Genesys Mobile Services API Reference

Genesys Mobile Engagement 8.1.2

1/20/2022

Table of Contents

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

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.

Important

Scenarios have been moved to the Service Admin UI Help.

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).

Operation

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

Response

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

HTTP code	200
HTTP message	OK

Example

The following example stores:

- Key1, Key2, Key3 and FileKey

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

Operation

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

Result
The above data is now stored 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**

Method	POST		
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.
Body: 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.
Body: Not used			

Response

HTTP code	200
HTTP message	OK

Example

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

Operation

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

Result

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

Query (one key)

Queries 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
Body: Not used			

Response

HTTP code	200
HTTP message	OK

Example

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

Operation

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

Result

```
Value1
```

Query (one binary key)

Queries 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
Body: Not used			

Response

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

Example

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

Operation

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

Delete

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

Operation

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

Response

HTTP code	200
HTTP message	OK

Example

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

Operation

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

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

Overview

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.

API

Query

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

Operation

Method	GET		
URL	/genesys/1/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.

Notification API

Overview

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

This set of APIs is used to manage notifications between applications and Genesys systems. It is event driven, that is, consumers subscribe to an event and provide an indication of how the notification should be delivered, and events are published to the system. For the 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":"${type}"},
   "expire":30,
   "filter":"${filter}"
}
```

Here:

- **subscriberId** – The id of subscriber (mandatory).
- **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 Push Notification Service for details on language.
- **notificationDetails** - (mandatory) Describes all the information needed for delivering the event to concrete subscriber
  - type - (Mandatory) this parameter defines what type of notification mechanism that the application wants to use. Valid values are **ios** (to-apple), **android** (to-android-device), gcm (to-android-gcm-device), **comet** (to-cometd-client), **httpcb** (callback POST to provided url) and **orscb** (callback to ORS) (see more information here Push Notification Service).
  - deviceId - (Mandatory) id of device to deliver message to (in the case of http or ORS callback - see the details here PushNotificationService.
  - 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":"${mediaType}",
  "notificationDetails":
  {
    "deviceId":"${devideId}",
    "type":"${type}",
    "properties":
    {
     "debug":"{true/false}"
    }
  }
}
```

- tag – (mandatory) The message tag.
- message– (optional) Some string. It may contain string representation of ANY data: notification service is message-agnostic; it ALWAYS interprets message as String. If no message is specified, the empty string will be sent to subscribers. The only restriction on **message** format is: it must not crash the JSON parser which attempts to parse the request body. If this happens - 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.
- devideId - (mandatory) The id of the device (for example, Android device id or iPad id).
- type - (mandatory) Yype 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 wildcart ‘*’ after last ‘.’ (or only ‘*’ – denotes subscription to all events), the last char can not be '.'. So, the channels like the following are allowed:

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

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

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

APIs

The standard `InternalServerError` with code 500, or `BAD_REQUEST` with code 400, can be returned as response to each request, so it 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

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

Response

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

HTTP code	200
HTTP message	OK

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

HTTP code	400
HTTP message	BAD REQUEST

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

HTTP code	404
HTTP message	NOT FOUND

Delete Subscription

This call cancels/terminates a given subscription.

Operation

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

Returns

Nothing

HTTP code	200
HTTP message	OK

Delete subscription for given subscriber

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

Operation

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

Returns

Nothing

HTTP code	200
HTTP message	OK

Publish Event

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

Operation

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

Response

Nothing

HTTP code	200
HTTP message	OK

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

HTTP code	400
HTTP message	BAD REQUEST

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

HTTP code	503
HTTP message	SERVICE UNAVAILABLE

Chat API

Overview

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

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

Basic Services (Genesys Mobile Services-Based):

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

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

Sequence Diagrams

- Chat Immediate
- Chat Delayed

Structures

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

Chat Interaction API Resources

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

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

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

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

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

Attribute Descriptions:

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

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

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

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

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

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

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

Attributes description:

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

- Type of user: `{"AGENT", "CLIENT", "EXTERNAL"}`

Service API Resources

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

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

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

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

COMETD Based Chat API

CometD handshake request:

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

Response:

```
Content-Type:        application/json;charset=UTF-8
[{"id":"0","minimumVersion":"1.0","supportedConnectionTypes":["callback-polling","long-
polling"],"successful":true,"channel":"/meta/
handshake","ext":{"ack":true},"clientId":"d1jw6puyen98c1baidgmlcivxs","version":"1.0"}]
```

CometD connect request:

```
POST http://localhost:8080/genesys/cometd
Content-Type:        application/json; charset=UTF-8
gms_user:        BuzzBrain
{"channel":"/meta/
connect","clientId":"d1jw6puyen98c1baidgmlcivxs","id":"1","connectionType":"long-polling"}
```

Response:

```
Content-Type:        application/json;charset=UTF-8
[{"id":"1","successful":true,"advice":{"interval":0,"reconnect":"retry","timeout":60000},"channel":"/meta/
connect"}]
```

CometD subscribe:

```
POST http://localhost:8080/genesys/cometd
Content-Type:        application/json; charset=UTF-8
gms_user:        BuzzBrain
[{"channel":"/meta/
subscribe","subscription":"/_genesys","clientId":"eurw5v1ezp5yn17z3cdrf3jxsc","id":"2"}]
```

Response:

```
Content-Type:        application/json;charset=UTF-8
[{"id":"3","successful":true,"channel":"/meta/connect"}]
```

CometD polling:

```
POST http://localhost:8080/genesys/cometd
Content-Type:        application/json; charset=UTF-8
gms_user:        BuzzBrain
{"clientId":"eurw5v1ezp5yn17z3cdrf3jxsc","id":"4","channel":"/meta/
connect","connectionType":"long-polling"}
```

Response:

```
Content-Type:        application/json;charset=UTF-8
[{"id":"4","successful":true,"channel":"/meta/connect"}]
```

... continue cometd polling ...

Create service session:

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

Response:

```
Content-Type:        application/json;charset=UTF-8
{"_chatIxnAPI-CREATE-URL":"/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/
chat","_id":"0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f"}
```

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

```
POST http://localhost:8080/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat
Content-Type:        application/x-www-form-urlencoded; charset=UTF-8
gms_user:        BuzzBrain
_verbose=true¬ify_by=comet&firstName=Buzz&lastName=Brain⊂ject=French&email=b.b%40gmail.com
```

Response:

```
Content-Type:        application/json;charset=UTF-8
{"message":"Messages will sent over CometD service channel \/_genesys"}
```

... continute cometd polling ...:

CometD polling:

```
POST http://localhost:8080/genesys/cometd
Content-Type:        application/json; charset=UTF-8
gms_user:        BuzzBrain
{"clientId":"eurw5v1ezp5yn17z3cdrf3jxsc","id":"32","channel":"/meta/
connect","connectionType":"long-polling"}
```

Response:

```
Content-Type:        application/json;charset=UTF-8
[{"data":{"id":"ccd229401d5f11e200000805691661bf","message":{"chatSessionId":"000BWa7RECRK001D","transcriptPosi
service is
available","startedAt":"2012-10-23T22:19:42Z","chatIxnState":"TRANSCRIPT","transcriptToShow":[]},"tag":"service
connect"}]
```

Polling agent 'Joined' message through CometD:

```
POST http://localhost:8080/genesys/cometd
Content-Type:        application/json; charset=UTF-8
gms_user:        BuzzBrain
{"clientId":"eurw5v1ezp5yn17z3cdrf3jxsc","id":"33","channel":"/meta/
connect","connectionType":"long-polling"}
```

Response:

```
Content-Type:        application/json;charset=UTF-8
[{"data":{"id":"3cc44da01d6011e200000805691661bf","message":{"chatSessionId":"000BWa7RECRK001D","transcriptPosi
service is
available","startedAt":"2012-10-23T22:19:42Z","chatIxnState":"TRANSCRIPT","transcriptToShow":[["Notice.Joined",
joined the
session","21","AGENT"]]},"tag":"service.chat.refresh.0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f"},"channel":"/_genesy
connect"}]]
```

Polling agent 'StartTyping' message through CometD:

```
POST http://localhost:8080/genesys/cometd
Content-Type:        application/json; charset=UTF-8
gms_user:        BuzzBrain
{"clientId":"eurw5v1ezp5yn17z3cdrf3jxsc","id":"34","channel":"/meta/
connect","connectionType":"long-polling"}
```

Response:

```
Content-Type:        application/json;charset=UTF-8
[{"data":{"id":"3cc44da01d6011e200000805691661bf","message":{"chatSessionId":"000BWa7RECRK001D","transcriptPosi
service is
available","startedAt":"2012-10-23T22:19:42Z","chatIxnState":"TRANSCRIPT","transcriptToShow":[["Notice.TypingSt
typing","212","AGENT"]]},"tag":"service.chat.refresh.0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f"},"channel":"/_genesy
connect"}]]
```

Polling agent chat message through CometD:

```
POST http://localhost:8080/genesys/cometd
Content-Type:        application/json; charset=UTF-8
gms_user:        BuzzBrain
{"clientId":"eurw5v1ezp5yn17z3cdrf3jxsc","id":"35","channel":"/meta/
connect","connectionType":"long-polling"}
```

Response:

```
Content-Type:        application/json;charset=UTF-8
[{"data":{"id":"3fc6b8301d6011e200000805691661bf","message":{"chatSessionId":"000BWa7RECRK001D","transcriptPosi
service is
available","startedAt":"2012-10-23T22:19:42Z","chatIxnState":"TRANSCRIPT","transcriptToShow":[["Message.Text","
connect"}]]
```

Send client chat message:

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

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

Response:

```
Content-Type:       application/json;charset=UTF-8
{"message":"Messages will sent over CometD service channel \/_genesys"}
```

Client message is being echoed back through CometD channel as a response to "refresh" or "send" request:

```
POST http://localhost:8080/genesys/cometd
Content-Type:       application/json; charset=UTF-8
gms_user:       BuzzBrain
{"clientId":"eurw5v1ezp5yn17z3cdrf3jxsc","id":"36","channel":"/meta/
connect","connectionType":"long-polling"}
```

Response:

```
Content-Type:       application/json;charset=UTF-8
[{"data":{"id":"472e2a901d6011e200000805691661bf","message":{"chatServerLoadBalancerAlias":"349","clientTimeZon
SimpleSamples812/ChatHA/
ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=349","_chatIxnAPI_REFRESH_FROM_START_URL":"/genesys/
1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
refresh?transcriptPosition=1","_chatIxnAPI_REFRESH_URL":"/genesys/1/service/
0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
refresh","chatSessionId":"000BWa7RECRK001D","isTLSrequired":"false","_chatIxnAPI_DISCONNECT_URL":"/genesys/
1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
disconnect","userId":"015D508717FE0027","chatServiceMessage":"Chat service is
available","_chatIxnAPI_STOP_TYPING_URL":"/genesys/1/service/
0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
stopTyping","_chatIxnAPI_START_TYPING_URL":"/genesys/1/service/
0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
startTyping","transcriptToShow":[["Message.Text","BuzzB","hello
agent","230","CLIENT"]],"_chatIxnAPI_SEND_URL":"/genesys/1/service/
0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
send","chatServerHost":"demosrv.genesyslab.com"},"tag":"service.chat.refresh.0da34a9d-0c5e-46a6-953c-6cc1aa82bf
connect"}]
```

Disconnect chat session:

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

Response:

```
Content-Type:       application/json;charset=UTF-8
{
 "chatIxnState" : "DISCONNECTED",
 "transcriptPosition" : "6",
 "chatServiceMessage" : "Chat was finished",
 "chatSessionId" : "000BWa7RECRK001D",
 "userId" : "015D508717FE0027",
 "secureKey" : "29a6758abacdd33e",
 "checkChatServiceLoadBalancerPath" : "/WebAPI812/SimpleSamples812/ChatHA/
ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=349",
 "chatServerLoadBalancerAlias" : "349",
 "chatServerHost" : "demosrv.genesyslab.com",
 "chatWebApiPort" : "4853",
 "isTLSrequired" : "false",
 "clientTimeZoneOffset" : "-420",
 "_chatIxnAPI_SEND_URL" : "/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/
send",
 "_chatIxnAPI_REFRESH_URL" : "/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/
chat/refresh",
 "_chatIxnAPI_START_TYPING_URL" : "/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/
ixn/chat/startTyping",
 "_chatIxnAPI_STOP_TYPING_URL" : "/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/
chat/stopTyping",
 "_chatIxnAPI_DISCONNECT_URL" : "/genesys/1/service/0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/
chat/disconnect",
 "_chatIxnAPI_REFRESH_FROM_START_URL" : "/genesys/1/service/
0da34a9d-0c5e-46a6-953c-6cc1aa82bf7f/ixn/chat/refresh?transcriptPosition=1"
}
```

Quick Start Examples

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

Using CometD to Receive Event Updates

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

CometD handshake request

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

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

CometD /meta/connect subscription request

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

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

```
 Content-Length: 116
[{"id":"1","successful":true,"advice":{"interval":0,"reconnect":"retry","timeout":60000},"channel":"/meta/
connect"}]
```

CometD /_genesys subscription request

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

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

CometD long polling request

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

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

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

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

Create a Service:

Request:

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

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:23:29 GMT
Content-Type: application/json
{"_id":"EKUJPKAQ197CFA6SJQKTJ03DBG00001M"}
```

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

Request:

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

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:26:26 GMT
Content-Type: application/json
Content-Length: 185
{
   "message":    {
      "_id": "EKUJPKAQ197CFA6SJQKTJ03DBG00001M",
      "_status": "waiting",
      "_dialog": "waiting_for_agent.html"
   },
   "tag": "a2c.advanced.service.statuschanged.EKUJPKAQ197CFA6SJQKTJ03DBG00001M"
}
```

Repeat request until status changes:

Request:

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

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:28:25 GMT
Content-Type: application/json
Content-Length: 186
{
   "message":    {
      "_id": "EKUJPKAQ197CFA6SJQKTJ03DBG00001M",
      "_status": "available",
      "_dialog": "agent_available.html"
   },
   "tag": "a2c.advanced.service.agentavailable.EKUJPKAQ197CFA6SJQKTJ03DBG00001M"
}
```

Create chat interaction using same sessionid:

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

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

Parameter Name	Mandatory	Description
		"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	false	If using a CometD connection for the asynchronous receiving of chat messages, then supply this parameter with the value "comet".

Request:

```
POST http://localhost:8080/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001M/ixn/chat HTTP/
1.1
Accept-Encoding: gzip,deflate
Content-Type: application/x-www-form-urlencoded
notify_by=comet&firstName=Vasya&lastName=Pupkin&email=Vasya.Pupkin@genesyslab.com⊂ject=test
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:30:10 GMT
Content-Type: application/json
Content-Length: 119
{
 "chatIxnState" : "CONNECTED",
 "transcriptPosition" : "1",
 "chatServiceMessage" : "Chat service is available"
}
```

Refresh chat transcript and show messages to the user:

Request:

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

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:33:00 GMT
Content-Type: application/json
Content-Length: 367
{"_id":"B2FS3346K151548QMEAFD89TE8000EBJ","comet_channel":"/_genesys"}
```

Send user's message:

Request:

```
POST http://localhost:8080/genesys/1/service/EKUJPKAQ197CFA6SJQKTJ03DBG00001M/ixn/chat/
refresh HTTP/1.1
```

```
Accept-Encoding: gzip,deflate
Content-Type: application/x-www-form-urlencoded
message=hello agent
```

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 08:34:38 GMT
Content-Type: application/json
Content-Length: 241
{"_id":"B2FS3346K151548QMEAFD89TE8000EBJ","comet_channel":"/_genesys"}
```

Disconnect user from chat:

Request:

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

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 07:43:07 GMT
Content-Type: application/json
Content-Length: 114
{
 "chatIxnState" : "DISCONNECTED",
 "transcriptPosition" : "9",
 "chatServiceMessage" : "Chat was finished"
}
```

Chat Interaction APIs

Start Chat

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

Operation

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

Method	POST

Body Properties: The following are the properties:

- _verbose - This will allow the application to get all the detail attributes associated with the chat session in the corresponding response.
- notify_by - If specified should be "comet".
- firstName - User's first name. Optional.
- lastName - User's last name. Optional.
- email - User's email address. Optional.
- subject - Subject of the chat conversation.
- userDisplayName - Nickname displayed in the chat conversation. Optional.

Response

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

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

Example

Request:

```
POST http://localhost:8080/genesys/1/service/9d6c31d3-1121-4ba9-91e1-b93c0fa6e32f/ixn/
chat?_verbose=true HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/<code>x-www-form-urlencoded</code>
firstName=Vasya&lastName=Pupkin&email=Vasya.Pupkin@genesyslab.com⊂ject=test
```

Response:

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

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

Refresh Chat

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

Operation

Method	POST		
URL	/genesys/1/service/***{service_id}***/**ixn**/***chat***/**refresh**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{service_id}	string	yes	The identifier of the service that the chat session is associated with.
Body: The body will be `x-www-form-urlencoded` form consisting of different items representing the key/value pairs associated with the request.			
Body Properties: The following are the properties: • transcriptPosition – This optional property indicates the current position in the chat session that the current user is in. This property is ignored when notify_by = comet when starting the chat session (ixn/chat) • message – This optional property is a chat message that will be added to the chat session/transcript. • _verbose - This optional property will allow the application to get all the detail attributes associated with the chat session in the corresponding response.			

Response

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

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

Example Request:

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

Response:

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

Start Typing

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

Operation

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

Method	POST
• _verbose - This will allow the application to get all the detail attributes associated with the chat session in the correspondingresponse.	

Response

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

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

Example Request:

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

Response:

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

Stop Typing

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

Operation

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

Response

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

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

Example Request:

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

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 07:38:58 GMT
```

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

Disconnect from chat session

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

Operation

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

Response

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

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

Example Request:

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

Response:

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

Basic Chat Service API

Create basic chat service

This API allows the application to create basic chat service session and then initiate chat interaction immediately or when user is ready. **Note:** If agent availability need to be checked before chat interaction is started - use one of the advanced sessions (for example: request-chat-poll)

Operation

Method	POST		
URL	/genesys/1/service/request-chat		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
'request-chat'	String	yes	Name of the preconfigured basic chat service
Body: The body will be x-www-form-urlencoded form consisting of different items representing the key/value pairs associated with the request.			

Method	POST
Body Properties: The following are the properties: • _verbose - This will allow the application to get all the detail attributes associated with the chat session in the corresponding response. • ... - Any other business data attributes can also be passed.	

Response

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

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

Example Request:

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

Response:

```
HTTP/1.1 200 OK
Date: Sun, 10 Jun 2012 07:49:46 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Transfer-Encoding: chunked
Server: Jetty(7.6.0.v20120127)
{
"_chatIxnAPI-CREATE-URL":"/genesys/1/service/81f0ef4e-99dd-43ea-8366-8d27a2cbd605/ixn/chat",
"_id":"81f0ef4e-99dd-43ea-8366-8d27a2cbd605"
}
```

Service API

Overview

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

API

Create

This API creates and initiates a service. It will support the creation and initiation of an service that is configured in Genesys Mobile Services.

Operation

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

Response

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

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

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

HTTP code	404
HTTP message	Not Found

Example

The following example starts a request-interaction service:

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

Operation

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

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

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

Service Specific Request

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

Important: This is only to be used for services which support such 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.
Body: The body can be either a MultiPart form or x-www-form-urlencoded form consisting of different items representing the key/value pairs associated with the given service request. In the case of MultiPart, the values can be strings or files but with urlencoded the values can be only strings.			

Response

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

Example

See the Chat Interaction APIs for an example.

Notes

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.

Callback Services API

Overview

The URI parameter, {callback-execution-name}, is required for all of the Callback APIs. The value for this parameter equals the name of a Callback service configured in your GMS application or cluster. For example, if you have a section "service.sales_callback" of type Callback configured, the {callback-execution-name} will be equal to "sales_callback".

API

Start-Callback

The Start-Callback API initiates a callback request. It validates the request by doing the following:

- Checks parameters, in general (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.

Description	Start-Callback		
Method	POST		
URL	http://host:port/{base-web-application}/service/callback/{callback-execution-name}		
Name	**Type**	**Mandatory**	**Description**
URI Parameters			
{callback-execution-name}	string	yes	This is the name of the callback execution service of 'ors' type that is provisioned in GMS.
Body (JSON content)			
_customer_number	string	yes	This is the number to call back.
_desired_time	string	no	This is the desired time to have the callback. Format is ISO 8601 "yyyy-MM-ddTHH:mm:ss.SSSZ" for example: "2013-05-28T15:30:00.000Z"
<property>	string	no	Any properties key/values to be attached. Key/Values may be used in Orchestration execution service. Keys without an underscore prefix are User Attached Data.
Response Body (JSON content)			
Name	**Type**	**Mandatory**	**Description**
_id	string	yes	The service id for which a successful callback request was registered.

HTTP code	200
HTTP message	OK

Example Operation

```
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 for this operation is different if immediate/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 service, by doing the following:

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

Description	Cancel-Callback		
Method	DELETE		
URL	http://host:port/{base-web-application}/service/callback/{callback-execution-name}/{service_id}		
Name	**Type**	**Mandatory**	**Description**
URI Parameters			
{service_id}	string	yes	This is the service id returned from the initial start callback response.
{callback-execution-name}	string	yes	This is the name of the callback execution service of 'ors' type that is provisioned in GMS.
Body (JSON content)			
Response Body (JSON content)			
Name	**Type**	**Mandatory**	**Description**

HTTP code	200
HTTP message	OK

Example Operation

```
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"
}
```

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 cancelled. 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.

Description	Reschedule-Callback		
Method	PUT		
URL	http://host:port/{base-web-application}/service/callback/{callback-execution-name}/{service_id}		
Name	**Type**	**Mandatory**	**Description**
URI Parameters			
{service_id}	string	yes	This is the service id returned from initial start callback response.
{callback-execution-name}	string	yes	This is the name of the callback execution service of 'ors' type that is provisioned in GMS.
Body (JSON content)			
_new_desired_time	string	yes	The new time for which to reschedule the callback.
Response Body (JSON content)			
Name	**Type**	**Mandatory**	**Description**
See example.	string	yes	• If accepted, none.

Description	Reschedule-Callback		
			• If not, a list of possible times around the desired_time.

HTTP code	200
HTTP message	OK

Example: Operation Successful

```
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
```

Example: 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
      }
}
```

Query-Callback

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

Important: Currently, only "customer_number" is indexed.

Description	Query-Callback		
Method	GET		
URL	http://host:port/{base-web-application}/service/callback/{callback-execution-name}?{property=value}		
Name	**Type**	**Mandatory**	**Description**
URI Parameters			
{callback-execution-name}	string	yes	This is the name of the callback execution service of 'ors' type that is provisioned in GMS.
{property=value}	string	yes	This is the property name used to query the callback. Currently, the only property to search is by "customer_number".
Body (JSON content)			
Response Body (JSON content)			
Name	**Type**	**Mandatory**	**Description**
See example.	string	yes	• If accepted, a list of service ids of the currently outstanding callback requests. • If not, an error code indicating the reason.

HTTP code	200
HTTP message	OK

Example Operation

```
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",
        "url": "/1/service/callback/BasicCallback/a550a12e-ca77-4146-98d0-58960e0939f7"
    },
    {
        "_id": "4a1ea889-1ef7-432d-a543-cff96b4a2daf",
        "desired_time": "2013-05-27T15:10:00.000Z",
        "url": "/1/service/callback/BasicCallback/4a1ea889-1ef7-432d-a543-cff96b4a2daf"
    }
]
```

Query-Availability

By specifying a desired time, the client can check if an office is open, and if open, if there is available time slot.

Options:

- business_hours: to match office open/close slots
- request-time-bucket: the bucket size returned
- max-request-by-time-bucket

Description	Query-Availability		
Method	GET		
URL	http://host:port/{base-web-application}/service/callback/{callback-execution-name}/availability?{timestamp=value}		
Name	**Type**	**Mandatory**	**Description**
URI Parameters			
{callback-execution-name}	string	yes	This is the name of the callback execution service of 'ors' type that is provisioned in GMS.
{timestamp=value}	string	no	This is the desired time to have the callback. Format is ISO 8601 "yyyy-MM-ddTHH:mm:ss.SSSZ" For example: "2013-05-28T15:30:00.000Z"
Body (JSON content)			
Response Body (JSON content)			
Name	**Type**	**Mandatory**	**Description**
See example.	string	yes	• If accepted, a list of available slots with remaining possible requests. • If not, an error code indicating the reason.

HTTP code	200
HTTP message	OK

Example Operations

```
GET http://localhost:8080/genesys/1/service/callback/BasicCallback/
availability?timestamp=2013-06-15T09:00:00.000Z
Results success:
   200 OK
   {
      "2013-06-15T08:00:00.000Z": 5,
      "2013-06-15T09:45:00.000Z": 5,
      "2013-06-15T09:00:00.000Z": 4,
      "2013-06-15T08:30:00.000Z": 5,
      "2013-06-15T08:45:00.000Z": 5,
      "2013-06-15T09:15:00.000Z": 4,
      "2013-06-15T08:15:00.000Z": 5,
      "2013-06-15T09:30:00.000Z": 5
   }

GET http://localhost:8080/genesys/1/service/callback/BasicCallback/
availability?timestamp=2013-06-15T12:00:00.000Z
Results error:
   400 BAD REQUEST
   {
      "message": "Office is closed at 2013-06-15T12:00:00.000Z. No time slots proposed.",
      "exception": "com.genesyslab.gsg.services.callback.CallbackExceptionAvailability",
      "availability":
      {
      }
   }
```

ADMIN - Query-Callback

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

Description	Query-Callback		
Method	GET		
URL	http://host:port/{base-web-application}/admin/callback/queues?target={target_name}&end_time={iso_end_time}		
Name	**Type**	**Mandatory**	**Description**
URI Parameters			
{iso_end_time}	string	no	This is the maximum time for which to query callback requests. If not specified, requests that are due in next 24 hours are returned. The format is ISO 8601 "yyyy-MM-ddTHH:mm:ss.SSSZ".

Description	Query-Callback		
			For example: "2013-05-28T15:30:00.000Z"
{target}	string	no	This is the name of a callback execution service. If not specified, the queues for all services are returned. For example: BasicCallback.
{max}	integer	no	This is maximum number of requests to return for each queue. If not specified, 500 maximum requests per queue are returned.
Body (JSON content)			
Response Body (JSON content)			
Name	**Type**	**Mandatory**	**Description**
See example.	string	yes	If accepted, a tree list of target queues and requests.

HTTP code	200
HTTP message	OK

Example Operation

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

Result

```
200 OK

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

Sample Errors

Number Exceptions

```
   {
      "message": "Customer Number [12345] is not allowed. Check option 'exceptions' :
details={Callback_exceptions=exception2}",
      "exception": "com.genesyslab.gsg.services.callback.CallbackServiceException"
   }
```

Business Hours Exceptions

```
   {
      "message": "{"status":"closed","reason":"closed_day_of_week"}",
      "exception": "com.genesyslab.gsg.services.callback.CallbackServiceException"
   }

   {
      "message": "{"status":"closed","reason":"closed_no_match"}",
      "exception": "com.genesyslab.gsg.services.callback.CallbackServiceException"
   }
```

Queue Full Exceptions

```
   {
      "message": "Too many requests around this desired time : 2013-05-24T18:03:29.952Z",
      "exception": "com.genesyslab.gsg.services.callback.CallbackServiceException"
   }

   {
      "message": "Too many requests at desired time [2013-05-28T15:30:00.000Z,
2013-05-28T15:45:00.000Z]. Proposing time slots.",
      "exception": "com.genesyslab.gsg.services.callback.CallbackAvailabilityException",
      "availability":
      {
          "2013-05-28T16:00:00.000Z": 4,
          "2013-05-28T16:45:00.000Z": 5,
          "2013-05-28T15:00:00.000Z": 5,
          "2013-05-28T15:45:00.000Z": 5,
          "2013-05-28T15:15:00.000Z": 5,
          "2013-05-28T16:30:00.000Z": 5,
          "2013-05-28T16:15:00.000Z": 5
```

```
        }
    }
```

Push Notification Service

Overview

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

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

In addition to discussing these different types of notification, this page also describes 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":"*"}
```

Android Notification

GCM Service

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

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

Keystore/Truststore Configuration Hints

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

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

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

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

C2DM Service

Note: Google has deprecated the C2DM Service, and no new users are being accepted. Please use the GCM Service, described above.

Android notification relies on the Android Cloud to Device Messaging (C2DM) service, described here: http://code.google.com/android/c2dm/. C2DM notifications are made on behalf of an account that is registered in Google services and described by the configuration options for the Genesys Mobile

Services Application object. Some key points about C2DM to take into consideration when creating your applications:

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

Keystore/Truststore Configuration Hints

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

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

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

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

Client Application Implementation

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

- When the application starts, it must register itself in the C2DM service by specifying the Google Services account that it will receive notifications from. The account name must be configurable because it will be unique for each customer.
- The push service uses *data.message=<message_body>* in the service-to-C2DM POST request body. When the Android client application receives a notification, it should use "message" as the key to extract the passed information. Sample code for message extraction is provided below:

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

Client Application Implementation

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

Apple Notification

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

Client Application Implementation

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

CometD Notification

Note: Available in 8.1.100.28.

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

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

CometD handshake request

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

HTTP/1.1 200 OK
```

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

CometD /meta/connect subscription request

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

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

CometD /_genesys subscription request

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

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

CometD long polling request

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

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

Localization of push messages

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

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

Localization file format is described here.

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

See more details on configuring the push section.

Orchestration Server Callback Notification

Subscription

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

Notification Propagation

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

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

Configuration

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

Providers

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

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

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

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

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

Support of OS specific capabilities associated with the notification message

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

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

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

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

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

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

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

Localization File

Overview

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

Localization file

Format

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

Arguments

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

Simple style	Expanded style

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

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.