This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

Chat Server Administration Guide

Push notifications via GMS to HTTP server

4/4/2025

Push notifications via GMS to HTTP server

Starting with version 8.5.311.06, the Chat solution allows you to request push (in other words, unsolicited) notifications through Genesys Mobile Server (GMS) to an HTTP server even when a customer-facing chat web application (Chat Widget) communicates with GMS via "Chat API Version 2". Previously, this was only possible with "Chat API Version 2 with CometD".

To enable this functionality, do the following:

Application	Instructions
GMS	1. Deploy GMS using Cluster Application 2. Configure GMS for Custom HTTP notification 3. Configure GMS with push_notification_include_payload (optional)
Chat Server	1. Add new configuration option flex-push-on-join in the **settings** section with value `true`. This forces Chat Server to acknowledge the push notification subscription during the creation of a chat session. 2. Ensure that option flex-push-enabled is set to `true`, and option flex-push-timeout is set with a larger value (for example, "86400 seconds"). For more information, see Async Requirements. 3. Review the schedule for resending push notifications, when using **GCTI_GMS_PushResend**, defined by the configuration options, flex-push-resend-attempts and flex-push-resend-delay. 4. Adjust, if needed, the value for configuration option flex-push-content. In addition to `session-id` and `user-id`, it is now possible to receive `app-dbid` and `secure-key in push notifications.`
Customer-facing chat web application	1. The web application must supply a set of mandatory key-value pairs in the userdata for the "Request Chat" HTTP method (using a `userData[key-name]` notation): • **GCTI_Chat_PushSubscribe** with the value `true`. This enables push notifications in Chat Server when “Chat API Version 2” is used . • **GCTI_GMS_NodeGroup** with the GMS

Application	Instructions
	cluster name. If the GMS version is 8.5.213.03 or greater, this key-value pair is not required, as it is automatically provided by GMS to Chat Server. • **GCTI_GMS_PushDeviceId** with a unique device ID. This ID is returned in the push notifications as `deviceId`. • **GCTI_GMS_PushDeviceType** with the value `customhttp`. This defines the type of push notification used. • **GCTI_GMS_NotifyRequestor** with value `true`. This forces Chat Server to send push notifications to GMS about the customer's own activity. • **GCTI_GMS_PushIncludePayload** with value `true`. This forces GMS to include the payload (in other words, the chat transcript event content) with a custom-http push notification. Without providing this key-value pair, GMS sends only the `deviceId` (provided in **GCTI_GMS_PushDeviceId**) in the push notification, which can prevent the distribution of sensitive content. When reliable delivery is requested by **GCTI_GMS_PushResend**, this key-value pair must be provided however, in this case, no event-specific payload is provided in the push notification (it only contains some ad hoc data that can be used to send a “Refresh” request). 2. Chat Server provides the ability to request a reliable delivery of push notifications. For that, the web application must additionally supply the **GCTI_GMS_PushResend** key-value pair with value `true` in the userdata. This forces Chat Server to activate the mechanism of resending push notifications according to a schedule defined in the configuration. Chat Server will start resending push notifications if no "refresh" (in other words "pull transcript update") request is being received within the amount of time specified by option flex-push-resend-delay. See below for more information about reliable push notifications delivery. 3. The web application can additionally supply a set of key-value pairs in the userdata: • **GCTI_GMS_PushProvider** Must be provided if you specified the configuration for the non-default provider in

Application	Instructions
	GMS. • **GCTI_GMS_PushDebug** Must be provided if you specified the debug mode for the provider configured in GMS. • **GCTI_GMS_ClientChannel** Must be provided if you want to include the GMS service name in obfuscated secure-key in the push notification.

Additional notes

- It is important to provide adequate throughput of the Web Server which processes the `customhttp` notification. The latency (in other words, the processing time for a single HTTP POST request) must be as low as possible as GMS sends all notifications sequentially. The next request is only sent after a reply from the previous one. For example, if the latency is 5 milliseconds on average, then a single GMS node is able to send 200 notifications per second. Enabling `GCTI_GMS_PushResend` could increase the volume of notifications, so it must be taken into account.
- If push notifications are enabled, Chat Server tries to find the GMS node in the GMS cluster (specified by **GCTI_GMS_NodeGroup**) and to associate that found node for further notifications (until the node is disconnected). Starting with version 8.5.311.06, if no GMS node is available (in other words, registered in Chat Server) in a given cluster, Chat Server selects another GMS cluster to seek for an available GMS node. Otherwise, if no other cluster and/or node is available, Chat Server attempts to find an available node the next time an activity is generated in the chat session or upon chat session restoration in HA mode.
- If reliable delivery of a push notification is not requested by sending **GCTI_GMS_PushResend**, no attempts to resubmit the same push notification will be made in case of a delivery failure between the GMS and HTTP server, and between Chat Server and GMS. The following log messages are logged in the event of this error condition:
  - **In GMS:** `"Dbg 09900 [com.genesyslab.PCT.invoker.default] DC Chat Server Persistent Listener: Event 17 was not (GMS is not running in full mode or incompatible Chat Server version) pushed for delivery to customhttp for device..."`
  - **In Chat Server:** `"Trc 59758 push-flex: could not send notification - no gms node found in group=..."`

Sample configuration for custom HTTP notifications in GMS

```
[chat]
enable_notification_mode=true
push_notification_include_payload=true

[push]
customhttp.url=http://<hostname>:<port>/<path>
pushEnabled=comet,customhttp
```

Important

Ensure that the `[push]` section does not contain the option *customhttp.message*. If it is present, the value of this option overrides the content of push notification.

Reliable push notifications delivery

When requesting reliable delivery for a push notification (in other words, when **GCTI_GMS_PushResend**=`true`):

- All push notifications are of **type**:`PushUrl` and **participantId**: `0` (which is not a valid participant ID).
- No payload is provided in the push notification. Instead, each push notification must be considered a trigger to send a “Refresh” request to GMS in order to obtain the newly published events in the chat session.

The following is the sample JSON which is delivered in the HTTP request for a push notification.

```
{
  "message":{

"secureKey":"G1xBGx9aTUYVBEECD0UZAVwTQEQDFgRZFVJTXEI3QSFFIShAHyVcRUI2GUJXXUEeAikkNSNTJFddQRc=",
    "chatId":"deprecated",
    "nextPosition":17,
    "messages":[
      {
        "from":{
          "nickname":"",
          "participantId":0,
          "type":"Client"
        },
        "index":0,
        "text":"PUSH-NOTIFICATION",
        "type":"PushUrl",
        "utcTime":1568662361000,
        "userData":{
          "notify-attempt":"0",
          "notify-position":"16",
          "secure-key":"c6c9a6d96dc14cef5f94",
          "app-dbid":"131",
          "user-id":"007D5D7FE31F001B",
          "session-id":"00020aEQFW6V0029"
        }
      }
    ],
    "alias":"0",
    "chatEnded":false,
    "userId":"deprecated",
    "statusCode":0,
    "monitored":false
  },
  "deviceId":"a1a23456789123456789"
}
```

Important field descriptions

Field	Description
participantId	Always `0` and must be ignored.
notify-position	Contains the starting position of content not retrieved. It can have a value of `-1` meaning that chat participant has been removed from the chat session.
notify-attempt	Contains the number of attempts to deliver the push notification.
secure-key	Secure key to be used with GMS REST API. The presence depends on flex-push-content.
app-dbid	App DBID (or alias) to be used with GMS REST API. The presence depends on flex-push-content.
user-id	User ID to be used with GMS REST API. The presence depends on flex-push-content.
session-id	Session ID to be used with GMS REST API. The presence depends on flex-push-content.
chatEnded	If the value is `true` it means the chat session is finished.

Warning

Starting with version 8.5.311.06, the `secure-key` for REST API requests is provided in the `userData` based on the value of the configuration option flex-push-content. The `secureKey` provided in `message` must be ignored by the REST API client, and only used for the CometD API.