



API Reference

v0.1

Send Dings (push notifications), manage channels and subscribers, schedule broadcasts, and track engagement. One base URL, JSON in and out.

Base URL <https://go.ding.fyi>

Authentication

Nearly every endpoint requires a **Firestore ID token** sent as a Bearer token. Sign in with a Firebase Auth account for this project, grab the ID token from the client SDK with `user.getIdToken()`, and pass it on every request:

Authorization: Bearer <FIREBASE_ID_TOKEN>



ID tokens expire after **1 hour** — refresh them with the Firebase SDK rather than caching one long-term. Requests without a valid token get 401 `{"error":"Authentication required"}`. CORS is open (*), so you can call the API straight from a browser app. Public (no-token) endpoints: `/health`, `GET /notifications/:id`, `/join/:id`, and the billing result pages.



No account yet? Ask the DING team to provision you a Firebase Auth user, then sign in at <https://go.ding.fyi/auth> — the web console stores a fresh ID token in `localStorage` (`firebaseIdToken`) you can lift for quick curl testing.

Sending a Ding

A Ding is stored in Firestore and delivered over FCM to a channel's subscribers. Delivery is subject to channel ownership and your plan's recipient cap.

POST**/broadcast** **Channel owner**

Send a Ding to all subscribers of a channel. For any channel other than the system `ding` channel, only the channel owner may broadcast.

FIELD	TYPE	DESCRIPTION
<code>topic optional</code>	string	Channel ID to broadcast to. Defaults to <code>ding</code>
<code>title optional</code>	string	Push title. Defaults to <code>"Broadcast Alert"</code>
<code>message optional</code>	string	Push body. Basic HTML is kept only if it contains a link, otherwise stripped to plain text
<code>imageUrl optional</code>	string	Publicly reachable image URL shown with the notification
<code>attachments optional</code>	string[]	Array of attachment URLs
<code>sponsor optional</code>	object	<code>{ "link": "...", "text": "Sponsored", "altText": "" }</code> — applied only when <code>link</code> is present; clicks are tracked per user

EXAMPLE

```
curl -X POST https://go.ding.fyi/broadcast \
-H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json" \
-d '{"topic":"my-channel","title":"Doors open!","message":"Come on in 🚪"}'
```

RESPONSE 200

```
{
  "success": true,
  "notification": { "id": "...", "title": "Doors open!", "channel": "my-channel", "openCount": 0 },
  "broadcast": { "success": true, "messageId": "projects/.../messages/..." }
}
```



Plan-based recipient cap (per Ding, not per month): Free = 100 recipients, Starter = 1,000, Unlimited = no cap. If your channel has more subscribers than your cap, only the first N receive the push and the response's broadcast object switches to `{"softLimited": true, "sent": ..., "failed": ..., "skipped": ...}`. The Ding is always saved to inboxes regardless of push outcome. Errors: 403 if you don't own the channel, 404 if it doesn't exist.

POST /add

 **Auth**

Legacy: create a Ding on the system `ding` channel and push it to everyone. Accepts `title`, `message`, `imageUrl`, `attachments` (defaults: `"Proximity Alert"` / `"Wristband outside of designated area."`). Prefer `/broadcast`.

Scheduled Dings

Queue a Ding for future delivery. Scheduled items live in a `pending` state until a server-side scheduler fires them, after which they appear under sent items.

POST /schedule-notification

 **Channel owner**

Schedule a future broadcast. Non-`ding` channels must exist and be owned by you.

FIELD	TYPE	DESCRIPTION
message required	string	Ding body
scheduledAt required	string	ISO timestamp; must parse to a future time
title optional	string	Defaults to <code>"Broadcast Alert"</code>
channel optional	string	Channel ID, defaults to <code>ding</code>
timeType optional	string	<code>"local"</code> to fire in each subscriber's timezone, anything else = <code>"global"</code>
imageUrl / attachments optional	—	Same as <code>/broadcast</code>

RESPONSE 200

```
{ "success": true, "scheduledNotification": { "id": "...", "status": "pending", "scheduledAt": "...
```

GET /scheduled-notifications

 **Auth**

List *your* pending scheduled Dings. Returns `{"success": true, "scheduled": [...]}`.

GET `/api/sent-scheduled-notifications` **Auth**

List your scheduled Dings that have already fired (`status: "sent"`) — useful for syncing a Sent box. Returns `{"success": true, "sent": [...]}`.

POST `/update-scheduled-notification` **Auth**

Reschedule a pending Ding you created. Body: `{"id": "...", "scheduledAt": "...", "timeType?": "local"|"global"}` — only the schedule is mutable. `403` if you're not the creator.

POST `/delete-scheduled-notification` **Auth**

Cancel a scheduled Ding you created. Body: `{"id": "..."}`.

Notifications

Reading and managing delivered Dings. List responses come from a server-side cache with a 5-minute TTL (writes refresh it immediately).

GET `/notifications` **Auth**

List notifications, newest first. Targeted notifications addressed to another user are filtered out. Items include engagement fields: `openCount`, `readCount`, `sponsorClickCount`, and `reactions` (map of emoji → user IDs).

EXAMPLE

```
curl https://go.ding.fyi/notifications -H "Authorization: Bearer $TOKEN"
```

GET `/notifications/:id` **Public**

Fetch one notification. Content-negotiated: API clients get JSON (enriched with `channelName` and `channelAvatar`); browsers (`Accept: text/html`) get a shareable landing page that deep-links into the app.

POST **/update-note**

 **Auth**

Attach free-text notes to a notification. Body: `{"id": "...", "notes": "..."} .`

POST **/delete**

 **Auth**

Delete one notification (and its read-tracking data). Body: `{"id": "..."} .` Only the notification's creator or the channel owner may delete — otherwise `403` .

Engagement & Stats

Per-user read tracking, emoji reactions, sponsor-click analytics, and sender-facing stats.

POST **/api/notifications/:id/open**

 **Auth**

Mark a notification opened/read by the calling user. Idempotent per user — only the first open increments the counters. Optional body: `{"source": "app"} .` Returns the updated `openCount` / `readCount` .

POST **/api/notifications/:id/react**

 **Auth**

Toggle an emoji reaction. Body: `{"emoji": "👍"} .` Calling again with the same emoji removes your reaction. Returns the full `reactions` map.

POST **/api/notifications/:id/sponsor-click**

 **Auth**

Record a sponsor-link click. Optional body: `{"link": "...", "sponsorText": "...", "source": "app"} .` (falls back to the notification's sponsor data). Not idempotent — every call counts a click.

GET **/api/notifications/:id/stats**

 **Sender/owner**

Full analytics for a Ding: counts, per-user opens, sponsor-click users, and the 50 most recent sponsor clicks. Only the sender or the channel owner may view — otherwise `403` .

GET `/api/notifications/read-states?ids=id1,id2`

 **Auth**

Bulk read-state check: pass up to **50** comma-separated IDs, get back `{"success": true, "readIds": [...]}` — the subset the calling user has opened.

Channels

Channels group subscribers. Each channel gets a shareable join URL and QR code at creation. Channel IDs are the lowercased name without the leading `#`.

GET `/api/channels`

 **Auth**

List channels. Private channels are hidden unless you own them. Query params:

PARAM	DESCRIPTION
<code>subscribed=true</code>	Only channels the calling user is subscribed to
<code>channelId=<id></code>	Look up a single channel by ID (case-insensitive)

POST `/api/channels`

 **Auth**

Create a channel you own; you're auto-subscribed as owner. Returns `201` with the full channel including `joinUrl` and a `qrCode` data URL, or `409` if the name is taken.

FIELD	TYPE	DESCRIPTION
<code>name</code> required	string	Letters, numbers, dashes only. Leading <code>#</code> optional
<code>description</code> optional	string	Shown on the join page
<code>avatarUrl</code> optional	string	Channel avatar image URL
<code>isPrivate</code> optional	boolean	Hide from discovery (default <code>false</code>)
<code>socialLinks</code> optional	object	Key → URL map

DELETE `/api/channels/:id`

 **Owner only**

Permanently delete a channel you own, plus **all of its notifications and scheduled Dings**. `403` if you're not the owner. No undo.

POST `/api/channels/:id/subscribe`

 **Auth**

Subscribe the calling user; also subscribes their registered FCM tokens to the channel's push topic. Optional body: `{"timezone": "America/New_York"}` (used if your profile has no timezone). Idempotent.

POST `/api/channels/:id/unsubscribe`

 **Auth**

Remove the calling user and detach their device tokens from the channel's push topic. No body required.

GET `/api/users/me/subscriptions`

 **Auth**

Compact list of channels you're subscribed to — `{"channels": [{"id", "name", "avatarUrl"}]}`. Handy for re-subscribing FCM topics on a new device. (Not billing — see [Plans & Billing](#).)

GET `/join/:id`

 **Public**

Shareable HTML landing page for a channel with an "Open in app" deep link. This is the URL encoded in each channel's QR code — meant for humans, not API clients.

Device Tokens

Mobile clients register their FCM token so pushes reach the device.

POST `/register-token`

 **Auth**

Register an FCM device token for the calling user and subscribe it to the topics of every channel they already follow. Body: `{"token": "..."}` . Duplicates are ignored.

POST /deregister-token

 **Auth**

Remove a token (e.g. on sign-out) and unsubscribe it from all channel topics. Body: `{"token": "..."}.`

Profile & Account

GET /api/profile

 **Auth**

Your full profile, including plan and subscription fields. `404` with `{"success": false, "profile": null}` if no profile exists yet.

POST /api/profile

 **Auth**

Create or update your profile. First-time creation also triggers a welcome Ding, a welcome email, and auto-subscription to the `#ding` channel.

FIELD	TYPE	DESCRIPTION
firstName required	string	First name
lastName required	string	Last name
age required	string	One of <code>12-17</code> , <code>18-24</code> , <code>25-34</code> , <code>35-44</code> , <code>45-54</code> , <code>55-64</code> , <code>65+</code>
location required	string	Free-text location
timezone optional	string	IANA timezone, e.g. <code>America/New_York</code>

GET /api/profiles/:uid

 **Auth**

Limited public profile of any user: name, age range, location, timezone. No email.

PATCH /api/timezone

 **Auth**

Update just your timezone. Body: `{"timezone": "America/New_York"}`. Propagates to your subscriber entries across channels in the background.

GET **/api/my-data**

 **Auth**

GDPR-style data export: your profile, subscription summary, owned channels, and usage stats in one JSON payload.

DELETE **/api/users/me**

 **Irreversible**

Permanently delete your account and **all** associated data — owned channels, notifications, scheduled Dings, subscription records, and the Firebase Auth user. If the response includes `"clientAuthDeleteRequired": true`, the client must finish deleting the Auth user itself.

POST **/api/support**

 **Auth**

File a support ticket. Body: `{"name", "email", "subject", "message", "category"}`. Returns `{"success": true, "ticketId": "..."}.`

GET **/health**

 **Public**

Liveness check. Returns `{"status": "healthy", "timestamp": "..."}.`

Plans & Billing

Paid plans raise the per-Ding recipient cap (Free 100 → Starter 1,000 → Unlimited ∞). Purchases go through Apple/Google in-app purchase (synced via `/api/subscriptions/sync`) or Stripe on the web.

GET **/api/subscriptions/plans**

 **Auth**

The plan catalog: keys, product IDs, tiers, intervals, prices, entitlements, and whether Stripe direct purchase is available per plan.

GET **/api/subscriptions/me**

 **Auth**

Your effective plan, subscription status, and entitlements. Query params: `includeTransactions` (default true; pass `false` to skip) and `limit` (1–50, default 10) for recent purchase transactions.

POST `/api/subscriptions/stripe/checkout`

 **Auth**

Create a Stripe Checkout session for a plan. Body identifies the plan (e.g. `{"planKey": "starter_monthly"}`). Redirect the user to the returned `url`; plan activation happens via webhook after payment.

POST `/api/subscriptions/stripe/portal`

 **Auth**

Create a Stripe Billing Portal session (manage/cancel a web subscription). Returns `{"url": "..."}`. Requires an existing Stripe customer.

GET `/api/subscriptions/stripe/status`

 **Auth**

Server Stripe configuration summary — use to detect test vs. live mode and whether web checkout is enabled.

POST `/api/subscriptions/sync`

 **Auth**

Report a native in-app purchase or restore for server-side verification. Requires `platform` (`"ios"` | `"android"`) plus plan and store receipt/token fields. Returns verification status, updated entitlements, and the recorded transaction. `402` if verification fails.

 **Internal endpoints** you'll see in traffic but should never call directly: `POST /api/webhooks/stripe` (signature-verified Stripe events) and `POST /api/webhooks/apple` (App Store Server Notifications), plus the human-facing `/billing/direct`, `/billing/success`, and `/billing/cancel` pages.

Sharing

POST `/api/share/email`

 **Auth**

Send a rich HTML share email for a channel or Ding. Body: `{"toEmail", "shareType", "channelName", "link"}` required; `dingTitle`, `dingMessage`, `dingTimestamp`, `attachments` optional. Your email address is never exposed to the recipient.

POST

[/api/share/qr-code](#) Channel owner

Email a channel's invite QR code. Body: `{"channelId", "channelName", "inviteUrl"}` required, `toEmail` optional (defaults to your email). `403` unless you own the channel.

Errors

Errors are JSON with a single `error` field.

STATUS	MEANING
400	Malformed JSON body or failed validation (the <code>error</code> message says which field)
401	Missing, invalid, or expired Firebase ID token
402	In-app purchase verification failed (<code>/api/subscriptions/sync</code>)
403	Authenticated but not allowed — usually a channel-owner or creator-only action
404	Notification, channel, or profile not found
409	Channel name already exists
503	Backing service unavailable — retry later
500	Server or database error — retry, then flag it to the team



Unauthenticated **browser** requests to protected routes are redirected (302) to `/auth` instead of getting a 401 — send `Accept: application/json` from API clients to always get JSON. JSON request bodies are capped at 64 KB on most endpoints (2 MB for subscription sync).