Push Notifications Documentation

Author: Josh S. Sakweli, Backend Lead Team **Last Updated**: 2026-10-28 **Version**: v1.1 (group calls and group messages; live on staging: Firebase and Apple VoIP are switched on; not yet verified on a real phone)

Base URL: `https://dev.api.nexgate.co/api/v1` (staging) — `http://localhost:8765/api/v1` (local)

Short Description: How NexGate reaches a phone or browser when the app is in the background or closed: what the app registers, what each push looks like, and exactly what Android, iOS and web must do with it. Every push is one of **five contracts**. The server sends facts (who, which chat, what text); **the app decides how it looks**.

Hints:

- Register the device token after **every** login and every time Firebase or PushKit gives you a new one.
- Delete the token on logout, or the next person on that phone gets this person's pushes.
- Read `contract` first. Ignore a contract you don't know, and ignore keys you don't know.
- All values in a push are **strings** (a Firebase rule): `"ringSeconds": "45"`, not `45`.
- Times are ISO-8601 UTC with a `Z` (`2026-10-28T09:14:03Z`).
- All three apps use the **same Firebase project:** `nextgate-484910`. A token from another project never receives anything.

Contents

- Part 0 — Is this the right document?
- Part 1 — How a push travels (the big picture)
- Part 2 — Setup per platform
- Part 3 — Endpoints (register, remove, and the call endpoints a push leads to)
- Part 4 — The five contracts (what arrives, key by key)
- Part 5 — Flows (chat, call, answered elsewhere, read elsewhere, system)
- Part 6 — App open vs closed: what to show
- Part 7 — Android guide
- Part 8 — iOS guide
- Part 9 — Web guide
- Part 10 — What the server already does for you
- Part 11 — Limits, known gaps and who does what

- Integration checklist

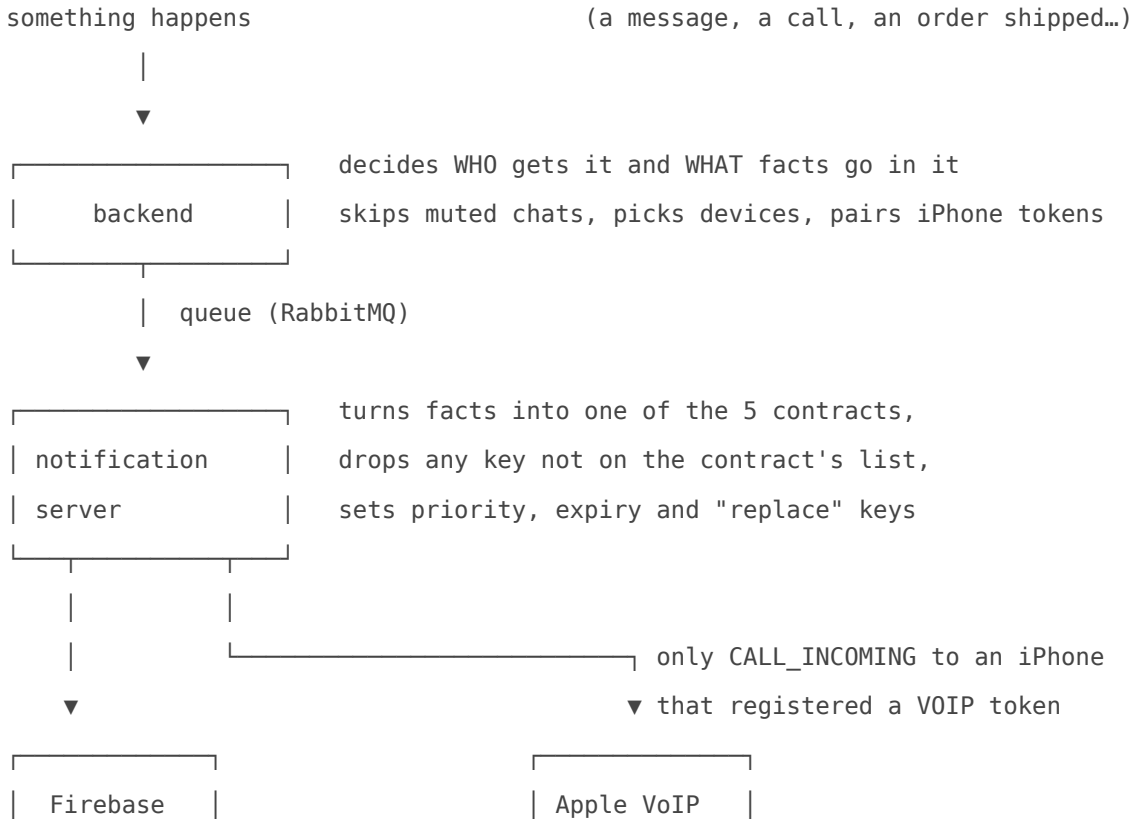
Part 0 — Is this the right document?

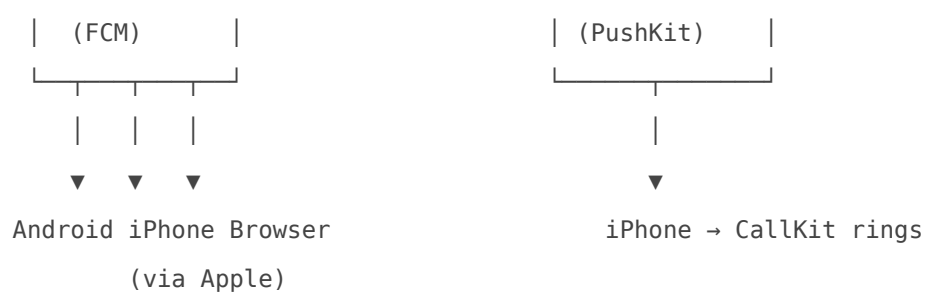
You are building…	Use
Lock-screen / background notifications, ringing when the app is closed	**This document**
The □□ bell list and its badge	`notifications_inapp_bell_api_doc.md`
Chat while the app is open (messages, typing, ticks)	`chatbox-api-doc.md` (XMPP + live events)
The call screen once the call is answered	`calls_meetings_api_doc.md`

The one rule that decides everything:

- App **open** → you already get everything live (XMPP for chat, `GET /api/v1/events` for the rest). A push may arrive too; **the app decides** whether to show anything (Part 6).
- App **in the background or closed** → the push is the only thing that reaches you.

Part 1 — How a push travels





- **Only the backend decides** who gets a push. **Only the notification server** talks to Firebase and Apple. Apps never call Firebase or Apple to send anything.
- One message reaches **every** registered device of the user (phone, tablet, browser), even while the user is connected somewhere else. The app on each device decides what to show.
- If Firebase or Apple says a token is dead, the server deletes it. The app gets it back by registering again (Part 2).

Part 2 — Setup per platform

2.1 What each team gets from the backend lead

Team	Gets	Nothing else needed
Android	`google-services.json` (package `com.nexgate.nexgate`)	
iOS	`GoogleService-Info.plist` (bundle `co.nexgate.ios`)	The Apple push key is already on the server, and the server accepts Xcode, TestFlight and App Store builds
Web	`firebaseConfig` (web app config) and the **VAPID public key**	

2.2 Registration flow (all platforms)

```
            │
            ├── iPhone only: get the PushKit token ──► POST /notifications/devices
            │                                          { token, platform:"IOS", kind:"VOIP",
deviceId }
            ▼
  Firebase / PushKit gives a NEW token later  ──►  POST again (same deviceId — old one is
replaced)
            │
            ▼
  user logs out  ──►  DELETE /notifications/devices/{token}   (every token of this device)
                      then delete the token locally (Firebase deleteToken)
```

`deviceId` is the same device id the app already sends at login (`deviceId` in the auth check request). Keep it stable for the life of the install. It ties an iPhone's two tokens together, so a call rings that iPhone once.

When to register: on every app start while logged in, after every login, and in the "new token" callback. It is safe to call many times with the same token.

Part 3 — Endpoints

1. Register a device token

Purpose: Tell the backend where this device receives pushes.

Endpoint: POST `{base_url}/notifications/devices`

Access Level: Protected (logged-in user)

Authentication: Bearer token

Request Headers:

Header	Type	Required	Description
Authorization	string	Yes	`Bearer <accessToken>`
Content-Type	string	Yes	`application/json`

Request JSON Sample (Android):

```
{
  "token": "fcm-token-from-firebase",
  "platform": "ANDROID",
  "kind": "PUSH",
  "deviceId": "6f1c2a9e-device"
}
```

Request JSON Sample (iPhone — two calls, same `deviceId`):

```
{ "token": "fcm-token-from-firebase", "platform": "IOS", "kind": "PUSH", "deviceId": "6f1c2a9e-device" }
```

```
{ "token": "pushkit-token-hex", "platform": "IOS", "kind": "VOIP", "deviceId": "6f1c2a9e-device" }
```

Request Body Parameters:

Parameter	Type	Required	Description	Validation
`token`	string	Yes	The Firebase registration token, or the PushKit token for `VOIP`	not blank
`platform`	string	Yes	Where the token came from	enum: `ANDROID`, `IOS`, `WEB` (send exactly one of these)
`kind`	string	No	`PUSH` = Firebase token (default). `VOIP` = iPhone PushKit token, used only to ring for calls	enum: `PUSH`, `VOIP`
`deviceId`	string	Yes for `VOIP`, send always	Stable id of this install, the same one sent at login	max 128 characters

Rules:

- Same token sent again → nothing new is created; it is refreshed.
- The same token registered by another account (shared phone) → it **moves** to the new account.
- A new token with the same `deviceId` and `kind` → the old token of that device is deleted. This is why `deviceId` matters: without it, old tokens pile up until Firebase reports them dead.

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Device registered",
  "action_time": "2026-10-28T09:14:03",
  "data": null
}
```


Error Response JSON Sample:


```
{
  "success": false,
  "httpStatus": "UNPROCESSABLE_ENTITY",
  "message": "Validation failed",
  "action_time": "2026-10-28T09:14:03",
  "data": { "voipPaired": "deviceId is required for a VOIP token" }
}
```


Standard Error Types:

- `401 UNAUTHORIZED`: missing or expired access token — register again after the token refresh
- `422 UNPROCESSABLE_ENTITY`: blank `token`, `deviceId` over 128 characters, or `VOIP` without `deviceId`

---

2. Remove a device token

Purpose: Stop pushes to this device for this user (logout).

Endpoint: **DELETE** `{base_url}/notifications/devices/{token}`

Access Level: □□ Protected

Authentication: Bearer token

Path Parameters:

Parameter	Type	Required	Description	Validation

`token`	string	Yes	The token to remove. URL-encode it (Firebase tokens contain `:`)	must belong to the caller, otherwise nothing happens

Success Response JSON Sample:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Device removed",
  "action_time": "2026-10-28T09:14:03",
  "data": null
}
```

Always `200`, even if the token was not found, so logout never gets stuck on it. **Call it before** the access token is thrown away — after logout the request has no user. iPhone: delete both the `PUSH` and the `VOIP` token.

---

3. Endpoints a push leads to

These live in `calls_meetings_api_doc.md` and the bell doc; listed here so you know which one each push needs.

Method	Endpoint	Used when
POST	`/calls/{callId}/end`	User taps **Decline** on a ringing call (the same call also hangs up or cancels)
POST	`/calls/{callId}/join`	Only if you do **not** have `url` + `token` (e.g. learned about the call from the live stream). A push already carries them
GET	`/calls/{callId}`	After a wake-up: is this call still ringing? (catches a cancel you missed)
GET	`/events`	Live stream (SSE) while the app is open: `call.updated`, `notification.added`, …
GET	`/notifications/...`	The bell list — a `SYSTEM` push with an unknown target opens it

---

Part 4 — The five contracts

Every push carries these three keys:

Key	Meaning
`contract`	`CHAT_MESSAGE`, `CALL_INCOMING`, `CALL_ENDED`, `READ_ELSEWHERE` or `SYSTEM`
`v`	Contract version, `"1"` today. New keys can appear without a new version
`type`	The server's notification type, e.g. `CHAT_NEW_MESSAGE`, `ORDER_SHIPPED`

Contract	Shows something?	Urgent	Expires	Replaces
`CHAT_MESSAGE`	yes	yes	—	—
`CALL_INCOMING`	rings	yes	after `ringSeconds` (45)	—
`CALL_ENDED`	**silent**	yes	60 s	an undelivered ring of the same call
`READ_ELSEWHERE`	**silent**	no	24 h	an older read push of the same chat
`SYSTEM`	yes	when `priority` is `HIGH` / `URGENT`	—	—

4.1 `CHAT_MESSAGE` — someone sent a message

```
{
  "contract": "CHAT_MESSAGE", "v": "1", "type": "CHAT_NEW_MESSAGE",
  "messageId": "m-81", "conversationId": "c-12", "conversationType": "GROUP",
  "messageType": "IMAGE", "senderJid": "su_ali@dev.chat.nexgate.co",
  "senderName": "Ali", "senderAvatar": "https://…/small.jpg",
  "groupId": "g-3", "groupName": "Family", "groupAvatar": "https://…/g.jpg",
  "preview": "□□Photo · at the beach", "sentAt": "2026-10-28T09:14:03Z"
}
```

Key	Always?	Meaning
`messageId`	yes	Skip it if you already have this message
`conversationId`	yes	Group notifications by it; open this chat on tap

Key	Always?	Meaning
`conversationType`	yes	`PERSONAL`, `GROUP`, `COMMERCE` or `SECRET`
`sentAt`	yes	When it was sent
`messageType`	not SECRET	`TEXT`, `IMAGE`, `VIDEO`, `AUDIO`, `VOICE_NOTE`, `FILE`, `PRODUCT_REF`, `ORDER_REF`, `OFFER_REF`
`senderJid`	not SECRET	Who sent it
`senderName`	not SECRET	User display name or shop name
`senderAvatar`	when set	Small image link
`preview`	not SECRET	Ready to show: the text, or `□□Photo`, `□□Voice note`, `□□Photo · caption`… Max 120 characters
`groupId`, `groupName`, `groupAvatar`	GROUP only	The group, for title and icon
`replyToYou`	only when `"true"`	Someone replied to **your** message. Show it **even if the chat is muted**

Secret chats carry only `messageId`, `conversationId`, `conversationType`, `sentAt`. Show "New message" and nothing else.

Chat	Title	Line
Personal / shop	`senderName`	`preview`
Group	`groupName`	`senderName: preview`
Secret	NexGate	New message

A muted chat gets no push — except a reply to your own message, which arrives with `replyToYou: "true"`.

Which picture where (group messages carry both):

	Main picture	Small picture / per line
Android	`groupAvatar` (large icon), `groupName` as conversation title	`senderAvatar` on each line (`MessagingStyle` person)
iOS	`groupAvatar` (communication notification, group image)	sender name in the text
Web	`groupAvatar` as `icon`	—
No `groupAvatar`	`senderAvatar`, or the group's initials	—

4.2 `CALL_INCOMING` — ring

```
{
  "contract": "CALL_INCOMING", "v": "1", "type": "CHAT_INCOMING_CALL",
  "callId": "k-55", "conversationId": "c-12", "kind": "VIDEO",
  "callerJid": "su_ali@dev.chat.nexgate.co", "callerName": "Ali", "callerAvatar": "https://…",
  "roomName": "call-k-55", "url": "wss://…livekit…", "token": "eyJ…",
  "ringSeconds": "45", "expiresAt": "2026-10-28T09:14:48Z",
  "groupId": "g-3", "groupName": "Family", "groupAvatar": "https://…/g.jpg",
"participantCount": "5"
}
```

Key	Meaning
`callId`	The call
`conversationId`	The chat the call belongs to
`kind`	`AUDIO` or `VIDEO`
`callerJid`, `callerName`, `callerAvatar`	Who is calling
`groupId`, `groupName`, `groupAvatar`	Group calls only. **Ring as the group**: name + picture of the group, "Ali is calling" underneath
`participantCount`	Everyone invited, caller included (`"2"` on a 1:1)
`roomName`, `url`, `token`	Everything needed to **answer at once** (LiveKit). Token valid 10 minutes
`ringSeconds`	How long to ring
`expiresAt`	**Do not ring** if this time has passed (a late push)

4.3 `CALL_ENDED` — stop ringing (silent)

```
{ "contract": "CALL_ENDED", "v": "1", "type": "CHAT_CALL_ENDED",
  "callId": "k-55", "conversationId": "c-12", "outcome": "ANSWERED_ELSEWHERE",
  "answeredBy": "su_me@dev.chat.nexgate.co" }
```

Key	Meaning
`callId`	Stop ringing for this call, remove its notification
`conversationId`	The chat
`outcome`	`COMPLETED`, `DECLINED`, `MISSED`, `CANCELLED`, `BUSY`, `FAILED`, `ANSWERED_ELSEWHERE`
`answeredBy`	Only with `ANSWERED_ELSEWHERE`: who picked up

Key	Meaning
`callActive`	Group calls only, `"true"`: **your** ringing stopped (you declined, or it rang 45 s), but the call goes on — show "Join" in the chat
`durationSeconds`	Only when the call actually happened

If you are already in this call, ignore it — never hang up on a `CALL_ENDED` with `ANSWERED_ELSEWHERE`. A missed call also gets its own `SYSTEM` push (`CHAT_MISSED_CALL`, "Missed call from Ali").

4.4 `READ_ELSEWHERE` — read on another device (silent)

```
{ "contract": "READ_ELSEWHERE", "v": "1", "type": "CHAT_READ_ELSEWHERE", "conversationId": "c-12" }
```

Remove every notification of this chat from this device.

4.5 `SYSTEM` — everything else (orders, shipping, follows, payments…)

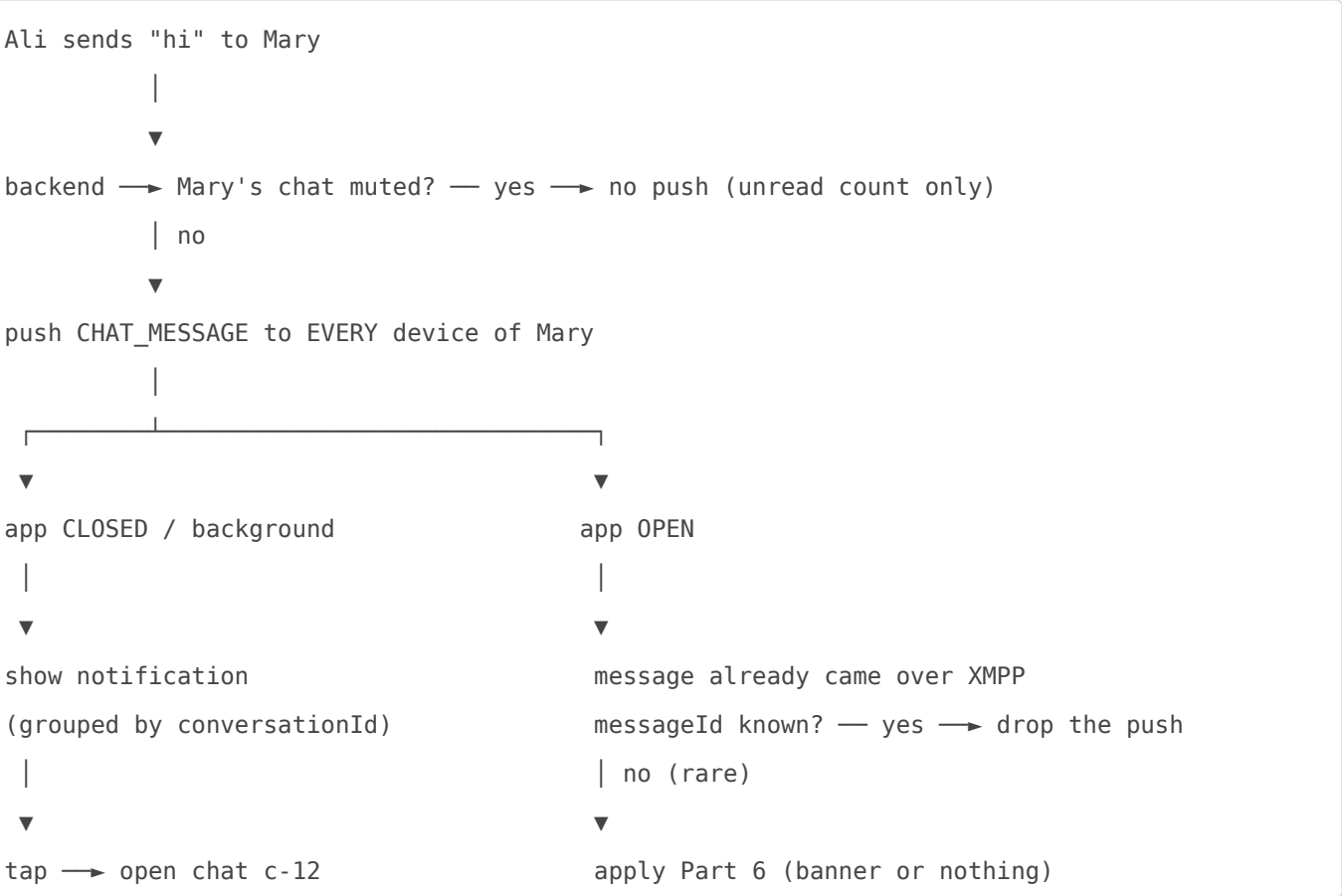
```
{
  "contract": "SYSTEM", "v": "1", "type": "ORDER_SHIPPED",
  "title": "Your order is on the way", "body": "Order #1042 left the shop",
  "targetType": "ORDER", "targetId": "o-1042", "imageUrl": "https://…",
  "priority": "HIGH", "sentAt": "2026-10-28T09:14:03Z"
}
```

Key	Meaning
`title`	Ready-made title
`body`	Second line (may be empty — then title only)
`targetType`	What a tap opens — same values as the bell doc, Part 3
`targetId`	The id to open (missing for targets that need none)
`imageUrl`	A picture, when the type has one
`priority`	`LOW`, `NORMAL`, `HIGH`, `URGENT` — HIGH/URGENT get sound + heads-up
`sentAt`	When it was sent

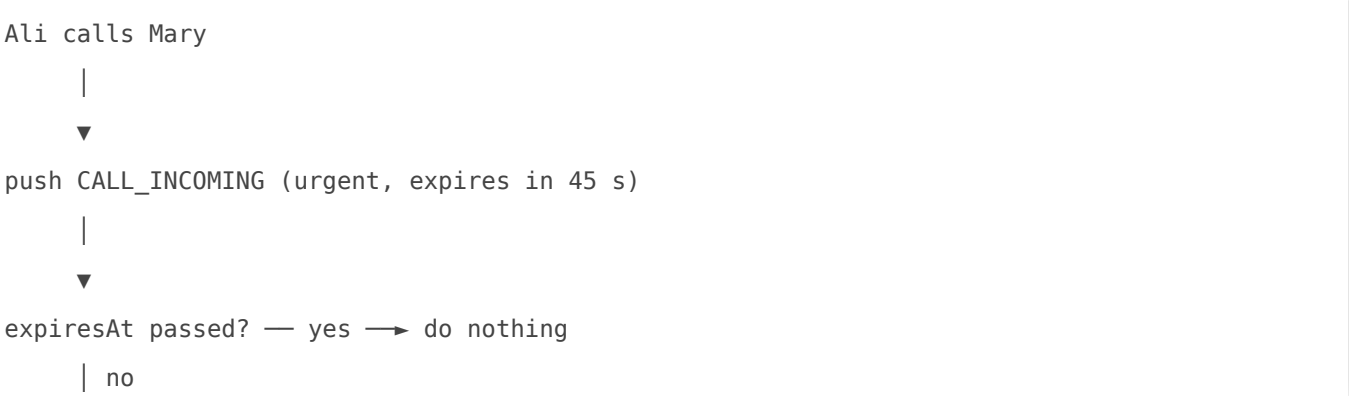
Unknown `targetType` → open the bell list. Not every notification type sends a push; that is decided per type on the server. The app just handles what arrives.

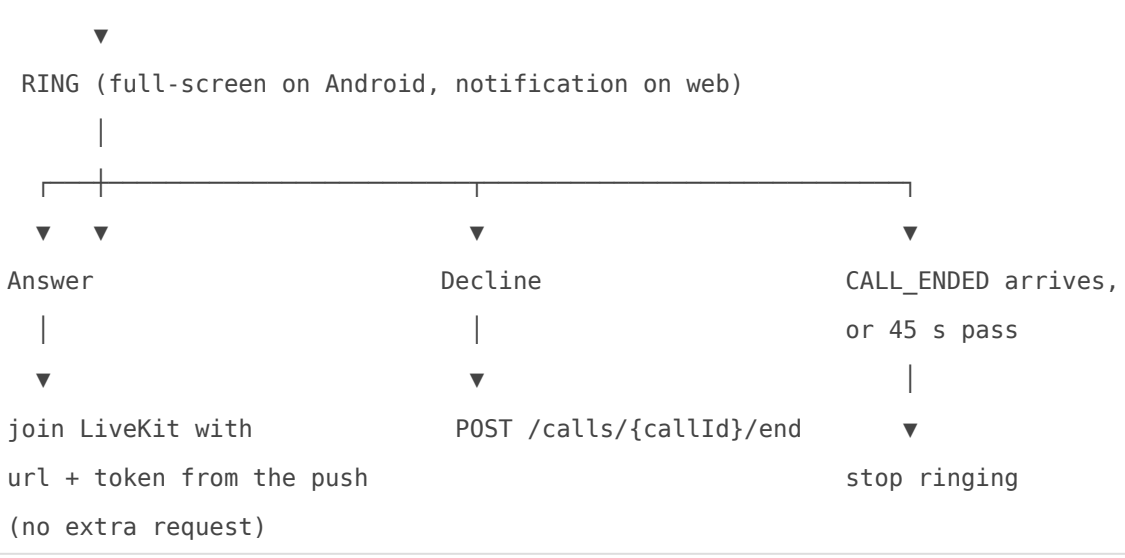
Part 5 — Flows

5.1 Chat message

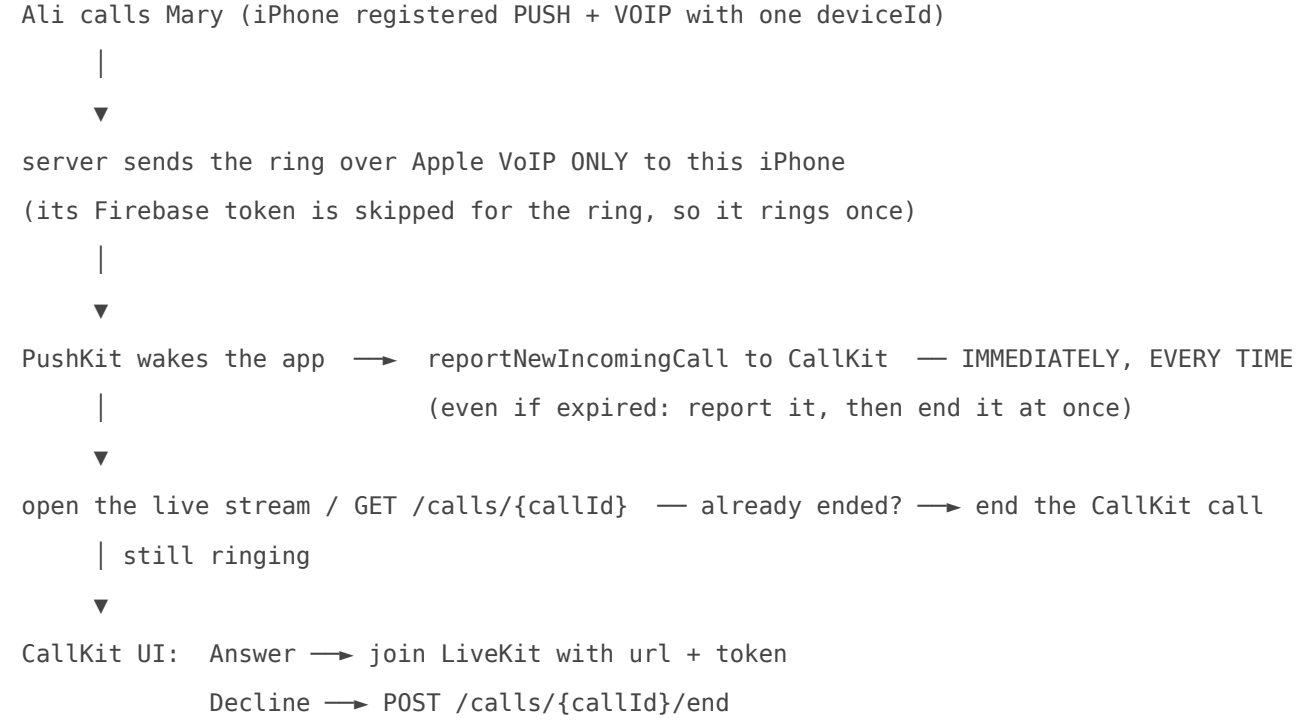


5.2 Incoming call — Android and web



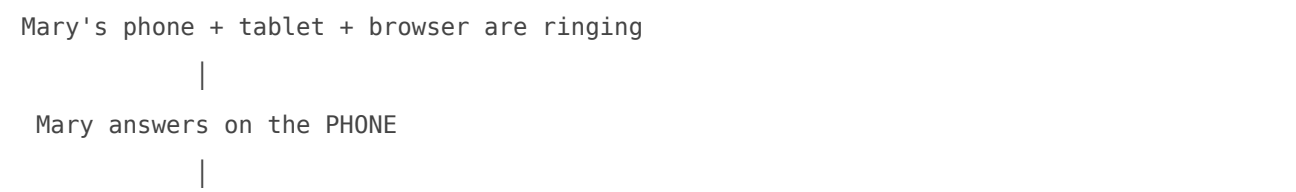


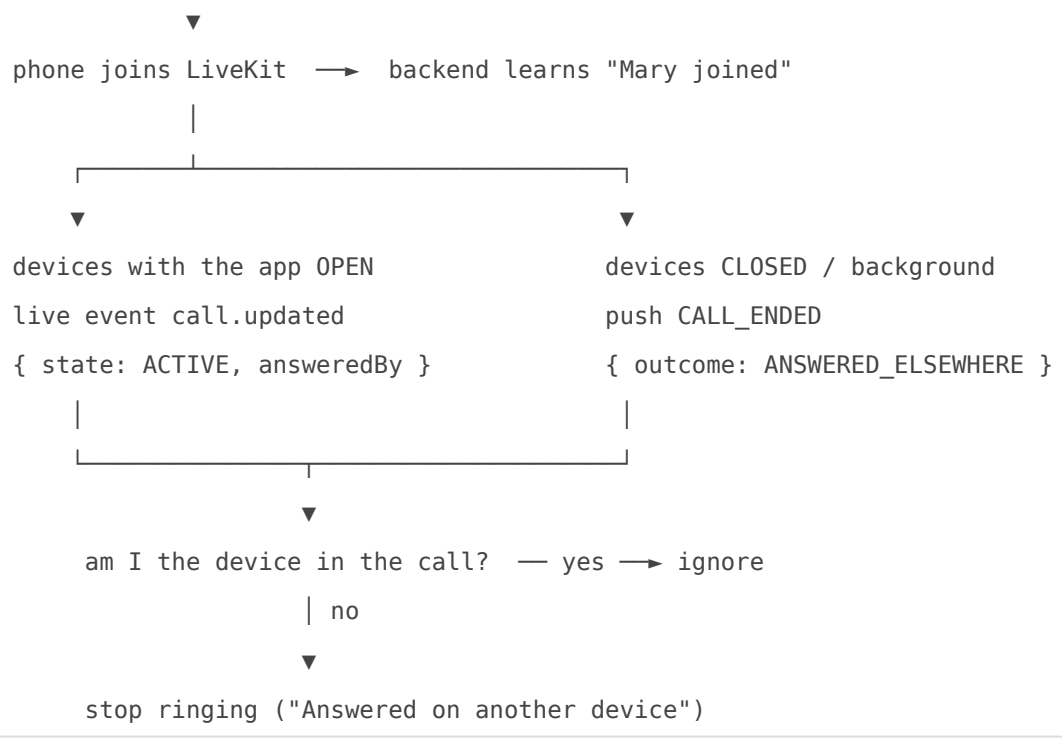
5.3 Incoming call — iPhone



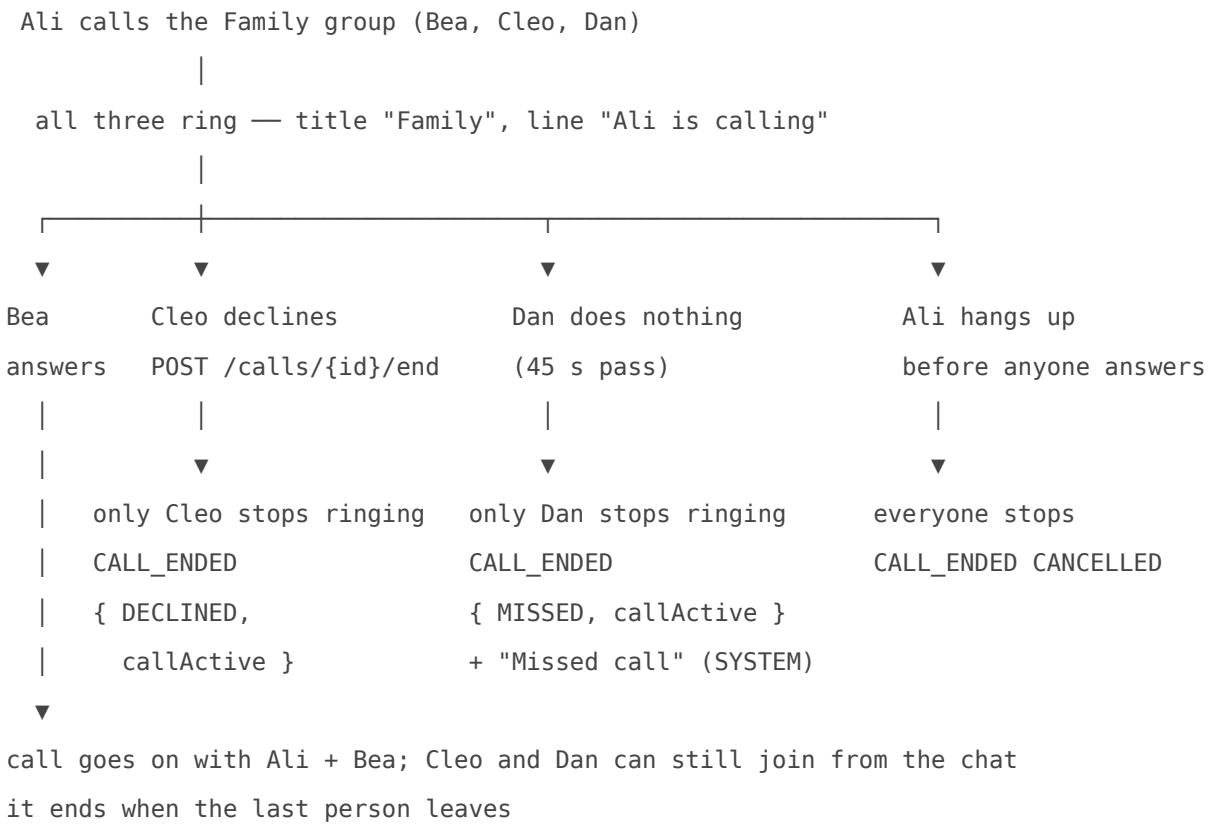
An iPhone **without** a `VOIP` token gets a normal Firebase notification ("Ali is calling") instead — no ringing.

5.4 Answered on another device





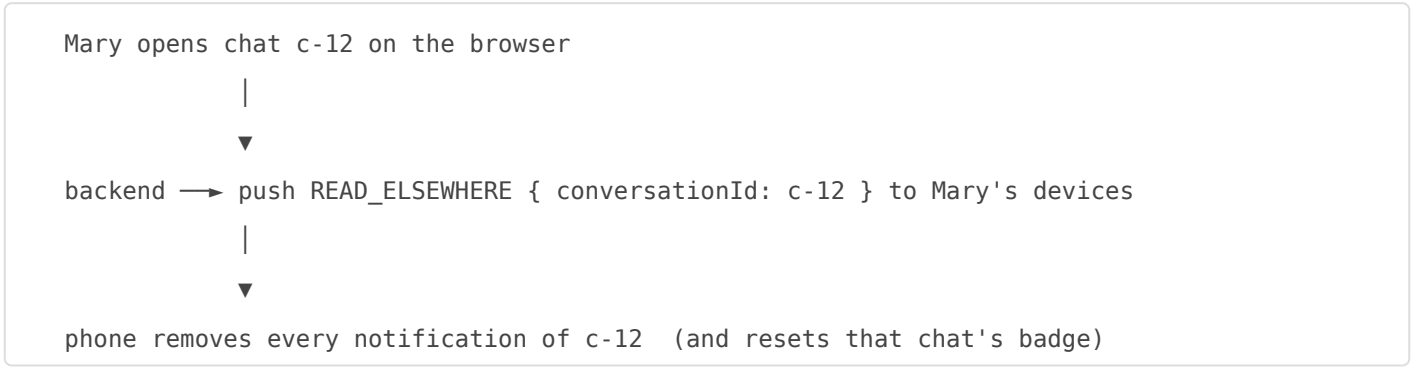
5.4b Group call



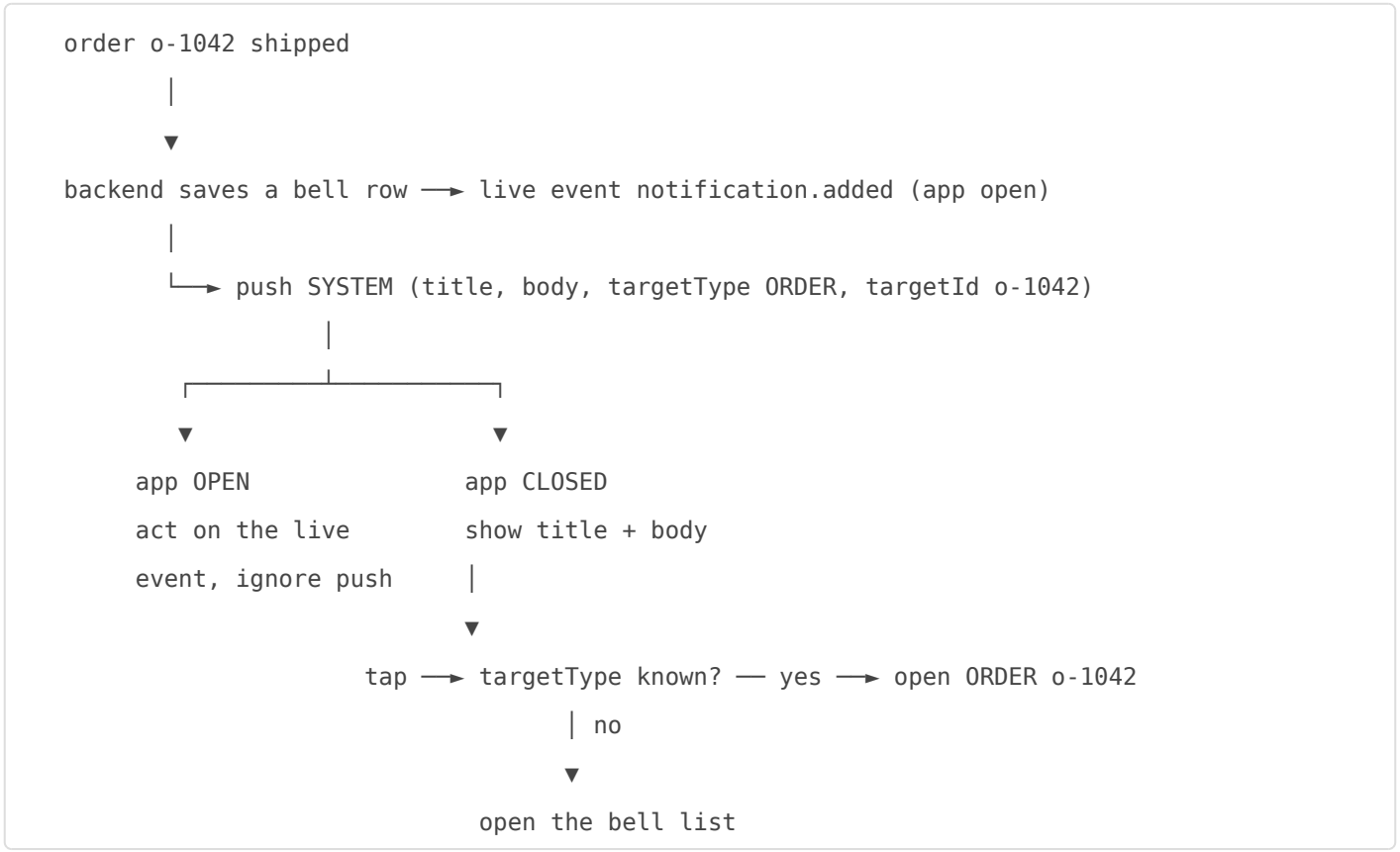
- Declining a group call **never** ends it for the others. If everyone declines before anyone answers, the call ends (`DECLINED`) for the caller too.
- The same `POST /calls/{callId}/end` is Decline (before you joined) and Leave (after you joined).

- Anyone who never picked up and did not decline gets "Missed call" — also when the call ended before their 45 s.

5.5 Read on another device



5.6 System notification



Part 6 — App open vs closed: what to show

The server cannot know which screen you are on, so the app decides.

Where the user is	New chat message	System notification
Anywhere else in the app	In-app banner + sound; tap opens the chat	Banner if `priority` HIGH/URGENT, otherwise only the bell badge
Chat list	Banner + sound; the chat moves to the top	Same as above
Inside chat **A**, message from **A**	**No banner** — it appears in the thread (soft sound)	Same as above
Inside chat **A**, message from **B**	Banner + sound	Same as above
On the exact screen it is about (that order)	—	No banner; refresh the screen
Muted chat	Unread count only — but a push with `replyToYou: "true"` is shown	—
Incoming call	**Full-screen ringing, always**	—

Never show the same thing twice. When the app is open you get each message over XMPP **and** as a push. Keep the `messageId`s you have shown (the last few hundred are enough); drop a push whose `messageId` you know. For system notifications, when the app is open act on the live `notification.added` event and ignore the push.

Background / closed:

Contract	Android	iOS	Web
`CHAT_MESSAGE`	App builds the notification	iOS shows title + line; extension adds the photo	Service worker builds it
`CALL_INCOMING`	Full-screen ringing	CallKit (VoIP push)	Notification only — a closed browser cannot ring
`CALL_ENDED`	Stop ringing, remove it	Silent; remove the call notification / end CallKit call	Close the notification
`READ_ELSEWHERE`	Remove the chat's notifications	Silent; remove them	Close them
`SYSTEM`	App builds it	iOS shows `title` + `body`	Service worker builds it

---

Part 7 — Android guide

What arrives: every push is **data-only**. Android never draws anything by itself — `onMessageReceived` is called for every push, app open or closed, and the app builds every notification.

7.1 Setup

- `google-services.json` from project `nextgate-484910`, Firebase Messaging SDK.
- Android 13+: ask `POST_NOTIFICATIONS` at a good moment (after login, with a one-line reason), not on first launch.
- Create **notification channels** once at start:

Channel	Importance	Used for
`chats`	HIGH (sound + heads-up)	`CHAT_MESSAGE`
`calls`	HIGH, ringtone, full-screen	`CALL_INCOMING`
`alerts`	HIGH	`SYSTEM` with `HIGH` / `URGENT`
`general`	DEFAULT	other `SYSTEM`

7.2 Handling (`FirebaseMessagingService`)

```
onMessageReceived(data)
   switch data["contract"]
     CHAT_MESSAGE   → skip if messageId known; skip if this chat is open (Part 6);
                      MessagingStyle notification, id = conversationId hash,
                      add the line to the existing one; person icon = senderAvatar
     CALL_INCOMING  → skip if expiresAt passed; CallStyle notification with
                      full-screen intent (or ConnectionService); ring until
                      ringSeconds / CALL_ENDED; keep url + token for Answer
     CALL_ENDED     → cancel the call notification for callId; stop ringtone
                      (ignore if this device is IN the call)
     READ_ELSEWHERE → cancel all notifications of conversationId
     SYSTEM         → channel by priority; title + body; imageUrl as BigPicture;
                      tap intent carries targetType + targetId
     else           → ignore
onNewToken(token)   → POST /notifications/devices
```

7.3 Best practice

- Finish `onMessageReceived` fast (under ~10 s). Load avatars with a short timeout and show the notification without the picture if it is slow.
- Urgent pushes (chat, call) are delivered high-priority. **Always show a notification for them** — Android lowers the priority of apps that receive urgent pushes and show nothing.
- Calls: Android 14+ restricts full-screen intents to calling apps; declare `USE_FULL_SCREEN_INTENT` and check `canUseFullScreenIntent()`, fall back to a heads-up

notification. Use a `phoneCall` foreground service during the call.
- Some phones (Xiaomi, Oppo, Huawei…) kill background apps. Show a one-time hint to allow "autostart" / remove battery optimisation if calls are missed.

Part 8 — iOS guide

What arrives:

- `CHAT_MESSAGE`, `SYSTEM`, and `CALL_INCOMING` without VoIP → a normal **alert** (title + body already set by the server, sound, `mutable-content: 1`, `thread-id` = chat or target type). The data keys sit next to `aps` in `userInfo`.
- `CALL_ENDED`, `READ_ELSEWHERE` → **silent** background pushes (`content-available: 1`).
- `CALL_INCOMING` to an iPhone with a `VOIP` token → **PushKit** push; the data keys are top-level in the payload.

8.1 Setup

- `GoogleService-Info.plist` from `nextgate-484910`; capabilities: **Push Notifications**, **Background Modes → Remote notifications + Voice over IP**.
- Ask notification permission after login with a reason, register for remote notifications, pass the APNs token to Firebase, then register the **Firebase** token with us (`kind: PUSH`).
- `PKPushRegistry` with type `.voIP` → register that token with us (`kind: VOIP`, same `deviceId`).
- A **Notification Service Extension** (for avatars and pictures).

8.2 Handling

```
Notification Service Extension (alert pushes)
   CHAT_MESSAGE → download senderAvatar / groupAvatar (short timeout), attach;
                  optional: communication notification (INSendMessageIntent) to show the face
   SYSTEM       → download imageUrl, attach
   always call the contentHandler before the time limit (~30 s)

userNotificationCenter(willPresent:)   app OPEN → apply Part 6; return [] to hide
userNotificationCenter(didReceive:)    tap → CHAT_MESSAGE open conversationId;
                                             SYSTEM open targetType/targetId, unknown → bell
application(didReceiveRemoteNotification:) silent pushes
   CALL_ENDED     → end the CallKit call / remove the notification for callId
                    (ignore if this device is in the call)
```

```
   READ_ELSEWHERE → remove delivered notifications with thread-id = conversationId


pushRegistry(didReceiveIncomingPushWith:) VoIP
   → CXProvider.reportNewIncomingCall(callId, callerName, hasVideo: kind == VIDEO)
     BEFORE the completion handler, EVERY TIME
   → expired or already ended? report it, then end it straight away
   → Answer: join LiveKit with url + token     Decline: POST /calls/{callId}/end
```

8.3 Best practice

- **Report every VoIP push to CallKit.** If the app receives one and shows no call, iOS kills the app and, after a few times, stops delivering VoIP pushes to it.
- **Silent pushes are best-effort.** iOS may delay or drop them (low battery, Low Power Mode, an app the user swiped away). So never rely on them alone: after any wake and on every app open, check the call (`GET /calls/{callId}`) and refresh unread counts. For "answered elsewhere" while CallKit rings, also listen to the live stream once the app is awake.
- Xcode builds use Apple's sandbox, TestFlight / App Store use production. Nothing to configure — the server tries both.
- The simulator cannot receive VoIP pushes; calls must be tested on a real iPhone.

Part 9 — Web guide

What arrives: data-only, like Android. Nothing shows unless the site or the service worker shows it.

9.1 Setup

- Firebase JS SDK with the `firebaseConfig` we give you.
- A service worker file `/firebase-messaging-sw.js` at the site root.
- Ask permission **only after a user click** ("Turn on notifications"), never on page load — browsers block or hide the prompt otherwise, and a "Block" is hard to undo.
- `getToken(messaging, { vapidKey })` → `POST /notifications/devices` with `platform: "WEB"`, `kind: "PUSH"` and a `deviceId` kept in local storage for this browser.

9.2 Handling

```
page open         onMessage(payload)         → apply Part 6 (in-page banner, no system
notification)
```

```
page closed /     service worker
background        onBackgroundMessage(payload)
                    CHAT_MESSAGE   → showNotification(title, { body, icon: senderAvatar,
                                       tag: conversationId, renotify: true, data })
                    CALL_INCOMING  → showNotification("Ali is calling", { tag: callId,
                                       requireInteraction: true, data })   (no ringing)
                    CALL_ENDED     → close notifications with tag = callId
                    READ_ELSEWHERE → close notifications with tag = conversationId
                    SYSTEM         → showNotification(title, { body, image: imageUrl, data })
notificationclick → focus an open tab or open the URL for the chat / target
```

9.3 Best practice

- Using `tag` = `conversationId` makes a new message **replace** the chat's notification instead of stacking.
- Safari on iPhone only receives web push when the site is **added to the home screen** (iOS 16.4+).
- A token can change when the user clears site data; register on every load while logged in.

Part 10 — What the server already does for you

You do not need to…	Because the server…
Skip muted chats	never sends a push for a muted chat (only a reply to you gets through, marked `replyToYou`)
Stop a group ring	stops each person's ring on its own: decline, 45 s, or the call ending
Send a "cancel" for a late ring	lets the ring expire after `ringSeconds`, and a `CALL_ENDED` replaces a ring not yet delivered
Ring an iPhone twice	sends the ring over VoIP only, skipping that device's Firebase token
Hide secret-chat text	sends no name and no text for secret chats
Collapse many "read" pushes	collapses them per chat
Clean up dead tokens	deletes a token Firebase or Apple rejects (app re-registers on next start)
Format previews	sends `preview` ready to show (`□□Photo · caption`, max 120 characters)

You do not need to…	Because the server…
Translate titles	`SYSTEM` `title` / `body` come from our templates

Part 11 — Limits, known gaps and who does what

Item	Status
Staging	Firebase + Apple VoIP **on** since 2026-10-28
Production	**Off** until this release ships
Real-device test	**Not done yet.** Each app registers a token on staging, then we send a message and make a call while the backend lead watches the server log
Silent pushes on iOS	Best-effort by Apple's design — always re-check on wake (8.3)
Web ringing	Not possible; a closed browser can only show a notification

Who	Does
Backend	Sends the five contracts; keys, Firebase project, server logs during testing
Android	Channels, `onMessageReceived` for all five, full-screen call, token register/remove
iOS	Service Extension, PushKit + CallKit, silent push handling, two tokens with one `deviceId`
Web	Service worker, permission on click, token register/remove

Something not behaving as written here? Send the backend lead the time, the user and the device — the server log shows every push it sent and what Firebase or Apple answered.

Integration Checklist

- ☐ Same Firebase project `nextgate-484910` in the app
- ☐ Register the token after login, on every start and on refresh — always with `deviceId`
- ☐ Delete every token of the device on logout, **before** dropping the access token
- ☐ iPhone: also register the PushKit token as `kind: VOIP`; report every VoIP push to CallKit
- ☐ Switch on `contract`; ignore unknown contracts and keys

☐ Chat: group by `conversationId`, skip known `messageId`s, secret = "New message" only

☐ Group chat: group picture + name as title, sender on each line; `replyToYou` shows even when muted

☐ Group call: ring as `groupName` / `groupAvatar`; `CALL_ENDED` with `callActive` = stop ringing, show "Join"

☐ Apply the Part 6 table when the app is open

☐ Calls: skip if `expiresAt` passed; ring until `ringSeconds` or `CALL_ENDED`; Answer uses `url` + `token`; Decline = `POST /calls/{callId}/end`

☐ `ANSWERED_ELSEWHERE` / `call.updated` ACTIVE: stop ringing unless this device is in the call

☐ `READ_ELSEWHERE` clears that chat's notifications

☐ `SYSTEM`: tap opens `targetType` + `targetId`; unknown → bell list

☐ After any wake and on app open: re-check calls and unread counts (don't trust silent pushes alone)

Revision #2
Created 28 October 2026 10:09:23 by Admin Qbit
Updated 28 October 2026 11:55:27 by Admin Qbit