentication credentials in the `auth` parameter of the operation. There are two ways to provide credentials in an `auth` object:

- In an open form containing the username and password fields of a user defined in the Configuration Server.
- In an encoded form using encoded fields, similar to the Basic Authentication header, which is a Base64-encoded composite string of "username:password".

Name	Type	Description
POST /genesys/1/admin/callback/reportcancelled		
BODY Parameters		
callback_reason <i>required</i>	string	The reason for the cancellation. For example, CANCELLED_BY_ADMIN.
exported_properties	string	List of properties to export for



Name	Type	Description
		each selected record. For example: ["_service_id,_desired_time"]. If this parameter is empty or missing, the following properties will be exported by default: _desired_time, _service_name, _customer_number, urs_virtual_queue, _vq_for_outbound_calls, and target.

## Response

<b>HTTP code</b>	200
<b>HTTP message</b>	OK
<b>Response Body (JSON content)</b>	CSV-formatted results for exported records: <property-1>,<property-2>,...,<property-n>  <record-1-property1>,<record-1-property2>,...,<record-1-property-n>  ...  <record-n-property1>,<record-n-property2>,...,<record-n-property-n>

## Errors

<b>HTTP code</b>	400
<b>HTTP message</b>	Callback reason is missing.
<b>HTTP code</b>	204
<b>HTTP message</b>	No record found.

Name	Value
<b>500 Internal Server Error</b>	
<b>Response body (JSON Content)</b>	
code	50050
phrase	UNKNOWN_ERROR
message	"Error processing callback {message} "
exception	com.genesyslab.gsg.services.callback.CallbackExceptionUnknown

Name	Value
properties	{ "message": <message caught> }

## Example

### Operation

```
POST /genesys/1/admin/callback/reportcancelled
{
  "callback_reason": "CANCELLED_BY_ADMIN",
  "exported_properties": []
}'
```

### Response:

```
Access-Control-Allow-Credentials →true
Access-Control-Allow-Origin →chrome-extension://aicmkgpgakddgnaphhhpliifpcfhicfo
Access-Control-Expose-Headers →
Content-Disposition →attachment; filename="report.csv"
_desired_time,_service_name,_customer_number,_target,_vq_for_outbound_calls,_urs_virtual_queue
2017-07-04T22:00:00.000Z,callback-
gms,5115,Billing@Stat_Server.GA,GMS_Callback_VQ_OUT,GMS_Callback_VQ
```

### Operation

```
POST /genesys/1/admin/callback/reportcancelled
```

```
HTTP/1.1
Connection: keep-alive
Content-Type: application/json
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
50.0.2661.102 Safari/537.36
Cookie: JSESSIONID=1ff4o2zwehsbx6fzdfwb66jsa
Authentication: Basic=...
```

```
{
  "callback_reason": "CANCELLED_BY_ADMIN",
  "exported_properties": ["_service_id,_desired_time"]
}
```

### Response

```
desired_time,customer_number,exported_properties1,exported_properties2
2017-05-11T12:22:00+00:00,3329284556,exported_value1,exported_value2
2017-05-11T12:21:00+00:00,3329284576,exported_value1,exported_value2
2017-05-10T07:21:00+00:00,3329284577,exported_value1,exported_value2
```

## Implement Preview and Disposition Scenarios

If you implement a custom agent desktop and wish to integrate the Preview and Disposition scenarios

to your Callback application, you need to configure Preview and Disposition options in your Callback service. After you do this, your Custom Agent Application will receive the following UserEvent events from Orchestration Server:

- **CallbackInvitationEvent**—The Callback invitation that contains the attached data for the preview. The invitation includes the list of actions that the agent can perform—accept, reject, or cancel. Your Agent application displays the actions and the attached data for the preview to the agent, then submits a Preview Response to your Callback service.
- **CallbackDispositionEvent**—The Callback disposition event that provides the URL to which you submit the disposition selected by the agent. Your Agent application then submits a Disposition Response to your Callback service through this URL.

For a complete description, refer to the [Callback Solution Guide](#).

# Digital Channels API

The Digital Channels API allows Genesys Mobile Engagement to work with Genesys digital channels such as chat and email. Before you can start using these APIs, you need to configure them by creating services in your GMS configuration. This information has been moved in the [Configuring the Digital Channels API](#) section of the Deployment Guide.

The following APIs and responses are supported:

- [Chat API Version 2](#)
- [Chat API Version 2 for CometD](#)
- [Email API](#)
- [Open Media API](#)
- [API Responses](#)

To get the list of options that you can configure for these APIs, refer to the *Genesys Mobile Engagement Configuration Options Guide*:

- [Chat service options](#)
- [Email service options](#)
- [Open Media service options](#)

## Chat API Version 2

Use this API for Web Chat (replacement for eServices WebAPI Chat). For more information, refer to:

- [API Responses](#)
- [Configuring the Digital Channels API](#)

### Request Chat

URI	HTTP Method
/genesys/2/chat/{serviceName}	POST

HTTP Header	Value
Content-Type	application/x-www-form-urlencoded
OR	
Content-Type	multipart/form-data

Parameter Name	Sample Value	Description	Required/Optional
nickname	"JohnDoe"	customer's nickname (either nickname or both firstName and lastName should be supplied)	Required
firstName	"John"	first name of the customer (either nickname or both firstName and lastName should be supplied)	Required
lastName	"Doe"	last name of the customer (either nickname or both firstName and lastName should be supplied)	Required
subject	"Help with account"	subject as entered by the customer	Optional
emailAddress	"jdoe@gmail.com"	email address of the customer	Optional
userData[attachedDataKey1]	value1	any attached data that client wants to add to	Optional

Parameter Name	Sample Value	Description	Required/Optional
		chat	
userData[importantKey2]	value2	any attached data that client wants to add to chat	Optional

## Example

http://localhost:8080/genesys/2/chat/customer-support

### Input

firstName=First  
lastName=Last  
subject=Subject+to

### Output

```
{
  "statusCode":0,
  "alias":"117",
  "userId":"007555677B20000A",
  "secureKey":"4ee15d7e1c343c8e",
  "messages":[
    {
      "from":{"nickname":"First Last","participantId":1,"type":"Client"},
      "index":1,
      "type":"ParticipantJoined",
      "utcTime":1432845088000
    }
  ],
  "chatId":"00048aAPEQJ8000U"
}
```

## Send Message

URI	HTTP Method
/genesys/2/chat/{serviceName}/{chatId}/send	POST

HTTP Header	Value
Content-Type	application/x-www-form-urlencoded

Parameter Name	Sample Value	Description	Required/Optional
message	"I need help with account"	text message to send	Required
userId	"007553863DC30029"	user ID	Required

Parameter Name	Sample Value	Description	Required/Optional
secureKey	"8b31761b2154884c"	secure key	Required
alias	"117"	host alias	Required
messageType	"text"	any arbitrary type that the user wants to set	Optional
transcriptPosition	8	Including this parameter enables return of the transcription and it sets the position in the transcript from where it should be retrieved	Optional

## Example

http://localhost:8080/genesys/2/chat/customer-support/00048aAPEQJ8000S/send

### Input

userId=00755567761A0008  
alias=117  
secureKey=09fa6e8a8691c229  
transcriptPosition=8 message=hello

### Output

```
{
  "messages": [
    {
      "from": {
        "nickname": "First Last",
        "participantId": 1,
        "type": "Client"
      },
      "index": 7,
      "text": "hello",
      "messageType": null,
      "type": "Message",
      "utcTime": 1432845541000
    },
    {
      "from": {
        "nickname": "First Last",
        "participantId": 1,
        "type": "Client"
      },
      "index": 8,
      "text": "hello",
      "messageType": null,
      "type": "Message",
      "utcTime": 1432845548000
    }
  ],
  "chatEnded": false,
  "statusCode": 0,
  "alias": "117",
  "secureKey": "3e2a69d421ae2672",
  "userId": "007555677CB4000D"
}
```

## Start Typing

URI	HTTP Method
/genesys/2/chat/{serviceName}/{chatId}/startTyping	POST

---

HTTP Header	Value
Content-Type	application/x-www-form-urlencoded

Parameter Name	Sample Value	Description	Required/Optional
userId	"007553863DC30029"	user ID	Required
secureKey	"8b31761b2154884c"	secure key	Required
alias	"117"	host alias	Required
message	"message ..... "	message if any	Optional

## Example

http://localhost:8080/genesys/2/chat/customer-support/00048aAPEQJ8000S/startTyping

### Input

userId=007555D17270020  
alias=117  
secureKey=cccfa34d89441faf

### Output

```
{
  messages: null
  chatEnded: false
  statusCode: 0
  alias: "117"
  secureKey: "cccfa34d89441faf"
  userId: "007555D17270020"
}
```

## Stop Typing

URI	HTTP Method
/genesys/2/chat/{serviceName}/{chatId}/stopTyping	POST

HTTP Header	Value
Content-Type	application/x-www-form-urlencoded



Parameter Name	Sample Value	Description	Required/Optional
userId	"007553863DC30029"	user ID	Required
secureKey	"8b31761b2154884c"	secure key	Required
alias	"117"	host alias	Required
message	"message ..... "	message if any	Optional

## Example

http://localhost:8080/genesys/2/chat/customer-support/00048aAPEQJ8000S/stopTyping

### Input

userId=0075555D17270020  
alias=117  
secureKey=cccfa34d89441faf

### Output

```
{
  messages: null
  chatEnded: false
  statusCode: 0
  alias: "117"
  secureKey: "cccfa34d89441faf"
  userId: "0075555D17270020"
}
```

## Refresh Chat

Refresh Chat requests a transcript of events from the specified chat. The value of the **transcriptPosition** parameter determines which events are returned:

- If **transcriptPosition** is set to **0**, **none** of the events from the chat are returned.
- If **transcriptPosition** is set to **1**, **all** of the events from the chat are returned.
- Otherwise, the request returns any new events that have occurred since the event at the position number indicated in the **transcriptPosition** parameter.

In addition to its usefulness in returning the above information, this request can be used to let Chat Server know that the web client that sent the request is still alive.

**Note:** Genesys recommends waiting at least 3 to 5 seconds between calls to the Refresh service.

URI	HTTP Method
/genesys/2/chat/{serviceName}/{chatId}/refresh	POST

HTTP Header	Value
Content-Type	application/x-www-form-urlencoded

Parameter Name	Sample Value	Description	Required/Optional
userId	"007553863DC30029"	user ID	Required
secureKey	"8b31761b2154884c"	secure key	Required
alias	"117"	host alias	Required
transcriptPosition	0 (no messages) 1 (all messages) 2 (all messages starting from 2nd message)	index position in the transcript starting from which the messages should be retrieved	Optional ( <b>All</b> messages are retrieved if no value is provided, which defaults to 1)
message	"Text that user is typing ..."	<p>For use with Typing Preview at the agent chat window. Text that the user has entered so far in the chat box at the time this request is being made. This text will be submitted to the chat server along with notification TypingStarted.</p> <p>Property "typing_preview" must be enabled as described in <a href="#">Chat Services Options</a>.</p>	<p>Applies only to Typing Preview</p> <ul style="list-style-type: none"> <li>If no message is submitted (absent from JSON input), no <b>TypingStarted</b> request is submitted to the chat server</li> <li>If message is submitted (even if it is empty/blank but present in JSON input) a <b>TypingStarted</b> request is submitted along with a <b>MessageText</b> containing this text.</li> </ul> <p>Field is ignored for a regular refresh transcript call.</p>

## Sample responses

### No new events

```
{
  "messages": [],
  "chatEnded": false,
  "statusCode": 0,
  "alias": "249",
  "secureKey": "45327f306556129f",
  "userId": "00F9568B2DA601BA",
  "tenantName": "Resources",
  "nextPosition": 5
}
```

## New message from agent

```
{
  "messages": [
    {
      "from": {
        "nickname": "AgentNick",
        "participantId": 3,
        "type": "Agent"
      },
      "index": 7,
      "text": "hello",
      "type": "Message",
      "utcTime": 1451961875000
    }
  ],
  "chatEnded": false,
  "statusCode": 0,
  "alias": "249",
  "secureKey": "45327f306556129f",
  "userId": "00F9568B2DA601BA",
  "tenantName": "Resources",
  "nextPosition": 8
}
```

## Agent has left; session is closed

(Note that the transcript has a **type** of ParticipantLeft and that **chatEnded** is true.)

```
{
  "messages": [
    {
      "from": {
        "nickname": "AgentNick",
        "participantId": 3,
        "type": "Agent"
      },
      "index": 9,
      "type": "ParticipantLeft",
      "utcTime": 1451961917000
    },
    {
      "from": {
        "nickname": "Maria",
        "participantId": 1,
        "type": "Client"
      },
      "index": 10,
      "type": "ParticipantLeft",
      "utcTime": 1451961917000
    }
  ],
  "chatEnded": true,
  "statusCode": 0,
  "alias": "249",
  "secureKey": "45327f306556129f",
  "userId": "00F9568B2DA601BA",
  "tenantName": "Resources",
  "nextPosition": 1
}
```

## Typing Preview

The message:

- Is the entire value of the text box in which the user is typing.
- Only needs to be sent if it has changed from the previous time it was sent.
- Only needs to be checked for changes every 10 seconds or so. So, for example, if the **Refresh** service is called every 5 seconds, the text box can be checked for changes every 2 iterations.

## Example

<http://localhost:8080/genesys/2/chat/customer-support/00048aAPEQJ8000W/refresh>

### Input

```
userId=007555677CB4000D
alias=117
secureKey=3e2a69d421ae2672
transcriptPosition=20
```

### Output

```
{
  "messages": [
    {
      "from": {
        "nickname": "system",
        "participantId": 2,
        "type": "External"
      },
      "index": 20,
      "text": "agent will be with you shortly ...",
      "messageType": null,
      "type": "Message",
      "utcTime": 1432845749000
    }
  ]
}
```

```
{
  "from": {"nickname": "system", "participantId": 2, "type": "External"},
  "index": 21, "text": "agent will be with you shortly...",
  "messageType": null,
  "type": "Message",
  "utcTime": 1432845767000
},
{
  "chatEnded": false,
  "statusCode": 0,
  "alias": "117",
  "secureKey": "3e2a69d421ae2672",
  "userId": "007555677CB4000D",
  "nextPosition": 22
}
```

## Disconnect

URI	HTTP Method
/genesys/2/ chat/{serviceName}/{chatId}/disconnect	POST

HTTP Header	Value
Content-Type	application/x-www-form-urlencoded

Parameter Name	Sample Value	Description	Required/Optional
userId	"007553863DC30029"	user ID	Required
secureKey	"8b31761b2154884c"	secure key	Required
alias	"117"	host alias	Required

## Example

http://localhost:8080/genesys/2/chat/customer-support  
/00048aAPEQJ8000S/disconnect

### Input

userId=0075555D17270020  
alias=117  
secureKey=cccfa34d89441faf

### Output

```
{
  messages: null
  chatEnded: null
}
```

```
    statusCode: 0
    alias: null
    secureKey: null
    userId: null
  }
```

## Push URL

URI	HTTP Method
/genesys/2/chat/{serviceName}/{chatId}/pushUrl	POST

HTTP Header	Value
Content-Type	application/x-www-form-urlencoded

Parameter Name	Sample Value	Description	Required/Optional
userId	"007553863DC30029"	user ID	Required
secureKey	"8b31761b2154884c"	secure key	Required
alias	"117"	host alias	Required
pushUrl	" <a href="#">http://url.to.see</a> "	URL	Required

## Example

<http://localhost:8080/genesys/2/chat/customer-support/00048aAPEQJ8000S/pushUrl>

### Input

```
userId=0075555D17270020
alias=117
secureKey=cccfa34d89441faf
pushUrl=url.to.see
```

### Output

```
{
  messages: null
  chatEnded: false
  statusCode: 0
  alias: "117"
  secureKey: "09fa6e8a8691c229"
  userId: "00755567761A0008"
}
```

## Update Nickname

URI	HTTP Method
/genesys/2/ chat/{serviceName}/{chatId}/updateNickname	POST

HTTP Header	Value
Content-Type	application/x-www-form-urlencoded

Parameter Name	Sample Value	Description	Required/Optional
userId	"007553863DC30029"	user ID	Required
secureKey	"8b31761b2154884c"	secure key	Required
alias	"117"	host alias	Required
nickname	John Doe 2	new nickname	Required

## Example

http://localhost:8080/genesys/2/chat/customer-support  
/00048aAPEQJ8000S/updateNickname

### Input

userId=00755567761A0008  
alias=117  
secureKey=09fa6e8a8691c229  
nickname=newName

### Output

```
{
  messages: null
  chatEnded: false
  statusCode: 0
  alias: "117"
  secureKey: "09fa6e8a8691c229"
  userId: "00755567761A0008"
}
```

## Custom Notice

URI	HTTP Method
/genesys/2/ chat/{serviceName}/{chatId}/customNotice	POST

---

HTTP Header	Value
Content-Type	application/x-www-form-urlencoded

Parameter Name	Sample Value	Description	Required/Optional
userId	"007553863DC30029"	user ID	Required
secureKey	"8b31761b2154884c"	secure key	Required
alias	"117"	host alias	Required
message	"message ..."	message, if any	Optional

## Example

http://localhost:8080/genesys/2/chat/customer-support/00048aAPEQJ8000S/customNotice

### Input

userId=00755567761A0008  
alias=117  
secureKey=09fa6e8a8691c229  
message=custom+message

### Output

```
{
  messages: null
  chatEnded: false
  statusCode: 0
  alias: "117"
  secureKey: "09fa6e8a8691c229"
  userId: "00755567761A0008"
}
```

## Update User Data

URI	HTTP Method
/genesys/2/chat/{serviceName}/{chatId}/updateData	POST

HTTP Header	Value
Content-Type	application/x-www-form-urlencoded

---

Parameter Name	Sample Value	Description	Required/Optional
userId	"007553863DC30029"	user ID	Required
secureKey	"8b31761b2154884c"	secure key	Required
alias	"117"	host alias	Required
userData[key1]	value1	user data to be updated	Required
userData[key2]	value2	user data to be updated	Required

## Example

http://localhost:8080/genesys/2/chat/customer-support/00048aAPEQJ8000S/updateData

### Input

(URL encoded, raw)  
userId=0075556DEA820000&alias=117&secureKey=7d1f6f3ff2e60cd6  
&userData%5Bkey1%5D=value1&userData%5Bkey2%5D=value2

(Form, user input, unencoded)  
userId=0075556DEA820000  
alias=117  
secureKey=7d1f6f3ff2e60cd6  
userData[key1]=value1  
userData[key2]=value2

### Output

```
{
  messages: null
  chatEnded: false
  statusCode: 0
  alias: "117"
  secureKey: "09fa6e8a8691c229"
  userId: "00755567761A0008"
}
```

## Chat Session File Management

### Updated in 8.5.110.07

Starting with Release 8.5.106, the GMS Digital Channels API allows agents and customers to exchange files during their chat sessions using these requests:

- **Get Limits**—Check for allowable file types and file size—or other constraints on file uploads—before sending an upload request.
- **Upload File**—Upload a file.
- **Download File**—Download a previously uploaded file.



- **Delete File**—Delete a previously uploaded the file.

## Get Limits

Use this optional request to avoid wasting network and CPU overhead by checking for allowable file types or maximum file size—or other constraints on file uploads—before sending an upload request.

URI	HTTP Method
/genesys/2/chat/{serviceName}/{chatId}/file/limits	POST

HTTP Header	Value
Content-Type	<ul style="list-style-type: none"><li>• multipart/form-data</li><li>• application/x-www-form-urlencoded (starting in 8.5.110.07)</li></ul>

## Input Parameters

Parameter Name	Sample Value	Description	Required/ Optional	Source
userId	"007553863DC30029"	user ID	Required	Form
secureKey	"8b31761b2154884c"	secure key	Required	Form
alias	"117"	host alias	Required	Form

## Output Parameters

Parameter	Description
download-attempts	Maximum number of times a specific file can be downloaded during this session
upload-max-files	Maximum number of files a client can upload during this session
upload-max-file-size	Maximum allowable file size of a single uploaded file
upload-max-total-size	Maximum allowable file size of all uploaded files
upload-need-agent	Indicates whether an agent needs to accept a chat session before an upload is allowed
upload-file-types	Colon-delimited list of file extensions indicating

Parameter	Description
	which types of files can be uploaded
used-upload-max-files	Current number of files uploaded during this session
used-upload-max-total-size	Current total size of all files uploaded during this session
used-download-attempts	Current number of downloaded files during this session
delete-file	Indicates whether <b>upload-max-files</b> is decremented after a file has been deleted

## Example

http://localhost:8080/genesys/2/chat/customer-support/00048aAPEQJ8000U/file/limits

## Input

userId=007553863DC30029  
secureKey=8b31761b2154884c  
alias=117

## Output

```
{
  "messages": null,
  "chatEnded": false,
  "statusCode": 0,
  "alias": "240",
  "secureKey": "799c5ff0faccd5bb",
  "userId": "00F05702F26B0006",
  "userData": {
    "download-attempts": "10",
    "upload-max-files": "10",
    "delete-file": "odd",
    "upload-max-file-size": "100000000",
    "used-download-attempts": "0",
    "used-upload-max-total-size": "0",
    "upload-need-agent": "true",
    "used-upload-max-files": "0",
    "upload-max-total-size": "500000000",
    "upload-file-types": "bmp:csv:doc:docx:exe:gif:htm:jpg:pdf:png:ppt:pptx:tif:txt:xls:xlsx:zip"
  }
}
```

## Upload File

Use this request to upload a file into a chat session so it can be shared with an agent or a customer. You can then use the **file-id** value in the response to delete or download the file, as needed.

URI	HTTP Method
/genesys/2/chat/{serviceName}/{chatId}/file	POST

HTTP Header	Value
Content-Type	multipart/form-data

Parameter Name	Sample Value	Description	Required/ Optional	Source
userId	"007553863DC30029"	user ID	Required	Form
secureKey	"8b31761b2154884c"	secure key	Required	Form
alias	"117"	host alias	Required	Form
file	"afile.txt"	file to be uploaded into the session	Required	Form
userData[file-description]	"March invoice"	This value can be used by the agent desktop and the client UI to provide additional file info	Optional	Form
userData[userKey1]	"User value 1"	Collection of key-value pairs that provides file metadata	Optional	Form

## Example

http://localhost:8080/genesys/2/chat/customer-support/00048aAPEQJ8000U/file

## Input

```
userId=007553863DC30029
secureKey=8b31761b2154884c
alias=117
file=<some file to upload>
```

## Output

```
{
  "messages": [],
  "chatEnded": false,
  "statusCode": 0,
  "alias": "240",
  "secureKey": "77cd1c487b67fefb",
  "userId": "00F057DB0FC30001",
  "chatId": "0001KaBWNVUV000K",
  "userData": {
```

```
    "file-id": "00F057DB0FF10005"
  },
  "nextPosition": 6
}
```

## Download File

Use this request to download a file that was uploaded into a chat session either by an agent or a customer. The `fileId` parameter can be retrieved from the previous transcript event (see response for Refresh Chat request) indicating a file was upload.

HTTP Method	URI
POST	/genesys/2/ chat/{serviceName}/{chatId}/file/{fileId}/download

HTTP Header	Value
Content-Type	<ul style="list-style-type: none"><li>multipart/form-data</li><li>application/x-www-form-urlencoded (starting in 8.5.110.07)</li></ul>

Parameter Name	Sample Value	Description	Required/ Optional	Source
userId	"007553863DC30029"	user ID	Required	Form
secureKey	"8b31761b2154884c"	secure key	Required	Form
alias	"117"	host alias	Required	Form
fileId	"00048aAPEQJ8ABCD"	File to be downloaded from the session	Required	URL

## Example

```
http://localhost:8080/genesys/2/chat/customer-support  
/00048aAPEQJ8000U/file/00048aAPEQJ8ABCD
```

### Input

```
userId=007553863DC30029  
secureKey=8b31761b2154884c  
alias=117
```

### Output

If the operation is successful, the contents of the file are downloaded. In case of an error, the server returns an HTML 500 response that includes the **referenceId** of the request.

The following HTML snippet shows how to download a file using an HTML form:

```
var form = $("<form method='POST' enctype='multipart/form-data'
target='_blank'
action='http://GMS_HOST:GMS_PORT/genesys/2/chat/customer-
support/0001JaBUS5FD002U/file/00F057C8BFA20052/download'>
<input name='alias' value='240' type='hidden'><input name='secureKey'
value='6b46a7a8910f21be' type='hidden'><input name='userId'
value='00F057C8B6B7004D' type='hidden'></form>");

form.submit();
```

## Delete File

Use this request to delete a file that was uploaded into a chat session by a customer if allowed by Chat Server settings. Web User has no permission to delete file uploaded by the agent.

URI	HTTP Method
/genesys/2/ chat/{serviceName}/{chatId}/file/{fileId}/delete	POST

HTTP Header	Value
Content-Type	<ul style="list-style-type: none"> <li>multipart/form-data</li> <li>application/x-www-form-urlencoded (starting in 8.5.110.07)</li> </ul>

Parameter Name	Sample Value	Description	Required/ Optional	Source
userId	"007553863DC30029"	user ID	Required	Form
secureKey	"8b31761b2154884c"	secure key	Required	Form
alias	"117"	host alias	Required	Form
fileId	"00048aAPEQJ8ABCD"	The file to be downloaded from the session	Required	URL

## Example

```
http://localhost:8080/genesys/2/chat/customer-support
/00048aAPEQJ8000U/file/00048aAPEQJ8ABCD/delete
```

**Input**

```
userId=007553863DC30029  
secureKey=8b31761b2154884c  
alias=117
```

**Output**

```
{  
  "messages": [],  
  "chatEnded": false,  
  "statusCode": 0,  
  "alias": "240",  
  "secureKey": "ad437d2338d09d09",  
  "userId": "00F057DB125E000A",  
  "chatId": "0001KaBWNVUV0011",  
  "nextPosition": 7  
}
```

# Chat API Version 2 with CometD

## Introduced in 8.5.109

Versions	Changes
8.5.109	Genesys Mobile Enterprise supports the Digital Channels Chat API V2 over CometD channel. Most of the time, requests, responses, parameters names, and error codes are identical between the REST and CometD APIs.
8.5.110	GMS supports CometD WebSocket Transport. To enable Secure WebSocket, deploy GMS with an <b>SSL certificate</b> .
8.5.112	<ul style="list-style-type: none"> <li>Digital Channels Chat API V2 over CometD now supports File Transfer operations.</li> <li>Chat API with CometD usability is enhanced and request parameters such as <code>alias</code>, <code>userId</code>, and <code>chatId</code> are now deprecated. To ensure backward compatibility, GMS nodes will still include those keys with bogus values in notification events.</li> </ul>
8.5.114	<ul style="list-style-type: none"> <li>Chat V2 API over CometD now supports Firebase Cloud Messaging push notifications, that allow delivering notifications of new chat events in the outgoing chat session to a mobile terminal, while the main application is offline or in background.</li> <li>Chat V2 API over CometD now supports native push notifications through Apple Push Notification Service. APNS support is limited to <b>Legacy Binary Provider API</b>.</li> </ul>

## Prerequisites for the CometD API

Server	Minimum version	Configuration details
GMS	8.5.109.05+	<ul style="list-style-type: none"> <li>GMS can run in chat only or in full mode.</li> <li>To enable the notification mode, set to <code>true</code> the option <code>enable_notification_mode</code> in</li> </ul>

		<p>the chat section.</p> <ul style="list-style-type: none"> <li>If you use a load balancer in front of several GMS nodes, enable the sticky sessions.</li> </ul> <p>The option <code>enable_notification_mode</code> is not dynamic. You cannot modify it at runtime.</p>
Chat Server	8.5.108+	<ul style="list-style-type: none"> <li>All the connected Chat Server instances need to be deployed in N+1 configuration with session restoration.</li> <li>Enable the notification mode by setting to <code>true</code> the option <code>flex-push-enabled</code> in the settings section.</li> <li>Configure a <b>digital channel</b></li> </ul>

### Important

Configuration changes to the connected Chat Server instance disable this instance for serving notifications.

- Do all changes when GMS is offline. Note that you don't need to stop the Chat Server to process the configuration changes.
- If you absolutely need to keep GMS online, remove the connection to the Chat Server instance from the GMS cluster, do configuration changes on the instance, and add the connection to the Chat Server again in the GMS cluster.

## How to use the CometD API

When you are using the CometD API, both client and server applications exchange messages with a JSON payload. The client application can request an operation from the GMS server, and the GMS server responds with notifications. GMS also delivers unsolicited notifications when an event happens during the chat session. All operations are asynchronous, which means that notifications will respond and confirm the requested operations.



## Handshake with GMS

The first step is a handshake with the GMS node, using the full GMS URI; for instance:  
`http://<gms-hostname>:8080/genesys/cometd`

## Subscribe for a channel

After a successful handshake, you need to either subscribe to or add a listener for your channel. The channel name depends on the **digital channel** that you created before. It includes the chat service name and is formatted as follows: `/service/chatV2/<chat service>`.

For instance, if you create and configure the customer-support chat service, the associated channel will be: `/service/chatV2/customer-support`

Example of handshake and channel listener:

```
var gmsCometURL = http://my.gms.host:8080/genesys/cometd
var chatChannel = /service/chatV2/customer-support

function handleChatEvent( message ) { ... }

cometd.addListener( chatChannel, handleChatEvent )
cometd.configure({
  url: gmsCometURL,
  logLevel: 'debug'
});
cometd.handshake()
```

Example of handshake and subscription:

```
var gmsCometURL = http://my.gms.host:8080/genesys/cometd
var chatChannel = /service/chatV2/customer-support

function handleChatEvent( message ) { ... }

function _handleHandshake( handshake ) {
  if ( true === handshake.successful ) {
    cometd.batch( function() { cometd.subscribe( chatChannel, handleChatEvent ); } );
  }
}

cometd.addListener( '/meta/handshake', _handleHandshake )
cometd.configure({
  url: gmsCometURL,
  logLevel: 'debug'
});
cometd.handshake()
```

## Enable Push Notifications

### Introduced in 8.5.114

## Configure Firebase Cloud Messaging in GMS

Refer to the **Push Notification Service** documentation to configure your Firebase Cloud Messaging

provider. You can define several different **provider** sections and associate any of them when starting a chat service with the `requestChat` and `requestNotifications` API queries.

- If you do not add provider information to the user data, the system will use the default Firebase Cloud Messaging options that you configured in the push section of the GMS configuration.
- If you specify a non-default provider in the user data of the query, the system will use the Firebase Cloud Messaging options that are configured in the `[push.provider.{ProviderName}]` section, where `{ProviderName}` is the string to provide in the `push_notification_provider` user data of the `requestChat` operation.

You can also use `fcm.title` and `fcm.body` options to extend your configuration by creating an event-level `[push.provider.{ProviderName}.event]` section and then, a specific service name and event. The following example shows how to configure in GMS two chat services for Bank's Saving Account and use localized messages in English and Russian:

## [+] See the configuration example

```
[chat.savings-english]
endpoint = Environment:SavingsEnglish

[chat.savings-russian]
endpoint = Environment:SavingsRussian

[push.provider.bankoperations]
pushEnabled=ios,fcm
fcm.apiKey=****
apple.keystore=/var/genesys/gms/appleKeystore.p12
apple.keystorePassword=****

[push.provider.bankoperations.event]
fcm.body="Please open app for more details"

[push.provider.bankoperations.event.chat.savings-english.ParticipantJoined]
fcm.title="Agent has joined an waiting"

[push.provider.bankoperations.event.chat.savings-english.Message]
fcm.title="You got new message from us"
fcm.body="Please answer us soon!"

[push.provider.bankoperations.event.chat.savings-russian.ParticipantJoined]
fcm.title="Агент присоединился и ждет"
fcm.body="Ответьте нам поскорее"

[push.provider.bankoperations.event.chat.savings-russian.Message]
fcm.title="У Вас новое сообщение!"
fcm.body="Ответьте нам поскорее"
```

If you do not configure FCM title and body options, the system will use the English default values.

Event Name	title default value	body default value	Comments
ParticipantJoined	"Chat participants changed"	"%s joined chat"	%s is substituted with the party name
ParticipantLeft	"Chat participants changed"	"%s left chat"	

Event Name	title default value	body default value	Comments
Message	"%s sent new message"	The message sent by the participant.	
PushUrl	"%s suggests Web page"	URL pushed by the participant.	
FileUploaded	"%s suggests to download file"	"Open app for more info"	
IdleAlert	"Chat will close soon"	"Please use app if you want to continue"	
IdleClose	"Chat is over"	"Your chat session was ended due to inactivity"	

### Important

The FileDeleted, CustomNotice, and Notice events are not pushed.

## Provide Firebase Cloud Messaging Parameters in User Data

When your client application submits `requestChat` and `requestNotifications` queries, the request must include the `push_notification_*` in its user data. These data identify the mobile terminal and your client app must provide the exact same data each time it re-establishes a CometD connection and issues the `requestNotifications` operation.

### Important

The `push_notification_*` user data are detailed in this page in the [requestChat](#) and [requestNotifications](#) sections.

## Firebase Push Notification Limitations

- Only one session per client is supported. If the same user tries to login into the mobile app from two devices, only one device will be able to manage the chat session in progress.
- The client app must create the chat session over Chat V2 with CometD in order to receive notifications. Furthermore, the client app must use Chat V2 API with CometD whenever it is sent in foreground. On background switchovers, Genesys recommends to perform a CometD disconnection or to unload the app from memory to preserve the device's battery.
- As stated above, default notifications are submitted in English only. To support additional languages,

you must provision the `fcml.title` and `fcml.body` options at the provider service event level, or intercept notifications on the terminal, process them in the background to change text, and present a new notification to the user.

## Receive Notifications

To start receiving notifications, you can either:

- Use the `requestChat` operation to request a new chat for your channel.
- Request notifications for an existing chat through the `requestNotifications` operation. In this scenario, you must specify the channel where the chat was created.

If the `requestChat` operation succeeds, the client receives a notification about joining the session and, later, further notifications on each session event.

- Your client application should read the parameters named `chatId`, `userId`, `secureKey`, and `alias` from the notification response, and reuse them with each subsequent request message.
- Each chat event has an index matching the event order.
- All unsolicited notifications will have exactly one event. Notifications sent as a response to a request may have one or no events. The notifications sent in response to the `requestChat` and `requestNotifications` requests may have 0, 1, or more events.

When your client application processes events, it needs to check the index of each event received and compare it with the index of the last processed event. If the index is identical or smaller, the event can be ignored. If the index of the new received event is greater than the index of the last processed event, the client application should update the UI with the new event, as appropriate, and remember the new index as the last processed event.

### Important

Starting with release 8.5.112, the request parameters such as `alias`, `userId`, and `chatId` are deprecated.

## Handling disconnections

In case of disconnection, the client application should perform a new handshake if necessary, and use the `requestNotifications` operation with the following information:

- The latest known transcript position
- Identical `userId`, `chatId`, and `secureKey` parameters

If the chat session is still ongoing, the client will receive a notification with a new (possibly the same) `alias` to use, and the list of the chat events that happened since the last known position.

## Authentication

If you enable Basic Authentication for the chat service accessed through the CometD API, you must provide the authentication credentials in the auth parameter of the requestChat operation. There are two ways to provide credentials in an auth object:

- In an open form containing the username and password fields
- In an encoded form using encoded fields, similar to the Basic Authentication header, which is a Base64-encoded composite string of "username:password"

Here is an open-form example of a request message with authentication credentials:

```
var channel = "/service/chatV2/customer-support";
var requestChatData = {
  "operation": "requestChat",
  "firstName": "Joan",
  "lastName": "Smith",
  "subject": "Savings Account",
  "userData": {
    "key1": "value1",
    "key2": "value2"
  },
  "auth": {
    "username": "user1",
    "password": "pass1"
  }
}
cometd.publish( channel, requestChatData);
```

Here is an encoded-form example of a request message with authentication credentials:

```
var channel = "/service/chatV2/customer-support";
var requestChatData = {
  "operation": "requestChat",
  "firstName": "Joan",
  "lastName": "Smith",
  "subject": "Savings Account",
  "userData": {
    "key1": "value1",
    "key2": "value2"
  },
  "auth": {
    "encoded": "dXNlcjE6cGFzc2E="
  }
}
cometd.publish( channel, requestChatData);
```

## CometD API

### Request Chat

**Modified in 8.5.112**

Use this method to request a new chat session and start receiving the associated notifications. If the requestChat operation is successful, you will receive a notification response data object that contains the parameters required for subsequent messages and operations:

- chatId—ID for the new chat session
- userId—User ID used for the chat session
- secureKey—Secure key for this session
- alias—Chat host alias used for the session

See also the [Subscribe](#) and [Authentication](#) sections for further use cases and details.

### Operation

Operation	requestChat		
Parameter	Type	Mandatory	Description
nickname	string	yes	Customer's nickname. Genesys recommends that you supply either nickname or both firstName and lastName parameters. For example: "JohnDoe"
lastName	string	no	Last name of the customer. Genesys recommends that you supply either nickname or both firstName and lastName parameters. For example: "Doe"
subject	string	no	The subject as entered by the customer. For example: "Help with account"
emailAddress	string	no	The email address of the customer. For example: "jdoe@gmail.com"
userData	JSON-formatted string	no	Any attached data that the client wants to add to the chat session. For example: { "key": "value", ... }

#### Important

Starting in 8.5.112, the request parameters such as alias, userId, and chatId are deprecated.

Starting in 8.5.114, you can add the following user data (that must be identical in the requestNotifications query) to enable Firebase Cloud Messaging push notifications.

User Data Key	Value	Required	Description
push_notification_deviceid	ID	Yes	Unique deviceid or token received from FCM or APNS. Your client app must provide the exact same data each time it re-establishes a CometD connection and issues the requestNotifications operation.
push_notification_type	fcm	Yes	<ul style="list-style-type: none"> <li>"fcm" for Google Firebase Messaging client</li> </ul> <p>Your client app must provide the exact same data each time it re-establishes a CometD connection and issues the requestNotifications operation.</p>
push_notification_debug	"true" "false" (default)	No	If true, debug mode for Firebase Cloud Messaging or Apple notification service will be used. Note if your Apple certificate is available for debug mode only, set this flag to true; otherwise, the device will not receive notifications.
push_notification_language	ISO language code	No	Reserved for now. Defines the language for the client application. The language is en_US by default.
push_notification_provider	string	No	<p>Defines the name of the GMS provider configuration section to use for this chat session. If you do not provide a name, the system will use the default provider defined in the push section.</p> <div> <b>Important</b>  Genesys recommends to </div>

User Data Key	Value	Required	Description
			use short strings that include short words and exclude any special symbols or punctuation.

## Example

The following is an example of a request.

```
var channel = "/service/chatV2/customer-support";
var requestChatData = {
  "operation": "requestChat",
  "firstName": "Joan",
  "lastName": "Smith",
  "subject": "Savings Account",
  "userData": {
    "key1": "value1",
    "key2": "value2"
  }
}
cometd.publish(channel, requestChatData);
```

The following is an example of the notification response.

```
{
  "statusCode": 0,
  "alias": "117",
  "userId": "007555677B20000A",
  "secureKey": "4ee15d7e1c343c8e",
  "messages": [
    {
      "from": {
        "nickname": "Joan Smith",
        "participantId": 1,
        "type": "Client"
      },
      "index": 1,
      "type": "ParticipantJoined",
      "utcTime": 1432845088000
    }
  ],
  "chatId": "00048aAPEQJ8000U",
  "nextPosition": 2
}
```

## Request Notifications

### Modified in 8.5.112

Use this operation to request notifications to be delivered for the existing chat session, after the CometD channel has been disconnected.



## Operation

Operation	requestNotifications		
Parameter	Type	Mandatory	Description
chatId <b>Deprecated in 8.5.112</b>	Long	yes	Chat ID that GMS sent in response to the requestChat operation. For example: "00048aAPEQJ800U"
userId <b>Deprecated in 8.5.112</b>	Long	yes	User ID that GMS sent in response to the requestChat operation. For example: "007555677B20000A"
secureKey	Long	yes	Secure key that GMS sent in response to the requestChat operation. For example: "4ee15d7e1c343c8e"
alias <b>Deprecated in 8.5.112</b>	string	yes	Chat host alias.
transcriptPosition	integer	no	The transcript event position from which the client application would like to receive the previous events. If you set this option to 0 or if you don't set this option, the client will receive all the events.

**Important**

Starting in 8.5.112, the request parameters such as alias, userId, and chatId are deprecated.

Starting in 8.5.114, you can add the following user data (that must be identical in the requestChat query) to enable Firebase Cloud Messaging push notifications.

User Data Key	Value	Required	Description
push_notification_deviceid ID		Yes	Unique deviceId or token received from FCM or APNS. Your client app must provide the exact same data each time it re-establishes a CometD connection and issues the

User Data Key	Value	Required	Description
			requestNotifications operation.
push_notification_type	fcm	Yes	<ul style="list-style-type: none"> <li>"fcm" for Google Firebase Messaging client</li> </ul> <p>Your client app must provide the exact same data each time it re-establishes a CometD connection and issues the requestNotifications operation.</p>
push_notification_debug	"true" "false" (default)	No	If true, debug mode for Firebase Cloud Messaging or Apple notification service will be used. Note if your Apple certificate is available for debug mode only, set this flag to true, otherwise, the device will not receive notifications.
push_notification_language	ISO language code	No	Reserved for now. Defines the language for the client application. The language is en_US by default.
push_notification_provider	string	No	<p>Defines the name of the GMS provider configuration section to use for this chat session. If you do not provide a name, the system will use the default provider defined in the push section.</p> <div> <b>Important</b>            Genesys recommends to use short strings that include short words and exclude any special symbols or punctuation.         </div>

## Example

Example of request:

```
var channel = "/service/chatV2/customer-support";
var requestNotificationData = {
```

```

    "operation": "requestNotifications",
    "alias": "117",
    "chatId": "00048aAPEQJ8000U",
    "userId": "007555677B20000A",
    "secureKey": "4ee15d7e1c343c8e",
    "transcriptPosition": "2"
  }
  cometd.publish( channel, requestNotificationData );

```

Example of received notification:

```

{
  "messages": [
    {
      "from": {
        "nickname": "Joan Smith",
        "participantId": 1,
        "type": "Client"
      },
      "index": 2,
      "text": "Hello, ...",
      "messageType": "text",
      "type": "Message",
      "utcTime": 1432845541000
    },
  ],
  "chatEnded": false,
  "statusCode": 0,
  "alias": "117",
  "secureKey": "3e2a69d421ae2672",
  "userId": "007555677CB4000D",
  "chatId": "00048aAPEQJ8000U",
  "nextPosition": 2
}

```

## Send message

Use this operation to deliver or send messages from the client application to the chat session.

Operation	sendMessage		
Parameter	Type	Mandatory	Description
message	string	yes	Text message to send. For example: "I need help with my account."
chatId <b>Deprecated in 8.5.112</b>	long	yes	The chat ID. For example: "00048aAPEQJ800U"
userId <b>Deprecated in 8.5.112</b>	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias <b>Deprecated in</b>	string	yes	Chat host alias. For example: "117"

Operation	sendMessage		
<b>8.5.112</b>			
messageType	string	no	Any arbitrary type that the user wants to set. For example: "text"

## Example

Example of request:

```
var channel = "/service/chatV2/customer-support";
var sendData = {
  "operation": "sendMessage",
  "message": "Hello, ...",
  "chatId": "00048aAPEQJ8000U",
  "userId": "007555677B20000A",
  "secureKey": "4ee15d7e1c343c8e"
}
cometd.publish( channel, sendData );
```

Example of notification response:

```
{
  "messages": [
    {
      "from": {
        "nickname": "Joan Smith",
        "participantId": 1,
        "type": "Client"
      },
      "index": 2,
      "text": "Hello, ...",
      "messageType": "text",
      "type": "Message",
      "utcTime": 1432845541000
    },
  ],
  "chatEnded": false,
  "statusCode": 0,
  "alias": "117",
  "secureKey": "3e2a69d421ae2672",
  "userId": "007555677CB4000D",
  "chatId": "00048aAPEQJ8000U",
  "nextPosition": 3
}
```

## Start Typing

Use this operation to indicate that the client has started typing a message. You can include a partial message, if desired, for typing preview.

### Operation

Operation	startTyping		
Parameter	Type	Mandatory	Description

Operation	startTyping		
chatId <b>Deprecated in 8.5.112</b>	long	yes	The chat ID. For example: "00048aAPEQJ800U"
userId <b>Deprecated in 8.5.112</b>	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias <b>Deprecated in 8.5.112</b>	string	yes	Chat host alias. For example: "117"
message	string	no	Optional preview typing message to send. For example: "I need help with my account."

## Example

Example of request:

```
var channel = "/service/chatV2/customer-support";
var startTypingData = {
  "operation": "startTyping",
  "message": "Hello, ...",
  "chatId": "00048aAPEQJ8000U",
  "userId": "007555677B20000A",
  "secureKey": "4ee15d7e1c343c8e"
}
cometd.publish( channel, sendData );
```

Example of notification response:

```
{
  "messages": [
    {
      "from": {
        "nickname": "John Smith",
        "participantId": 1,
        "type": "Client"
      },
      "index": 3,
      "text": "hello, I ha",
      "type": "TypingStarted",
      "utcTime": 1493509617000
    }
  ],
  "chatEnded": false,
  "statusCode": 0,
  "alias": "117",
  "secureKey": "3e2a69d421ae2672",
  "userId": "007555677CB4000D",
  "chatId": "00048aAPEQJ8000U",
  "nextPosition": 4
}
```

## Stop Typing

Use this operation to indicate that the client has stopped typing a message. You can include a partial message, if desired, for typing preview.

### Operation

Operation	stopTyping		
Parameter	Type	Mandatory	Description
chatId <b>Deprecated in 8.5.112</b>	long	yes	The chat ID. For example: "00048aAPEQJ800U"
userId <b>Deprecated in 8.5.112</b>	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias <b>Deprecated in 8.5.112</b>	string	yes	Chat host alias. For example: "117"
message	string	no	Optional preview typing message to send. For example: "I need help with my account."

### Example

Example of request message:

```
var channel = "/service/chatV2/customer-support";
var sendData = {
  "operation": "stopTyping",
  "message": "Hello, ...",
  "chatId": "00048aAPEQJ8000U",
  "userId": "007555677B20000A",
  "secureKey": "4ee15d7e1c343c8e"
}
cometd.publish( channel, sendData );
```

Example of notification response data object:

```
{
  "messages": [
    {
      "from": {
        "nickname": "John Smith",
        "participantId": 1,
        "type": "Client"
      },
      "index": 4,
      "text": "hello, I have a question",
      "type": "TypingStopped",
      "utcTime": 1493509618000
    }
  ]
}
```

```
],
"chatEnded": false,
"statusCode": 0,
"alias": "117",
"secureKey": "3e2a69d421ae2672",
"userId": "007555677CB4000D",
"chatId" : "00048aAPEQJ8000U",
"nextPosition": 5
}
```

## Disconnect

Use this operation to finish the chat session. The notification response does not include a secureKey or userId parameter. Your application needs to clean up chatId, userId, and secureKey values and forget about them. Any subsequent operation with these IDs will generate error notification responses.

### Operation

Operation	disconnect		
Parameter	Type	Mandatory	Description
chatId <b>Deprecated in 8.5.112</b>	long	yes	The chat ID. For example: "00048aAPEQJ800U"
userId <b>Deprecated in 8.5.112</b>	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias <b>Deprecated in 8.5.112</b>	string	yes	Chat host alias. For example: "117"

### Example

Example of request message:

```
var channel = "/service/chatV2/customer-support";
var disconnectData = {
  "operation": "disconnect",
  "alias": "117",
  "chatId": "00048aAPEQJ8000U",
  "userId": "007555677B20000A",
  "secureKey": "4ee15d7e1c343c8e"
}
cometd.publish( channel, disconnectData );
```

Example of notification response:

```
{
  "messages": [],
  "chatEnded": true,
  "statusCode": 0,
}
```

```
"alias": "117",  
"chatId": "00048aAPEQJ8000U",  
"nextPosition": 1  
}
```

## Push URL

Use this operation to push a web page to the agent.

### Operation

Operation	pushUrl		
Parameter	Type	Mandatory	Description
chatId <b>Deprecated in 8.5.112</b>	long	yes	The chat ID. For example: "00048aAPEQJ8000U"
userId <b>Deprecated in 8.5.112</b>	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias <b>Deprecated in 8.5.112</b>	string	yes	Chat host alias. For example: "117"
pushUrl	string	yes	URL to push to the agent. For example: "http://www.genesys.com"

### Example

Example of request.

```
var channel = "/service/chatV2/customer-support";  
var pushUrlData = {  
  "operation": "pushUrl",  
  "pushUrl": "http://www.genesys.com",  
  "chatId": "00048aAPEQJ8000U",  
  "userId": "007555677B20000A",  
  "secureKey": "4ee15d7e1c343c8e"  
}  
cometd.publish( channel, pushUrlData );
```

Example of notification response:

```
{  
  "messages": [  
    {  
      "from": {  
        "nickname": "John Smith",  
        "participantId": 1,  
        "type": "Client"  
      },  
      "index": 2,  
    },  
  ],  
}
```



```

        "text": "http://www.google.com",
        "type": "PushUrl",
        "utcTime": 1493250733000
    }
],
"chatEnded": false,
"statusCode": 0,
"alias": "117",
"secureKey": "3e2a69d421ae2672",
"userId": "007555677CB4000D",
"chatId" : "00048aAPEQJ8000U",
"nextPosition": 3
}

```

## Update Nickname

Use this operation to update the customer's nickname.

### Operation

Operation	updateNickname		
Parameter	Type	Mandatory	Description
chatId <b>Deprecated in 8.5.112</b>	long	yes	The chat ID. For example: "00048aAPEQJ800U"
userId <b>Deprecated in 8.5.112</b>	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias <b>Deprecated in 8.5.112</b>	string	yes	Chat host alias. For example: "117"
nickname	string	yes	New nickname. For example: "John Doe 2"

### Example

Example of request:

```

var channel = "/service/chatV2/customer-support"
...
var updateNicknameData = {
  "operation": "updateNickname",
  "nickname" : "MyNewNickname",
  "chatId" : "00048aAPEQJ8000U",
  "userId": "007555677B20000A",
  "secureKey": "4ee15d7e1c343c8e"
}
cometd.publish( channel, updateNicknameData );

```

Example of notification response:

```
{
  "messages": [{
    "from": { "nickname": "MyNewNickname", "participantId": 1, "type": "Client" },
    "index": 8,
    "text": "MyNewNickname",
    "type": "NicknameUpdated",
    "utcTime": 1493936549000
  }],
  "chatEnded": false,
  "statusCode": 0,
  "alias": "117",
  "secureKey": "3e2a69d421ae2672",
  "userId": "007555677CB4000D",
  "chatId": "00048aAPEQJ8000U",
  "nextPosition": 9
}
```

## Custom Notice

Use this operation to send a custom notice to the agent. The agent will need a customized desktop that can process this notice.

### Operation

Operation	customNotice		
Parameter	Type	Mandatory	Description
chatId <b>Deprecated in 8.5.112</b>	long	yes	The chat ID. For example: "00048aAPEQJ8000U"
userId <b>Deprecated in 8.5.112</b>	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias <b>Deprecated in 8.5.112</b>	string	yes	Chat host alias. For example: "117"
message	string	no	Optional message.

### Example

Example of request message:

```
var channel = "/service/chatV2/customer-support"
...
var customNoticeData = {
  "operation": "customNotice",
  "message": "ORDER UPDATE",
  "chatId": "00048aAPEQJ8000U",
  "userId": "007555677B20000A",
  "secureKey": "4ee15d7e1c343c8e"
}
cometd.publish( channel, customNoticeData );
```

Example of notification response:

```
{
  "messages": [
    {
      "from": {
        "nickname": "John Smith",
        "participantId": 1,
        "type": "Client"
      },
      "index": 8,
      "text": "ORDER UPDATE",
      "type": "CustomNotice",
      "utcTime": 1493510733000
    }
  ],
  "chatEnded": false,
  "statusCode": 0,
  "alias": "117",
  "secureKey": "3e2a69d421ae2672",
  "userId": "007555677CB4000D",
  "chatId": "00048aAPEQJ8000U",
  "nextPosition": 9
}
```

## Update User Data

Use this operation to modify or add attached data to the chat session.

### Operation

Operation	updateData		
Parameter	Type	Mandatory	Description
userId <b>Deprecated in 8.5.112</b>	long	yes	The user ID. For example: "007555677B20000A"
secureKey	long	yes	Secure key. For example: "4ee15d7e1c343c8e"
alias <b>Deprecated in 8.5.112</b>	string	yes	Chat host alias. For example: "117"
userData	JSON-formatted string	no	Any attached data that the client wants to add to the chat session. For example: { "key": "value", ... }

### Example

Example of request message:

```
var channel = "/service/chatV2/customer-support"
...
var updateUDData = {
```

```
"operation":"updateData",
"chatId" : "00048aAPEQJ8000U",
"userId":"007555677B20000A",
"secureKey":"4ee15d7e1c343c8e",
"userData" : { "key3" : "value3", "key4" : "value4" }
}
cometd.publish( channel, updateUDData );
```

Example of notification response:

```
{
"messages":[],
"chatEnded":false,
"statusCode":0,
"alias":"117",
"secureKey":"3e2a69d421ae2672",
"userId":"007555677CB4000D",
"chatId" : "00048aAPEQJ8000U",
"nextPosition" : 9
}
```

## Chat Session File Management

### Updated in 8.5.112

Starting with release 8.5.112, the GMS Digital Channels API allows agents and customers to exchange files during their chat sessions using these requests:

- **Get Limits**—Check for allowable file types and file size—or other constraints on file uploads—before sending an upload request.
- **Upload File**—Upload a file.
- **Download File**—Download a previously uploaded file.
- **Delete File**—Delete a previously uploaded the file.

File Management requests are all POST requests submitted to a single URL endpoint:

```
POST /genesys/2/chat-ntf
```

Each request must include at least the `operation` and `secureKey` parameters.

### Get Limits

Use this optional request to avoid wasting network and CPU overhead by checking for allowable file types or maximum file size—or other constraints on file uploads—before sending an upload request.

HTTP Header	Value
Content-Type	<ul style="list-style-type: none"><li>• multipart/form-data</li><li>• application/x-www-form-urlencoded</li></ul>

## Input Parameters

Parameter Name	Sample Value	Description	Required/Optional
operation	"fileGetsLimits"	user ID	Required
secureKey	"8b31761b2154884c"	secure key	Required

## Output Parameters

Parameter	Description
download-attempts	Maximum number of times a specific file can be downloaded during this session
upload-max-files	Maximum number of files a client can upload during this session
upload-max-file-size	Maximum allowable file size of a single uploaded file
upload-max-total-size	Maximum allowable file size of all uploaded files
upload-need-agent	Indicates whether an agent needs to accept a chat session before an upload is allowed
upload-file-types	Colon-delimited list of file extensions indicating which types of files can be uploaded
used-upload-max-files	Current number of files uploaded during this session
used-upload-max-total-size	Current total size of all files uploaded during this session
used-download-attempts	Current number of downloaded files during this session
delete-file	Indicates whether <b>upload-max-files</b> is decremented after a file has been deleted

## Example

http://localhost:8080/genesys/2/chat-ntf

operation=fileGetsLimits  
secureKey=8b31761b2154884c

## Output

```
{
  "messages": [
    {
      "from": {
        "nickname": "JohnS",
        "participantId": 1,
        "type": "Client"
      },
      "text": "file-client-cfg-get",
      "type": "Notice",
      "utcTime": 1497840553000,
      "userData": {
        "download-attempts": "10",
        "upload-max-files": "3",
        "delete-file": "odd",
        "upload-max-file-size": "2097152",
        "used-download-attempts": "0",
        "used-upload-max-total-size": "0",
        "upload-need-agent": "true",
        "used-upload-max-files": "0",
        "upload-max-total-size": "5242880",
        "upload-file-types":
"bmp:csv:doc:docx:gif:htm:jpg:pdf:png:ppt:pptx:tif:txt:xls:xlsx"
      }
    },
    ],
    "chatEnded": false,
    "statusCode": 0,
    "secureKey": "799c5ff0faccd5bb"
  }
}
```

## Upload File

Use this request to upload a file into a chat session so it can be shared with an agent or a customer. You can then use the **file-id** value in the response to delete or download the file, as needed.

HTTP Header	Value
Content-Type	multipart/form-data

Input parameters

Parameter Name	Sample Value	Description	Required/Optional	Source
operation	fileUpload	secure key	Required	Form
secureKey	"8b31761b2154884c"	secure key	Required	Form
file	"afile.txt"	file to be uploaded into the session	Required	Form
userData[file-description]	"March invoice"	This value can be used by the agent desktop and the client UI to provide	Optional	Form

Parameter Name	Sample Value	Description	Required/ Optional	Source
		additional file info		
userData[userKey1]	"User value 1"	Collection of key-value pairs that provides file metadata	Optional	Form

## Example

http://localhost:8080/genesys/2/chat-ntf

```
operation=fileUpload
secureKey=8b31761b2154884c
file=<some file to upload>
```

## Output

```
{
  "chatEnded": false,
  "statusCode": 0,
  "secureKey": "77cd1c487b67fefb",
  "userData": {
    "file-id": "00F057DB0FF10005"
  }
}
```

## Download File

Use this request to download a file that was uploaded into a chat session either by an agent or a customer. The `fileId` parameter can be retrieved from the previous transcript event (see the response for Refresh Chat request) indicating that a file was upload.

HTTP Header	Value
Content-Type	<ul style="list-style-type: none"><li>multipart/form-data</li><li>application/x-www-form-urlencoded</li></ul>

Parameter Name	Sample Value	Description	Required/ Optional	Source
operation	fileDownload		Required	Form
secureKey	"8b31761b2154884c"	secure key	Required	Form
fileId	"00048aAPEQJ8ABCD"	File to be downloaded from the session	Required	URL

## Example

http://localhost:8080/genesys/2/chat-ntf

```
operation=fileDownload
secureKey=8b31761b2154884c
fileId=00048aAPEQJ8ABCD
```

## Output

If the operation is successful, the contents of the file are downloaded. In case of an error, the server returns an HTML 500 response that includes the **referenceId** of the request.

The following HTML snippet shows how to download a file using an HTML form:

```
var form = $(
"<form method='POST' enctype='multipart/form-data' target='_blank'
action='http://GMS_HOST:GMS_PORT/genesys/2/chat-ntf'>
  <input name='secureKey' value='6b46a7a8910f21be' type='hidden'>
<input name='fileId' value='00F057C8B6B7004D' type='hidden'>
  <input name='operation' value='download' type='hidden'>/form>");
form.submit();
```

## Delete File

Use this request to delete a file that was uploaded into a chat session by a customer if allowed by Chat Server settings. Web User has no permission to delete file uploaded by the agent.

HTTP Header	Value
Content-Type	<ul style="list-style-type: none"><li>multipart/form-data</li><li>application/x-www-form-urlencoded</li></ul>

Parameter Name	Sample Value	Description	Required/ Optional	Source
operation	"fileDelete"	user ID	Required	Form
secureKey	"8b31761b2154884c"	secure key	Required	Form
fileId	"00048aAPEQJ8ABCD"	The file to be deleted.	Required	URL

## Example

POST /genesys/2/chat-ntf

```
operation=fileDelete
fileId=007553863DC30029
```



secureKey=8b31761b2154884c

### Output

```
{
  "chatEnded": false,
  "statusCode": 0,
  "secureKey": "ad437d2338d09d09"
}
```

# Email API Version 2

## Updated in 8.5.110.07

- See [Configuring Digital Channels](#) for email configuration details
- See [Email options reference](#) for the list of options available for your email service.

HTTP Method		URI		
POST		/genesys/2/email/{serviceName}		
HTTP Header		Value		
Content-Type		<ul style="list-style-type: none"> <li>• multipart/form-data</li> <li>• application/x-www-form-urlencoded (starting in 8.5.110.07)</li> </ul>		
Parameter Name	Sample Value	Description	Required/Optional	Comments
firstName	"John"	first name of the customer	Required	
lastName	"Doe"	last name of the customer	Required	
fromAddress	"jdoe@gmail.com"	email address of the customer	Required	
subject	"Help with account"	email subject	Required	
text	"I need help with my account"	email body as plain text	Required	
mailbox	"support@company.com"	the address to which the email is to be delivered	Optional	This value takes priority over any configured "mailbox" in the WebAPI application on Config Server
userData	key1=value1	User data — can be submitted multiple times	Optional	
userData	key2=value2	User data — can be submitted multiple times	Optional	
files		files to be attached to the email	Optional	

Parameter Name	Sample Value	Description	Required/ Optional	Comments
		<b>Important</b> Use the multipart/form-data Content-Type if you need to attach files. With the application/x-www-form-urlencoded Content-Type, this parameter is ignored.		

Constraints	Default Value	Configuration Parameter	Comments
Max no. of files	10	Options / email / max_files	
Max size of the content in bytes	10485760 (for 10MB) or 26214400 (for 25MB).	Options / email / max_size	
File Types	pdf,doc,txt,jpg,png,gif,bmp,zip	Options / email / file_types	If the parameter "file_types" is configured, it completely overrides the default value

Configuration Parameter	Comments
Options / email / mailbox	The address to which the email is to be delivered. If no value is set in the incoming request and if this value is set on Config Server then the Config Server value will be set to the mailbox value from the request to Email Server. If neither of these values is set, then Email Server uses its default value.
Options / email / tenant	Tenant name which has been assigned to email servers

## Request

```

Content-Type: multipart/form-data;
boundary="WebKitFormBoundaryE19zNvXGzXaLvS5C"
----WebKitFormBoundaryE19zNvXGzXaLvS5C
Content-Disposition: form-data; name="firstName"

Name
----WebKitFormBoundaryE19zNvXGzXaLvS5C
Content-Disposition: form-data; name="lastName"

LastName
----WebKitFormBoundaryE19zNvXGzXaLvS5C
Content-Disposition: form-data; name="fromAddress"

john@doe.com

```

```
----WebKitFormBoundaryE19zNvXGzXaLvS5C
Content-Disposition: form-data; name="subject"
```

```
Hi hello
```

```
----WebKitFormBoundaryE19zNvXGzXaLvS5C
Content-Disposition: form-data; name="text"
```

```
This is the text
```

```
----WebKitFormBoundaryE19zNvXGzXaLvS5C
```

## Response

```
HTTP 200 - success
```

```
{
  "statusCode": 0,
  "interactionId": "0000KaA0C8XH003X"
}
```

```
HTTP 200 - failure (server down, network failure, etc. Try again)
```

```
{
  "statusCode": 1
}
```

# Open Media API

GMS supports OpenMedia API starting with 8.5.110. This API enables you to manage an Open Media interaction: you can create, retrieve, update, or stop an Open Media interaction by submitting one of the REST queries detailed below.

- See the [API Responses](#) page for further details about the returned status code.
- See [Configuring Digital Channels](#) for open media configuration details
- See [Open Media Service Options](#) for the list of options available for your open media service(s).

## Create an Open Media Interaction

Creates an Open Media interaction. In your service configuration, you can define the default Interaction Type, Subtype, and Media Type to assign to created interactions. You can overwrite these defaults in your REST queries if you enable the `allow_overwrite` option in your configuration. See [Open Media Service Options](#) for further details.

### Operation

POST /genesys/2/openmedia/{serviceName}			
HTTP Header	Value		
<b>Content-Type</b>	application/x-www-form-urlencoded		
Parameter Name	Type	Mandatory	Description
URI Parameters			
serviceName	string	Yes	Service to use to create the Open Media interaction.
Body Parameters			
externalId	string	No	External ID to link to the new interaction; for example: "xy932". This ID is passed to Interaction Server in the UserData with the key "ExternalId".
interactionType	string	No	Interaction Type to assign to the new interaction, for example Inbound. The Interaction Type must match one of the values

POST /genesys/2/openmedia/{serviceName}			
			<p>listed in the <b>Business Attributes &gt; Interaction Type</b> section of your configuration.</p> <p>By default, the Interaction Type used to create the interaction is defined in the interaction_type option of your Open Media service configuration. The system considers the interactionType parameter only if the allow_overwrite option is set to true in your service configuration. See the <a href="#">options</a> section for more information.</p>
interactionSubType	string	No	<p>Interaction Subtype to assign to the new interaction, for example InboundNew. The Interaction Subtype must match one of the values listed in the <b>Business Attributes &gt; Interaction Subtype</b> section of your configuration.</p> <p>By default, the Interaction Subtype used to create the interaction is defined in the interaction_subtype option of your Open Media service configuration. The system considers the interactionSubtype parameter only if the allow_overwrite option is set to true in your service configuration. See the <a href="#">options</a> section for more information.</p>
mediaType	string	No	<p>Media Type to assign to the new interaction, for example workitem. The Media Type must match one of the values listed in the <b>Business Attributes &gt; Media Type</b> section of your configuration.</p> <p>By default, the Media Type used to create the interaction is defined in the media_type option of your Open Media service configuration. The system considers the</p>

POST /genesys/2/openmedia/{serviceName}			
			mediaType parameter only if the allow_overwrite option is set to true in your service configuration. See the <a href="#">options</a> section for more information.
userData[<key-name>]	string	No	User data to attach to the interaction. You can submit multiple user data. For example: userData[OrderNumber]=uABC1234567

## Examples

The following request includes the following encoded parameters:

```
externalId=xy932&userData[OrderNumber]=ABC123456789&userData[TestKey2]=Test value&userData[KeyToDelete]=value to delete
```

```
POST /genesys/2/openmedia/customer-support
HTTP/1.1
Host: localhost:8080
Content-Type: application/x-www-form-urlencoded
Cache-Control: no-cache
```

```
externalId=xy932&userData%5BOrderNumber%5D=ABC123456789&userData%5BTestKey2%5D=Test+value&userData%5BKeyToDelete%5D=value to delete
```

## Response

```
HTTP 200 - success
{
  "statusCode": 0,
  "interactionId": "01QG0Q2S5S50G000"
}
HTTP 200 - failure (server down, network failure, etc. Try again)
{
  "statusCode": 1
}
```

## Get an Open Media Interaction

Retrieves an Open Media Interaction by its interactionId (default) or by its externalId.

## Operation

POST /genesys/2/openmedia/{serviceName}/{id}/get	
HTTP Header	Value
Content-Type	application/x-www-form-urlencoded

POST /genesys/2/openmedia/{serviceName}/{id}/get			
Parameter Name	Type	Mandatory	Description
URI Parameters			
serviceName	string	Yes	Service to use to create the Open Media interaction.
id	string	Yes	ID to search. By default, the interactionId is used for search. For example: "xy932".
Body Parameters			
searchByExternalId	true, false	No	true to search by externalId. By default, if you omit this option's value, this option is false and an interaction ID is used for search.

## Examples

### Request

```
POST /genesys/2/openmedia/customer-support/01QG0Q2S5S50G000/get HTTP/1.1
Host: localhost:8080
Content-Type: application/x-www-form-urlencoded
Cache-Control: no-cache
```

### Response

```
HTTP 200 - success
{
  "statusCode": 0,
  "interactionProperties": {
    "interactionType": "Inbound",
    "interactionId": "01QG0Q2S5S50G000",
    "interactionReceivedAt": "2017-05-31T04:43:52Z",
    "interactionIsHeld": false,
    "interactionSubmittedBy": "GMS-DC-API:gms1:1",
    "interactionMediatype": "promotion",
    "interactionTenantId": 1,
    "interactionQueue": "ATS Inbound Queue",
    "interactionSubmitSeq": "1049036",
    "interactionMovedToQueueAt": "2017-05-31T04:43:52Z",
    "interactionState": "Queued",
    "interactionSubmittedAt": "2017-05-31T04:43:52Z",
    "interactionPlacedInQueueAt": "2017-05-31T04:51:39Z",
    "interactionUserData": [
      {
        "key": "KeyToDelete",
        "type": "str",
        "value": "value to delete"
      },
      {
        "key": "OrderNumber",
        "type": "str",
        "value": "ABC123456789"
      }
    ]
  }
}
```



```
    },
    {
      "key": "TestKey2",
      "type": "str",
      "value": "Test value"
    },
    ...
  ],
  "interactionSubtype": "InboundNew",
  "interactionIsLocked": false,
  "interactionIsOnline": false,
  "interactionPlaceInQueueSeq": "1050198"
}
```

## Response

HTTP 200 - Failure

```
{
  "statusCode": 2,
  "errors": [
    {
      "code": 248,
      "advice": "Specified interaction Id[01QG0Q2S5S50G000] was not found"
    }
  ]
}
```

## Update an Open Media Interaction

Enables you to update two values and/or delete two key-value pairs at the same time. You need to provide at least one key to update or delete.

## Operation

POST /genesys/2/openmedia/{serviceName}/{id}/update			
HTTP Header	Value		
Content-Type	application/x-www-form-urlencoded		
Parameter Name	Type	Mandatory	Description
URI Parameters			
serviceName	string	Yes	Service that manages the Open Media interaction.
id	string	Yes	ID of the updated interaction. By default, the interactionId is used. For example: "xy932".
Body Parameters			
change[<key-1>]	string	No	Adds or modifies the value for the given user

POST /genesys/2/openmedia/{serviceName}/{id}/update			
			data key. For example:  change[OrderNumber]=uABC123456789  You must provide at least one key to change or delete.
change[<key-n>]	string	No	Adds or modifies the value for the given user data key. For example:  change[TestKey]=myValue  You must provide at least one key to change or delete.
delete[<key-1>]	empty	No	Key of the user data to delete. You do not need to provide a value; You can leave it empty. If you provide a value, it will be ignored.
delete[<key-n>]	empty	No	Key of the user data to delete. You do not need to provide a value; You can leave it empty. If you provide a value, it will be ignored.

## Examples

In this example, the requested update to be encoded is:

change[OrderNumber]=uABC123456789&change[TestKey2]=uTest+value&delete[KeyToDelete]=

```
POST genesys/2/openmedia/test/0000KaA0C8XH003Y/stop
POST /genesys/2/openmedia/customer-support/01QG0Q2S5S50G000/update HTTP/1.1
Host: localhost:8080
Content-Type: application/x-www-form-urlencoded
Cache-Control: no-cache
```

change%5BOrderNumber%5D=uABC123456789&change%5BTestKey2%5D=uTest+value&delete%5BKeyToDelete%5D=

## Response

HTTP 200 - success

```
{
  "statusCode": 0
}
```

HTTP 200 - failure

```
{
  "statusCode": 2,
  "errors": [
    {
      "code": 248,
      "advice": "Interaction with specified id[01QG0Q2S5S50G000] was not found"
    }
  ]
}
```

```
}
```

## Stop an Open Media Interaction

### Operation

POST /genesys/2/openmedia/{serviceName}/{id}/stop			
HTTP Header	Value		
Content-Type	application/x-www-form-urlencoded		
Parameter Name	Type	Mandatory	Description
URI Parameters			
serviceName	string	Yes	Service that created the Open Media interaction.
id	string	Yes	ID of the interaction to stop. By default, the interactionId is used. For example: "xy932".
Body Parameters			
reasonSystemName	string	No	Reason for stopping the interaction. This reason will be attached to the event sent by Interaction Server; for example: "Cancel"
reasonDescription	string	no	Reason description for stopping the interaction. This description will be attached to the event sent by Interaction Server; for example: "Not needed"

### Example

The following operation stops the interaction with the following encoded reason and description:  
reason=RSN-123&description=ORDER+CANCELLED

```
POST genesys/2/openmedia/test/0000KaA0C8XH003Y/stop
POST /genesys/2/openmedia/customer-support/01QG0Q2S5S50G000/stop HTTP/1.1
Host: localhost:8080
Content-Type: application/x-www-form-urlencoded
Cache-Control: no-cache

reason%3DRSN-123%26description%3DORDER%2BCANCELLED
```

### Response

```
HTTP 200 - success
```

```
{
  "statusCode": 0
}
HTTP 200 - failure
{
  "statusCode": 2,
  "errors": [
    {
      "code": 248,
      "advice": "Interaction with specified id[01QG0Q2S5S50G0001] was not found"
    }
  ]
}
```

# API Responses

All Digital Channels API responses contain:

- An HTTP status code
- Either an API **statusCode** or one or more API error codes.

The following sections provide more details about these responses.

## HTTP 200

Name	Description
200 OK	
<b>Response Body</b> (JSON content)	
<code>statusCode</code> <i>required</i>	<p>In all responses with an HTTP status code of 200:</p> <ul style="list-style-type: none"><li>• An API <b>statusCode</b> value of 0 indicates that the operation was successful and all fields in the response have valid values.</li><li>• An API <b>statusCode</b> value of 1 indicates that there was an error and the client can keep trying until it receives a <b>statusCode</b> of 0 or until it decides to give up. <b>Note:</b> When you receive this response, you cannot assume that the values of any other fields are valid.</li><li>• An API <b>statusCode</b> value of 2 indicates that:<ul style="list-style-type: none"><li>• There was an error</li><li>• Repeating requests will not be successful</li><li>• The chat may have ended—read the value of the <b>chatEnded</b> flag. <b>Note:</b> When you receive this response, you cannot assume that the values of any other fields are valid.</li></ul></li></ul>

## Digital Channels Chat V2 Response Format

The positive response (HTTP status 200) from the GMS node is identical for any Chat V2 API request. It includes some fields that you can use in subsequent requests such as **alias**, **userId**, **chatId**,

**securekey**, **nextPosition**, as well as chat transcripts available in the **messages** array. The Chat transcript may be empty or include one or more events.

Name	Type	Description
200 OK		
<b>Response Body</b> (JSON content)		
statusCode <i>required</i>	Integer 0,1,2	<ul style="list-style-type: none"> <li>0 indicates that the operation was successful and all fields in the response have valid values.</li> <li>1 indicates that there was an error and the client can keep trying until it receives a <b>statusCode</b> of 0 or until it decides to give up. <b>Note:</b> When you receive this response, you cannot assume that the values of any other fields are valid.</li> <li>2 indicates that there was an error.</li> </ul>
alias	string	Identifies the Chat Server instance that served this request. When provided, use it in subsequent API requests. This alias may change several times within the same session.
chatEnded	boolean false, true	If true, indicates that the current chat session has ended. If false, the chat session is still ongoing.
chatId	string	Identifies current Chat Session id. When provided, use it in subsequent API requests. This value doesn't change during the chat session.
messages	array of Chat transcripts	Contains full or partial chat transcript. See the Chat Transcript section below.
nextPosition	integer	Indicates which event index to use to request a transcript in the next operation (used in Refresh Chat, and other requests)
secureKey	string	Secured key for this session; use this key (valid only for this session) in each request. Make sure to protect this key.
userId	string	Identifies this chat party (self).

Name	Type	Description
		When provided, use it in subsequent API requests.

## Party Resource

The `from` field of the chat event describes the party who generated this event. This resource matches the following format:

Party Resource

Name	Type	Description
nickname	string	The preferable calling [nick] name of the chat participant.
participantId	integer 1,2...	Identifier of this chat party. This identifier differs from the <code>userId</code> . The same party can join and leave several times the chat session. Each time that the party joins, it is assigned a new <code>partyId</code> even if it is the same agent.
type	string "Agent" "Client" "External"	Describes the type of participant.  "External" means that the event was not generated by a customer or an agent, but rather by an ESP request, received for example from the routing strategy.

## Chat Transcript

The Chat Transcript consists of a list of chat events. Each chat event has an index assigned to the event when it happens that defines the order of events. You can request full or partial transcripts using the Chat Refresh operation. Note that you do not receive all of the Chat events, so the index difference between two events in the transcript can exceed one. Each chat event is generated by some chat party.

### Important

Semantics of event text and `userData` in the event may vary, depending on the event type. For further details, see the query description in your Digital Channel API (Chat, Email, or Open Media).

Chat Transcript Resource

Field	Type	Description
from	Party object	Describes the chat party who generated this event.
index	integer	Identifies serial number of this event within chat session, starts with 1 for the first event.
text	string	If available, contains the text associated with this event.
type	ParticipantJoined ParticipantLeft Message TypingStarted TypingStopped NicknameUpdated PushUrl FileUploaded FileDeleted CustomNotice Notice IdleAlert IdleClose	Identifies the chat event type.
utcTime	integer	Indicates the event timestamp.
userData	map<string,string>	If available, contains additional information for this event.

## Example

```
200 OK
{
  'alias': '117',
  'chatEnded': False,
  'chatId': '00001aCY6K8U003T',
  'messages': [{ 'from': { 'nickname': 'FirstStep',
                           'participantId': 1,
                           'type': 'Client'},
                 'index': 1,
                 'type': 'ParticipantJoined',
                 'utcTime': 1509117363000},
               { 'from': { 'nickname': 'TestName',
                           'participantId': 2,
                           'type': 'Agent'},
                 'index': 2,
```



```

      'type': 'ParticipantJoined',
      'utcTime': 1509117368000},
    {'from': {'nickname': 'FirstStep',
              'participantId': 1,
              'type': 'Client'},
      'index': 3,
      'text': 'First message',
      'type': 'Message',
      'utcTime': 1509117368000
    }],
    'nextPosition': 4,
    'secureKey': 'e4071047872b7820630f',
    'statusCode': 0,
    'userId': '007559F34DB30001'
  }
}

```

## HTTP 400

Validation errors with an HTTP status code of 400 happen when any of the values necessary to complete an operation is missing. They have two different formats, depending on which release you are using:

### GME version 8.5.103.08 and lower

```

{
  error: "validationError"
  fields:
  {
    nickname: "either nickname or firstName and
    lastName should be supplied"
    tenantName: "invalid"
  }
}

```

### GME version 8.5.103.09 and higher

```

// array of all validation errors
{ "errors" : [
  // missing required parameter _firstName_
  {"code":102},
  // missing required parameter _lastName_
  {"code":103},
  // total file size exceeded,configured value of 300kB
  {"code":202,"advice":"300000"},
  // attempt to upload file of incorrect type
  {"code":203,"
advice":"jpg,zip,txt,pdf,exe,bmp,gif,dll,png,doc"}
]}

```

## Web API Validation Error Codes

code	description	chat	email	openmedia
100-199	Error messages in this range indicate that a required parameter is missing from a request.			
100	RESERVED			
101	<b>tenant</b> is missing.			
102	<b>firstName</b> is missing.	x	x	
103	<b>lastName</b> is missing.	x	x	
104	<b>subject</b> is missing.	x	x	
105	<b>userData</b> is missing.	x		
151	<b>alias</b> is missing.	x		
152	<b>userId</b> is missing.	x		
153	<b>secureKey</b> is missing.	x		
154	<b>chatId</b> is missing.	x		
161	<b>nickname</b> is missing.	x		
162	<b>message</b> is missing.	x		
163	<b>pushUrl</b> is missing.	x		
181	<b>fromAddress</b> is missing.		x	
182	<b>text</b> is missing.		x	
191	<b>mediaType</b> is missing.			x
192	<b>interactionType</b> is missing.			x
193	<b>interactionSubtype</b> is missing.			x
200-299	Error messages in this range indicate that one or more of the parameters specified in a request is outside of the range configured in the Genesys Mobile Engagement application options. These errors include an <b>advice</b> field that displays the allowable range or limit.			
200	RESERVED			
201	The total <b>number</b> of uploaded files is too high	x	x	
202	The total <b>size</b> of the uploaded files	x	x	

code	description	chat	email	openmedia
	is too large			
203	The uploaded file has an incorrect <b>filetype</b>	x	x	
240	An unexpected error occurred. The advice field includes the ReferenceID that helps to identify the error cause in the GMS log file.	x	x	x
241	The specified file is not present in the current session.	x		
248	The interaction with the specified ID was not found.			x
249	The Media Server was not able to execute the request.	x		
250-299	Error messages in this range are not generated by Genesys Mobile Engagement but are provided directly by the connected Media Server (for example, Chat Server).	x		
250	RESERVED			
261	File Upload is not possible until an agent joins the session.	x		
264	File was already deleted.	x		
300-399	Error messages in this range indicate that a request parameter is malformed or invalid (for example, an invalid email address).			
300	RESERVED			
306	<b>serviceName</b> URL parameter is invalid (that is, not configured in the Genesys Mobile Engagement application). The	x	x	

---

code	description	chat	email	openmedia
	HTTP status will be set to 404 - Not Found instead of 400 - Bad Request.			
364	<b>emailAddress</b> contains an invalid email address.	x		
381	<b>fromAddress</b> contains an invalid email address.		x	
383	<b>mailbox</b> contains an invalid email address.		x	

# Phone Number Validation API

This Phone Number Validation API wraps the [Google's common Java, C++ and JavaScript library](#) for the purposes of parsing, formatting, and validating international phone numbers. GMS 8.5.108.02. integrates Version 7.2.8 and uses the [Apache License Version 2.0](#).

The API supports the following features:

- Parsing the phone number for a syntactic check (number, plus signs, extensions, and so on);
- Splitting the phone number into parts (country code, extensions, and so on);
- Verifying local patterns such as: US=(650) 466-1100, FR=06 04 12 04 05, DE=089 125016040;
- Translating numbers to enable automated systems to use either localized or E164 or international form (useful to identify unique patterns);
- Providing some hints on advanced information such as timezone, city, carrier;
- Providing information in the case of premium numbers.

You can pass a country code in the parameters of your queries. See below for further details.

## Important

Note that if the validation parsing fails, the response will be an Error 400.

## API Query

GET /genesys/1/phonenumbers?number={number}			
Name	Type	Mandatory	Description
URI Parameters			
number	string	yes	Phone number to validate. For example, 0604120405.
country	string	no	Two-letter country code as defined in <a href="#">ISO 3166</a> . If you do not specify country in your query, US will be used by default to parse the phone number. For example, in the default configuration, the phone number 06 04 12 04 05 is invalid unless you

GET /genesys/1/phonenumbers?number={number}			
			set country=FR in your query.
geocodingLocale	string	no	<p>Language used for displaying the area descriptions. For example, if you validate the phone number "06 04 12 04 05":</p> <ul style="list-style-type: none"> <li>geocodingLocale=FR would return "location": "France"</li> <li>geocodingLocale=DE would return "location": "Frankreich"</li> </ul>

## Response

Name	Type	Mandatory	Description
HTTP code	200		
HTTP message	OK		
Response Body (JSON content)			
language-specified	string	yes	Specified language set in the geocodingLocale request parameter.
number-specified	string	yes	Validated number.
country-default	string	yes	Specified or default country used for the response.
country-code	string	yes	Country code part; for example, 33 for France.
region-code	string	yes	The region where the phone number is originating from. For example, "IT" for Italy.
country-code-source	string	yes	<p>Information used to detect country code.</p> <ul style="list-style-type: none"> <li>FROM_NUMBER_WITH_PLUS_SIGN</li> <li>FROM_NUMBER_WITH_IDD</li> <li>FROM_NUMBER_WITHOUT_PLUS_SIGN</li> </ul>

Name	Type	Mandatory	Description
			<ul style="list-style-type: none"> <li>FROM_DEFAULT_COUNTRY</li> </ul>
extension	string	yes	Extension part. For example, "32" if the number is "0604120405#32". Note that # should be url-encoded as %23 in the request.
number-national-short	string	yes	Short number for country. For example, 604120405 for a French mobile number.
number-national	string	yes	Number formatted using the national format
number-E164	string	yes	Number formatted using the E.164 format.
number-original	string	yes	Number formatted using the original phone number format that was parsed.
number-international	string	yes	Number formatted using the international format
is-possible-number	string	yes	true if the number looks valid from considering the length information.
is-valid-number	string	yes	true if the number looks valid from considering the length and prefix information.
is-premium	string	no	In case of a valid number, true if the number is a premium rate number.
italian-leading-zero	string	yes	In case of an Italian number, true if the number has leading zero.
number-type	string	no	Type of the number based on the number parsing to determine if the number matches Fixed-line, Mobile, Toll-free, Premium Rate, Shared Cost, VoIP, and Personal Numbers (whenever feasible). The possible types are:

Name	Type	Mandatory	Description
			<ul style="list-style-type: none"><li>• UNKNOWN</li><li>• FIXED_LINE</li><li>• MOBILE</li><li>• FIXED_LINE_OR_MOBILE</li><li>• TOLL_FREE</li><li>• PREMIUM_RATE</li><li>• SHARED_COST</li><li>• VOIP</li><li>• PERSONAL_NUMBER</li><li>• PAGER</li><li>• UAN</li><li>• VOICEMAIL</li></ul>
location	string	no	Geographical information related to a phone number.
time-zone	string	no	Geographical time information related to a phone number.
carrier	string	no	Carrier information related to a phone number.
is-possible-short-number	string	no	If the number is impossible or invalid, might return information on a possible usage as a short number.
is-valid-short-number	string	no	When number is not possible short number, might return info on valid usage as short number.
is-possible-number-with-reason	string	no	When the number is impossible, gives the reason why. For example: "T00_SHORT"

## Examples

The following example returns a successful validation result for a French mobile number.

```
GET http://localhost:8010/genesys/1/phonenumbers?number=0604120405&country=FR&ocodingLocale=DE
```

Result



200 OK

```
{
  "language-specified": "de",
  "country-code": "33",
  "extension": "",
  "number-type": "MOBILE",
  "number-national-short": "604120405",
  "number-national": "06 04 12 04 05",
  "number-E164": "+33604120405",
  "number-specified": "0604120405",
  "number-original": "06 04 12 04 05",
  "is-possible-number": "true",
  "italian-leading-zero": "false",
  "carrier": "SFR",
  "country-code-source": "FROM_DEFAULT_COUNTRY",
  "number-international": "+33 6 04 12 04 05",
  "country-default": "FR",
  "is-valid-number-for-region": "true",
  "number-region": "FR",
  "location": "Frankreich",
  "is-valid-number": "true",
  "time-zone": "[Europe/Paris]"
}
```

The following example returns a successful validation result for a German fixed line number.

GET http://localhost:8010/genesys/1/  
phonenumber?number=%2B4989125016040&country=DE&ocodingLocale=FR

Result

200 OK

```
{
  "language-specified": "fr",
  "country-code": "49",
  "extension": "",
  "number-type": "FIXED_LINE",
  "number-national-short": "89125016040",
  "number-national": "089 125016040",
  "number-E164": "+4989125016040",
  "number-specified": "+4989125016040",
  "number-original": "+49 89 125016040",
  "is-possible-number": "true",
  "italian-leading-zero": "false",
  "country-code-source": "FROM_NUMBER_WITH_PLUS_SIGN",
  "number-international": "+49 89 125016040",
  "country-default": "DE",
  "is-valid-number-for-region": "true",
  "number-region": "DE",
  "location": "Munich",
  "is-valid-number": "true",
  "time-zone": "[Europe/Berlin]"
}
```

The following example returns a successful validation result for a US fixed line number.

GET http://localhost:8010/genesys/1/  
phonenumber?number=%2B16504661100&country=US&ocodingLocale=FR

Result

200 OK

```
{
  "language-specified": "fr",
  "country-code": "1",
  "extension": "",
  "number-type": "FIXED_LINE_OR_MOBILE",
  "number-national-short": "6504661100",
  "number-national": "(650) 466-1100",
  "number-E164": "+16504661100",
  "number-specified": "+16504661100",
  "number-original": "+1 650-466-1100",
  "is-possible-number": "true",
  "italian-leading-zero": "false",
  "carrier": "",
  "country-code-source": "FROM_NUMBER_WITH_PLUS_SIGN",
  "number-international": "+1 650-466-1100",
  "country-default": "US",
  "is-valid-number-for-region": "true",
  "number-region": "US",
  "location": "California",
  "is-valid-number": "true",
  "time-zone": "[America/Los_Angeles]"
}
```

## Errors

A **400 Bad request** error is generated if the phone number is invalid. For example:

GET <http://localhost:8010/genesys/1/phonenumbers?number=bad&country=DE&encodingLocale=de>

Result

400 Bad Request

```
{
  "exception": "com.genesyslab.gsg.GSGException",
  "message": "Invalid Phone Number bad for Country DE (parsing impossible)"
}
```

# Stat Service API

## Modified in 8.5.114

This API retrieves statistics from the Genesys Statistics Server (Stat Server) and URS.

### Important

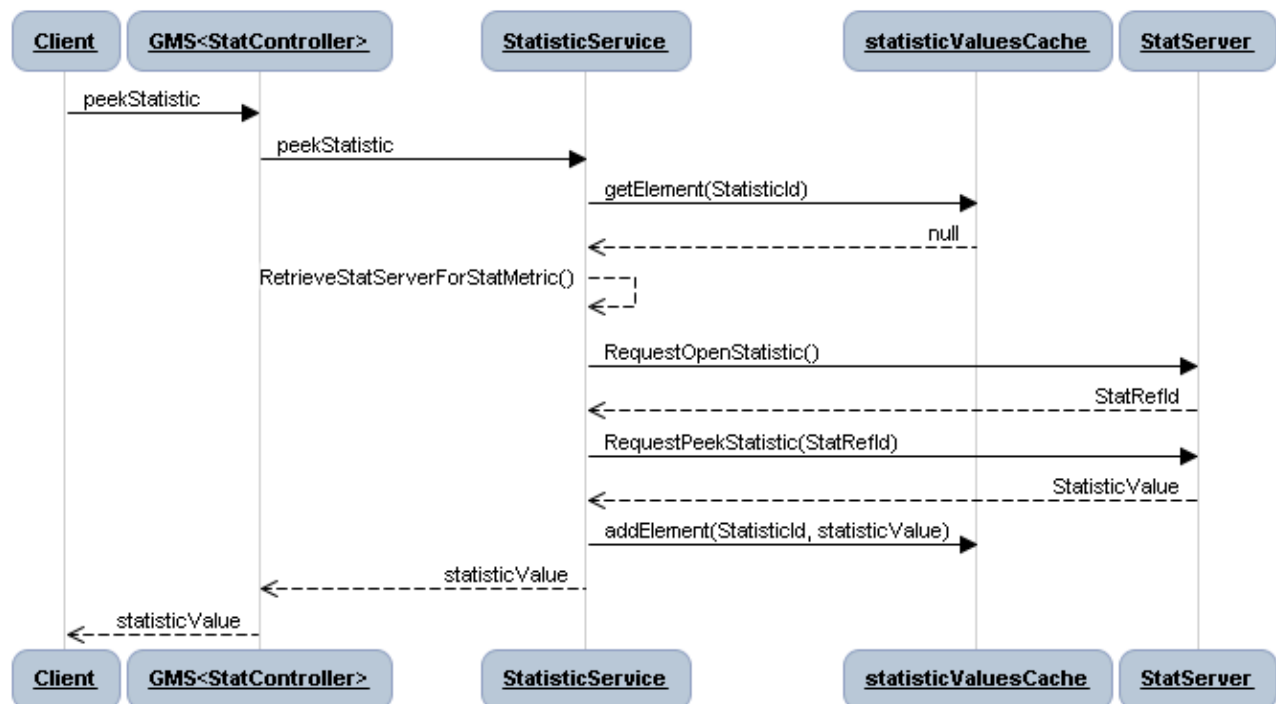
You need an Authentication header for this service.

## Stat Request APIs

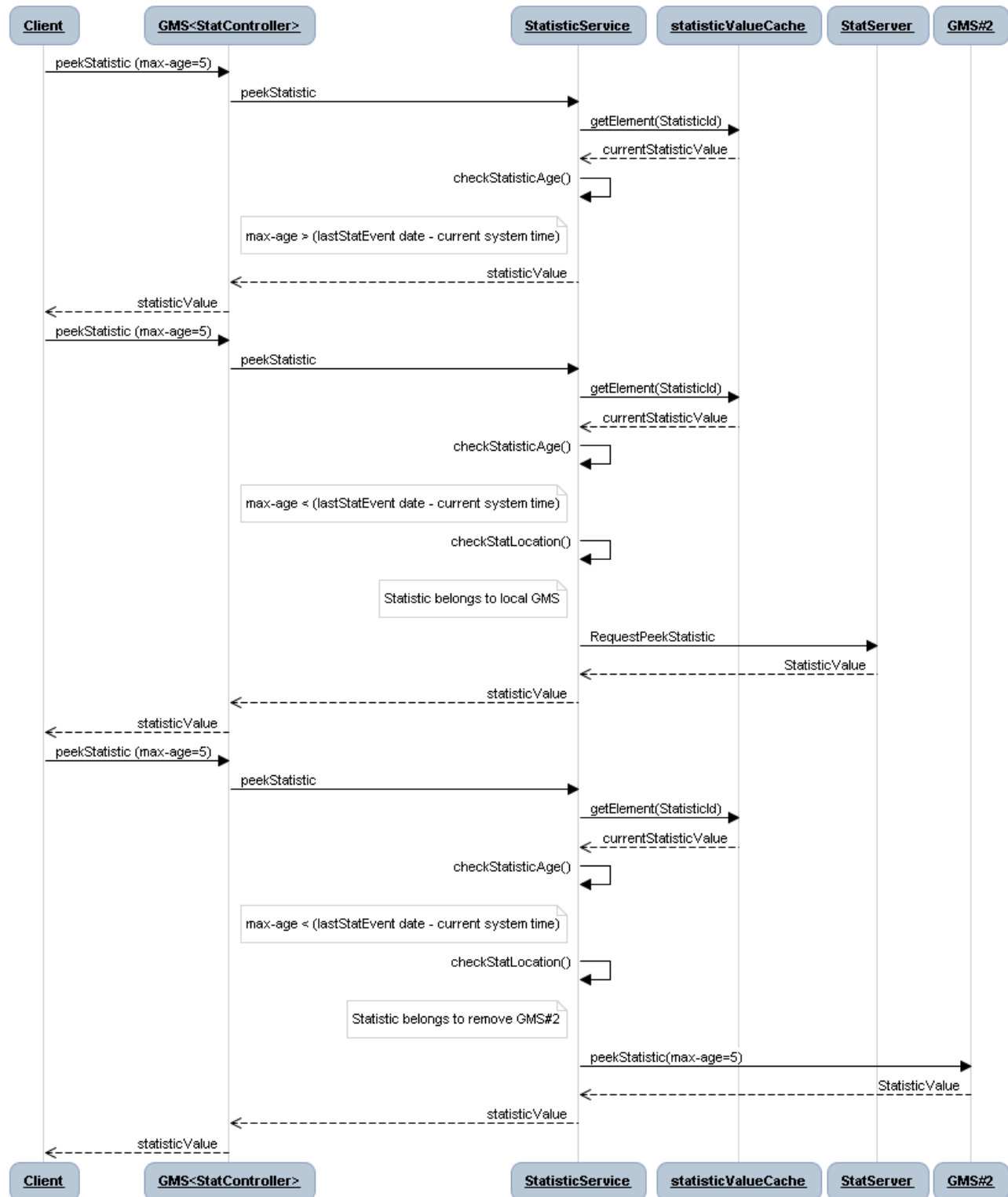
### Sequence Diagrams

#### [+] Show Peek Stat Sequence (Statistic is not already opened)

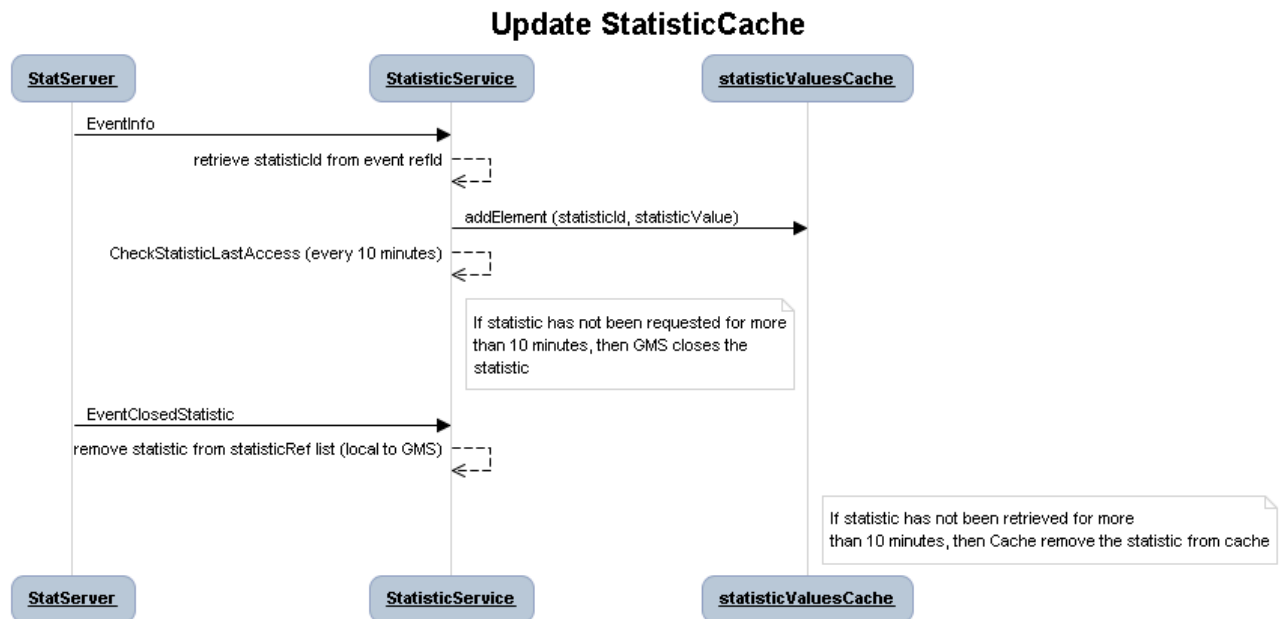
#### PeekStat Sequence (Statistic not already opened)



**[+] Show Peek Stat Sequence (Statistic already exists)**

**PeekStat Sequence (Statistic already exist)**

## [+] Show Peek Stat Sequence (Update Cache)



## Accessing the APIs

The Stat Server API exposes two interfaces:

- **"genesys/1/internal\_statistic"** — for internal access with no authentication control.
- **"genesys/1/statistic"** — for external use, which uses Basic Access Authentication (BA) to authenticate users.

As a standard protocol, the username and password for BA can be passed in the URL, as shown here:

`http://username:password@127.0.0.1:8080/genesys/1/statistic`

This API provides the following queries:

- **PeekStat**
- **URS Stat**

**Important**

See the [Extended API](#) to query multiple statistics.

## PeekStat Request

This request fetches one statistic value.

### Operation

Method	POST		
URL	/genesys/1/statistic		
Parameter	Type	Mandatory	Description
Header Parameters			
Cache-Control : max-age=XX	A number of seconds	no	The max-age value used to check if GMS has to recalculate the statistic. If the peek statistic time window is greater than max-age, GMS recalculates the statistic value. If the value is not present, it returns the latest value in cache.
Body			
objectId, objectType, tenant, metric (or statisticType), filter, notificationMode, timeProfile, timeRange, timeRange2		yes	<p>The body can be either a MultiPart form or an x-www-form-urlencoded form consisting of different items representing the key/value pairs associated with the statistic (objectId, objectType, tenant, tenantPassword, metric, notificationMode, filter, timeProfile, timeRange, timeRange2).</p> <ul style="list-style-type: none"><li>• NotificationMode can be NoNotification, Reset, or Immediate.</li><li>• filter is the business attribute value to use to filter the results.</li></ul>

Response

HTTP code	200
HTTP Message	OK
HTTP Header	Content-Type: application/json;charset=UTF-8
Body	A JSON object representing a structure with value

If a problem occurs during operation, the following status codes are returned:

HTTP code	HTTP Message	HTTP Body
400	Bad Request	message: {"message":"Notification Mode is unknown","exception":"com.genesyslab.gsg.serv
401	Unauthorized	Access to API refused
403	Forbidden	message: {"message":"Object type not valid","exception":"com.genesyslab.gsg.services
403	Forbidden	message: {"message":"Agent 'Kippola' (Tenant 'Environment') not found","exception":"com.genesyslab.gsg.service
403	Forbidden	message: {"message":"Wrong parameter","exception":null}
500	Server Error	message: {"message":"Metric (TotalLoginTime;) not found on running StatServer","exception":"com.genesyslab.gsg.se
500	Server Error	message: {"message":"Statistic Service bad parameter for tenant","exception":"com.genesyslab.gsg.servic

Important

You can make this request without using authentication by replacing **/genesys/1/statistic** with **/genesys/1/internal\_statistic**.

Example

Request

```
$ curl -v -u default:password -X POST --data "metric=TotalLoginTime&objectType=Agent&objectId=KSippola&tenant=Environment" http://localhost:8080/genesys/1/statistic
```



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

## Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
{"value":4963}
```

## URS Stat Request

This API allows you to query URS statistics with GMS.

To use this request, you must add the following configuration information:

- URS version must be 8.1.400.12 or higher.
- In your URS application, allow HTTP requests:
  1. Open your URS application in Genesys Administrator or Configuration Manager.
  2. In the Server Info tab, create an HTTP listening port.
- In the reporting section of your GMS application, create the `_urs_url` option and set it to the URL of URS.

You can also create a **builtin** service to retrieve URS statistics with the `urs-stat` template. When you create your service, select the "urs-stat" type in the Admin UI service panel. See the [Admin UI Help](#) for further details. Then, to retrieve the statistics, you can make the following request:

```
GET http://<gms_home>:<gms_port>/genesys/1/service/<urs_service>
```

## Operation

Method	GET		
URL	/genesys/1/statistic/urs/ <i>parameters</i>		
Parameter	Type	Mandatory	Description
Header Parameters			
Cache-Control : max-age=XX	A number of seconds	no	The max-age value used to check if GMS has to recalculate the statistic. If the peek statistic time window is greater than max-age, GMS recalculates the statistic value. If the value is not present, it returns the latest value in cache.
URL Parameters			
parameters		no	For information on these parameters, either refer

Method	GET		
			to the <a href="#">URS documentation</a> or send the following HTTP request to URS: http://URS_host:URS_HTTP_port/urs/help/

Response

HTTP code	200
HTTP Message	OK
HTTP Header	Content-Type: application/json;charset=UTF-8
Body	A JSON object representing a structure with value

If a problem occurs during operation, the following status codes are returned:

HTTP code	HTTP Message	HTTP Body
401	Unauthorized	message: "Authentication failed: username and/or password is incorrect"
500	Server error	message: {"message": "Wrong request", "exception": "org.apache.http.HttpException"}
500	Server error	message: {"message": "Call not found", "exception": "org.apache.http.HttpException"}

**Note:** You can make this request without using authentication by replacing **/genesys/1/statistic/urs/** with **/genesys/1/internal\_statistic/urs/**.

Example

Request

```
$ curl -v -u default:password "http://localhost:8080/genesys/1/statistic/urs/stat/targetstate?tenant=Environment&KSippola.A&json=&ext="
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "dnsall":6,
  "voice":1,
  "status":4,
  "loading":0,
  "dns":
  [
    {
      "type":38,
      "number":"KSippola",
      "switch":"","
      "status":4,
```

```
        "loading":0,
        "ready":1,
        "media":1,
        "loggedin":0
    },
    {
        "type":36,
        "number":"KSippola",
        "switch":"",
        "status":2,
        "loading":0,
        "ready":0,
        "media":1,
        "loggedin":0
    },
    {
        "type":61,
        "number":"KSippola",
        "switch":"",
        "status":4,
        "loading":0,
        "ready":1,
        "media":1,
        "loggedin":0
    },
    {
        "type":1,
        "number":"7001",
        "switch":"SIP_Switch",
        "status":4,
        "loading":0,
        "ready":1,
        "media":0,
        "loggedin":0
    },
    {
        "type":1,
        "number":"7001_IM",
        "switch":"SIP_Switch",
        "status":8,
        "loading":0,
        "ready":0,
        "media":0,
        "loggedin":0
    },
    {
        "type":38,
        "number":"7001_IM",
        "switch":"SIP_Switch",
        "status":8,
        "loading":0,
        "ready":0,
        "media":0,
        "loggedin":0
    }
],
"login":"SIP1",
"place":"SIP_Server_Place1",
"ready":1,
"agent":"KSippola",
"dnsrdy":3
}
```

## Extended Stat Request API

- [PeekStat Request: Retrieve configured statistics](#)
- [PeekStat Request: Querying Multiple Statistic Values in a Single Request](#)
- [PeekStat Request: Filtering Multiple Statistic Values in a Single Request](#)

### PeekStat Request: Retrieve configured statistics

Starting with 8.5.101, you can provision statistics in the **stat** [section](#) of your configuration options instead of passing properties as parameters of the statistics requests.

For instance, you can define **stat1** in your configuration as follows:

```
[stat.stat1]
_type=builtin
_service=statistic
metric=TotalLoginTime
notificationMode=Immediate
objectId=KSippola
objectType=Agent
tenant=Environment
filter=Bronze
```

Then, you can retrieve the statistics for **stat1** with a simple statistics query:

POST <http://localhost:8080/genesys/1/statistic/stat1>

Response:

```
{"value":41889}
```

### Operation

Method	POST		
URL	/genesys/1/statistic/ <b>stat_name</b>		
Parameter	Type	Mandatory	Description
Header			
Cache-Control: max-age	Integer	no	The max-age value in seconds of the statistic. If the peek statistic time window is greater than max-age, GMS recalculates the statistic value. If max-age is not

Method	POST		
			specified, GMS returns the latest value in the cache.
URL parameters			
stat_name	String	yes	Name of the statistic defined in the stat section of your configuration.
Body			
objectId, objectType, tenant, metric (or statisticType), filter, notificationMode, timeProfile, timeRange, timeRange2		yes	<p>The body can be either a MultiPart form or an x-www-form-urlencoded form consisting of different items representing the key/value pairs associated with the statistic (objectId, objectType, tenant, tenantPassword, metric, notificationMode, filter, timeProfile, timeRange, timeRange2).</p> <ul style="list-style-type: none"> <li>NotificationMode can be NoNotification, Reset, or Immediate.</li> <li>filter is the business attribute value to use to filter the results.</li> </ul>

## Response

<b>HTTP code</b>	200
<b>HTTP Message</b>	OK
<b>HTTP Header</b>	Content-Type: application/json;charset=UTF-8
<b>Body</b>	A JSON object representing the statistic.

If a problem occurs during operation the following status codes are returned:

HTTP code	HTTP message	Body
400	Bad request	message:

HTTP code	HTTP message	Body
		{"message": "stat.total-time not found", "exception": "com.genesyslab.gsg.s"} check GMS options
401	Unauthorized	Access to API refused

## PeekStat Request: Querying Multiple Statistic Values in a Single Request

This request gets the current values of several peek statistic objects.

### Operation

Method	POST		
URL	/genesys/1/statistics		
Parameter	Type	Mandatory	Description
<b>Body</b>			
objectId, objectType, tenant, metric (or statisticType), filter, notificationMode, timeProfile, timeRange, timeRange2		yes	<p>The body can be either a MultiPart form, an x-www-form-urlencoded form, or a JSON object consisting of different items representing the key/value pairs associated with the statistic (objectId, objectType, tenant, tenantPassword, metric, notificationMode, filter, timeProfile, timeRange, timeRange2).</p> <ul style="list-style-type: none"> <li>NotificationMode can be NoNotification, Reset, or Immediate.</li> <li>filter is the business attribute value to use to filter the results.</li> </ul>

Response

HTTP code	200
HTTP Message	OK
HTTP Header	Content-Type: application/json;charset=UTF-8
Body	A JSON object representing a structure with value

If a problem occurs during operation the following status codes are returned:

HTTP code	HTTP message	Body
200	OK	<p>You get an error message instead of one of the expected values if the requested statistic is incorrect; for example: message: {"stat1":8940,"stat2":"do not have 5 semicolon delimiters"}</p> <p>This means that, instead of getting a 400 Bad request response, you are able to retrieve statistics, even if one of them is incorrect.</p>
400	Bad request	message: message: {"message":"Error for stat1: Agent 'Kippola' (Tenant 'Environment') not found","exception":"com.genesyslab.gsg.s
401	Unauthorized	Access to API refused
403	Forbidden	message: {"message":"Wrong parameter","exception":null}
500	Server Error	message: {"message":"Metric (CurrentGrouargetState) not found on running StatServer","exception":"com.genesyslab.g

Important

You can make this request without using authentication by replacing **/genesys/1/statistics** with **/genesys/1/internal\_statistics**.

Examples with Multiple Statistics

The following example uses application/x-www-form-urlencoded:

Request

```
$ curl -v -u default:password -X POST --data "stat1=TotalLoginTime;Agent;KSippola;Environment;Immediate;&stat2=CurrentGroupTargetState;GroupAgents;Billing;E http://localhost:8080/genesys/1/statistics
```

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

## Response

HTTP/1.1 200 OK

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

```
{
  "stat1":248,
  "stat2":
  [
    {
      "agentDbId":102,
      "placeDbId":102,
      "agentId":"KSippola",
      "placeId":"SIP_Server_Place1",
      "extensions":
      [
        {"VOICE_MEDIA_STATUS":4},
        {"AGENT_VOICE_MEDIA_STATUS":4},
        {"DEVCAP":"AAIAAAAAAAEABXZvaWNlAAAAAAAEEAAAABVxTnEgAAAAAAQAAAAIABXZvaWNlAAAAAAAEEAAAABVxTnEgAEY2hhdAAAAAA
AAAAAFcU5xIAAAAA"}
      ],
      "mediaCapacityList":
      [
        {"mediaType":"chat",
          "currentInteractions":0,
          "maxInteractions":1,
          "currentMargin":1,
          "timestamp":1460961964},
        {"mediaType":"vmail",
          "currentInteractions":0,
          "maxInteractions":0,
          "currentMargin":0,
          "timestamp":1460554482},
        {"mediaType":"voice",
          "currentInteractions":0,
          "maxInteractions":1,
          "currentMargin":1,
          "timestamp":1460961964},
        {"mediaType":"webcallback",
          "currentInteractions":0,
          "maxInteractions":1,
          "currentMargin":1,
          "timestamp":1460961964}
      ],
      "dnstatusExList":
      [
        {"dnId":"7001",
          "gswDnTypes":1,
          "mediaCapacityList":
          [
            {"mediaType":"voice",
              "currentInteractions":0,
              "maxInteractions":1,
              "currentMargin":1,
              "timestamp":1460987666}
          ],
          "multimedia":0,
          "status":4,
          "switchId":"SIP_Switch",

```



```

        "time":1460554482},
    {"dnId":"7001_IM",
     "gswDnTypes":1,
     "mediaCapacityList":
     [
       {"mediaType":"voice",
        "currentInteractions":0,
        "maxInteractions":0,
        "currentMargin":0,
        "timestamp":1460987666},
       {"mediaType":"chat",
        "currentInteractions":0,
        "maxInteractions":255,
        "currentMargin":0,
        "timestamp":1460987666}
     ],
     "multimedia":1,
     "status":8,
     "switchId":"SIP_Switch",
     "time":1460553961}
  ]
}

```

The following example uses JSON:

### Request

```

$ curl -v -u default:password -H "Content-Type: application/json" -X POST --data
'{"stat1":{"objectId":"KSippola","objectType":"Agent","tenant":"Environment","tenantPassword":"","metric":"Total"}'
http://localhost:8080/genesys/1/statistics

```

Content-Type: application/json

### Response

```

HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8

```

```

{
  "stat1":3426,
  "stat3":
  [
    {"agentDbId":102,
     "placeDbId":102,
     "agentId":"KSippola",
     "placeId":"SIP_Server_Place1",
     "extensions":
     [
       {"VOICE_MEDIA_STATUS":4},
       {"AGENT_VOICE_MEDIA_STATUS":4},
     ]
    },
    {"DEVCAP":"AAIAAAAAAAEABXZvaWNlAAAAAAAEAAAABVxTzkwAAAAAAQAAAAIABXZvaWNlAAAAAAAVxTzkwAEY2hhdAAAAAA
    AAAAAFcU85MAAAAA"},
    {"mediaCapacityList":
     [
       {"mediaType":"chat",
        "currentInteractions":0,

```

```

        "maxInteractions":1,
        "currentMargin":1,
        "timestamp":1460961964},
    {"mediaType":"vmail",
     "currentInteractions":0,
     "maxInteractions":0,
     "currentMargin":0,
     "timestamp":1460554482},
    {"mediaType":"voice",
     "currentInteractions":0,
     "maxInteractions":1,
     "currentMargin":1,
     "timestamp":1460961964},
    {"mediaType":"webcallback",
     "currentInteractions":0,
     "maxInteractions":1,
     "currentMargin":1,
     "timestamp":1460961964}
  ],
  "dnstatusExList":
  [
    {"dnId":"7001",
     "gswDnTypes":1,
     "mediaCapacityList":
     [
       {"mediaType":"voice",
        "currentInteractions":0,
        "maxInteractions":1,
        "currentMargin":1,
        "timestamp":1460990867}
     ],
     "multimedia":0,
     "status":4,
     "switchId":"SIP_Switch",
     "time":1460554482},
    {"dnId":"7001_IM",
     "gswDnTypes":1,
     "mediaCapacityList":
     [
       {"mediaType":"voice",
        "currentInteractions":0,
        "maxInteractions":0,
        "currentMargin":0,
        "timestamp":1460990867},
       {"mediaType":"chat",
        "currentInteractions":0,
        "maxInteractions":255,
        "currentMargin":0,
        "timestamp":1460990867}
     ],
     "multimedia":1,
     "status":8,
     "switchId":"SIP_Switch",
     "time":1460553961}
  ]
}

```

## PeekStat Request: Filtering Multiple Statistic Values in a Single Request

### Important

This is a JSON query. This JSON query is available since 8.5.004.05.

### Operation

<b>Method</b>	POST		
<b>URL</b>	/genesys/1/statistics		
<b>Parameter</b>	<b>Type</b>	<b>Mandatory</b>	<b>Description</b>
<b>Body:</b> A JSON item representing the key/value pairs associated with statistics (objectId, objectType, tenant, tenantPassword, metric, notificationMode, filter).			
<ul style="list-style-type: none"><li>filter is the business attribute value to use to filter the results.</li><li>notificationMode can be set to NoNotification, Reset, or Immediate.</li></ul>			

### Response

<b>HTTP code</b>	200
<b>HTTP Message</b>	OK
<b>HTTP Header</b>	Age: containing the age (in seconds) of the statistic value.
<b>Body</b>	A JSON array of key/value pairs where key is the key stat (see request) and value is the statistic value.

If a problem occurs during subscription, the following status codes are returned:

HTTP code	HTTP Message	HTTP Body
400	Bad Request	The provided filter is incorrect; for example: { "message": "Filter name is not found", "exception": "com.genesyslab.gsg.s" }
403	Forbidden	Part of the submitted data is incorrect, for example: { "message": "Place 'SIP_Server_Place' (Tenant 'Environment') not found", "exception": "com.genesyslab.gsg.s" }

HTTP code	HTTP Message	HTTP Body
500	Internal server error	If Stat Server is not connected, it returns, for example, {"message": "Statistic Service unavailable", "exception": "com.genesyslab

## Example with Filter

### Query

http://localhost:8080/genesys/1/statistics

Input: JSON File

```
{
  "stat1": {
    "objectId": "KSippola",
    "objectType": "Agent",
    "tenant": "Environment",
    "tenantPassword": "",
    "metric": "TotalLoginTime",
    "filter": "Bronze"
  },
  "stat2": {
    "objectId": "9002@SIP_Switch",
    "objectType": "Queue",
    "tenant": "Environment",
    "metric": "ExpectedWaitTime"
  }
}
```

### Response

```
HTTP/1.1 200 OK
Date: Thu, 19 Jun 2014 09:06:43 GMT
Content-Type: application/json; charset=UTF-8
Content-Type: application/json
Content-Length: 25

{"stat1":0,"stat2":10000}
```

## EWT APIs

Starting in 8.5.112, new Estimated Waiting Time (EWT) APIs are available to query EWT. They use different calculation types to provide the Estimated Waiting Time. These APIs return either a JSON object presentation that describes the information in the virtual queue or a collection of such objects (one per every matched virtual queue the call is in). GMS retrieves all of the results from URS.

## Important

In your GMS configuration, add a connection to an active URS to enable this service.

To provide Estimated Waiting Time, three methods are available:

- **ewt**—URS checks how fast interactions go through the virtual queue and how many interactions are still pending. URS ignores the current agent availability and does not immediately adjust the EWT if, for example, all of the agents handling the queue suddenly logout.
- **ewt2**—Similarly to the first method, URS checks how fast interactions go through the virtual queue and how many interactions are pending. Additionally, URS takes into account the agents who have historically been handling interactions of the Virtual Queue. If all of these agents logout, URS notices and adjust the EWT value to a very high number like 10000 for example.
- **ewt3**—Query EWT from Stat Server.

## Query-EWT for a Virtual Queue

**Introduced in:** 8.5.109

**Modified in:** 8.5.112

You can query Estimated Wait Time statistics if you have configured a Virtual Queue for your Callback service by using the `_urs_virtual_queue` option. Three calculation types are available and use a URS query to retrieve the Estimated Wait Time in seconds.

- **ewt**

`http://<urshost>:<ursport>/urs/call/max/lvq?name=VQ_Name&aqt=urs&tenant=TenantName`

- **ewt2** (added in 8.5.112)

`http://<urshost>:<ursport>/urs/call/max/lvq?name=VQ_Name&aqt=urs2&tenant=TenantName`

- **ewt3** (added in 8.5.112)

`http://<urshost>:<ursport>/urs/call/max/lvq?name=VQ_Name&aqt=stat&tenant=TenantName`

## Operations

Name	Type	Mandatory	Description
<ul style="list-style-type: none"><li>• <b>GET /genesys/1/ewt/{VQ-Name}</b></li><li>• <b>GET /genesys/1/ewt2/{VQ-Name}</b></li><li>• <b>GET /genesys/1/ewt3/{VQ-Name}</b></li></ul>			
URI Parameters			

Name	Type	Mandatory	Description
{VQ-Name}	string	Yes	Alias name of a virtual queue configured in a service. If you are using a version older than 8.5.114, the VQ alias name must not include spaces. If your VQ alias name includes spaces, you should replace spaces with underscores, which renames the alias. For example, <code>GMS Chat VQ Multimedia Switch</code> should be renamed <code>GMS_Chat_VQ_Multimedia_Switch</code> .

## Response

<b>HTTP code</b>	200
<b>HTTP message</b>	OK
<b>Response Body (JSON content)</b>	
<b>Name</b>	<b>Description</b>
See object description below.	JSON-formatted string of information per virtual queue. The <code>ewt</code> parameter provides the Estimated Waiting Time in seconds. If you did

The JSON objects describing information in the virtual queue have the following properties:

Attribute	Description
time	UTC timestamp when the call entered in the virtual queue.
wt	Time in seconds that the call is in the virtual queue (effectively <code>CurrentTime-time</code> ).
calls	Current number of all calls in the virtual queue.
wcalls	Number of waiting calls in the virtual queue.
pos	Position of the call in the virtual queue.
priority	Calls priority in the virtual queue. Absent if scaling is used.
aqt	Average quitting rate of calls from the virtual queue. May be skipped if unknown.
ewt	Expected waiting time for the call.
hit	Percentage of calls distributed into the virtual queue. Its value is -1 if no calls were distributed yet.

In case of error, you will receive:

<b>HTTP code</b>	400
<b>HTTP message</b>	Not found

<b>HTTP code</b>	204
<b>HTTP message</b>	OK, but the record is empty.

## Examples

GET http://localhost:8080/genesys/1/ewt/VQ\_GMS\_Callback

```
200 OK
{
  "hit": 0,
  "ewt": 28.4,
  "calls": 9,
  "pos": 1,
  "aqt": 10000,
  "wpos": 10,
  "time": 1506021690,
  "wt": 0,
  "wcalls": 9
}
```

GET http://localhost:8080/genesys/1/ewt2/VQ\_GMS\_Callback

```
200 OK
{
  "hit": 0,
  "ewt": 126.8,
  "calls": 9,
  "pos": 3,
  "aqt": 10000,
  "wpos": 10,
  "time": 1506021690,
  "wt": 0,
  "wcalls": 9
}
```

GET http://localhost:8080/genesys/1/ewt3/VQ\_GMS\_Callback

```
200 OK
{
  "hit": 0,
  "ewt": 35.2,
  "calls": 9,
  "pos": 10,
  "aqt": 10000,
  "wpos": 10,
  "time": 1506021690,
  "wt": 0,
  "wcalls": 9
}
```

## Query-EWT for All Virtual Queues

### Introduced in: 8.5.112

You can query Estimated Wait Time statistics if you have configured at least a Virtual Queue for one of your Callback service by using the `_urs_virtual_queue` option. Three calculation types are available and use a URS query to retrieve the Estimated Waiting Time in seconds.

- `ewt`

`http://<urshost>:<ursport>/urs/call/max/lvq?aqt=urs&tenant=TenantName`

- `ewt2` (added in 8.5.112)

`http://<urshost>:<ursport>/urs/call/max/lvq?aqt=urs2&tenant=TenantName`

- `ewt3` (added in 8.5.112)

`http://<urshost>:<ursport>/urs/call/max/lvq?aqt=stat&tenant=TenantName`

## Operations

- **GET** `/genesys/2/ewt`
- **GET** `/genesys/2/ewt2`
- **GET** `/genesys/2/ewt3`

## Response

HTTP code	200
HTTP message	OK
Response Body (JSON content)	
Name	Description
JSON array	JSON array of objects. Each object describes the information per virtual queue. See below for details.

The JSON objects describing information in the virtual queue have the following properties:

Attribute	Description
<code>time</code>	UTC timestamp when the call entered in the virtual queue.
<code>wt</code>	Time in seconds that the call is in the virtual queue (effectively <code>CurrentTime-time</code> ).
<code>calls</code>	Current number of all calls in the virtual queue.
<code>wcalls</code>	Number of waiting calls in the virtual queue.



Attribute	Description
pos	Position of the call in the virtual queue.
priority	Calls priority in the virtual queue. Absent if scaling is used.
aqt	Average quitting rate of calls from the virtual queue. May be skipped if unknown.
ewt	Expected waiting time.
hit	Percentage of calls distributed into the virtual queue. Its value is -1 if no calls were distributed yet.

In case of error, you will receive:

<b>HTTP code</b>	400
<b>HTTP message</b>	Not found
<b>HTTP code</b>	204
<b>HTTP message</b>	OK, but the record is empty.

## Examples

GET http://localhost:8080/genesys/2/ewt

```
200 OK
{
  "VQ_GMS_Callback_Out": {
    "time": 1506021728,
    "wt": 0,
    "calls": 0,
    "wcalls": 0,
    "pos": 1,
    "wpos": 1,
    "aqt": 9.999,
    "ewt": 9.999,
    "hit": 0
  },
  "VQ_GMS_Callback": {
    "time": 1506021728,
    "wt": 0,
    "calls": 9,
    "wcalls": 9,
    "pos": 10,
    "wpos": 10,
    "aqt": 10000,
    "ewt": 100000,
    "hit": 0
  },
  "VQ_CB": {
    "time": 1506021728,
    "wt": 0,
    "calls": 5,
    "wcalls": 5,
    "pos": 6,
    "wpos": 6,
    "aqt": 10000,
  }
}
```

```
    "ewt": 60000,  
    "hit": 0  
  }  
}
```

GET http://localhost:8080/genesys/2/ewt2

200 OK

```
{  
  "VQ_GMS_Callback_Out": {  
    "time": 1506021728,  
    "wt": 0,  
    "calls": 0,  
    "wcalls": 0,  
    "pos": 1,  
    "wpos": 1,  
    "aqt": 9.999,  
    "ewt": 9.999,  
    "hit": 0  
  },  
  "VQ_GMS_Callback": {  
    "time": 1506021728,  
    "wt": 0,  
    "calls": 9,  
    "wcalls": 9,  
    "pos": 10,  
    "wpos": 10,  
    "aqt": 10000,  
    "ewt": 100000,  
    "hit": 0  
  },  
  "VQ_CB": {  
    "time": 1506021728,  
    "wt": 0,  
    "calls": 5,  
    "wcalls": 5,  
    "pos": 6,  
    "wpos": 6,  
    "aqt": 10000,  
    "ewt": 60000,  
    "hit": 0  
  }  
}
```

GET http://localhost:8080/genesys/2/ewt3

200 OK

```
{  
  "VQ_GMS_Callback_Out": {  
    "time": 1506021728,  
    "wt": 0,  
    "calls": 0,  
    "wcalls": 0,  
    "pos": 1,  
    "wpos": 1,  
    "aqt": 9.999,  
    "ewt": 9.999,  
    "hit": 0  
  },  
  "VQ_GMS_Callback": {  
    "time": 1506021728,  
    "wt": 0,  
    "calls": 9,  
    "wcalls": 9,  
    "pos": 10,  
    "wpos": 10,  
    "aqt": 10000,  
    "ewt": 100000,  
    "hit": 0  
  }  
}
```

```
    "wcalls": 9,  
    "pos": 10,  
    "wpos": 10,  
    "aqt": 10000,  
    "ewt": 100000,  
    "hit": 0  
  },  
  "VQ_CB": {  
    "time": 1506021728,  
    "wt": 0,  
    "calls": 5,  
    "wcalls": 5,  
    "pos": 6,  
    "wpos": 6,  
    "aqt": 10000,  
    "ewt": 60000,  
    "hit": 0  
  }  
}
```

## Configuration

### Stat Server Connection

In order to use the Stat Server API, there must be a connection to Stat Server in the GMS Application. You can create the connection in Configuration Manager on the *Connections* tab. See [Creating and Configuring the GMS Application Cluster Object](#).

You can add several Stat Servers in the *Connection* tab; this feature is used in case of different statistic definitions in the Stat Servers. GMS will open the statistic on the Stat Server to which the statistic belongs to. In the case of the same statistic definition in Stat Servers, GMS will take the first Stat Server that contains the statistic definition.

### ADDP Setting

Open Stat Server in the GMS *Connection* tab and set the connection protocol to *addp*. Set the values for the Local Timeout and Remote Timeout, and then select the Trace mode. See [Implementing ADDP](#) for more information.

#### Important

The ADDP traces are only visible on the Stat Server side.

The following example shows an ADDPtrace in Stat Server logs:

```
-AP[10] -<-2168 @15:37:30.4540  
-Ap[10] ->-2168 @15:37:30.4560
```

## High Availability

If Stat Servers (defined in GSM connections) are configured to use High Availability (HA) (Primary/Backup Stat Server), in the case of a lost connection with the primary Stat Server, the Stat Server API will switch to the backup Stat Server.

## Subscribe to Statistic Server Event Notifications

You can receive event notifications sent by the Statistics Server by using the CometD channel. To do so:

1. Register for GSM CometD channel.
2. Subscribe for a list of statistics through the new API: **POST /genesys/2/statistics**.

### Important

This Statistics API currently supports polling mode.

## Register for GSM CometD Channel

Create a client that will listen to CometD events sent by GSM, as shown in the following Node JS example:

```
var faye = require("faye");
var request = require('request');
// subscribe GSM comet
var referenceId;
var client = new faye.Client("http://localhost:8080/genesys/cometd");
client.setHeader("gms_user", "123456");
var subscr = client.subscribe("/_genesys", function (message) {
    var stats = message["message"]
    var statReferenceId = stats["statReferenceId"]
    ///...
    client.disconnect();
});
```

## Subscribe to Statistics through the Statistics API

Use the API to subscribe to a list of statistics and retrieve for each of them the reference ID of the Stat Server. Then, you will receive CometD event notifications associated with these reference IDs.

### Important

You must subscribe with the same gms\_user ID as the one used for the CometD client.

## Operation

POST /genesys/2/statistics			
Parameter	Type	Mandatory	Description
Header Parameters			
Content-type		Yes	application/json; charset=UTF-8
gms_user	String (UUID)	Yes	The user ID that has been used in CometD at registration time.
Body: The body is a JSON structure that lists the statistics to get notifications for.			
For example:			
<pre>{   "stat1": {     "objectId": "kmilburn",     "objectType": "Agent",     "tenant": "Environment",     "tenantPassword": "",     "metric": "TotalLoginTime",     "filter": ""   },   "stat2": {     "objectId": "KSippola",     "objectType": "Agent",     "tenant": "Environment",     "tenantPassword": "",     "metric": "TotalLoginTime",     "filter": "Bronze"   } }</pre>			

## Response

The response contains the reference IDs from the Stat Server subscribed statistics.

HTTP code	200
HTTP Message	OK
Body	<p>A JSON array of key/value pairs, where:</p> <ul style="list-style-type: none"><li>• The key is the name of the subscribed statistics.</li><li>• The value is the reference ID of the Stat Server subscribed statistics.</li></ul>

For example:

200 OK  
{ "stat1":1958339548, "stat2":1329710246 }

## Errors

HTTP Code	500
HTTP Message	Server Error
Body	<ul style="list-style-type: none"><li>• {"message":"Could not read JSON: Unexpected character ('\"' (code 34)): [...]"}.</li><li>• {"message":"Error for stat1: Supplied Object type does not belong to specified Stat Type.", "exception":"com.genesyslab.gsg.services.statistic.StatServerErrorException"}.</li></ul>

## Example

The following node JS sample shows how to subscribe statistics with this API.

```
var request = require('request');
// send request to subscribe stat using gms_user of comet subscription
request({
  url: 'http://localhost:8080/genesys/2/statistics', //URL to hit
  headers: {
    'Content-Type': 'application/json',
    'gms_user': '123456'
  },
  method: 'POST',
  json: true, // output data is JSON
  json: { // input data is JSON too
    "stat2": {
      "objectId": "KSippola",
      "objectType": "Agent",
      "tenant": "Environment",
      "tenantPassword": "",
      "metric": "TotalLoginTime",
      "filter": ""
    }
  }
}, function (error, response, body) {
  if (error) {
    ///...
  } else {
    referenceId = body["stat2"];
    ///...
  }
});
```

As a result of the subscription, you would receive the following reference IDs:

200 OK  
{ "stat1":1958339548, "stat2":1329710246 }

As a result, CometD Polling responses include these reference IDs too:

```
[
  {
    "data": {
      "id": "8f8dc3a0d00311e5845a0de0f4fa8277",
      "message": {
        "statValue": "0", "statReferenceId": "1958339548",
        "tag": "service.statistic.push.1958339548",
        "channel": "/_genesys"},
      "id": "55", "successful": true, "channel": "/meta/connect"}
    }
  ]
[
  {
    "data": {
      "id": "9560da60d00311e5845a0de0f4fa8277",
      "message": {
        "statValue": "0", "statReferenceId": "1329710246",
        "tag": "service.statistic.push.1329710246",
        "channel": "/_genesys"},
      "id": "56", "successful": true, "channel": "/meta/connect"}
    }
  ]
]
```

## Limitations

- CometD requests TTL is 120 seconds (by default).
- The statistics filter subscription is set to 10000 seconds, then the subscription is removed and you must send back the same stat request before the end of TTL subscription.
- You cannot define a statistic in Configuration Server for the GMS application that will be automatically subscribed at GMS startup. You must subscribe to the statistics as detailed above.

# Push Notification Service

## Overview

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

- [HttpCallback Notification](#)
- [Firebase Cloud Notification](#)
- [Android Notification \(deprecated\)](#)
- [Apple Notification](#)
- [Orchestration Server Callback Notification](#)
- [Windows Push Notification](#)

In addition to discussing these different types of notification, this page also describes [Notification Propagation](#). For details about the configuration options available for various types of notification, see the [push Section](#) in the Configuration Options Reference.

## 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 [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": "*"}
```

## Firebase Cloud Messaging Notification

### Introduced in: 8.5.114

Due to recent changes in Google Cloud Messaging, GMS now supports Firebase Cloud Messaging (FCM).

- Refer to the official documentation to [Set Up a Firebase Cloud Messaging Client App on Android](#).
- You will need to retrieve an API Key through the [Firebase Console](#).

To configure Native Push Notification through Firebase Cloud Messaging, you can either specify an `apiKey` or create a dedicated section to secure passwords (recommended for production environments) in the push section of your GMS application.

### Development

```
[push]
fcm.apiKey=<serverKey>
pushEnabled=fcm
```

### Production

```
[push]
fcm=fcmsection
pushEnabled=fcm
```

```
[fcmsection]
password=***** (<serverKey>)
```

- If you define these options in the push section, they will be used as default settings if you do not specify provider information in your API queries or service configuration.
- You can also define these options for non-default providers in the `[push.provider.{ProviderName}]` section, where `{ProviderName}` is the string you need to provide in the user data of your API queries or in your service configuration.

For example, you can configure `_provider_name={ProviderName}` for a given Callback service, or in a Chat V2 scenario, you could pass this provider name in the `push_notification_provider` user data of the `requestChat` operation.

GMS allows the following options for the provider-level configuration:

- `fcm.apiKey`
- `debug.fcm.apiKey`

Use `fcm.title` and `fcm.body` options to extend your configuration by creating an event-level `[push.provider.{ProviderName}.event]` section and then, a specific service name and event:

- [push.provider.{ProviderName}.event.{ServiceName}]
- [push.provider.{ProviderName}.event.{ServiceName}.{EventName}]

The following example shows how to configure in GMS two chat services for Bank's Saving Account and use localized messages in English and Russian. This configuration sample is applicable for Chat V2 service only, as event names are different for Chat V1 service even if you can configure it similarly.

```
[chat.savings-english]
endpoint = Environment:SavingsEnglish

[chat.savings-russian]
endpoint = Environment:SavingsRussian

[push.provider.bankoperations]
pushEnabled=ios, fcm
fcm.apiKey=****
apple.keystore=/var/genesys/gms/appleKeystore.p12
apple.keystorePassword=****

[push.provider.bankoperations.event]
fcm.body="Please open app for more details"

[push.provider.bankoperations.event.chat.savings-english.ParticipantJoined]
fcm.title="Agent has joined an waiting"

[push.provider.bankoperations.event.chat.savings-english.Message]
fcm.title="You got new message from us"
fcm.body="Please answer us soon!"

[push.provider.bankoperations.event.chat.savings-russian.ParticipantJoined]
fcm.title="Агент присоединился и ждет"
fcm.body="Ответьте нам поскорее"

[push.provider.bankoperations.event.chat.savings-russian.Message]
fcm.title="У Вас новое сообщение!"
fcm.body="Ответьте нам поскорее"
```

## Android Notification (Deprecated)

### GCM Service

**Deprecated in: 8.5.114**

### Important

GCM is now deprecated. You can no longer create new GCM projects. However, previous projects are still available.

Android GCM notification relies on the new Google Cloud Messaging (GCM) service, described [here](#). GCM notifications are made on behalf of an apiKey that is created in [Google services](#) 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 allow HTTPS connections to the following addresses to use this functionality:
  - <http://developer.android.com/google/gcm/index.html>.

### Important

When subscribing for notifications via GCM, it is important to ensure that the device's *registration ID* is provided as **notificationDetails.deviceId**. This registration ID must be obtained by registering the device with GCM servers through the Google Services API. For specific client implementation details, please refer to:

- <http://developer.android.com/google/gcm/client.html>

## 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:
  - <http://developer.android.com/google/gcm/index.html>.
2. Import those certificates into the keystore you plan to use.

### Important

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)

## Apple Notification

### Modified in 8.5.114

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: [Apple Push Notification Service](#).

Starting in 8.5.114, Digital Channels Chat V2 API with CometD supports native push notifications through Apple Push Notification Service. APNS support is limited to [Legacy Binary Provider API](#). To establish a TLS session with APNs, install an Entrust Secure CA root certificate on the provider's server. If the server is running macOS, this root certificate is already part of the keychain. On other systems, the certificate might not be available. You can download this certificate from the [Entrust SSL Certificates](#) website.

## 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 relies on the corresponding ORS Service.

## 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 configuration 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 be 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.



## Sensitive Options

You can set sensitive options in different sections of your application configuration. In the following example, the configuration structure allows to set the password for apple in a new section called apple-desc.

```
[push]
apple=apple-desc;
debug-apple=debug-apple-desc
```

```
[apple-desc]
password=*****
apple.keystore=xxxxxxxxxx
[debug-apple-desc]
password=*****
apple.keystore=xxxxxxxxxx
```

In case of:

- IOS (APNS): The password option is used instead of apple.keystorePassword.
- Android (GCM): The password option is used instead of android.gcm.apiKey.
- Microsoft (WNS): The password option is used instead of wns.clientSecret.

You do not need to change the name of other options. Genesys recommends that you create a new section to set the mandatory options and that you keep optional options in the push section. See the [options' detail](#) for more information.

## Windows Notification

WNS notification relies on the new [Windows Push Notification Services](#) (WNS).

Before you can send notifications using WNS, your app must be registered with the Windows Store Dashboard. This will provide you with credentials for your app that your cloud service will use in authenticating with WNS. These credentials consist of a Package Security Identifier (SID) and a secret key. To perform this registration, go to the Windows Dev Center and select Dashboard. This SID and secret key need to be set by the configuration options for the Genesys Mobile Services Application object.

---

# Callback Push Notifications for Android

Genesys Mobile Services (GMS) employs various mechanisms to achieve asynchronous messaging (push notifications). For Android devices, it is a combination of GCM/C2DM, and Comet. Likewise, iOS devices employ APNs and Comet. The scope of this article is limited to how push notifications are handled in Android devices, particularly for the Callback application.

Note: For iOS devices, see [CallbackPushNotificationsforiOS](#).

## Procedure

[Implementing a GCM client](#) is well documented by Google. Alternately, you can also refer to the [Genesys Mobile Services Android Sample](#) for a Genesys implementation of a GCM BroadcastReceiver.

Push notifications can be divided into two distinct parts:

1. Chat (implements push notifications over Comet)
2. Callback (implemented over GCM)

For Chat, a Bayeux Client is created to listen to all push notifications related to Chat. The default channel for Chat is `/_genesys`. The format of Chat push notifications can be seen in the [Chat \(Comet\) section](#).

For Callback push notifications, refer to the [Genesys Mobile Services Android Sample](#) for reference.

Processing of GCM notifications is a three-step procedure:

1. Obtain Service ID and Action identifier from GCM Intent.
2. Issue HTTP POST to GMS with specified action to obtain action data as JSON.
3. Execute action using data provided by GMS.

The data contained within the GCM Intent is structured as follows:

```
Intent intent;  
Bundle extras = intent.getExtras();  
String message = extras.getString("message");  
System.out.println(message);
```

```
-----  
Result:
```

```
{  
  "_id": "$(_id)",  
  "_action": "$(_action)",  
}
```

In the case of the Genesys sample client, the `GenesysCloudMessageReceiver` repackages the GCM Intent into an application-specific Intent:

```
Intent newIntent = new Intent(context, GenesysSampleActivity.class);
newIntent.setAction(Globals.ACTION_GENESYS_CLOUD_MESSAGE);
newIntent.putExtra(Globals.EXTRA_MESSAGE, extras.getString("message"));
newIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
context.startActivity(newIntent);
```

This intent is then handled by the `GenesysSampleActivity (handleIntent() > interpretCloudMessage())` where the encapsulated data is used to form an HTTP POST with the following URL:

```
$(ServerURL)$(URLPath)$( _id)/$( _action)
for example,
http://135.34.145.123:8080/genesys/1/service/3SQI3S31693JL9
L3R00506T40C000U73/get-dialog-start-chat
```

Identifier	Description	Example Values
\$(ServerURL)	URL to Genesys Mobile Services host	http://135.34.145.123:8080
\$(URLPath)	Path to Services API	/genesys/1/service/
\$( _id)	GMS-issued Service ID	3SQI3S31693JL9L3R00506T40C000U73
\$( _action)	Callback action to perform	get-dialog-user-confirmation-provide_code-true get-dialog-user-confirmation-provide_code-false get-dialog-start-chat connect-inbound connect-outbound wait-for-agent

The response of the HTTP POST request contains JSON, which describes an action to perform and/or UI elements to display in the client application. Each of these requests are referred to as Dialogs.

## Dialogs (REST)

The following examples are JSON structures returned by the GMS Callback service to the client application. The contents of the JSON response depends on the Callback action performed (as described in the [Procedure section](#)).

Refer to the [Genesys Mobile Services Android Sample](#) for examples on how these JSON responses can be interpreted as actions (for example: Call agent, Display menu, Display dialog) and/or UI elements (for example: Confirmation dialogs or Menu items).

### get-dialog-user-confirmation-provide\_code-true

```
{
  "_dialog_id": "0",
  "_label": "Agent is available right now",
  "_user_action_url":
"$ (ExtURLBase)/1/service/$(ServiceID)/not-used",
  "_method": "POST",
  "_action": "DisplayMenu",
  "_expires": "$(Date)",
```

```

    "_resource_url":
    "${ExtURLBase}/1/service/${ServiceID}/get-dialog-user-confirmation",
    "_content": [
        {
            "_group_name": "Are you ready?",
            "_group_content": [
                {
                    "_dialog_id": "1",
                    "_label": "Yes, I'm ready to talk",
                    "_action": "MenuItem",
                    "_user_action_url":
                    "${ExtURLBase}/1/service/${ServiceID}/connect",
                    "_method": "POST",
                    "_id_to_jump_before": "exit://",
                    "_confirmation_dialog": {
                        "_text":
                        "You will hear tones immediately after call is connected. This is normal.",
                        "_dialog_type": "Notification",
                        "_dismiss_timeout": 2
                    }
                }, {
                    "_dialog_id": "2",
                    "_label": "No, try again in 5 minutes",
                    "_action": "MenuItem",
                    "_user_action_url":
                    "${ExtURLBase}/1/service/${ServiceID}/snooze",
                    "_method": "POST",
                    "_id_to_jump_before": "exit://"
                }, {
                    "_dialog_id": "3",
                    "_label": "Cancel, my problem has been solved",
                    "_action": "MenuItem",
                    "_user_action_url":
                    "${ExtURLBase}/1/service/${ServiceID}/cancel",
                    "_method": "POST",
                    "_id_to_jump_before": "exit://"
                }
            ]
        }
    ]
}

```

### get-dialog-user-confirmation-provide\_code-false

```

{
    "_dialog_id": "0",
    "_label": "Agent is available right now",
    "_user_action_url":
    "${ExtURLBase}/1/service/${ServiceID}/not-used",
    "_method": "POST",
    "_action": "DisplayMenu",
    "_expires": "${Date}",
    "_resource_url":
    "${ExtURLBase}/1/service/${ServiceID}/get-dialog-user-confirmation",
    "_content": [
        {
            "_group_name": "Are you ready?",
            "_group_content": [
                {
                    "_dialog_id": "1",
                    "_label": "Yes, I'm ready to talk",
                    "_action": "MenuItem",
                    "_user_action_url":

```

```

"$(ExtURLBase)/1/service/$(ServiceID)/connect",
  "_method": "POST",
  "_id_to_jump_before": "exit://",
}, {
  "_dialog_id": "2",
  "_label": "No, try again in 5 minutes",
  "_action": "MenuItem",
  "_user_action_url":
"$(ExtURLBase)/1/service/$(ServiceID)/snooze",
  "_method": "POST",
  "_id_to_jump_before": "exit://"
}, {
  "_dialog_id": "3",
  "_label": "Cancel, my problem has been solved",
  "_action": "MenuItem",
  "_user_action_url":
"$(ExtURLBase)/1/service/$(ServiceID)/cancel",
  "_method": "POST",
  "_id_to_jump_before": "exit://"
}
}
]
}
}
}

```

### get-dialog-start-chat

```

{
  "_dialog_id": "1",
  "_action": "StartChat",
  "_label": "Start Chat",
  "_start_chat_url":
"$(ExtURLBase)/1/service/$(ServiceID)/ixn/chat",
  "_comet_url": "$(CometURL)",
  "_user_header": "$(GMSUser)",
  "_id_to_jump_before": "exit://",
  "_chat_parameters": {
    "subject": "None"
  },
  "_id": "$(ServiceID)"
}

```

### connect-inbound

```

{
  "_dialog_id": "0",
  "_label": "Connecting ...",
  "_action": "DialNumber",
  "_tel_url": "n/a",
  "_access_code": "n/a",
  "_id": "$(ServiceID)"
}

```

### connect-outbound

```

{
  "_dialog_id": "0",
  "_action": "ConfirmationDialog",
  "_text": "You will receive the call shortly",
  "_ok_title": "Ok",
  "_id": "$(ServiceID)"
}

```

```
}
```

### wait-for-agent

```
{
  "_dialog_id": "0",
  "_action": "ConfirmationDialog",
  "_text": "We will notify you when agent is available",
  "_ok_title": "Ok",
  "_id": "${ServiceID}"
}
```

## Push Notifications

### Chat (Comet)

#### Message Receipt

```
{
  "data": {
    "message": {
      "startedAt": "2014-05-02T16:27:38Z",
      "chatIxnState": "TRANSCRIPT",
      "chatSessionId": "000FRa9NYM9A001K",
      "transcriptToShow": [ ["Message.Text", "Stan", "Hello.", "8", "CLIENT"] ],
      "transcriptPosition": "2",
      "chatServiceMessage": "Chat service is available"
    },
    "id": "b2e607a0d21611e3000010932938a0ff",
    "tag": "service.chat.refresh.3SQIS3S1693JL9L3R00506T40C000UL4"
  },
  "channel": "/_genesys"
}
```

#### Party Joined/Left

```
{
  "data": {
    "message": {
      "startedAt": "2014-05-02T16:27:38Z",
      "chatIxnState": "TRANSCRIPT",
      "chatSessionId": "000FRa9NYM9A001K",
      "transcriptToShow": [ ["Notice.Joined", "Kristi Sippola",
"has joined the session", "17", "AGENT"] ],
      "transcriptPosition": "3",
      "chatServiceMessage": "Chat service is available",
    },
    "id": "b7dd6460d21611e3000010932938a0ff",
    "tag": "service.chat.refresh.3SQIS3S1693JL9L3R00506T40C000UL4"
  },
  "channel": "/_genesys"
}
```

#### Typing Started/Stopped

```
{
```

```
{
  "data": {
    "message": {
      "startedAt": "2014-05-02T16:27:38Z",
      "chatIxnState": "TRANSCRIPT",
      "chatSessionId": "000FRa9NYM9A001K",
      "transcriptToShow": [
        ["Notice.TypingStarted",
        "Kristi Sippola", "is typing", "20", "AGENT"]],
      "transcriptPosition": "4",
      "chatServiceMessage": "Chat service is available",
    },
    "id": "b91bd7d0d21611e3000010932938a0ff",
    "tag": "service.chat.refresh.35QIS3S1693JL9L3R00506T40C000UL4"
  },
  "channel": "/_genesys"
}
```

## Notes

Identifier	Description	Values
\$(TranscriptType)	Type of event to display in the chat log.	Message.Text Notice.TypingStarted Notice.TypingStopped Notice.Joined Notice.Left
\$(Timestamp)	UTC Time format	YYYY-MM-DDTHH:MM:SSZ
\$(TranscriptPosition)	Line Number	Some integer.
\$(ChatIxnState)	State of chat interaction.	TRANSCRIPT DISCONNECTED

# Callback Push Notifications for iOS

This article is for developers who wish to develop an iOS client application for Genesys Mobile Services (GMS) Callback Services.

GMS employs various mechanisms to achieve asynchronous messaging (push notifications). For iOS devices, Apple Push Notification service (APN) and Comet are used. For Android devices, GCM/C2DM and Comet are used. This article describes how push notifications should be handled in iOS devices. The [Genesys Mobile Services iOS Sample](#) provides example code for the concepts discussed.

## Important

For Android devices, see [CallbackPushNotifications for Android](#).

## Push Notifications in iOS Applications

Push notifications are used for two purposes in GMS iOS applications:

- Chat
- Callback

### Chat

The GMS chat implementation uses Comet push notifications for the text message exchange. GMS implements a Comet server that the mobile client connects to when a chat session is opened.

The iOS sample code includes a Comet library. The library includes a Comet client class called `DDCometClient`. The iOS sample application `GMSChatViewController` class illustrates how to use `DDCometClient` to connect to the server and to send and receive chat messages. The default channel used for chat is `"/_genesys"`.

### Callback

The GMS Callback functions utilize APN for push notifications to iOS applications. The processing of APN notifications operates as follows:

1. Obtain the Service ID and Action identifier from the notification “message” component (the “message” component is in JSON format).
2. Issue an HTTP POST to GMS with specified action. The response includes further action data in JSON format.
3. Execute action using data provided by GMS.



The APN notification contains several components. The “aps” component specifies the confirmation dialog to display to the user. You can refer to Apple developer documentation for more information on this topic. The iOS sample application `GMSAppDelegate` `didReceiveRemoteNotification` method provides an example. The “message” component provides the data from GMS and has the following format:

```
{
  "_id": "$(_id)",
  "_action": "$(_action)",
}
```

The `_id` and `_action` parameters are extracted and used to construct a URL for a POST to GMS. The iOS sample application `GMSAppDelegate` `processAPN` method provides an example of this.

```
$(ServerURL)$(URLPath)$(_id)/$(_action)
such as
http://135.34.145.123:8080/genesys/1/service
/3SQI3S31693JL9L3R00506T40C000U73/get-dialog-start-chat
```

Identifier	Description	Example Values
<code>\$(ServerURL)</code>	URL to Genesys Mobile Services host	<code>http://135.34.145.123:8080</code>
<code>\$(URLPath)</code>	Path to Services API	<code>/genesys/1/service/</code>
<code>\$(_id)</code>	GMS-issued Service ID	<code>3SQI3S31693JL9L3R00506T40C000U73</code>
<code>\$(_action)</code>	Callback action to perform	<code>get-dialog-user-confirmation-provide_code-true</code> <code>get-dialog-user-confirmation-provide_code-false</code> <code>get-dialog-start-chat</code> <code>connect-inbound</code> <code>connect-outbound</code> <code>wait-for-agent</code>

The response of the HTTP POST request contains JSON data, which describes an action to perform and/or UI elements to display in the client application. Each of these requests will be referred to as Dialogs.

## Dialogs (REST)

The following are example JSON structures returned by the GMS Callback service to the client application. The contents of the JSON response depends on the Callback action performed (as described in the [Callback section](#)).

Refer to the [Genesys Mobile Services iOS Sample](#) for examples on how these JSON responses can be interpreted as actions (for example, Call agent, Display menu, Display dialog) and/or UI elements (for example, Confirmation dialogs or Menu items).

### get-dialog-user-confirmation-provide\_code-true

```
{
  "_dialog_id": "0",
  "_label": "Agent is available right now",
  "_user_action_url":
    "$(ExtURLBase)/1/service/$(ServiceID)/not-used",
}
```

```

    "_method": "POST",
    "_action": "DisplayMenu",
    "_expires": "${Date}",
    "_resource_url":
    "${ExtURLBase}/1/service/${ServiceID}/get-dialog-user-confirmation",
    "_content": [
      {
        "_group_name": "Are you ready?",
        "_group_content": [
          {
            "_dialog_id": "1",
            "_label": "Yes, I'm ready to talk",
            "_action": "MenuItem",
            "_user_action_url":
              "${ExtURLBase}/1/service/${ServiceID}/connect",
            "_method": "POST",
            "_id_to_jump_before": "exit://",
            "confirmation_dialog": {
              "_text": "You will hear tones immediately
                        after call is connected. This is normal.",
              "_dialog_type": "Notification",
              "_dismiss_timeout": 2
            }
          }, {
            "_dialog_id": "2",
            "_label": "No, try again in 5 minutes",
            "_action": "MenuItem",
            "_user_action_url":
              "${ExtURLBase}/1/service/${ServiceID}/snooze",
            "_method": "POST",
            "_id_to_jump_before": "exit://"
          }, {
            "_dialog_id": "3",
            "_label": "Cancel, my problem has been solved",
            "_action": "MenuItem",
            "_user_action_url":
              "${ExtURLBase}/1/service/${ServiceID}/snooze",
            "_method": "POST",
            "_id_to_jump_before": "exit://"
          }
        ]
      }
    ]
  }
}

```

#### get-dialog-user-confirmation-provide\_code-false

```

{
  "_dialog_id": "0",
  "_label": "Agent is available right now",
  "_user_action_url":
    "${ExtURLBase}/1/service/${ServiceID}/not-used",
  "_method": "POST",
  "_action": "DisplayMenu",
  "_expires": "${Date}",
  "_resource_url":
    "${ExtURLBase}/1/service/${ServiceID}/get-dialog-user-confirmation",
  "_content": [
    {
      "_group_name": "Are you ready?",
      "_group_content": [
        {
          "_dialog_id": "1",

```

```

        "_label": "Yes, I'm ready to talk",
        "_action": "MenuItem",
        "_user_action_url":
            "$(ExtURLBase)/1/service/$(ServiceID)/connect",
        "_method": "POST",
        "_id_to_jump_before": "exit://",
    }, {
        "_dialog_id": "2",
        "_label": "No, try again in 5 minutes",
        "_action": "MenuItem",
        "_user_action_url":
            "$(ExtURLBase)/1/service/$(ServiceID)/snooze",
        "_method": "POST",
        "_id_to_jump_before": "exit://"
    }, {
        "_dialog_id": "3",
        "_label": "Cancel, my problem has been solved",
        "_action": "MenuItem",
        "_user_action_url":
            "$(ExtURLBase)/1/service/$(ServiceID)/snooze",
        "_method": "POST",
        "_id_to_jump_before": "exit://"
    }
    ]
}
]
}

```

### get-dialog-start-chat

```

{
    "_dialog_id": "1",
    "_action": "StartChat",
    "_label": "Start Chat",
    "_start_chat_url":
        "$(ExtURLBase)/1/service/$(ServiceID)/ixn/chat",
    "_comet_url": "$(CometURL)",
    "_user_header": "$(GMSUser)",
    "_id_to_jump_before": "exit://",
    "_chat_parameters": {
        "subject": "None"
    },
    "_id": "$(ServiceID)"
}

```

### connect-inbound

```

{
    "_dialog_id": "0",
    "_label": "Connecting ...",
    "_action": "DialNumber",
    "_tel_url": "n/a",
    "_access_code": "n/a",
    "_id": "$(ServiceID)"
}

```

### connect-outbound

```

{
    "_dialog_id": "0",
    "_action": "ConfirmationDialog",

```

```
    "_text": "You will receive the call shortly",
    "_ok_title": "Ok",
    "_id": "${ServiceID}"
}
```

### wait-for-agent

```
{
  "_dialog_id": "0",
  "_action": "ConfirmationDialog",
  "_text": "We will notify you when agent is available",
  "_ok_title": "Ok",
  "_id": "${ServiceID}"
}
```

## Push Notifications

### Chat (Comet)

#### Message Receipt

```
{
  "data": {
    "message": {
      "startedAt": "2014-05-02T16:27:38Z",
      "chatIxnState": "TRANSCRIPT",
      "chatSessionId": "000FRa9NYM9A001K",
      "transcriptToShow": [ ["Message.Text", "Stan", "Hello.", "8", "CLIENT"] ],
      "transcriptPosition": "2",
      "chatServiceMessage": "Chat service is available"
    },
    "id": "b2e607a0d21611e3000010932938a0ff",
    "tag": "service.chat.refresh.3SQIS3S1693JL9L3R00506T40C000UL4"
  },
  "channel": "/_genesys"
}
```

#### Party Joined/Left

```
{
  "data": {
    "message": {
      "startedAt": "2014-05-02T16:27:38Z",
      "chatIxnState": "TRANSCRIPT",
      "chatSessionId": "000FRa9NYM9A001K",
      "transcriptToShow": [ ["Notice.Joined", "Kristi Sippola",
        "has joined the session", "17", "AGENT"] ],
      "transcriptPosition": "3",
      "chatServiceMessage": "Chat service is available",
    },
    "id": "b7dd6460d21611e3000010932938a0ff",
    "tag": "service.chat.refresh.3SQIS3S1693JL9L3R00506T40C000UL4"
  },
  "channel": "/_genesys"
}
```

## Typing Started/Stopped

```
{
  "data":{
    "message":{
      "startedAt":"2014-05-02T16:27:38Z",
      "chatIxnState":"TRANSCRIPT",
      "chatSessionId":"000FRa9NYM9A001K",
      "transcriptToShow":[["Notice.TypingStarted",
        "Kristi Sippola","is typing","20","AGENT"]],
      "transcriptPosition":"4",
      "chatServiceMessage":"Chat service is available",
    },
    "id":"b91bd7d0d21611e3000010932938a0ff",
    "tag":"service.chat.refresh.3SQIS3S1693JL9L3R00506T40C000UL4"
  },
  "channel":"/_genesys"
}
```

## Notes

Identifier	Description	Values
\$(TranscriptType)	Type of event to display in the chat log.	Message.Text Notice.TypingStarted Notice.TypingStopped Notice.Joined Notice.Left
\$(Timestamp)	UTC Time format	YYYY-MM-DDTHH:MM:SSZ
\$(TranscriptPosition)	Line Number	Some integer.
\$(ChatIxnState)	State of chat interaction.	TRANSCRIPT DISCONNECTED

# Chat API Version 1 Push Notification and Background State

## Mobile Application Background State Issues

If an iOS or Android mobile application is moved to the background state, the operating system does not close active TCP connections. So, if your app does not handle the foreground and background events, the server-side of your CometD connection isn't notified.

More specifically, if your app handles chat with GMS, the agent may send a new message which would result in sending the long poll response. Here is a list of the various scenarios which can happen according to your implementation.

- The Cometd server attempts to send long poll response but fails and closes socket of current long-poll request.
- The Cometd server sets a 10-second timer; if the app does not reconnect within this period, the app is considered to be gone and its context is destroyed.

Later, if the app returns to the foreground, it does not receive chat notifications from the Cometd server and the chat session appears to be broken.

To avoid these types of issues, your app should listen for events that both iOS and Android submit when your app enters the background (or foreground) state. See the Official documentation:

- Apple provides background instructions for iOS, [here](#)
- Android provides best background practices [here](#).

Then, your app should implement the Cometd Meta Disconnect message detailed below to handle background State issues.

## Cometd Meta Disconnect Messages

To handle background state, your app must send Cometd `/meta/disconnect` messages to GMS and later, if your app returns to foreground, it should establish a new Cometd session as detailed below.

1. When your app enters the background state, it sends a Comet/meta/disconnect message to GMS which includes the highest transcript position received by your app, as for example:

```
{
  "channel": "/meta/disconnect",
  "clientId": "71k6evjfbffqq9eir3jhn6udxz",
  "ext":
```

```

    {
      "transcriptPosition": "4"
    },
    "id": "4"
  }

```

2. If GMS receives a `/meta/disconnect` message, it flags your app as disconnected and sends a chat Notice.Custom event to the agent. The default value is `customer is not online`. You can configure this text.

1. Edit your GMS application (with Configuration Manager for example).
2. Navigate to **Options > Service.<service\_name>**.
3. Edit the `_agent_timeout_notification_message` value.

3. If the agent sends a new message while your app is disconnected, GMS starts a configurable timer. You can set this timer value to zero:

1. Edit your GMS application (with Configuration Manager for example).
2. Navigate to **Options > Service.<service\_name>**.
3. Edit the `_client_timeout_notification` value.

`</toggledisplay>`. When the timer expires, GMS sends a push notification (iOS apns, Android gcm, or http). Typically, the notification results in a dialog which allows the user to optionally return the app to the foreground.

4. If the app returns to the foreground, it must start a new Cometd session with GMS. GMS flags the app as connected and sends a Cometd notification that includes all chat transcript events which occurred after the app entered the background state, as for example:

```

[
  {
    "data":{
      "id":"fdf93730c1e411e4a5e8f1be76b28efd",
      "message":
        { "chatSessionId":"0001BaAFC4J8000N", "transcriptPosition":
          "6", "chatServiceMessage":"Chat service is available", "startedAt":
            "2015-03-03T20:36:12Z", "chatIxnState":"TRANSCRIPT",
            "transcriptToShow":[ ["Notice.TypingStarted","agentName","is typing","28","AGENT"],
              ["Message.Text","agentName","Hello","29","AGENT"] ] },
      "tag":
        "service.chat.refresh.180-655e104c-a6cf-447d-b94b-f132d49a35fe"
    },
    "channel":"/_genesys"
  },
  { "successful":true, "channel":"/meta/connect" }
]

```

## Important

- If your app returns to foreground before the timer expires, no push notification is sent, but your app must still start a new Cometd session and it will receive a Cometd notification (which includes all new transcript data) from GMS.
- If your app does not return to foreground after the first push notification is sent, GMS

sends additional push notification messages for each new agent event. Note that these events are sent immediately.

## Content of Push Notification Messages

GMS triggers push notification messages if the agent submits new chat events to the app flagged as disconnected. The following events can result in a notification:

- Notice.TypingStarted
- Notice.TypingStopped
- Notice.Joined
- Notice.Left
- Notice.PushUrl
- Notice.Custom
- Message.Text

Your application's configuration includes a filter to exclude events from triggering a push notification. The `filtering_chat_events` option in the push section sets the default value for this filter to `Notice.TypingStarted,Notice.TypingStopped`. You can still set a different value for your service.

1. Edit your GMS application (with Configuration Manager for example).
2. Navigate to **Options > Service.<service\_name>**.
3. Edit the `_filtering_chat_events` option to add new event types to exclude.

For the Android and HTTP push notification types, notification messages include the content of the filtered agent events. Currently, this is not supported for iOS push notification messages due to the 256-byte payload size limit for this notification type.

The following example of the push notification message includes the content of the filtered agent events.

```
"message": {
  "message": "New message from Agent",
  "serviceId": "180-e941460d-1eb0-4f0b-9022-b99ca9ee41f7",
  "lastTranscript": [
    {"Message.Text": "A line of text."}
    {"Message.Text": "Another line of text."}
  ]
}
```

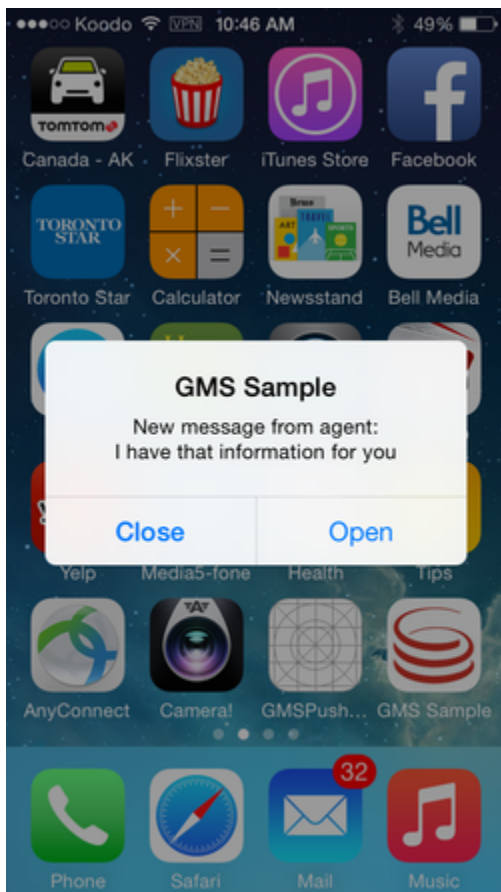
By default, the message attribute (or notification message) is set to `New message from Agent`, but you can change this text in your service options.



1. Edit your GMS application (with Configuration Manager for example).
2. Navigate to **Options > Service.<service\_name>**.
3. Edit the value of the `_client_timeout_notification_message` option.

The notification is tagged as `chat.newagentmessage`. In addition to the notification message, the notification content provides the `lastTranscript` text that can be displayed to the user.

You can display this the `lastTranscript` text in a popup a or banner notification to the user.



## Specific Configuration for Chat Push Notification

To customize your GMS configuration for push notification messages (for both iOS and Android), you can create a `push.provider.event.chat.newagentmessage` section in the **Options** tab (with Configuration Manager for instance). You can add there any of your Android or iOS options.

- This additional configuration is not mandatory.

### Tip

- Read more about options of the GMS push section [here](#).
- Read more about the OS specific capabilities associated with the notification message [here](#).

# 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.

Simple style	Expanded style
<pre>{   "message": "welcome",   "tag": "yourtag",   "mediaType": "localizestring",   "locArgs": {     "customer.lastname": "Doe"   } }</pre>	<pre>{   "message": "welcome",   "tag": "yourtag",   "mediaType": "localizestring",   "locArgs": {     "customer": {       "lastname": "Doe"     }   } }</pre>

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](#).