Genesys Mobile Services API Reference

Notification API

4/3/2026

Contents

Notification API

Overview

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

This set of APIs manages notifications between applications and Genesys systems. It is event-driven, that is, consumers subscribe to an event and indicate how to deliver the notification, 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, so queries need to provide **application/json** content type. Its absence or incorrect value can result in a 415 (Unsupported Media Type) error.

Subscription

The subscription data identifies 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}"
}
```

Property	Mandatory	Description
subscriberId	yes	The ID of subscriber.
authorization	no	Authorization parameter to add to the HTTP headers of the request if basic authorization is required on the custom HTTP channel.
expire	no	Time, in seconds, after which the subscription expires (optional; default value is configurable).
filter	yes	The filter which is applied to the tags of incoming events. If the filter matches the tag, the system publishes the event to the destination specified by the subscription. Note: The event is published to ALL subscriptions which match the filter.
providerName	no	Name of the provider for the given subscription. If not specified, the subscription is registered for the default provider.
language	no	Language used by this subscription. If missing, GMS handles the localized strings as normal messages. See Genesys Mobile Services PushNotificationService for details about languages.
notificationDetails	yes	Information needed for delivering the event to the subscriber. • **type**–(Mandatory) Notification mechanism that the application requires. Valid values are **ios**, **android**, **gcm**, **comet**, **httpcb** (callback POST to the provided URL), and **orscb** (callback to ORS), **wns** (to-wns-client) (see more information here). • **deviceId** – (Mandatory) ID of the device to deliver a message to (in the case of HTTP or ORS callback; see Push Notification Service for further details.

Property	Mandatory	Description
		• **properties** – (Optional) The String-String map of additional properties used for delivering the notification. If the information provided is incomplete from the publisher standpoint, GMS returns an error. • **debug** – true if the production or debug provider connection must send the notifications. The subscription is sent to the debug channel if the ${debug} value is **debug** or **true**.

Note: For **comet** subscriptions, the **gms_user** header is required. For example, here is a request for an android push notification subscription (note the absence of the *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

Internal components are required to publish the events. The Notification service matches the event to the subscription using the event's tag and subscriptions filters and notifies the subscriptions with matching filters. The event structure is formatted as follows:

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

```
}
```

Where:

Parameter	Mandatory	Description
tag	yes	The message tag.
message	no	A string message. It may contain the string representation of ANY data: the notification service is message-agnostic and always interprets the message as a string. If you do not add a message, an empty string is sent to the subscribers. Make sure that the **message** content is correctly formatted if you use some JSON representation string. If the message breaks the JSON parser which attempts to parse the request body, GMS replies with a `BAD_REQUEST` response.
mediaType	no	"**string**" for a simple string, "**localizestring**" for a string with a localized parameter, and **json** for a JSON payload in the message. See Localization File.
providerName	no	This is the name of the provider that this subscription is for. If not specified, the subscription is for the default provider.
notificationDetails	no	If missing, notification is sent to default subscribers. It allows sending the notification to a specific device.
devideId	yes	The ID of the device (for example, Android device ID or iPad ID).
type	yes	Type of notification (gcm, ios...).
properties	no	Additional properties.
debug	no	If true, displays the debug log for this notification.

Filters and Tags

The tag cannot be null or an empty string. The format of tag, specified in the event, matches the java package name alphanumeric string with '.' delimiters.

- Underscores are allowed.
- The first symbol may be number.

- Only English alphanumeric chars are allowed.

The filter cannot be null or an empty string. The format of filter entry is similar to the tag format, but in addition, it allows wildcard '*' after last '.' (or only '*' which indicates a subscription to all events) and the last character cannot be '.'. So, the channels formatted as follows are allowed:

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

> When publishing event, the tag is matched versus the filters of all active subscriptions and GMS notifies all matching subscriptions. For example, consider the Notification Event published with tag **ors.agentavailability.agent123.available**. Such notification will be propagated to the subscriptions with any of the 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 in some cases when the syntax of body is involved). **Notes:** All POST requests must use the media type **application/json**.

Create Subscription

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

Operation

POST /genesys/{api version}/notification/subscription
Body: JSON with subscription (see above)

Response

Success

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

Errors

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

HTTP code	400
HTTP message	BAD REQUEST

- If the subscription is being created for a push type which is not enabled for now, GMS returns the `NOT_FOUND` error.

HTTP code	404
HTTP message	NOT FOUND

Delete Subscription

This API cancels/terminates a given subscription.

Operation

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

Response

Success

HTTP code	200
HTTP message	OK

Error 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`

HTTP code	404
	.services.notification.SubscriptionNotFoundException"}

Delete subscription for given subscriber

This API cancels/terminates all the subscriptions for a given subscriber.

Operation

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

Returns

Success

HTTP code	200
HTTP message	OK

Error
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 .SubscriberNotFoundException"}

Publish Event

To publish an event or a message, your application can use one of the following workflows:

- Publish the event to the subscriber ID, if the subscription ID is not null.
- Publish the event to the device ID or type or [provider] if the device ID is not null.
- Publish the event to all subscriber IDs which registered to a channel like **service.chat.refresh.<service_ID>** using the tag parameter.

Operation

POST /genesys/{api version}/notification/publish	
Body: JSON event (see the example below.)	
message	The message to publish. For example a string or a JSON map of key/value pairs.
tag	The channel, for example, `service.chat.refresh.<service_ID>`", where the event should be published.
mediaType	The media type. The following values are supported: `MAP`, `STRING`, or `LOCALIZE_STRING`.
language	The language used to publish the message.
localized arguments	A map of key/value pairs for localization.
providerName	The prefix name of the push section defined in the GMS options: `push.provider.<provider name>`. If null, the system uses the default provider configured in the GMS push section.
notificationDetails.deviceId	The device ID (android or ios).
notificationDetails.type	Type of the device where the message is published. The supported values are: `ios, gcm/fcm, httpcb, orscb, comet, java, customhttp, wns`.
notificationDetails.properties	Additional properties for device type such as "debug":"{true/false}"
subscriptionId	Unique subscriber ID where the message should be published.

Example

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

```
POST /genesys/1/notification/publish HTTP/1.1
Host: 172.25.157.93:8080
gms_user: 4f295ba0c0f0a7f1e5ef068bf1d0732e0e70fda7c443081bb3cc5698fa
9a276c
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

Success

HTTP code	200
HTTP message	OK

Errors

In case of an incorrect request body syntax, GMS returns `BAD_REQUEST`.

HTTP code	400
HTTP message	BAD REQUEST

The error code 503 appears in one of the following conditions.

- Failure of HTTP Post action to the specified URL.
- The server did not return 200 status code.
- Network issues.
- Error from APNS and C2DM due to authorization issues or temporary service unavailability for C2DM.

HTTP code	503
HTTP message	SERVICE UNAVAILABLE