Genesys Mobile Services API Reference

Genesys Mobile Engagement 8.5.0

1/25/2022

Table of Contents

Genesys Mobile Services API Reference **3**
- New in This Document 4
- Storage API 5
  - Storage API HTML Sample 12
- Pattern Matcher API 14
- Node API 19
- Notification API 20
- Chat API 27
- Service API 53
- Callback Services API 63
- Stat Server API 83
- Push Notification Service 92
- Callback Push Notifications for Android 101
- Callback Push Notifications for iOS 107
- Localization File 113

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.
- Stat Service API — This API is used to interact with the Genesys Statistics Server (Stat Server) or Universal Routing Server (URS). The API provides the request so an application can get statistics related to the Contact Center.

Additional Information:

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

New in This Document

The following topics have been added or changed in the GMS 8.5.006.09 release:

- The Notification API page changed, to reflect a new authorization header.
- The Callback Services API page changed, to reflect an update to the `_customer_number` parameter, and to reflect an update to the Query-Availability API.
- The Start Chat API was updated with the `subsciptionID` parameter.

The following topics have been added or changed in the GMS 8.5.005.08 release:

- The Callback Services API page changed, to reflect updates in the Start Callback API and the Reschedule Callback API.

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 (_).

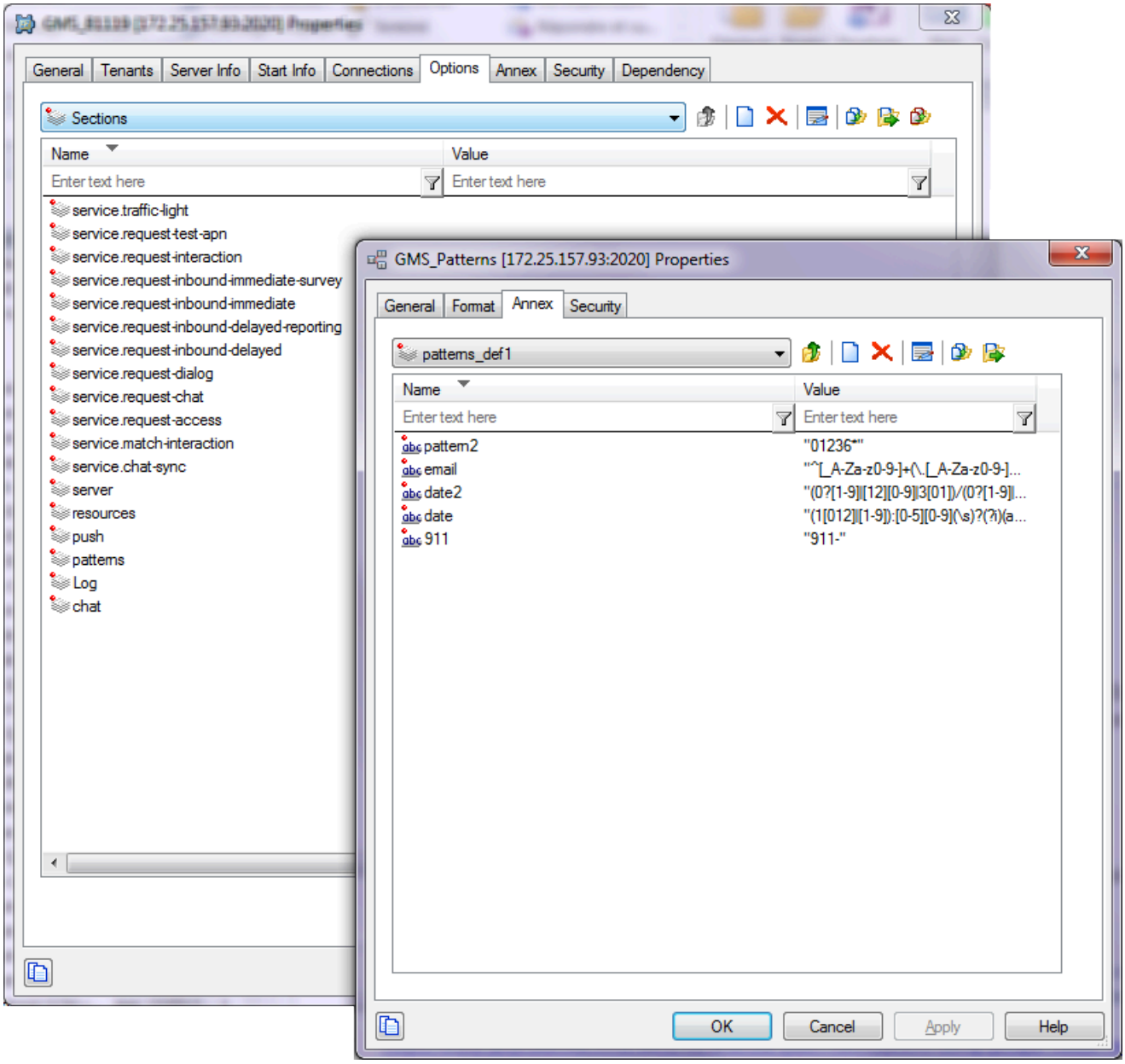
Storage API HTML Sample

```
<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>
```

Pattern Matcher API

Getting Started



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

First, configure a list of patterns or groups of list of patterns in your GMS Configuration object
service, in the GMS Service Management UI.

For example, the left-side screenshot shows the GMS configuration of the `patterns_def1` gro

Once you have some patterns defined, use the Pattern Matcher API queries to check the valid
your parameters.

Pattern Format

The exception patterns are regular expressions as defined in the Java Pattern Class.

List of API Queries

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

Verify Parameters Against General Patterns List

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

Operation

POST genesys/1/patterns
Body: The body can be either a MultiPart form or `x-www-form-urlencoded` form that consists of key/value pairs representing the strings to test. For example: `param1=<string to check>¶m2=<string to check>&...¶m-n=<string to check>`

Response

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

Example

The following example verifies `param1`, `param2`, and `param3`.

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

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

```
HTTP/1.1 200 OK
Date: Thu, 14 Mar 2013 16:13:06 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json;charset=UTF-8
Content-Length: 44

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

Verify Parameters Against a Specific Group in the Patterns List

Use this query to submit a list of parameters to verify against the patterns defined in a specific group

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

Operation

POST genesys/1/patterns/group/{groupName}			
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{groupName}	String	Yes	The group to which the patterns belong.
Body: The body can be either a MultiPart form or `x-www-form-urlencoded` form consisting of key/value pairs, representing the strings to test. For example: `param1=<string to check>¶m2=<string to check>&...¶m-n=<string to check>`			

Response

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

Example

The following example verifies if the `param1` and `param2` match the group of patterns "patterns_def1".

Operation

```
POST http://127.0.0.1:8080/genesys/1/patterns/group/patterns_def1
 HTTP/1.1
Host: 127.0.0.1:8080
Connection: keep-alive
Content-Length: 41
Cache-Control: no-cache
Origin: chrome-extension://fdmmgilgnpjigdojojpjoooidkmcomcm
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64)
 AppleWebKit/537.22 (KHTML, like Gecko)
Chrome/25.0.1364.152 Safari/537.22
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8,fr;q=0.6,fr-FR;q=0.4
```

```
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
param1=john.doe%40gmail.com¶m2=blabla
```

Result

```
HTTP/1.1 200 OK
Date: Thu, 14 Mar 2013 16:22:30 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json;charset=UTF-8
Content-Length: 30
Server: Jetty(8.1.8.v20121106)
{"param1":"email"}
```

The following response indicates that `param1` matches the `email` pattern of the **patterns_def1** group defined in Configuration Manager.

Sample Errors

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

```
HTTP/1.1 403 Forbidden
Date: Thu, 14 Mar 2013 16:23:35 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Length: 107
Server: Jetty(8.1.8.v20121106)
{"message":"Group (patterns_def1s) unknown.",
"exception":"com.genesyslab.gsg.services.pattern.MatchPattern"}
```

```
HTTP/1.1 403 Forbidden
 Date: Thu, 14 Mar 2013 16:24:32 GMT
 Pragma: no-cache
 Cache-Control: no-cache
 Cache-Control: no-store
 Content-Length: 142
 Server: Jetty(8.1.8.v20121106)
{"message":"Service (match-interaction) not configured
 to support exceptions.",
"exception":"com.genesyslab.gsg.services.pattern.MatchPattern"}
```

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: Do not use the Publish part of this API from a mobile device. The API is designed and intended for use only from Orchestration Server-based Services. In release 8.1.100.28, comet was added as a notification subscription for device_os parameters.

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

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

Structures

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

Subscription

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

Subscription Request

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

Here:

- **subscriberId** – The id of subscriber (mandatory).
- **authorization**- (Optional) If basic authorization is needed on the custom HTTP channel. The value of the **authorization** parameter will be added to the HTTP Headers request sent to the custom http channel.
- **expire** - This parameter defines the time, in seconds, after which the subscription expires (optional; default value is configurable).
- **filter** - (Mandatory) The filter which is applied to the tags of incoming events. If filter matches the tag - the event will be published to destination, specified by subscription. Note: event is published to ALL subscription which specify the matching filter. The format of filter see further.
- **providerName** – This is the name of the provider which this subscription is for (optional). If not specified, the subscription is for default provider.
- **language** - (Optional) Describes the language used by this subscription. If not present, GMS will treat localizedstring as a normal message. See Genesys Mobile Services Push Notification Service for details on language.
- **notificationDetails** - (mandatory) Describes all the information needed for delivering the event to concrete subscriber
  - **type** - (Mandatory) this parameter defines what type of notification mechanism that the application wants to use. Valid values are **ios** (to-apple), **android** (to-android-device), gcm (to-android-gcm-device), **comet** (to-cometd-client), **httpcb** (callback POST to provided url) and **orscb** (callback to ORS) (see more information here Push Notification Service).
  - **deviceId** - (Mandatory) id of device to deliver message to (in the case of http or ORS callback - see the details here Push Notification Service.
  - **properties** – (Optional) The String-String map of properties – additional properties 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) Type of the notification (gcm, ios...).
- properties - (optional).
- debug - (optional) Allows the display of the debug log for this notification.

Filters and Tags

The tag cannot be null or an empty string. The format of tag, specified in event, is like the java package name alphanumeric string with '.' delimiters. Underscores are allowed and first symbol may be number. Please note: at the moment only English alphanumeric chars are allowed. The filter can not be null or empty string. The format of filter entry is similar to the tag format, but in addition allowed 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

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

HTTP code	404
HTTP message	Not Found
Body	`{"message":"Subscription ID not found", "exception":"com.genesyslab.gsg.services.notification.Su`

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**

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

Returns

Nothing

HTTP code	200
HTTP message	OK

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

HTTP code	404
HTTP message	Not Found
Body	`{"message":"Subscriber ID not found", "exception":"com.genesyslab.gsg.services.notification.Su`

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)			

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

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

{
  "message": "Agent availability",
  "tag": "gms.notification.agentstatus",
  "mediaType": "STRING",
  "notificationDetails": {
    "deviceId": "XXXXXXXXXXXXXXXXXXXXXX",
    "type": "ios",
    "properties": {
      "apple.alertMessage.body": "Agent is available.",
```

```
      "apple.badge": 9,
      "apple.sound": "bingbong.aiff",

    }
  }
}
```

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

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

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/handshake
gms_user: b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673
Content-Type: application/json;charset=UTF-8
[{"version":"1.0","minimumVersion":"0.9","channel":"/meta/handshake",
"supportedConnectionType"":["long-polling","callback-polling"],
"advice":{"timeout":60000,"interval":0},"id":"1"}]
```

Response

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

CometD Subscribe

Request

```
POST http://localhost:8080/genesys/cometd/
gms_user: b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673
Content-Type: application/json;charset=UTF-8
[{"channel":"/meta/subscribe","subscription":"/_genesys","id":"2",
"clientId":"3vym301sjdtc218qabm5w0z8yb"}]
```

Response

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

CometD Connect

Request

```
POST http://localhost:8080/genesys/cometd/connect
gms_user: b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673
Content-Type: application/json;charset=UTF-8
[{"channel":"/meta/connect","connectionType":"long-polling","advice":{"timeout":0},
"id":"3","clientId":"3vym301sjdtc218qabm5w0z8yb"}]
```

Response

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

Create Service Session

Request

```
POST http://localhost:8080/genesys/1/service/request-chat
gms_user: b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673
Content-Type: application/x-www-form-urlencoded; charset=UTF-8

_verbose=true
```

Response

```
Content-Type: application/json;charset=UTF-8
{ "_chatIxnAPI-CREATE-URL":"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/chat",
  "_id":"4d1697a9-dda5-4742-8a6f-fbc01c25c640",
  "_data_id":"429-791eaee8-571e-4633-9a73-cc936336f8e2"
}
```

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

Request

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

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

Response

```
Content-Type: application/json;charset=UTF-8
{
"chatServerLoadBalancerAlias":"371",
"_chatIxnAPI_SEND_CUSTOM_URL":"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/
chat/customNotice",
"clientTimeZoneOffset":"120",
"transcriptPosition":"1",
"chatWebApiPort":"9002",
"chatIxnState":"CONNECTED",
"comet_channel":"/_genesys",
"secureKey":"1b21478a91a7d1dc",
"checkChatServiceLoadBalancerPath":"/WebAPI812/SimpleSamples812/ChatHA/
```

```
ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=371",
"_chatIxnAPI_REFRESH_FROM_START_URL":"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/
ixn/chat/refresh?transcriptPosition=1",
"_chatIxnAPI_REFRESH_URL":"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/chat/
refresh",
"chatSessionId":"000E5aA2A40P000Q",
"isTLSrequired":"false",
"_chatIxnAPI_DISCONNECT_URL":"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/
chat/disconnect",
"chatServiceMessage":"Chat service is available",
"userId":"0173542518870006",
"_chatIxnAPI_STOP_TYPING_URL":"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/
chat/stopTyping",
"_chatIxnAPI_START_TYPING_URL":"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/
chat/startTyping",
"_chatIxnAPI_SEND_URL":"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/chat/
send",
"chatServerHost":"demosrv.genesyslab.com"
}
```


Polling Agent 'Joined' Message Through CometD

Request

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

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

Response


```
Content-Type: application/json;charset=UTF-8
[
{"data":{"id":"7b239ff0455011e4b31cbb293fafe316",
"message":{"chatSessionId":"000E5aA2A40P000Q",
"transcriptPosition":"2","chatServiceMessage":"Chat service is available",
"startedAt":"2014-09-26T07:40:55Z",
"chatIxnState":"TRANSCRIPT",
"transcriptToShow":[ ["Notice.Joined","Kristi Sippola","has joined the
session","14","AGENT"]]},
"tag":"service.chat.refresh.4d1697a9-dda5-4742-8a6f-fbc01c25c640"},
"channel":"/_genesys"},
{"id":"4","successful":true,"channel":"/meta/connect"}
]
```


Polling Agent 'StartTyping' Message Through CometD

Request

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

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

Response

```
Content-Type: application/json;charset=UTF-8
[
{"data":{"id":"802c88e0455011e4b31cbb293fafe316",
"message":{"chatSessionId":"000E5aA2A40P000Q",
"transcriptPosition":"3","chatServiceMessage":"Chat service is available",
"startedAt":"2014-09-26T07:40:55Z",
"chatIxnState":"TRANSCRIPT",
"transcriptToShow":[["Notice.TypingStarted","Kristi Sippola","is
typing","22","AGENT"]]},
"tag":"service.chat.refresh.4d1697a9-dda5-4742-8a6f-fbc01c25c640"},
"channel":"/_genesys"},{"id":"5","successful":true,"channel":"/meta/connect"}
]
```

Polling Agent Chat Message Through CometD

Request

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

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

Response

```
Content-Type: application/json;charset=UTF-8
[
{"data":{"id":"816f9030455011e4b31cbb293fafe316",
  "message":{"chatSessionId":"000E5aA2A40P000Q",
  "transcriptPosition":"4",
  "chatServiceMessage":"Chat service is available",
  "startedAt":"2014-09-26T07:40:55Z",
  "chatIxnState":"TRANSCRIPT",
  "transcriptToShow":[["Message.Text","Kristi
Sippola","Hello","23","AGENT"]]},
  "tag":"service.chat.refresh.4d1697a9-dda5-4742-8a6f-fbc01c25c640"},
  "channel":"/_genesys"},
  {"id":"6","successful":true,"channel":"/meta/connect"}
  ]
```

Send Client Chat Message

Request

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

_verbose=true&message=Hi%20Verbose
```

Response

```
Content-Type: application/json;charset=UTF-8
{
"chatServerLoadBalancerAlias":"371",
"_chatIxnAPI_SEND_CUSTOM_URL":"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/
chat/customNotice",
"clientTimeZoneOffset":"120",
```

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


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

Request

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

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

Response


```
Content-Type: application/json;charset=UTF-8
[
{"data":{"id":"867b3840455011e4b31cbb293fafe316",
          "message":{"chatSessionId":"000E5aA2A40P000Q",
                 "transcriptPosition":"5",
                 "chatServiceMessage":"Chat service is available",
                 "startedAt":"2014-09-26T07:40:55Z",
                 "chatIxnState":"TRANSCRIPT",
                 "transcriptToShow":[["Message.Text","127.0.0.1","Hi
Verbose","32","CLIENT"]]},
  "tag":"service.chat.refresh.4d1697a9-dda5-4742-8a6f-fbc01c25c640"},
  "channel":"/_genesys"},
  {"id":"7","successful":true,"channel":"/meta/connect"}]
```


CometD Polling

Request

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

```
gms_user: b16416334828b1d26ef14f329628b55b5a8c631d8928a371a5584722dd7fb673
Content-Type: application/json;charset=UTF-8

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

Response

```
Content-Type: application/json;charset=UTF-8
[{"id":"8","successful":true,"advice":{"reconnect":"none"},"channel":"/meta/connect"}]
```

Disconnect Chat Session

Request

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

_verbose=true
```

Response

```
Content-Type: application/json;charset=UTF-8
{
"chatIxnState" : "DISCONNECTED",
"transcriptPosition" : "5",
"chatServiceMessage" : "Chat was finished",
"chatSessionId" : "000E5aA2A40P000Q",
"userId" : "0173542518870006",
"secureKey" : "1b21478a91a7d1dc",
"checkChatServiceLoadBalancerPath" :
"/WebAPI812/SimpleSamples812/ChatHA/ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=371",
"chatServerLoadBalancerAlias" : "371",
"chatServerHost" : "demosrv.genesyslab.com",
"chatWebApiPort" : "9002",
"isTLSrequired" : "false",
"clientTimeZoneOffset" : "120",
"_chatIxnAPI_SEND_URL" :
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/chat/send",
"_chatIxnAPI_REFRESH_URL" :
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/chat/refresh",
"_chatIxnAPI_START_TYPING_URL" :
"/genesys/1/service/4d1697a9-dda5-4742-8a6f-fbc01c25c640/ixn/chat/startTyping",

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

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

CometD Unsubscribe

Request

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

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

Response

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

CometD Disconnect

Request

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

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

Response

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

Quick Start Examples

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

Using CometD to Receive Event Updates

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

CometD handshake request

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

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

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

CometD /meta/connect subscription request

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

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

CometD /_genesys subscription request

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

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

CometD long polling request

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

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

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

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

Create a Service:

Request:

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

Response:

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

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

Request:

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

Response:

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

Repeat request until status changes:

Request:

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

Response:

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

Create chat interaction using same sessionid:

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

Parameter Name	Mandatory	Description
firstName	no	First name of the user. If provided will be attached as "fldnFirstName" to the chat interaction.
lastName	no	Last name of the user. If provided will be attached as "fldnLastName" to the chat interaction.
email	no	e-Mail address of the subject. If provided will be attached as "fldnEmailAddress" to the chat interaction.
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.			
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. • subscriptionID – ID of the subscription created to receive specific events on Comet channel disconnection. • 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 (if transcriptPosition input parameter is null):

```
HTTP/1.1 200 OK
Date: Sat, 09 Jun 2012 22:26:16 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Content-Length: 1225
Server: Jetty(7.6.0.v20120127)
{
    "chatIxnState": "TRANSCRIPT",
    "chatSessionId":"000BRa84KRFB00BK",
    "transcriptPosition": "5",
    "chatServiceMessage": "Chat service is available",
    "startedAt": "2012-06-09T22:26:17Z",
    "userId": "015E4FD3CD890036",
    "secureKey": "b306749dabfa1cf6",
    "checkChatServiceLoadBalancerPath":
"/WebAPI812/SimpleSamples812/ChatHA/ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=350",
    "chatServerLoadBalancerAlias": "350",
    "chatServerHost": "localhost",
    "chatWebApiPort": "4856",
    "isTLSrequired": "false",
    "clientTimeZoneOffset": "-420",
    "_chatIxnAPI_SEND_URL": "/genesys/1/service/{service_id}/ixn/chat/send",
    "_chatIxnAPI_REFRESH_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh",
    "_chatIxnAPI_START_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/startTyping",
    "_chatIxnAPI_STOP_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/stopTyping",
    "_chatIxnAPI_DISCONNECT_URL":
"/genesys/1/service/{service_id}/ixn/chat/disconnect",
    "_chatIxnAPI_REFRESH_FROM_START_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh?transcriptPosition=1"
}
```

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

```
HTTP/1.1 200 OK
Date: Sat, 09 Jun 2012 22:26:16 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Content-Length: 1225
Server: Jetty(7.6.0.v20120127)
{
    "chatIxnState": "TRANSCRIPT",
    "chatSessionId":"000BRa84KRFB00BK",
    "transcriptPosition": "5",
    "chatServiceMessage": "Chat service is available",
    "transcriptToShow":    [
             [
          "Notice.Joined",
```

```
          "ksippo",
          "has joined the session",
          "15",
          "AGENT"
       ],
             [
          "Message.Text",
          "VasyaP",
          "hello agent",
          "26",
          "CLIENT"
       ],
             [
          "Notice.TypingStarted",
          "ksippo",
          "is typing",
          "57",
          "AGENT"
       ],
             [
          "Message.Text",
          "ksippo",
          "hello customer",
          "61",
          "AGENT"
       ]
    ],
    "startedAt": "2012-06-09T22:26:17Z",
    "userId": "015E4FD3CD890036",
    "secureKey": "b306749dabfa1cf6",
    "checkChatServiceLoadBalancerPath":
"/WebAPI812/SimpleSamples812/ChatHA/ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=350",
    "chatServerLoadBalancerAlias": "350",
    "chatServerHost": "localhost",
    "chatWebApiPort": "4856",
    "isTLSrequired": "false",
    "clientTimeZoneOffset": "-420",
    "_chatIxnAPI_SEND_URL": "/genesys/1/service/{service_id}/ixn/chat/send",
    "_chatIxnAPI_REFRESH_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh",
    "_chatIxnAPI_START_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/startTyping",
    "_chatIxnAPI_STOP_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/stopTyping",
    "_chatIxnAPI_DISCONNECT_URL":
"/genesys/1/service/{service_id}/ixn/chat/disconnect",
    "_chatIxnAPI_REFRESH_FROM_START_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh?transcriptPosition=1"
}
```


Refresh Chat

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

Operation

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 (if transcriptPosition input parameter is null):

```
HTTP/1.1 200 OK
Date: Sat, 09 Jun 2012 22:26:16 GMT
Pragma: no-cache
```

```
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Content-Length: 1225
Server: Jetty(7.6.0.v20120127)
{
    "chatIxnState": "TRANSCRIPT",
    "chatSessionId":"000BRa84KRFB00BK",
    "transcriptPosition": "5",
    "chatServiceMessage": "Chat service is available",
    "startedAt": "2012-06-09T22:26:17Z",
    "userId": "015E4FD3CD890036",
    "secureKey": "b306749dabfa1cf6",
    "checkChatServiceLoadBalancerPath":
"/WebAPI812/SimpleSamples812/ChatHA/ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=350",
    "chatServerLoadBalancerAlias": "350",
    "chatServerHost": "localhost",
    "chatWebApiPort": "4856",
    "isTLSrequired": "false",
    "clientTimeZoneOffset": "-420",
    "_chatIxnAPI_SEND_URL": "/genesys/1/service/{service_id}/ixn/chat/send",
    "_chatIxnAPI_REFRESH_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh",
    "_chatIxnAPI_START_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/startTyping",
    "_chatIxnAPI_STOP_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/stopTyping",
    "_chatIxnAPI_DISCONNECT_URL":
"/genesys/1/service/{service_id}/ixn/chat/disconnect",
    "_chatIxnAPI_REFRESH_FROM_START_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh?transcriptPosition=1"
}
```

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

```
HTTP/1.1 200 OK
Date: Sat, 09 Jun 2012 22:26:16 GMT
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Content-Type: application/json
Content-Length: 1225
Server: Jetty(7.6.0.v20120127)
{
    "chatIxnState": "TRANSCRIPT",
    "chatSessionId":"000BRa84KRFB00BK",
    "transcriptPosition": "5",
    "chatServiceMessage": "Chat service is available",
    "transcriptToShow":    [
             [
          "Notice.Joined",
          "ksippo",
          "has joined the session",
          "15",
          "AGENT"
       ],
             [
          "Message.Text",
          "VasyaP",
          "hello agent",
          "26",
          "CLIENT"
```

```
        ],
              [
           "Notice.TypingStarted",
           "ksippo",
           "is typing",
           "57",
           "AGENT"
        ],
              [
           "Message.Text",
           "ksippo",
           "hello customer",
           "61",
           "AGENT"
        ]
    ],
    "startedAt": "2012-06-09T22:26:17Z",
    "userId": "015E4FD3CD890036",
    "secureKey": "b306749dabfa1cf6",
    "checkChatServiceLoadBalancerPath":
"/WebAPI812/SimpleSamples812/ChatHA/ChatLBServerInfo.jsp?chatServerLoadBalancerAlias=350",
    "chatServerLoadBalancerAlias": "350",
    "chatServerHost": "localhost",
    "chatWebApiPort": "4856",
    "isTLSrequired": "false",
    "clientTimeZoneOffset": "-420",
    "_chatIxnAPI_SEND_URL": "/genesys/1/service/{service_id}/ixn/chat/send",
    "_chatIxnAPI_REFRESH_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh",
    "_chatIxnAPI_START_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/startTyping",
    "_chatIxnAPI_STOP_TYPING_URL":
"/genesys/1/service/{service_id}/ixn/chat/stopTyping",
    "_chatIxnAPI_DISCONNECT_URL":
"/genesys/1/service/{service_id}/ixn/chat/disconnect",
    "_chatIxnAPI_REFRESH_FROM_START_URL":
"/genesys/1/service/{service_id}/ixn/chat/refresh?transcriptPosition=1"
}
```


Start Typing

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

Operation

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.			

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 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 a 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.

Query (All Keys)

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

Note: Introduced in 8.5.000.12.

Operation

Method	GET
URL	/genesys/1/service/**{id}**/storage

Method	GET		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			
{id}	string	yes	The id of the service.
Body: None			

Response

HTTP code	200
HTTP message	OK

Example

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

Operation

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

Request Method:GET
```

Result

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

Query (One Key)

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

Note: Introduced in 8.5.000.12.

Operation

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

Response

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

key value itself).

HTTP code	200
HTTP message	OK

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

HTTP code	404
HTTP message	Not Found

Example

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

Operation

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

Request Method:GET
```

Result

```
Value1
```

Create/Update (Create Storage on Service)

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

Note: Introduced in 8.5.000.12.

Operation

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

Response

HTTP code	200
HTTP message	OK

Example

The following example stores:

Key1, Key2, Key3, and FileKey

Operation

```
Request URL:http://localhost:8080/genesys/1/service/efef8eb61-1f24-593d-90da-0034aca34b55/
storage
Request Method:POST
Status Code:200 OK
Request Headersview source
Accept:*/*
Accept-Charset:ISO-8859-1,utf-8;q=0.7,*;q=0.3
Accept-Encoding:gzip,deflate,sdch
Accept-Language:en-US,en;q=0.8
Connection:keep-alive
Content-Length:13028
Content-Type:multipart/form-data; boundary=----WebKitFormBoundaryy16qocbN6tmPORZL
Host:localhost:8080
Origin:http://localhost:8080
User-Agent:Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.7 (KHTML, like Gecko) Chrome/
16.0.912.77 Safari/535.7
Request Payload
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="Key1"
Value1
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="Key2"
Value2
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="Key3"
Value3
------WebKitFormBoundaryy16qocbN6tmPORZL
Content-Disposition: form-data; name="FileKey"; filename="MyPic.png"
Content-Type: image/png
------WebKitFormBoundaryy16qocbN6tmPORZL--
```

Result

The above data is now stored.

```
HTTP 200 OK
```

Query Binary (One Binary Key)

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

Note: Introduced in 8.5.002.02

Operation

Method	GET		
URL	/genesys/1/service/**{id}**/storage/binary/**{key}**		
Parameter	**Type**	**Mandatory**	**Description**

Method	GET		
URI Parameters			
{id}	string	yes	The id of the service.
{key}	string	yes	The key of the specifically requested value.
Body: None.			

Response

HTTP code	200
HTTP message	OK

HTTP code	404
HTTP message	Not Found, if the binary key does not exists in user storage.

Example

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

Operation

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

Request Method:GET
```

Result

Binary stream content and its content type.

Delete One Key (Delete One Key Storage)

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

Note: Introduced in 8.5.002.02

Operation

Method	DELETE		
URL	/genesys/1/service/**{id}**/storage/**{key}**		
Parameter	**Type**	**Mandatory**	**Description**
URI Parameters			

Method	DELETE		
{id}	string	yes	The id of the service.
{key}	string	yes	The key to be deleted.
Body: None.			

Response

HTTP code	200
HTTP message	OK

Example

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

Operation

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

Request Method: DELETE
```

Result

OK

Delete All Keys (Delete All Keys Storage)

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

Note: Introduced in 8.5.002.02

Operation

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

Response

HTTP code	200
HTTP message	OK

Example

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

Operation

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

Request Method: DELETE
```

Result

OK

Delete (Delete Service)

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

Operation

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

Response

HTTP code	200
HTTP message	OK

Example

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

Operation

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

Request Method: DELETE
```

Result

OK

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

Getting Started

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

Accessing your Callback Service

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

```
http://<host>:<port>/{base-web-application}/service/callback/{callback-execution-name}
```

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

```
http://<host>:<port>/{base-web-application}/service/callback/callback-for-mobile
```

Overwriting Configuration in Queries

You can create variables in your configuration parameters, then overwrite the current configuration by setting these variables in your queries.

> **Tip**
> You should use this feature to avoid duplicating configuration for multiple services that handle the same functionality but use different queues.

To create a variable, all you need to do is to specify a string matching the following format in your service configuration: `$my_variable$`.

Then, you can use the parameter `my_variable=MYVALUE` in your REST queries; any occurrence of `$my_variable$` in this service will be replaced with this value.

Name	Value
_customer_number	Not Specified
_service	callback
_type	ors
_vq_name	vq_name_token

For instance, you can create the `_vq_name` parameter in the `callback-for-mobile` service a its value to the `$vq_name_token$` variable or use `$vq_name_token$` in your service configura

Then, if you wish to create a callback request for the `callback-for-mobile` service using th `MYVQNAMEVALUE` queue, you should use the following query:

```
POST /genesys/1/service/callback/callback-for-mobile
HTTP/1.1
Host: 127.0.0.1:8080
Cache-Control: no-cache
Content-Type: application/x-www-form-urlencoded
_customer_number=01822256&vq_name_token=MYVQNAMEVALUE
```

When GMS receives the query information, it uses `vq_name_token = MYVQNAMEVALUE` instea default configuration set for `callback-for-mobile`.

List of API Queries

The Callback Services API provides the following REST queries:

- Start-Callback–Initiate a Callback request.
- Cancel-Callback–Cancel a Callback request.

- Query-Availability—Get the availability for a new callback request.
- Query-Callback—Query outstanding callbacks by properties.
- ADMIN - Query-Callback—Query outstanding callbacks by queue(s).

Start-Callback

Start-Callback 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	Name of the callback execution service provisioned in GMS.
Body (JSON content)			
_customer_number	string	yes	Number to call back. This parameter can also be

Description	Start-Callback		
			replaced by any parameter specified in the option `_mandatory_customer_lookup_keys` (comma separated list of attributes) that can identify a unique customer.
_desired_time	string	no	Desired time to have the callback. Format is ISO 8601 "yyyy-MM-ddTHH:mm:ss.SSSZ" For example: "2013-05-28T15:30:00.000Z" Default is current time. Note that the Callback is an *immediate* Callback based on the following rule: immediate = _desired_time > {current_time} + option(_request_execution_time_buffer) + computed(option(_request_queue_time_stat)) Examples: • `_desired_time` is in `1h`, `_request_execution_time_buffer=300` (5min), statistic set is "EstimatedWaitTime" returning, for example, 10min then the Callback is not immediate and will be submitted later for execution. • `_desired_time` is in `5min`, `_request_execution_time_buffer=120` (2min), statistic set is "EstimatedWaitTime" returning, for example, 5min then the Callback is immediate and is submitted for execution.
<property>	string	no	Any properties key/values to be attached.

Description	Start-Callback		
			Key/Values may be used in Orchestration execution service. Keys without an underscore prefix are User Attached Data.
_callback_state	string	no	Forces creation of Callback in a specified state. **Important:** This is for advanced users that handle Callback life-cycle externally to GMS. By default, the `_callback_state` value is either QUEUED or SCHEDULED depending if the Callback is processed as immediate or scheduled (respectively).
_urs_virtual_queue	string	no	Queue to use for this callback if several virtual queues are used for callback with identical configuration.
_request_queue_time_stat	string	no	Queue statistics. For example, "ExpectedWaitTime;Queue;8999@SIP_S Note: If this option is set, it always overwrites the parameter.
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 request, by doing the following:

- Validates that the request is still in the queue.
  - If not, returns the appropriate error.
  - If valid, removes the request from the scheduling queue.
- Checks state of the Callback request:
  - If _callback_state=QUEUED, a callback cancel event is submitted to the execution service.
- Callback request is marked _callback_state=COMPLETED with _callback_reason=CANCELLED.

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)			

Description	Cancel-Callback		
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"
}



DELETE http://localhost:8080/genesys/1/service/callback/callback-test/361-cf088d4e-88ab-452c-
ac1f-39086cc96cbe
Result 400 Bad Request
{
    "message": "Request already cancelled or completed : [361-cf088d4e-88ab-452c-
ac1f-39086cc96cbe]",
    "exception":
"com.genesyslab.gsg.services.callback.exceptions.CallbackExceptionInvalidOperation"
}
```

Reschedule-Callback

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

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

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

Description	Update-Callback		
Method	PUT		
URL	http://host:port/{base-web-application}/service/callback/{callback-execution-name}/{service_id}		
Name	**Type**	**Mandatory**	**Description**

Description	Update-Callback		
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	no	The new time for which to reschedule the callback. If provided and validated through office-hours, _callback_state will be automatically switched to "scheduled" or "immediate", discarding `_callback_state` property.
_callback_state	string	no	Possible values are: • SCHEDULED: The request is handled by the Callback Management service (there are no sessions started in ORS). While in this state, the request will be handled by Management when the specified `desired_time` is approaching. • QUEUED: Callbacks actively waiting for an agent in ORS/URS; agent is not assigned yet. • ROUTING - Agent is reserved but the call is not yet routed to the agent. • PROCESSING: The Callback is being handled by the assigned agents. • COMPLETED: The Callback was completed with

Description	Update-Callback		
			_callback_reason; for example, timed-out, cancelled, and so on. Note: The _callback_state is incompatible with the _new_desired_time property.
<other properties>	any	no	Properties to be updated in request.
Response Body (JSON content)			
Name	**Type**	**Mandatory**	**Description**
See example.	string	yes	• If accepted, none. • If not, a list of possible times around the desired_time.

HTTP code	200
HTTP message	OK

Example: Operation Successful

Successful Rescheduling

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

Failed Rescheduling

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

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
      }
}
```

Sample operation typicaly performed by ORS execution

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

Query-Callback

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

Notes:

- Outstanding Callback services are requests where `_callback_state` is one of the following values: `SCHEDULED, QUEUED, ROUTING, PROCESSING, COMPLETED.`
- Properties allowing the Callback request trackback are defined as comma-separated keys with service option `_customer_lookup_keys`.

Description	Query-Callback		
Method	GET		
URL	• http://host:port/{base-web-application}/service/callback/{callback-execution-name}?{property=value} • GET http://host:port/{base-web-application}/service/callback?{property=value}		
Name	**Type**	**Mandatory**	**Description**

Description	Query-Callback		
URI Parameters			
{callback-execution-name}	string	no since 8.5.101.03	This is the name of the callback execution service of 'ors' type that is provisioned in GMS.
{property=value}	string	yes	This is a property name used to query the callback. Several properties may be specified.
operand	string	no	Possible values are `AND` or `OR`. Default is `AND`. When multiple properties are provided, specifies which operation to perform on matched Callback requests: • `AND` means all properties must match; • `OR` means any property can match.
_callback_state Since 8.5.101.03	string	no	Specifies a unique state to filter onto. For example: • `_callback_state='COMPLETED'` filters callbacks and returns only callbacks in COMPLETED state. • `_callback_state='!COMPLETED'` filter callbacks and only return the ones that are not COMPLETED. Important The character "!" is used to negate a case. You can query the following callback states: • Intermediate states: • `PROCESSING</tt :> Customer is connected to an agent and`

Description	Query-Callback		
			`talking with this agent.` • `QUEUED`: Callback request has been submitted to the callback Queue. • `SCHEDULED`: Callback request is scheduled. • `ROUTING`: Customer phone is reached and waiting for an agent. • Final state: `COMPLETED`: The call is done. • Reason `FAIL_LOAD_MESSAGE_FILE`: Callback Service cannot load the strings resource file. • Reason `CANCELLED`: Callback Service received a cancel request for this callback. • Reason `NOT_AVAILABLE`: Callback Service exited with no specified reason. • Reason `FAIL_INTERACTION_DELETED`: The callback interaction was deleted prior to routing the interaction to the agent. • Reason `AGENT_CONNECTED`: Callback Service

Description	Query-Callback		
			successfully routed the interaction to the agent. • Reason `FAIL_TARGET_NOT_FOUND`: Callback Service cannot reserve the requested target to handle the request. • Reason `FAIL_ERROR`: Callback Service failed due to an unknown error. • Reason `FAIL_TIMEOUT_TTL`: Callback Service did not manage to handle the request in the specified time (`_ttl`). • Reason `FAIL_IXN_UNKNOWN_MEDIA_TYPE`: The media type of the interaction is not supported by Callback Service. Callback Service only processes `voice` and `chat` interactions. • Reason `FAIL_NO_CUSTOMER_NUMBER`: Customer number is missing. • Reason `FAIL_USER_NO_CONFIRM`: The user confirmation was not received although it was

Description	Query-Callback		
			required; this issue can occur if `_on_user_confirm_timeout` is not set to `CONNECT-ANYWAY`. • Reason `FAIL_QUEUEING`: The callback request could not be queued. • Reason `FAIL_AGENT_CONNECT`: The callback interaction could not be connected to the agent. • Reason `AGENT_PREVIEW_CANCEL_AFTER_<n>REJECTS`: The agent rejected the request '<n>' times. • Reason `FAIL_CALL_TO_CUSTOMER`: Replaces `FAIL_USER_UNREACHABLE` since GMS 8.5.102.14. Callback Service could not connect the customer. • Reason `FAIL_INCORRECT_CONFIG_MEDIA_TYPE`: The `_media_type` option is set to an incorrect value.Callback Service only processes `voice` and `chat` interactions. • Reason `FAIL_FAX_REACH`

Description	Query-Callback		
			ED: Callback Service could not connect the customer. The provided number was answered by a fax machine. • Reason SUBMIT_ERROR: GMS did not manage to submit the callback service request to Orchestration Server for processing. • Reason FAIL_USER_UNREACHABLE: Reported as FAIL_CALL_TO_CUSTOMER prior to GMS 8.5.102.14.
_desired_time_from Since 8.5.101.03	string	no	Specifies ISO timestamps. All callback matching lookup properties and scheduled before this time will be filtered out.
_desired_time_to Since 8.5.101.03	string	no	Specifies ISO timestamps. All callback matching lookup properties and scheduled after this time will be filtered out. (warning)
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

Description	Query-Callback		
			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",
      "_callback_state": "QUEUED",
      "_expiration_time": "2014-11-03T18:36:45.000Z",
      "_customer_number": "555-5461206",
      "url": "/1/service/callback/BasicCallback/a550a12e-ca77-4146-98d0-58960e0939f7"
    },
   {
      "_id": "4a1ea889-1ef7-432d-a543-cff96b4a2daf",
      "desired_time": "2013-05-27T15:10:00.000Z",
      "_callback_state": "SCHEDULED",
      "_expiration_time": "2014-11-03T18:36:45.000Z",
      "_customer_number": "555-5461206",
      "url": "/1/service/callback/BasicCallback/4a1ea889-1ef7-432d-a543-cff96b4a2daf"
   }
]
```

Query-Availability

Description	Availability REST Request		
Method	GET		
URL	/genesys/1/service/callback/<service-name>/availability		
Name	**Type**	**Mandatory**	**Description**
URI Parameters			
None.			
Body			
start	date	no	Start date is specified in `ISO 8601` format, using UTC as timezone: `yyyy-MM-ddTHH:mm:ss.SSSZ`.

Description	Availability REST Request		
			If not specified, it is assumed to be now.
timestamp	date	no	Alias to `start` parameter; kept for compatibility reasons.
number-of-days	integer	no	Used as an alternative to the end date. If neither end, nor `number-of-days` is specified, the end date is assumed to be the same as start date.
end	date	no	End date is specified in `ISO 8601` format, using UTC as timezone: `yyyy-MM-ddTHH:mm:ss.SSSZ`. If neither end, nor `number-of-days` is specified, the end date is assumed to be the same as start date.
max-time-slots	integer	no	Maximum number of time slots to be included in the response when the office is open and capacity is above zero. It can be used to improve performance of the query over a lengthy period of time.
Response Body			
See example below.			

Request example:

```
http://localhost:8080/genesys/1/service/callback/Callback_VQ/
availability?start=2014-12-03T15:00:00.000Z#ber-of-days=2
```

Response

The Callback controller provides a facet to the availability service, which uses the calendar service underneath. In the same manner as the calendar service takes three non-mandatory input parameters (start, number-of-days, end), the availability service should accept the same parameters and pass them on to the calendar service. The response contains a map of time slots and capacity counters. The slots are ordered in ascending order. Any time slots where capacity is full (for example, zero) are not provided in the response. In a similar way, if the office is closed, those time slots are not provided in the response.

```
{
    // All periods are ordered in ascending time order
```

```
    "2014-10-17T13:00:00.000Z":"5",
    "2014-10-17T13:10:00.000Z":"4",
    // there were no agents available between 13:20 and 13:30 UTC, hence the time slot is not
reported
    "2014-10-17T13:30:00.000Z":"5"
}
```

> **Important**
> Existing calendar configurations must be updated for the time zone definition. Instead of EST or PST time zones that were configured using Configuration Manager, you must use time zones as allowed in Java: http://en.wikipedia.org/wiki/List_of_tz_database_time_zones, such as `America/Toronto`, or `Europe/Paris`. You must also change the service option `_type` from `ors` to `builtin`.

ADMIN - Query-Callback

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

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

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". For example: "2013-05-28T15:30:00.000Z"
{target}	string	no	This is the name of a callback execution service. If not specified,

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

```
            "_desired_time": "2013-06-07T16:45:00.000Z",
            "_id": "8f68d4969d904d039ccf0101fac39283",
            "url": "/1/service/callback/AdvancedCallback/8f68d4969d904d039ccf0101fac39283"
        }
    ]
}
```

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

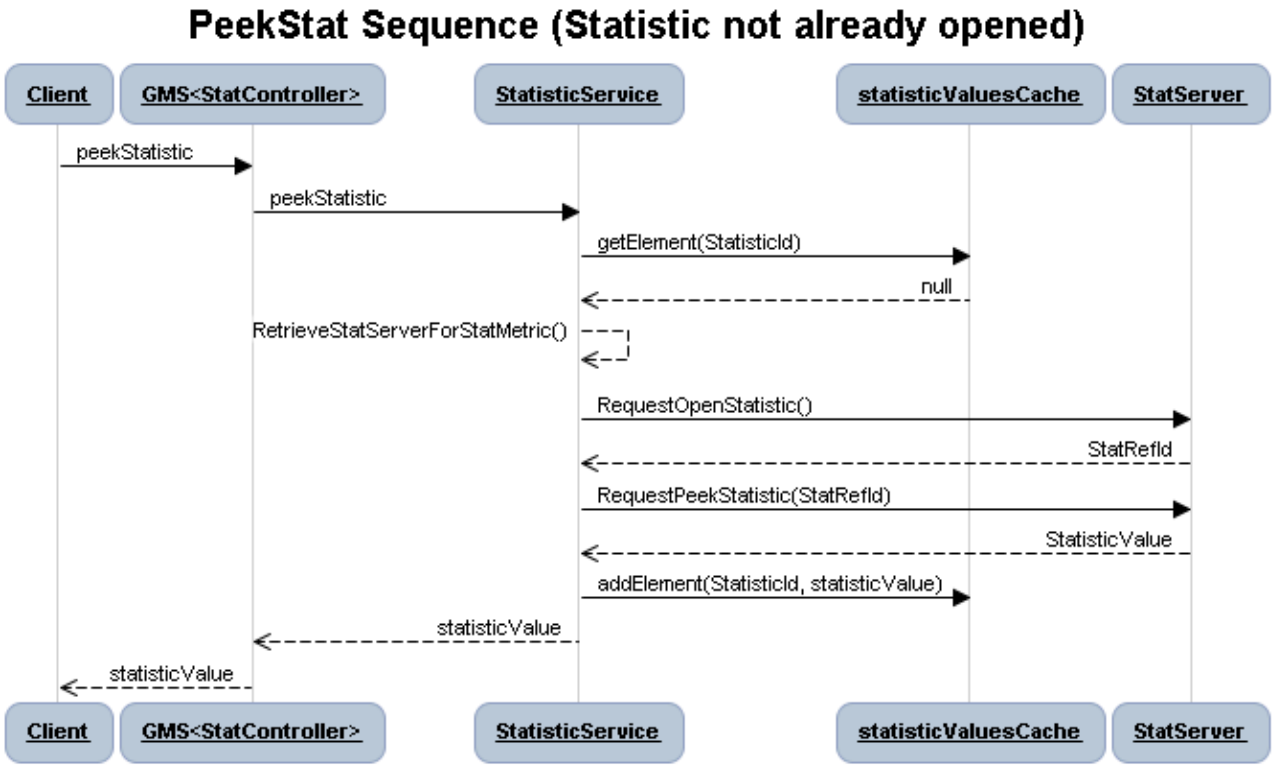
```
{
    "message": "Too many requests around this desired time : 2013-05-24T18:03:29.952Z",
    "exception": "com.genesyslab.gsg.services.callback.CallbackServiceException"
}

{
    "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
    }
}
```

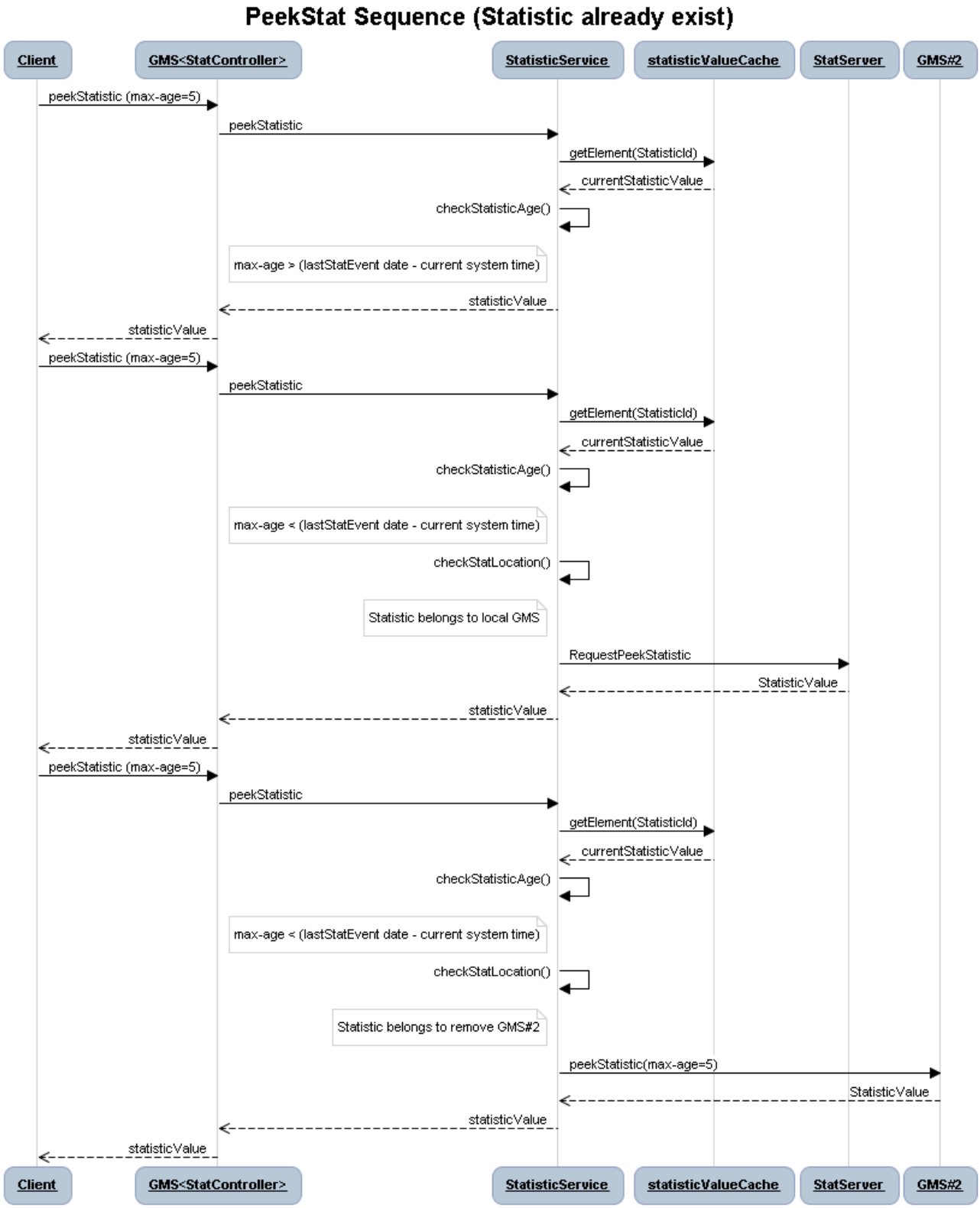
Stat Server API

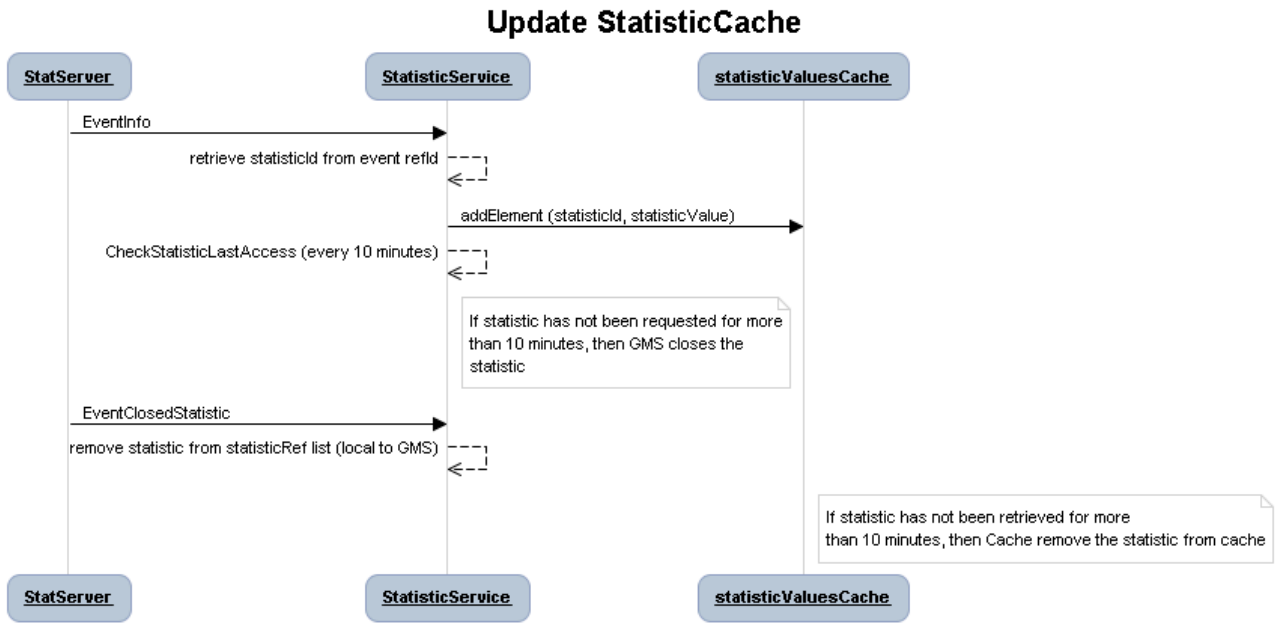
Overview

This API is used to interact with the Genesys Statistics Server (Stat Server). The API provides the request so an application can get statistics related to the Contact Center.

Sequences Diagrams




PeekStat Sequence (Statistic already exist)
Client
GMS<StatController>
StatisticService
statisticValueCache
StatServer
GMS#2
peekStatistic (max-age=5)
peekStatistic
getElement(StatisticId)
currentStatisticValue
checkStatisticAge()
max-age > (lastStatEvent date - current system time)
statisticValue
statisticValue
peekStatistic (max-age=5)
peekStatistic
getElement(StatisticId)
currentStatisticValue
checkStatisticAge()
max-age < (lastStatEvent date - current system time)
checkStatLocation()
Statistic belongs to local GMS
RequestPeekStatistic
StatisticValue
statisticValue
statisticValue
peekStatistic (max-age=5)
peekStatistic
getElement(StatisticId)
currentStatisticValue
checkStatisticAge()
max-age < (lastStatEvent date - current system time)
checkStatLocation()
Statistic belongs to remove GMS#2
peekStatistic(max-age=5)
StatisticValue
statisticValue
statisticValue
Client
GMS<StatController>
StatisticService
statisticValueCache
StatServer
GMS#2



API

The Stat Server API exposes two interfaces:

- "**genesys/1/internal_statistic**" - for internal access with no authentication control
- "**genesys/1/statistic**" - for external use, which uses Basic Access Authentication (BA) to authenticate users

As a standard protocol, username and password for BA can be passed in the URL, for example:

```
http://username:password@127.0.0.1:8080/genesys/1/statistic
```

PeekStat Request

This request gets the current value of the peek statistic.

Operation

Method	POST		
URL	"/genesys/1/statistic" or "genesys/1/internal_statistic"		
Parameter	**Type**	**Mandatory**	**Description**
Header Parameters			
Cache-Control : max-age=XX	A number of seconds	no	The max-age value used to check if GMS has to recalculate the statistic. If the peek statistic time window is greater than maxAge, GMS recalculates the statistic value. If the value is not present, it returns the latest value in cache.
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 statistic (objectId, objectType, tenant, tenantPassword, metric, notificationMode). NotificationMode can be NoNotification, Reset, or Immediate.			

Response

HTTP code	200
HTTP Message	OK
HTTP Header	Age: containing the age (in seconds) of the statistic value.
Body	A json object representing the current statistic value.

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

HTTP code	403
HTTP Message	Forbidden

HTTP code	403
Body	Error message stating that the statistic is not well defined, for example: {"message":"Place 'SIP_Server_Place' (Tenant 'Environment') not found","exception":"com.genesyslab.gsg.services.statistic.Sta

HTTP code	500
HTTP Message	Internal server error
Body	If Stat Server is not connected, it returns, for example, {"message":"Statistic Service unavailable","exception":"com.genesyslab.gsg.services.statis

Example

Request

```
POST http://127.0.0.1:8080/genesys/1/statistic HTTP/1.1
Host: 127.0.0.1:8080
Connection: keep-alive
Content-Length: 91
Authorization: Basic ZGVmYXVsdDpwYXNzd29yZA==
Origin: chrome-extension://fdmmgilgnpjigdojojpjoooidkmcomcm
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.22 (KHTML,
like Gecko) Chrome/25.0.1364.97 Safari/537.22
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8,fr;q=0.6,fr-FR;q=0.4
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

objectId=KSippola&objectType=Agent&tenant=Environment&tenantPassword=&metric=TotalLoginTime
```

The following request is another example with cache-control set to two seconds:

```
POST http://127.0.0.1:8080/genesys/1/statistic HTTP/1.1
Host: 127.0.0.1:8080
Connection: keep-alive
Content-Length: 91
Cache-Control: max-age=2
Authorization: Basic ZGVmYXVsdDpwYXNzd29yZA==
Origin: chrome-extension://fdmmgilgnpjigdojojpjoooidkmcomcm
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.22 (KHTML, like Gecko) Chrome/25.0.1364.97
Safari/537.22
Content-Type: application/x-www-form-urlencoded Accept: */*
Accept-Encoding: gzip,deflate,sdch Accept-Language: en-US,en;q=0.8,fr;q=0.6,fr-FR;q=0.4
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3

objectId=KSippola&objectType=Agent&tenant=Environment&tenantPassword=&metric=TotalLoginTime
```

Response

The following response is a json object representing the value for the statistic:

```
HTTP/1.1 200 OK
Pragma: no-cache
Cache-Control: no-cache
Cache-Control: no-store
Age: 9
Content-Length: 14
Content-Type: application/json
Server: Jetty(8.1.8.v20121106)

{"value":1025}
```

Extended API

PeekStat Request: Querying Multiple Statistic Values in a Single Request

This request gets the current values of several peek statistic objects.

Operation

Method	POST		
URL	/genesys/1/statistics		
Parameter	**Type**	**Mandatory**	**Description**
Body: The body is `x-www-form-urlencoded` form consisting of different items representing the key/value pairs associated with the statistic (`objectId, type, tenantName...`) • `keystat<1>`: the statistic object as `Metric;ObjectType;ObjectId;TenantName` • `keystat<2>`: the statistic object as `Metric;ObjectType;ObjectId;TenantName` • `...<n>`:			

Response

HTTP code	200
HTTP message	OK
Body	A JSON array of key/value pairs: `key` is the key stat (see request) and `value` is the statistic value.

If a problem occurs during subscription, it will return the following status codes:

HTTP code	404
HTTP message	Not Found
desc	Statistic is not defined on StatServer side.

HTTP code	500
HTTP message	A JSon exception
desc	The exception contains the StatServer message.

Example

The following example shows a peek statistic with multiple statistic objects:

Operation

```
Request URL:http://localhost:8080/genesys/1/statistics
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:108
Content-Type: application/x-www-form-urlencoded
Host:localhost:8080
Origin:http://localhost:8080

stat1=ExpectedWaitTime;Queue;9002@SIP_Switch;Environment&stat2=ExpectedWaitTime;Queue;9001@SI
```

Result
The above result is the array of statistic values.

```
HTTP 200 OK

{"stat1":10000,"stat2":10000}
```

Configuration

Stat Server Connection

In order to use the Stat Server API, there must be a connection to Stat Server in the GMS Application. You can create the connection in Configuration Manager on the *Connections* tab. See Creating and Configuring the GMS Application Cluster Object.

You can add several Stat Servers in the *Connection* tab; this feature is used in case of different statistic definitions in the Stat Servers. GMS will open the statistic on the Stat Server to which the statistic belongs to. In the case of the same statistic definition in Stat Servers, GMS will take the first Stat Server that contains the statistic definition.

ADDP Setting

Open Stat Server in the GMS *Connection* tab and set the connection protocol to *addp*. Set the values for the Local Timeout and Remote Timeout, and then select the Trace mode. See Implementing ADDP for more information.

Note: The addp traces are only visible on the Stat Server side. The following example shows an addp trace in Stat Server logs:

```
-AP[10]-<-2168 @15:37:30.4540

-Ap[10]->-2168 @15:37:30.4560
```

High Availability

If Stat Servers (defined in GMS connections) are configured to use High Availability (HA) (Primary/Backup Stat Server), in the case of a lost connection with the primary Stat Server, the Stat Server API will switch to the backup Stat Server.

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:
  - http://developer.android.com/google/gcm/index.html.

> **Important**
> When subscribing for notifications via GCM, it is important to ensure that the device's *registration ID* is provided as **notificationDetails.deviceId**. This registration ID must be obtained by registering the device with GCM servers through the Google Services API. For specific client implementation details, please refer to:
>
> - http://developer.android.com/google/gcm/client.html

Keystore/Truststore Configuration Hints

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

1. Use your web browser or another tool to retrieve the certificates required for the following addresses:
   - http://developer.android.com/google/gcm/index.html.
2. Import those certificates into the keystore you plan to use.

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

C2DM Service

> **Important**
> 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.

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

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 recommendations 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.

Callback Push Notifications for Android

Genesys Mobile Services (GMS) employs various mechanisms to achieve asynchronous messaging (push notifications). For Android devices, it is a combination of GCM/C2DM, and Comet. Likewise, iOS devices employ APNs and Comet. The scope of this article is limited to how push notifications are handled in Android devices, particularly for the Callback application.

Note: For iOS devices, see Callback Push Notifications for iOS.

Procedure

Implementing a GCM client is well documented by Google. Alternately, you can also refer to the Genesys Mobile Services Android Sample for a Genesys implementation of a GCM BroadcastReceiver.

Push notifications can be divided into two distinct parts:

1. Chat (implements push notifications over Comet)
2. Callback (implemented over GCM)

For Chat, a Bayeux Client is created to listen to all push notifications related to Chat. The default channel for Chat is `/_genesys`. The format of Chat push notifications can be seen in the Chat (Comet) section.

For Callback push notifications, refer to the Genesys Mobile Services Android Sample for reference.

Processing of GCM notifications is a three-step procedure:

1. Obtain Service ID and Action identifier from GCM Intent.
2. Issue HTTP POST to GMS with specified action to obtain action data as JSON.
3. Execute action using data provided by GMS.

The data contained within the GCM Intent is structured as follows:

```
Intent intent;
Bundle extras = intent.getExtras();
String message = extras.getString("message");
System.out.println(message);

---------------------------------------------
Result:
{
        "_id":"$(_id)",
        "_action":"$(_action)",
}
```

In the case of the Genesys sample client, the `GenesysCloudMessageReceiver` repackages the GCM Intent into an application-specific Intent:

```
Intent newIntent = new Intent(context, GenesysSampleActivity.class);
                newIntent.setAction(Globals.ACTION_GENESYS_CLOUD_MESSAGE);
                newIntent.putExtra(Globals.EXTRA_MESSAGE, extras.getString("message"));
                newIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                context.startActivity(newIntent);
```

This intent is then handled by the `GenesysSampleActivity (handleIntent() > interpretCloudMessage())` where the encapsulated data is used to form an HTTP POST with the following URL:

```
$(ServerURL)$(URLPath)$(_id)/$(_action)
for example,
http://135.34.145.123:8080/genesys/1/service/3SQI3S31693JL9L3R0O5O6T4OC000U73/get-dialog-start-chat
```

Identifier	Description	Example Values
$(ServerURL)	URL to Genesys Mobile Services host	http://135.34.145.123:8080
$(URLPath)	Path to Services API	/genesys/1/service/
$(_id)	GMS-issued Service ID	3SQI3S31693JL9L3R0O5O6T4OC000U73
$(_action)	Callback action to perform	get-dialog-user-confirmation-provide_code-true get-dialog-user-confirmation-provide_code-false get-dialog-start-chat connect-inbound connect-outbound wait-for-agent

The response of the HTTP POST request contains JSON, which describes an action to perform and/or UI elements to display in the client application. Each of these requests are referred to as Dialogs.

Dialogs (REST)

The following examples are JSON structures returned by the GMS Callback service to the client application. The contents of the JSON response depends on the Callback action performed (as described in the Procedure section).

Refer to the Genesys Mobile Services Android Sample for examples on how these JSON responses can be interpreted as actions (for example: Call agent, Display menu, Display dialog) and/or UI elements (for example: Confirmation dialogs or Menu items).

get-dialog-user-confirmation-provide_code-true

```
{
        "_dialog_id":"0",
        "_label":"Agent is available right now",
        "_user_action_url":"$(ExtURLBase)/1/service/$(ServiceID)/not-used",
        "_method":"POST",
        "_action":"DisplayMenu",
        "_expires": "$(Date)",
        "_resource_url":"$(ExtURLBase)/1/service/$(ServiceID)/get-dialog-user-confirmation",
```

```
        "_content":[
                {
                        "_group_name":"Are you ready?",
                        "_group_content":[
                                {
                                        "_dialog_id":"1",
                                        "_label":"Yes, I'm ready to talk",
                                        "_action":"MenuItem",
                                        "_user_action_url":"$(ExtURLBase)/1/
service/$(ServiceID)/connect",
                                        "_method":"POST",
                                        "_id_to_jump_before":"exit://",
                                        "_confirmation_dialog":{
                                                "_text":"You will hear tones immediately
after call is connected. This is normal.",
                                                "_dialog_type":"Notification",
                                                "_dismiss_timeout": 2
                                        }
                                },{
                                        "_dialog_id":"2",
                                        "_label":"No, try again in 5 minutes",
                                        "_action":"MenuItem",
                                        "_user_action_url":"$(ExtURLBase)/1/
service/$(ServiceID)/snooze",
                                        "_method":"POST",
                                        "_id_to_jump_before":"exit://"
                                },{
                                        "_dialog_id":"3",
                                        "_label":"Cancel, my problem has been solved",
                                        "_action":"MenuItem",
                                        "_user_action_url":"$(ExtURLBase)/1/
service/$(ServiceID)/cancel",
                                        "_method":"POST",
                                        "_id_to_jump_before":"exit://"
                                }
                        ]
                }
        ]
}
```


get-dialog-user-confirmation-provide_code-false


```
{
        "_dialog_id":"0",
        "_label":"Agent is available right now",
        "_user_action_url":"$(ExtURLBase)/1/service/$(ServiceID)/not-used",
        "_method":"POST",
        "_action":"DisplayMenu",
        "_expires": "$(Date)",
        "_resource_url":"$(ExtURLBase)/1/service/$(ServiceID)/get-dialog-user-confirmation",
        "_content":[
                {
                        "_group_name":"Are you ready?",
                        "_group_content":[
                                {
                                        "_dialog_id":"1",
                                        "_label":"Yes, I'm ready to talk",
                                        "_action":"MenuItem",
                                        "_user_action_url":"$(ExtURLBase)/1/
service/$(ServiceID)/connect",
                                        "_method":"POST",
                                        "_id_to_jump_before":"exit://",
                                },{
```

```
                                        "_dialog_id":"2",
                                        "_label":"No, try again in 5 minutes",
                                        "_action":"MenuItem",
                                        "_user_action_url":"$(ExtURLBase)/1/
service/$(ServiceID)/snooze",
                                        "_method":"POST",
                                        "_id_to_jump_before":"exit://"
                                },{
                                        "_dialog_id":"3",
                                        "_label":"Cancel, my problem has been solved",
                                        "_action":"MenuItem",
                                        "_user_action_url":"$(ExtURLBase)/1/
service/$(ServiceID)/cancel",
                                        "_method":"POST",
                                        "_id_to_jump_before":"exit://"
                                }
                        ]
                }
        ]
}
```


get-dialog-start-chat


```
{
        "_dialog_id": "1",
        "_action":"StartChat",
        "_label":"Start Chat",
        "_start_chat_url":"$(ExtURLBase)/1/service/$(ServiceID)/ixn/chat",
        "_comet_url":"$(CometURL)",
        "_user_header":"$(GMSUser)",
        "_id_to_jump_before":"exit://",
        "_chat_parameters":{
                "subject":"None"
        },
        "_id":"$(ServiceID)"
}
```


connect-inbound


```
{
        "_dialog_id":"0",
        "_label":"Connecting ...",
        "_action":"DialNumber",
        "_tel_url":"n/a",
        "_access_code":"n/a",
        "_id":"$(ServiceID)"
}
```


connect-outbound


```
{
        "_dialog_id":"0",
        "_action":"ConfirmationDialog",
        "_text":"You will receive the call shortly",
        "_ok_title":"Ok",
        "_id":"$(ServiceID)"
}
```


wait-for-agent


```
{
```

```
        "_dialog_id":"0",
        "_action":"ConfirmationDialog",
        "_text":"We will notify you when agent is available",
        "_ok_title":"Ok",
        "_id":"$(ServiceID)"
}
```


Push Notifications

Chat (Comet)

Message Receipt


```
{
        "data":{
                "message":{
                        "startedAt":"2014-05-02T16:27:38Z",
                        "chatIxnState":"TRANSCRIPT",
                        "chatSessionId":"000FRa9NYM9A001K",
                        "transcriptToShow":[["Message.Text","Stan","Hello.","8","CLIENT"]],
                        "transcriptPosition":"2",
                        "chatServiceMessage":"Chat service is available"
                },
                "id":"b2e607a0d21611e3000010932938a0ff",
                "tag":"service.chat.refresh.3SQIS3S1693JL9L3R00506T40C000UL4"
        },
        "channel":"/_genesys"
}
```


Party Joined/Left


```
{
        "data":{
                "message":{
                        "startedAt":"2014-05-02T16:27:38Z",
                        "chatIxnState":"TRANSCRIPT",
                        "chatSessionId":"000FRa9NYM9A001K",
                        "transcriptToShow":[["Notice.Joined","Kristi Sippola","has joined the
session","17","AGENT"]],
                        "transcriptPosition":"3",
                        "chatServiceMessage":"Chat service is available",
                },
                "id":"b7dd6460d21611e3000010932938a0ff",
                "tag":"service.chat.refresh.3SQIS3S1693JL9L3R00506T40C000UL4"
        },
        "channel":"/_genesys"
}
```


Typing Started/Stopped


```
{
        "data":{
                "message":{
                        "startedAt":"2014-05-02T16:27:38Z",
                        "chatIxnState":"TRANSCRIPT",
                        "chatSessionId":"000FRa9NYM9A001K",
```

```
                        "transcriptToShow":[["Notice.TypingStarted","Kristi Sippola","is
typing","20","AGENT"]],
                        "transcriptPosition":"4",
                        "chatServiceMessage":"Chat service is available",
                },
                "id":"b91bd7d0d21611e3000010932938a0ff",
                "tag":"service.chat.refresh.3SQIS3S1693JL9L3R00506T40C000UL4"
        },
        "channel":"/_genesys"
}
```


Notes

Identifier	Description	Values
$(TranscriptType)	Type of event to display in the chat log.	Message.Text Notice.TypingStarted Notice.TypingStopped Notice.Joined Notice.Left
$(Timestamp)	UTC Time format	YYYY-MM-DDTHH:MM:SSZ
$(TranscriptPosition)	Line Number	Some integer.
$(ChatIxnState)	State of chat interaction.	TRANSCRIPT DISCONNECTED

Callback Push Notifications for iOS

This article is for developers who wish to develop an iOS client application for Genesys Mobile Services (GMS) Callback Services.

GMS employs various mechanisms to achieve asynchronous messaging (push notifications). For iOS devices, Apple Push Notification service (APN) and Comet are used. For Android devices, GCM/C2DM and Comet are used. This article describes how push notifications should be handled in iOS devices. The Genesys Mobile Services iOS Sample provides example code for the concepts discussed.

Note: For Android devices, see Callback Push Notifications for Android.

Push Notifications in iOS Applications

Push notifications are used for two purposes in GMS iOS applications:

- Chat
- Callback

Chat

The GMS chat implementation uses Comet push notifications for the text message exchange. GMS implements a Comet server that the mobile client connects to when a chat session is opened.

The iOS sample code includes a Comet library. The library includes a Comet client class called `DDCometClient`. The iOS sample application `GMSChatViewController` class illustrates how to use `DDCometClient` to connect to the server and to send and receive chat messages. The default channel used for chat is `"/_genesys"`.

Callback

The GMS Callback functions utilize APN for push notifications to iOS applications. The processing of APN notifications operates as follows:

1. Obtain the Service ID and Action identifier from the notification “message” component (the “message” component is in JSON format).
2. Issue an HTTP P0ST to GMS with specified action. The response includes further action data in JSON format.
3. Execute action using data provided by GMS.

The APN notification contains several components. The “aps” component specifies the confirmation dialog to display to the user. You can refer to Apple developer documentation for more information on this topic. The iOS sample application `GMSAppDelegate didReceiveRemoteNotification` method provides an example. The “message” component provides the data from GMS and has the following format:

```
{
        "_id":"$(_id)",
        "_action":"$(_action)",
}
```

The `_id` and `_action` parameters are extracted and used to construct a URL for a POST to GMS. The iOS sample application `GMSAppDelegate` `processAPN` method provides an example of this.

```
$(ServerURL)$(URLPath)$(_id)/$(_action)
e.g.
http://135.34.145.123:8080/genesys/1/service/3SQI3S31693JL9L3R0O5O6T4OC000U73/get-dialog-start-chat
```

Identifier	Description	Example Values
$(ServerURL)	URL to Genesys Mobile Services host	http://135.34.145.123:8080
$(URLPath)	Path to Services API	/genesys/1/service/
$(_id)	GMS-issued Service ID	3SQI3S31693JL9L3R0O5O6T4OC000U73
$(_action)	Callback action to perform	get-dialog-user-confirmation-provide_code-true get-dialog-user-confirmation-provide_code-false get-dialog-start-chat connect-inbound connect-outbound wait-for-agent

The response of the HTTP POST request contains JSON data, which describes an action to perform and/or UI elements to display in the client application. Each of these requests will be referred to as Dialogs.

Dialogs (REST)

The following are example JSON structures returned by the GMS Callback service to the client application. The contents of the JSON response depends on the Callback action performed (as described in the Callback section).

Refer to the Genesys Mobile Services iOS Sample for examples on how these JSON responses can be interpreted as actions (for example, Call agent, Display menu, Display dialog) and/or UI elements (for example, Confirmation dialogs or Menu items).

get-dialog-user-confirmation-provide_code-true

```
{
        "_dialog_id":"0",
        "_label":"Agent is available right now",
        "_user_action_url":"$(ExtURLBase)/1/service/$(ServiceID)/not-used",
        "_method":"POST",
        "_action":"DisplayMenu",
        "_expires": "$(Date)",
        "_resource_url":"$(ExtURLBase)/1/service/$(ServiceID)/get-dialog-user-confirmation",
        "_content":[
                {
                        "_group_name":"Are you ready?",
```

```
                        "_group_content":[
                                {
                                        "_dialog_id":"1",
                                        "_label":"Yes, I'm ready to talk",
                                        "_action":"MenuItem",
                                        "_user_action_url":"$(ExtURLBase)/1/
service/$(ServiceID)/connect",
                                        "_method":"POST",
                                        "_id_to_jump_before":"exit://",
                                        "_confirmation_dialog":{
                                                "_text":"You will hear tones immediately
after call is connected. This is normal.",
                                                "_dialog_type":"Notification",
                                                "_dismiss_timeout": 2
                                        }
                                },{
                                        "_dialog_id":"2",
                                        "_label":"No, try again in 5 minutes",
                                        "_action":"MenuItem",
                                        "_user_action_url":"$(ExtURLBase)/1/
service/$(ServiceID)/snooze",
                                        "_method":"POST",
                                        "_id_to_jump_before":"exit://"
                                },{
                                        "_dialog_id":"3",
                                        "_label":"Cancel, my problem has been solved",
                                        "_action":"MenuItem",
                                        "_user_action_url":"$(ExtURLBase)/1/
service/$(ServiceID)/snooze",
                                        "_method":"POST",
                                        "_id_to_jump_before":"exit://"
                                }
                        ]
                }
        ]
}
```


get-dialog-user-confirmation-provide_code-false


```
{
        "_dialog_id":"0",
        "_label":"Agent is available right now",
        "_user_action_url":"$(ExtURLBase)/1/service/$(ServiceID)/not-used",
        "_method":"POST",
        "_action":"DisplayMenu",
        "_expires": "$(Date)",
        "_resource_url":"$(ExtURLBase)/1/service/$(ServiceID)/get-dialog-user-confirmation",
        "_content":[
                {
                        "_group_name":"Are you ready?",
                        "_group_content":[
                                {
                                        "_dialog_id":"1",
                                        "_label":"Yes, I'm ready to talk",
                                        "_action":"MenuItem",
                                        "_user_action_url":"$(ExtURLBase)/1/
service/$(ServiceID)/connect",
                                        "_method":"POST",
                                        "_id_to_jump_before":"exit://",
                                },{
                                        "_dialog_id":"2",
                                        "_label":"No, try again in 5 minutes",
                                        "_action":"MenuItem",
```

```
                                        "_user_action_url":"$(ExtURLBase)/1/
service/$(ServiceID)/snooze",
                                        "_method":"POST",
                                        "_id_to_jump_before":"exit://"
                                },{
                                        "_dialog_id":"3",
                                        "_label":"Cancel, my problem has been solved",
                                        "_action":"MenuItem",
                                        "_user_action_url":"$(ExtURLBase)/1/
service/$(ServiceID)/snooze",
                                        "_method":"POST",
                                        "_id_to_jump_before":"exit://"
                                }
                        ]
                }
        ]
}
```


get-dialog-start-chat


```
{
        "_dialog_id": "1",
        "_action":"StartChat",
        "_label":"Start Chat",
        "_start_chat_url":"$(ExtURLBase)/1/service/$(ServiceID)/ixn/chat",
        "_comet_url":"$(CometURL)",
        "_user_header":"$(GMSUser)",
        "_id_to_jump_before":"exit://",
        "_chat_parameters":{
                "subject":"None"
        },
        "_id":"$(ServiceID)"
}
```


connect-inbound


```
{
        "_dialog_id":"0",
        "_label":"Connecting ...",
        "_action":"DialNumber",
        "_tel_url":"n/a",
        "_access_code":"n/a",
        "_id":"$(ServiceID)"
}
```


connect-outbound


```
{
        "_dialog_id":"0",
        "_action":"ConfirmationDialog",
        "_text":"You will receive the call shortly",
        "_ok_title":"Ok",
        "_id":"$(ServiceID)"
}
```


wait-for-agent


```
{
        "_dialog_id":"0",
        "_action":"ConfirmationDialog",
        "_text":"We will notify you when agent is available",
```

```
        "_ok_title":"Ok",
        "_id":"$(ServiceID)"
}
```


Push Notifications

Chat (Comet)

Message Receipt


```
{
        "data":{
                "message":{
                        "startedAt":"2014-05-02T16:27:38Z",
                        "chatIxnState":"TRANSCRIPT",
                        "chatSessionId":"000FRa9NYM9A001K",
                        "transcriptToShow":[["Message.Text","Stan","Hello.","8","CLIENT"]],
                        "transcriptPosition":"2",
                        "chatServiceMessage":"Chat service is available"
                },
                "id":"b2e607a0d21611e3000010932938a0ff",
                "tag":"service.chat.refresh.3SQIS3S1693JL9L3R00506T40C000UL4"
        },
        "channel":"/_genesys"
}
```


Party Joined/Left


```
{
        "data":{
                "message":{
                        "startedAt":"2014-05-02T16:27:38Z",
                        "chatIxnState":"TRANSCRIPT",
                        "chatSessionId":"000FRa9NYM9A001K",
                        "transcriptToShow":[["Notice.Joined","Kristi Sippola","has joined the
session","17","AGENT"]],
                        "transcriptPosition":"3",
                        "chatServiceMessage":"Chat service is available",
                },
                "id":"b7dd6460d21611e3000010932938a0ff",
                "tag":"service.chat.refresh.3SQIS3S1693JL9L3R00506T40C000UL4"
        },
        "channel":"/_genesys"
}
```


Typing Started/Stopped


```
{
        "data":{
                "message":{
                        "startedAt":"2014-05-02T16:27:38Z",
                        "chatIxnState":"TRANSCRIPT",
                        "chatSessionId":"000FRa9NYM9A001K",
                        "transcriptToShow":[["Notice.TypingStarted","Kristi Sippola","is
typing","20","AGENT"]],
                        "transcriptPosition":"4",
```

```
                        "chatServiceMessage":"Chat service is available",
                },
                "id":"b91bd7d0d21611e3000010932938a0ff",
                "tag":"service.chat.refresh.3SQIS3S1693JL9L3R00506T40C000UL4"
        },
        "channel":"/_genesys"
}
```


Notes

Identifier	Description	Values
$(TranscriptType)	Type of event to display in the chat log.	Message.Text Notice.TypingStarted Notice.TypingStopped Notice.Joined Notice.Left
$(Timestamp)	UTC Time format	YYYY-MM-DDTHH:MM:SSZ
$(TranscriptPosition)	Line Number	Some integer.
$(ChatIxnState)	State of chat interaction.	TRANSCRIPT DISCONNECTED

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.