Genesys Mobile Services Deployment Guide

Mobile Push Notifications

3/31/2026

Mobile Push Notifications

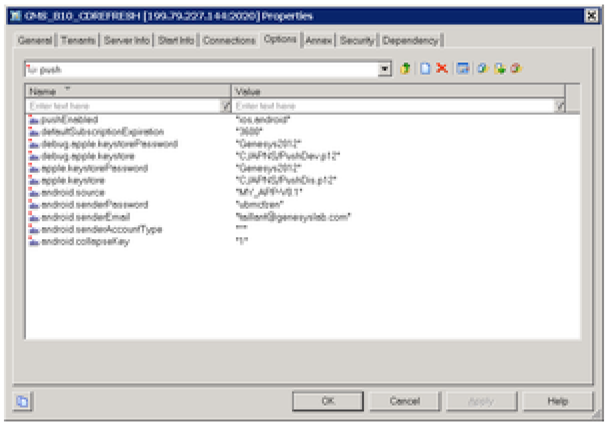
If you are using push notification service, the configuration detailed here should be implemented; however, these steps are not mandatory to complete your GMS installation. See also Push Notification Service for detailed information and examples.

Some services send native push messages to the mobile device. For this to work, both general and device-specific settings need to be configured correctly in the *push* section of your Genesys Mobile Services Application object.

Important

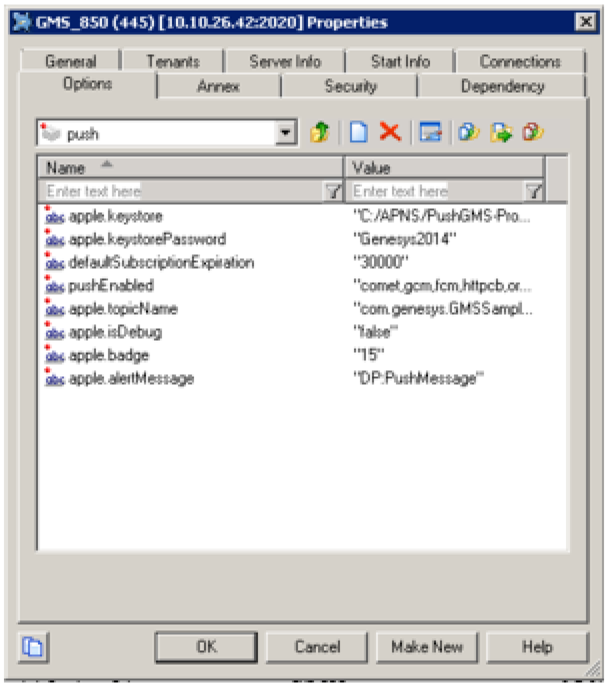
Options set in the *push* section determine how all push notifications are handled by Genesys Mobile Services, regardless of which service is sending the notification.

Note that it is possible to configure this native push notification service for more than one type of device by using a comma-delimited string in the *pushEnabled* option. In this case, be sure to configure the mandatory options for all available device types.



Name	Value
pushEnabled	"ios,android"
defaultSubscriptionExpiration	"3600"
debug.apple.keystorePassword	"Genesys2012"
debug.apple.keystore	"C:/APNS/PushDev.p12"
apple.keystorePassword	"Genesys2012"
apple.keystore	"C:/APNS/PushDis.p12"
android.source	"MY_APP-V0.1"
android.senderPassword	"ubmdten"
android.senderEmail	"taillant@genesyslab.com"
android.senderAccountType	""
android.collapseKey	"1"

GMS multi-device notification.png



Name	Value
apple.keystore	"C:/APNS/PushGMS-Pro...
apple.keystorePassword	"Genesys2014"
defaultSubscriptionExpiration	"30000"
pushEnabled	"comet,gcm,fcm,httpcb,or...
apple.topicName	"com.genesys.GMSSampl...
apple.isDebug	"false"
apple.badge	"15"
apple.alertMessage	"DP:PushMessage"

GMS iOS notification.png

Common Device Settings

- pushEnabled – Device operating system.
- defaultSubscriptionExpiration
- customhttp.url

Mandatory iOS Device Settings

Updated in 8.5.228

To establish a TLS session with APNs, install an Entrust Secure CA root certificate on the provider's server. If the server is running on macOS, this root certificate is already part of the keychain. On other systems, if the certificate is not available, download this certificate from the Entrust SSL Certificates website.

To configure Apple Push Notification, configure the following options in the `[push]` section your GMS application:

- pushEnabled=ios in the Push section
- apple.keystore

- apple.keystorePassword
- apple.topicName, where `apple.topicName` is the bundle ID of your apple app.
  For example, `com.genesys.GMSSampleApp` is the topic name of the `gmssample` app.

```
[push]
apple.keystore=<path of the downloaded p12 certificate>
apple.keystorePassword=<keystorePassword>
apple.topicName=<Bundle ID of your app>
pushEnabled=ios
```

If you do not specify provider information in your API queries or service configuration, GMS uses the above options defined in the push section by default.

You can also define these options for non-default providers in the `[push.provider.{ProviderName}]` section, where `{ProviderName}` is the string that you provide in the user data of your API queries or in your service configuration.

Important

To route push notification calls through a proxy, update the proxy.pac file and specify the file's location in your GMS application, under the [gms] section as shown here:

```
[gms]
http.proxy-auto-config-file=temp/ProxyPac.js
```

Mandatory Android Device Settings

Firebase Cloud Messaging Notification

Introduced in: 8.5.114

Due to recent changes in Google Cloud Messaging, GMS now supports Firebase Cloud Messaging (FCM).

- Refer to the official documentation to Set Up a Firebase Cloud Messaging Client App on Android.
- You will need to retrieve an API Key through the Firebase Console.
- If GMS is deployed behind a firewall, edit your rules to allow the server fcm.googleapis.com and range 5228-5230 for ports, as detailed in the FCM ports and your firewall section of the Firebase Cloud Messaging documentation.

To configure Native Push Notification through Firebase Cloud Messaging, you can either specify an apiKey or create a dedicated section to secure passwords (recommended for production environments) in the push section of your GMS application.

Development

```
[push]
fcm.apiKey=<serverKey>
pushEnabled=fcm
```

Production

```
[push]
fcm=fcmsection
pushEnabled=fcm

[fcmsection]
password=***** (<serverKey>)
```

- If you define these options in the push section, they will be used as default settings if you do not specify provider information in your API queries or service configuration.
- You can also define these options for non-default providers in the `[push.provider.{ProviderName}]` section, where {ProviderName} is the string you need to provide in the user data of your API queries or in your service configuration.

For example, you can configure _provider_name={ProviderName} for a given Callback service, or in a Chat V2 scenario, you could pass this provider name in the `push_notification_provider` user data of the requestChat operation.

GMS allows the following options for the provider-level configuration:

- fcm.apiKey
- debug.fcm.apiKey

Use fcm.title and fcm.body options to extend your configuration by creating an event-level `[push.provider.{ProviderName}.event]` section and then, a specific service name and event:

- [push.provider.{ProviderName}.event.{ServiceName}]
- [push.provider.{ProviderName}.event.{ServiceName}.{EventName}]

Deprecated —Mandatory Android GCM Device Settings

- android.gcm.apiKey — A valid Google API key value (Notifications are sent on behalf of this API key, see http://developer.android.com/guide/google/gcm/gs.html).
- android.gcm.retryNumber — The number of retries in case of service unavailability errors.

For additional detail about these options and the allowed values, see the push Section documentation.

Deprecated — Mandatory Android C2DM Device Settings

Important

Google has deprecated the C2DM Service, and no new users are being accepted. Please use the Firebase Cloud Messaging Service, described above).

- android.senderEmail — Name of a valid mail account. (Notifications are sent on behalf of this account.)
- android.senderPassword — Password of mail account specified in android.senderEmail.
- android.senderAccountType — Specified when initializing C2DM push service.
- android.source — Specified when sending push notifications.
- android.collapseKey — An arbitrary string used to collapse a group of like messages when the device is offline so that only the last message gets sent to the client.

Mandatory customhttp Settings

- customhttp.url — http://xxxx/xx
- pushEnabled — comet,gcm,customhttp,orscb,ios