QRD API & MCP — Developer Guide

Create, render, manage, and track QR codes programmatically over a simple REST API, or let an AI assistant do it through the QRD MCP server.

- **Base URL:** `https://qrd.cx` (replace with your deployment's URL if self-hosted)
- **API version:** `v1` (all endpoints are under `/api/v1`)
- **Format:** JSON in, JSON out (except image endpoints, which return binary)
- **Postman:** import `qrd-api.postman_collection.json`, then set the `baseUrl` and `apiKey` collection variables to try every endpoint.

---

1. Authentication

Every request must send an API key as a Bearer token:

```
Authorization: Bearer qrd_live_xxxxxxxxxxxxxxxxxxxx
```

Getting a key

1. Sign in to QRD and open **Settings** → **API & MCP**.
2. Click **Create key**, give it a label, and copy the token.
3. The full key is shown **once** — store it somewhere safe (a secrets manager, `.env`, etc.). If you lose it, revoke it and create a new one.

Keys act on behalf of a single organization. Anyone holding a key can use your quota and manage your codes, so treat them like passwords. Revoke compromised keys from the same screen.

---

2. Rate limits & plans

Each API call counts against a per-organization **daily quota**. Exceeding it returns `429 Too Many Requests`.

Plan	Daily requests	Active API keys	Max QR codes
Free	500	1	5
Pro	10,000	Unlimited	1,000
Enterprise	1,000,000	Unlimited	Unlimited

The API must be enabled for your deployment (`ENABLE_API_KEYS`). Your plan must include API access; otherwise create/list calls return `402 Payment Required`.

---

3. Quick start

```
# 1. Create a dynamic QR code (scans are tracked)
curl -X POST https://qrd.cx/api/v1/qr \
  -H "Authorization: Bearer qrd_live_..." \
  -H "Content-Type: application/json" \
  -d '{
        "title": "Storefront flyer",
        "targetUrl": "https://your-shop.com/deal",
        "type": "DYNAMIC",
        "tags": ["promo"],
        "appearance": { "foreground": "#101010", "size": 512 }
      }'

# The response includes the new code's "id" plus an inline "image" (PNG data URL).

# 2. Download the rendered image any time (uses the id from step 1)
curl -L "https://qrd.cx/api/v1/qr/QR_ID/image?format=png&size=512" \
  -H "Authorization: Bearer qrd_live_..." \
  -o qr.png
```

---

4. Concepts

Dynamic vs. static codes

- `DYNAMIC` **(default & recommended)** — the QR image encodes a short QRD link (`https://qrd.cx/r/<shortCode>`). Every scan is redirected through QRD, so it's **tracked** and you can change the destination later without reprinting.
- `STATIC` — the QR image encodes your `targetUrl` directly. Nothing is tracked and the destination can't be changed after printing.

The field `encodedUrl` in responses tells you exactly what a given code's image encodes.

Appearance

Optional visual settings, accepted when creating a code (stored on the code) and as query overrides on the image endpoint:

Field	Type	Notes
`foreground`	string	Hex color of the dark modules, e.g. `#101010`
`background`	string	Hex color of the light area, e.g. `#FFFFFF`
`size`	number	Image width in px, 64–2000
`margin`	number	Quiet-zone width in modules, 0–20
`errorCorrection`	string	`L`, `M` (default), `Q`, or `H`

---

5. Endpoints

All responses are wrapped as `{ "success": true, "data": ... }` on success or `{ "success": false, "error": "reason" }` on failure.

Create a QR code

`POST /api/v1/qr`

Body:

Field	Required	Description
`title`	yes	Display name (1–200 chars)
`targetUrl`	yes	Destination URL / payload
`type`	no	`DYNAMIC` (default) or `STATIC`
`tags`	no	Array of labels for filtering (max 20)
`appearance`	no	Appearance object (see above)

Returns `201` with the created code, including `encodedUrl` and an inline `image` (PNG data URL):

```
{
  "success": true,
  "data": {
    "id": "b1e5…",
    "shortCode": "Ab12Cd34",
    "title": "Storefront flyer",
    "targetUrl": "https://your-shop.com/deal",
    "type": "DYNAMIC",
    "status": "ACTIVE",
    "tags": ["promo"],
    "scans": 0,
    "encodedUrl": "https://qrd.cx/r/Ab12Cd34",
    "image": "…"
  }
}
```

List QR codes

`GET /api/v1/qr`

Query params: `page` (default 1), `perPage` (default 20, max 100), `search` (matches title or URL substring).

```
{
  "success": true,
  "data": {
    "items": [ /* QR records (no inline image) */ ],
    "total": 42, "page": 1, "perPage": 20, "totalPages": 3
  }
}
```

Get one QR code

`GET /api/v1/qr/:id`

Returns the single record, including `encodedUrl` and an inline `image`.

Render the QR image

`GET /api/v1/qr/:id/image`

Returns the actual image as binary (not JSON).

Query	Description
`format`	`png` (default) or `svg`
`foreground`, `background`, `size`, `margin`, `errorCorrection`	Per-request appearance overrides (see Appearance)

`Content-Type` is `image/png` or `image/svg+xml`. DYNAMIC codes encode the tracked short link; STATIC codes encode the raw target.

Update a QR code

`PATCH /api/v1/qr/:id`

Body (all optional): `title`, `targetUrl`, `status` (`ACTIVE` | `PAUSED` | `ARCHIVED`), `tags`. Returns the updated record.

> Tip: changing `targetUrl` on a `DYNAMIC` code re-points every already-printed QR — no reprint needed.

Delete a QR code

`DELETE /api/v1/qr/:id`

Permanently deletes the code and its scan history. Returns `{ "success": true, "data": { "id": "…" } }`. **This cannot be undone.**

Read scan events (tracking)

`GET /api/v1/qr/:id/scans`

Returns the most recent scan events for a code (newest first).

Query	Description
`limit`	Max events, default 100, max 500

Each scan event includes:

```
{
  "id": "…",
  "scannedAt": "2026-07-09T12:34:56.000Z",
  "country": "CA",
  "city": "Montreal",
  "referer": "https://instagram.com/",
  "userAgent": "Mozilla/5.0 …",
  "viewerTimeZone": "America/Toronto",
  "viewerLocalHour": 8,
  "resolvedTargetUrl": "https://your-shop.com/deal?utm_source=…"
}
```

Only `DYNAMIC` codes produce scan events (static codes are not routed through QRD).

---

6. Scheduled email reports

Subscribe email addresses to a recurring performance summary for your whole organization (total scans, conversions, top codes, and top countries over the period). Reports are sent **weekly** or **monthly**.

> Requires the reports feature to be enabled on the deployment (`ENABLE_REPORTS`) and an email provider configured for real delivery. Disabled deployments return `503` .

Create a report schedule

`POST /api/v1/reports`

Field	Required	Description
`recipients`	yes	1–20 email addresses
`cadence`	no	`weekly` (default) or `monthly`

```
curl -X POST https://qrd.cx/api/v1/reports \
  -H "Authorization: Bearer qrd_live_..." \
  -H "Content-Type: application/json" \
  -d '{ "recipients": ["owner@shop.com"], "cadence": "weekly" }'
```

```
{
  "success": true,
  "data": {
    "id": "…",
    "recipients": ["owner@shop.com"],
    "cadence": "weekly",
    "active": true,
    "lastSentAt": null,
    "createdAt": "2026-07-09T12:00:00.000Z"
  }
}
```

The first email goes out on the next scheduled run; subsequent emails follow the cadence.

List schedules

`GET /api/v1/reports` — returns all schedules for your organization.

Update a schedule

`PATCH /api/v1/reports/:id`

Body (all optional): `recipients` , `cadence` , `active` (set `false` to pause, `true` to resume). Returns the updated schedule.

Delete a schedule

`DELETE /api/v1/reports/:id` — stops and removes the schedule. Returns `{ "success": true, "data": { "id": "…" } }`.

> **Note on scope:** reports are organization-wide summaries, not per-QR-code. For per-code metrics, poll `GET /api/v1/qr/:id/scans`.

---

7. Errors

All errors are JSON: `{ "success": false, "error": "reason" }`.

Status	Meaning
400	Malformed request (e.g. invalid JSON)
401	Missing / invalid API key
402	Plan doesn't include API access, or a limit is hit
404	Code not found (or not in your organization)
422	Validation error (bad field values)
429	Daily quota exceeded
503	API disabled on this deployment

---

8. MCP server (use QRD from AI assistants)

The `qrd-mcp` package lets AI clients (Cursor, Claude Desktop, and other MCP hosts) create and manage QR codes using your API key.

Setup

The server is published on npm as `qrd-mcp` — no repo access, clone, or build required. Add it to your MCP client config and it runs via `npx`:

```
{
  "mcpServers": {
    "qrd": {
      "command": "npx",
      "args": ["-y", "qrd-mcp"],
      "env": {
        "QRD_API_KEY": "qrd_live_...",
        "QRD_API_BASE_URL": "https://qrd.cx"
      }
    }
  }
}
```

`QRD_API_KEY` is required. `QRD_API_BASE_URL` is optional and defaults to `https://qrd.cx`. (Requires Node.js 18+ available on the client machine.)

Tools

Tool	What it does
`qrd_create_qr`	Create a code (title, targetUrl, type, tags, appearance)
`qrd_render_qr`	Return the actual QR image (PNG or SVG) for an existing code
`qrd_update_qr`	Update title / targetUrl / status / tags
`qrd_delete_qr`	Permanently delete a code
`qrd_list_qr_codes`	List codes (pagination + search)

A typical AI flow: `qrd_create_qr` → `qrd_render_qr` to get a downloadable image.

---

9. Versioning

The current API version is `v1`. Backwards-incompatible changes will ship under a new version prefix (e.g. `/api/v2`). Additive fields may appear on existing responses without a version bump, so ignore unknown fields defensively.