Genesys Mobile Services API Reference

Notification API

5/9/2025

Contents

- 1 Notification API
  - 1.1 Overview
  - 1.2 Structures
  - 1.3 Filters and Tags
  - 1.4 APIs

Notification API

Overview

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

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

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

Structures

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

Subscription

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

Subscription Request

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

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

Where:

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

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

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

Subscription Response

If OK:

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

- returns the ID of created subscription.

Event

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

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

Where:

- tag – (mandatory) The message tag.
- message– (optional) Some string. It may contain string representation of ANY data: notification service is message-agnostic; it ALWAYS interprets message as String. If no message is specified, the empty string will be sent to subscribers. The only restriction on **message** format is: it must not crash the JSON parser which attempts to parse the request body. If this happens - a BAD_REQUEST response will be returned.
- mediaType - (optional) "**string**" for a simple string, "**localizestring**" for a string with localized parameter. See Localization File.
- providerName - (optional) This is the name of the provider that this subscription is for. If not specified, the subscription is for the default provider.
- notificationDetails - (optional) If not present, notification is sent to default subscribers. It allows sending the notification to a specific device.
- 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 wildcard '*' after last '.' (or only '*' – denotes subscription to all events), the last char can not be '.'. So, the channels like the following are allowed:

- * - subscription to all channels
- ors.* - subscriptions to all channels starting with ors.
- ors.events.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

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 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

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 .services.notification.SubscriptionNotFoundException"}

Delete subscription for given subscriber

This call cancels/terminates all subscription 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

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

Operation

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

Example

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

```
POST /genesys/1/notification/publish HTTP/1.1
Host: 172.25.157.93:8080
gms_user: 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 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