

Over-the-air (OTA) updates — deployment schema v0

From OTA.md

Table of contents

- [URL layout](#)
- [Versioning](#)
- [JSON files](#)
- [Optional HTTP bundle \(.otabundle.zip\)](#)
- [MQTT transport](#)
- [Publish from a release APK](#)
 - [Docker build + staging channel](#)
- [App configuration](#)
- [Checklist](#)

Source: OTA.md — section links work in PDF viewers that support internal anchors.

The app checks a **stable channel URL**, follows it to a per-build manifest, then downloads the update. Version comparison uses `major.minor.build` (packed into Android `versionCode` at build time). Deploy never computes `versionCode` by hand. The client runs checks and downloads in `OtaUpdateService` (a standalone service, not embedded in `cast/receiver` services). Launch checks enqueue work there; download uses a **foreground service** (`dataSync`) with a progress notification. MQTT is fully supported for OTA sources using retained topic payloads (`mqtt://.../mqttps://...`). The URI path is treated as the exact topic to subscribe.

URL layout

```
https://host/v0/ota/channel/stable.json
https://host/v0/ota/00/00.{major}/00.{major}.{minor}/android_cast_00.{major}.{minor}.{build}.otapkg
https://host/v0/ota/00/00.{major}/00.{major}.{minor}/android_cast_00.{major}.{minor}.{build}_sign.json
https://host/v0/ota/00/00.{major}/00.{major}.{minor}/android_cast_00.{major}.{minor}.{build}_manifest.json
https://host/v0/ota/00/00.{major}/00.{major}.{minor}/android_cast_00.{major}.{minor}.{build}.otabundle.zip
```

- `v0` — deployment schema version (default channel).
- `00` after `ota/` — OTA payload format revision within `v0`.
- Components are **zero-padded to 2 digits** in paths and filenames (`00.01.05`).

Versioning

In `app/build.gradle` (overridable via `local.properties`):

```
ota.major=0
ota.minor=1
ota.build=0
```

Packed rule (each part 0–99):

```
versionCode = major * 10000 + minor * 100 + build
```

Example: `0.1.5` → `versionCode 105`.

JSON files

```
v0/ota/channel/stable.json
{
  "schema": "v0",
  "manifestUrl": "https://host/v0/ota/00/00.01/00.01.05/android_cast_00.01.05_manifest.json"
}
***_manifest.json**
{
  "schema": "v0",
  "major": 0,
  "minor": 1,
  "build": 5,
  "versionName": "0.1.5",
  "apkUrl": "https://host/v0/ota/00/00.01/00.01.05/android_cast_00.01.05.otapkg",
  "signUrl": "https://host/v0/ota/00/00.01/00.01.05/android_cast_00.01.05_sign.json",
  "sizeBytes": 42100000,
  "bundleUrl": "https://host/v0/ota/00/00.01/00.01.05/android_cast_00.01.05.00.otabundle.zip",
  "bundleSha256": "...",
  "bundleSizeBytes": 42150000,
  "mandatory": false,
  "releaseNotes": ""
}
***_sign.json**
{
  "schema": "v0",
  "sha256": "..."
}
```

The app loads `sha256` from `sign.json` when the manifest omits it (APK hash inside the bundle or standalone `.otapkg`).

Optional HTTP bundle (`.otabundle.zip`)

`generate-ota-v0.sh` emits a **STORE** (uncompressed) zip so the APK is not re-deflated:

Entry inside zip	Content
<code>manifest.json</code>	Same as <code>*_manifest.json</code>
<code>sign.json</code>	Same as <code>*_sign.json</code>
<code>package.apk</code>	Release APK bytes

Check path (lightweight): `channel` → small `*_manifest.json` (+ optional `*_sign.json`) — no bundle download until the user installs.

Download path (HTTP): if `bundleUrl` is `http/https`, the app downloads one zip, verifies `bundleSha256`, unpacks, verifies `package.apk` against `sha256`, then runs the installer. If `bundleUrl` is absent, it falls back to separate `apkUrl` / `signUrl` fetches (required for MQTT).

MQTT: keep **per-artifact** retained topics (`manifest.json`, `sign.json`, `.otapkg`). Do not publish full bundles on MQTT (message size limits).

The app ignores `bundleUrl` when the scheme is not `HTTP(S)`.

MQTT transport

Use the same JSON payload shapes as HTTP, but publish them as **retained** messages on topics that match your URI paths:

- `mqtt://host:1883/v0/ota/channel/stable.json` -> retained payload is `stable.json` content
- `mqtt://host:1883/v0/ota/00/..._manifest.json` -> retained payload is manifest JSON
- `mqtt://host:1883/v0/ota/00/..._sign.json` -> retained payload is sign JSON
- `mqtt://host:1883/v0/ota/00/...otapkg` -> retained payload is raw APK/otapkg bytes

When `ota.trusted=true` and build is debug, strict checksum/sign enforcement is relaxed for that source. Release builds always remain strict.

Publish from a release APK

```
chmod +x scripts/generate-ota-v0.sh
./scripts/generate-ota-v0.sh app/build/outputs/apk/release/app-release.apk https://your-host:port ./ota-publish staging
```

Fourth argument is the **channel** file name (`stable`, `staging`, `next`, ...). Upload `ota-publish/v0/` to your web root so `https://your-host/v0/ota/...` resolves.

Docker build + staging channel

See `BUILD_DEPLOY.md`:

```
export OTA_BASE_URL=https://apps.f0xx.org
./scripts/docker-build-ota.sh
./scripts/deploy-ota-staging.sh # after setting OTA_DEPLOY_TARGET
```

App configuration

`local.properties` (not committed):

```
ota.channel.url=https://your-host/v0/ota/channel/stable.json
ota.major=0
ota.minor=1
ota.build=0
```

Or set the channel URL under **Developer settings** → **App updates (OTA)**.

Legacy `ota.manifest.url` still works as a direct manifest URL fallback.

Checklist

1. Bump `ota.major` / `ota.minor` / `ota.build` (or Gradle defaults).
2. Build signed release APK (`versionCode` is derived automatically).
3. Run `generate-ota-v0.sh` and upload `v0/ota/**` (including `.otabundle.zip`).
4. Confirm `stable.json` points at the new `*_manifest.json`.