

Android Cast — On-Demand Remote Access (Reverse Tunnel) Proposals

Generated from markdown — rev. 1–3 + appendices WG test/production

Table of contents

- Document history
- Context and constraints
- Executive summary
 - Initial proposal (rev. 1)
 - Refined after infra review (rev. 2)
- Stakeholder questions (rev. 2 — added to requirements)
- Problem statement
- Proposed control plane (heartbeat type: ra)
 - Device → BE (poll, random interval 1–7 min when enabled)
 - BE → device (only if whitelisted + operator requested session)
 - Disable (user unchecks box or admin revokes)
 - Field reference
 - State machine (simplified)
- Network topology — Gentoo FE → Alpine BE (rev. 2)
 - FE ↔ BE tuning (by track)
- Security model
 - Key handling options (rev. 1)
- Embedding in the Android app (rev. 2)
- Backend integration ease — Alpine BE (rev. 2)
- Licensing and open source (rev. 2)
- What “Custom file API” means (rev. 2 — clarified)
- Tooling alternatives — full survey (rev. 1)
- Tooling summary — infra-focused (rev. 2)
- Comparison matrix — initial scoring (rev. 1)
- Comparison matrix — FE proxy focus (rev. 2)
- WireGuard vs SSH — decision lens (rev. 2)
- Recommendations
 - Initial (rev. 1)
 - Refined (rev. 2 — Gentoo FE → Alpine BE)
- RBAC roles (BE)
- BE admin UI (crashes look-and-feel)
- MariaDB schema (sketch — rev. 1)
- Android app changes (sketch — rev. 1)
- Estimates (engineering)
- Open questions (merged)
- Rev. 3 — WG vs SSH Q&A + bastion deep-dive
 - Q1 — Exact FE/BE rules: WireGuard (DNAT) vs SSH (no DNAT required)
 - Q2 — Can we use ports other than :80? (heartbeats and everything else)
 - Q3 — Why is “Proxyable (FE→BE)” low for WG (2) and high for SSH (5)?
- Rev. 3 — Bastion Q&A
 - Q4 — How easy to implement? Command checklist (FE + BE)
 - Q5 — Bastion: file transfers only, or other operations too?
- Appendix 1 — WG test configuration (Linux laptop)
 - Network topology (v1 WireGuard)
 - Prerequisites (laptop)
 - A. HTTP E2E (no tunnel yet)
 - B. UDP / WireGuard from laptop (after open session)
 - C. Quick checks before blaming WG
- Appendix 2 — WG production configuration
 - BE config.php (remote_access)
 - BE (Alpine) — WireGuard interface
 - FE (Gentoo) — UDP DNAT (not nginx)
 - MariaDB / migrations
 - Production verify (on BE or laptop)
 - Android (field device)
- Source linkage

Source: 20260602_REVERSE_SSH_proposals_summary.md — section links work in PDF viewers that support internal anchors.

Document history

Rev	Focus	Summary
1	Initial research memo	On-demand reverse tunnel via heartbeat type: ra; compare WG, SSH, tinc, Tailscale, etc.; hybrid WireGuard outbound + SSH/SFTP bastion as first pick.
2	Stakeholder Q&A	Gentoo FE → Alpine BE path, embed WG/SSH in app, licensing, Custom file API clarified; refined pick reverse SSH/SFTP (Proxyable) over WG (Lightweight) for this nginx topology.
3	WG vs SSH + bastion deep-dive	Exact FE/BE commands (DNAT vs stream), port policy, Proxyable scoring explained, bastion scope (files vs full debug).

Both revisions are kept below. Where they differ, §14 Recommendations lists both and explains why rev. 2 refined the tunnel choice. §Rev. 3 adds operator-ready command checklists for the final pick.

Context and constraints

- **Need**: occasional remote debug / manual settings on **one field device** (e.g. edit codecs.json in app home, inspect logs) without a **persistent** device↔BE tunnel.
- **Stack**: minimalistic LAMP (nginx + php-fpm + MariaDB) on BE; Android app already posts to /api/heartbeat.php.
- **Traffic path (rev. 2)**: **Mobile app** → **Gentoo FE** (TLS, firewall, nginx reverse proxy) → **Alpine BE** (XEN HVM, project services). Heartbeat already follows this; any data tunnel must be tunable on **both** FE and BE.
- **Security**: no standing inbound connection from BE to device; user opt-in; admin whitelist; full audit trail; minimal secret exposure on wire.
- **Ops**: sessions expire on inactivity; 24h DB cleanup; user can disable instantly from app settings.
- **Convenience (rev. 2)**: SCP/SFTP-like file exchange is **not mandatory** but highly desirable for codecs.json and logs.

Executive summary

Initial proposal (rev. 1)

- **Recommended**: on-demand reverse tunnel orchestrated by BE heartbeat (type: ra) + **WireGuard or SSH over a single ephemeral port** on BE, with **per-session keys** and **PHP admin UI** matching crashes console look-and-feel.
- **Not recommended for v1**: always-on VPN mesh (tinc/Tailscale on every device), public inbound SSH to phones, or BE-initiated connections through carrier NAT.
- **Pattern**: device polls BE at **random 1–7 min** when user enables “Remote access”; BE responds only if device is **whitelisted**; device then **dials out** and attaches tunnel; operator connects to BE **bastion/port**, not to the phone directly.
- **Primary pick (rev. 1)**: WireGuard outbound + SSH/SFTP on bastion for operator (combines mobile simplicity with familiar file edit workflow for codecs.json).
- **Fallback (rev. 1)**: if UDP blocked on target networks, use **chisel/frp over HTTPS** on same ra vhost with identical control plane.

Refined after infra review (rev. 2)

- **Control plane**: unchanged (heartbeat type: ra, checkbox, whitelist, jittered poll, device dials out).
- **Tunnel choice (refined)**: prefer **reverse SSH (+ SFTP)** over WireGuard for v1 **on this stack**.
- **One word — SSH: Proxyable** — TCP (nginx stream) matches Gentoo FE → Alpine BE like existing HTTP proxy; SFTP gives SCP-like files.
- **One word — WireGuard: Lightweight** — excellent embeddable crypto, but **UDP does not ride nginx HTTP reverse-proxy**; FE needs UDP DNAT/firewall rules.
- **Custom file API**: optional third track — HTTPS upload/download via PHP (§10); good Phase-1 or narrow path without sshd.

Stakeholder questions (rev. 2 — added to requirements)

#	Question	Short answer
Q1	Is WireGuard “best”?	Best crypto/simplicity , not best FE→BE proxy fit for your stack.
Q2	Embed WG in app or separate app?	Embed via wireguard-android tunnel library + VpnService; no standalone WireGuard app.
Q3	Embed SSH or separate app?	Embed Dropbear/libssh2/MINA SSHD; no Termux.
Q4	BE integration: WG vs SSH?	SSH easier on Alpine (openssh, authorized_keys, -R port). WG needs wg set + UDP FE→BE.

Q5	Traffic Mobile → Gentoo FE → Alpine BE?	Heartbeat: proxy_pass :80. SSH: nginx stream. WG: UDP DNAT (not nginx location /).
Q6	Licensing / open source?	See §9; Tailscale server is main license caveat.
Q7	What is "Custom file API"?	HTTPS file transfer via PHP — not SCP, not a VPN; see §10.
Q8	SCP-like file exchange?	Native SFTP (SSH) ; similar UX via Custom file API; WG needs SSH/SFTP on top for SCP-like UX.

Problem statement

Aspect	Detail
Trigger	Single device in the field needs interactive access (debug, replace config, pull logs).
Anti-goals	No permanent VPN; no open listening port on mobile; no broad fleet exposure.
User consent	Checkbox in app settings ("allow remote access at my own risk").
Admin gate	Device ID must be on BE remote_access_devices whitelist before any session is offered.
Observability	Every enable/disable/connect/disconnect/timeout logged with reason and actor.
Path (rev. 2)	Signalling via FE→BE :80; tunnel endpoint on FE forwarded to BE.

Proposed control plane (heartbeat type: ra)

Reuse existing heartbeat endpoint; add branch for remote access.

Device → BE (poll, random interval 1–7 min when enabled)

```
{
  "heartbeat": {
    "type": "ra",
    "status": "enable",
    "device_id": "<stable device uuid>",
    "random": "<client session nonce / reconnect token>",
    "app_version": "0.1.0",
    "capabilities": ["ssh", "sftp", "files_app_home"]
  }
}
```

Also accepted (rev. 2): "capabilities": ["ssh_reverse", "sftp", "files_app_home"].

BE → device (only if whitelisted + operator requested session)

```
{
  "heartbeat": {
    "type": "ra",
    "action": "connect",
    "session_id": "uuid",
    "endpoint": "ra.apps.f0xx.org:443",
    "tunnel": "wireguard|ssh|ssh_reverse",
    "credentials": {
      "mode": "ephemeral_pubkey|otp",
      "username": "ra-<session_id>",
      "host_key_fingerprint": "...",
      "remote_forward": "127.0.0.1:<be_port>:127.0.0.1:8022"
    },
    "expires_at": 1717340000
  }
}
```

WireGuard branch (rev. 1): "tunnel": "wireguard" with peer keys in credentials. SSH branch (rev. 2): "tunnel": "ssh_reverse" as above.

Disable (user unchecks box or admin revokes)

```
{ "heartbeat": { "type": "ra", "status": "disable", "device_id": "...", "random": "..." } }
```

Field reference

Field	Purpose
random	Client nonce for matching pending BE session; rotation on reconnect.
session_id	DB primary key for audit + port/key binding.
status	enable / disable / poll (optional alias for enable).
action	BE command: connect, wait, deny, rotate.

State machine (simplified)

[user enables RA] → device polls → BE: wait (not whitelisted / no operator)
→ admin opens session for device_id → next poll → BE: connect + creds
→ device opens outbound tunnel → BE marks active → operator uses bastion
[user disables] → disable heartbeat → tunnel tear-down → BE inactive + audit row
[24h idle] → BE cron purges stale rows + closes ports

Network topology — Gentoo FE → Alpine BE (rev. 2)

[Android app]
■ HTTPS POST /api/heartbeat.php (type: ra)
▼
[Gentoo FE – apps.f0xx.org]
■ nginx: proxy_pass http://artc0.intra.raptor.org:80 (existing)
▼
[Alpine BE – artc0 / XEN HVM]
■ php-fpm + MariaDB (control plane)
■ sshd / ra-bastion (data plane – SSH track)
■ wg0 + UDP listener (data plane – WG track, if used)

[Operator laptop]
■ SSH/SFTP to FE or BE bastion port (never to phone IP)
▼
session forwarded to device's reverse tunnel

FE ↔ BE tuning (by track)

Track	Public entry (FE)	FE config	BE service	Notes
Heartbeat (all)	https://apps.f0xx.org/.../heartbeat.php	proxy_pass http://artc0:80	php-fpm	Already works.
Reverse SSH	ra.apps.f0xx.org:443 TCP or :2222	stream { proxy_pass artc0:22; }	sshd + Match User ra-*	Rev. 2 pick (Proxyable).
WireGuard	ra.apps.f0xx.org:51820/udp	iptables/nft DNAT → BE	wg-quick / dynamic peers	Rev. 1 pick candidate (Lightweight).
Custom file API	https://apps.f0xx.org/.../api/ra/*	existing HTTP proxy	php-fpm only	No stream/UDP rules.

Security model

Layer	Mechanism
Authorization	RBAC roles (§12) + per-device whitelist + user opt-in on device.
Authentication	Per-session ephemeral key pair or one-time port + password; no long-lived device secrets in APK.
Transport	TLS to BE for heartbeat; tunnel encrypted (WireGuard / SSH).
Exposure	BE listens on dedicated ra vhost/port; no inbound to mobile; optional IP allowlist for operators.
Audit	remote_access_events table: who, device, action, reason, ip, timestamps.
Cleanup	Inactivity timeout (e.g. 30–60 min); daily purge of records > 24h; credential invalidation on disable.
Reconnect	Same session_id + rotated random within grace window; new keys if expired.

Key handling options (rev. 1)

Mode	Description	Risk
Ephemeral WG keypair	BE generates keys per session; device installs peer config once	Low; keys short-lived
SSH host cert + user cert	Step-CA or internal CA signs 1h user cert	Medium ops; mature tooling
Pre-allocated device key	One WG pubkey per device in DB	Higher leak impact; simpler reconnect
OTP + reverse SSH	autossh -R random_port with password from BE	Simple; weaker than pubkey

Key ceremony (rev. 1): ephemeral WireGuard or SSH **user certificate** per session; optional device **identity key** (Ed25519) registered at whitelist time for authenticating poll requests only (not tunnel).

- Additional (rev. 2):**
- Terminate operator SSH on FE stream or require VPN to Gentoo before bastion.
 - Device **outbound-only** to ra.apps.f0xx.org:443 (TCP).
 - Per-session ForceCommand internal-sftp if shell not required.

Embedding in the Android app (rev. 2)

Solution	Separate app required?	How to embed	Android notes
WireGuard	No	wireguard-android tunnel AAR / GoBackend	VpnService; VPN icon while active.
SSH reverse	No	Dropbear + JNI, libssh2, Apache MINA SSHD	Foreground service; no VPN icon.
frp/chisel client	No	Static binary or Go mobile bind	Extra artifact.
Custom file API	No	OkHttp + JSON	Pure Java/Kotlin.

You do **not** need Termux or the standalone WireGuard app if libraries are embedded in **Android Cast**.

Backend integration ease — Alpine BE (rev. 2)

Aspect	Reverse SSH	WireGuard
Package on Alpine	openssh	wireguard-tools + kernel mod
Session open (PHP)	authorized_keys + -R port; optional ForceCommand internal-sftp	wg set wg0 peer ...
Session close	Remove key; kill session; free port	wg set ... remove
FE cooperation	nginx stream (TCP)	UDP DNAT
Operator access	sftp -P 443 ra-...@ra.apps.f0xx.org	wg-quick up or SSH over WG
Jump host f0xx.org:222	Natural extension	Separate UDP path

Licensing and open source (rev. 2)

Component	License	Ship in app / BE	Notes
WireGuard (kernel module)	GPL-2.0	BE kernel	Stock Alpine package.
wireguard-go / Android tunnel	MIT / Apache-2.0	App embed	Userspace embed.
OpenSSH	BSD-style	BE + client	Permissive.
Dropbear	Permissive (MIT-like)	App embed	Small client.
libssh2	BSD-3-Clause	App embed	
Apache MINA SSHD	Apache-2.0	App (Java)	SSH/SFTP in Java.
chisel	MIT	App/BE	
frp	Apache-2.0	App/BE	
tinc	GPL-2.0+	—	Copyright; mesh.
Tailscale client	BSD-3-Clause	—	Client OSS.
Headscale / Tailscale server	BSD / BSL	—	Self-host Headscale if needed.
Custom file API	Project code	BE PHP	No third-party tunnel license.

GPL-2.0 Android app: use permissively licensed clients (MIT/BSD/Apache); review before linking GPL client libs into APK.

What “Custom file API” means (rev. 2 — clarified)

Not a tunnel and **not** the SCP protocol. Application-layer remote file exchange on existing PHP:

- Endpoints: POST /api/ra/upload, GET /api/ra/download, GET /api/ra/list.
- Auth: short-lived token from heartbeat connect (same session as control plane).
- Scope: getFilesDir() / codecs.json unless Storage permission granted (broader scope needs second consent).
- Operator: crashes-themed **web UI** or curl — **SCP-like** push/pull without scp binary.
- Pros:** no FE stream/UDP; minimal attack surface (no shell); reuses LAMP.
- Cons:** no shell; implement listing/chunking for large logs.

Use when: Phase-1 before sshd bastion, or permanent narrow path alongside SSH for power users.

Original rev. 1 one-liner: “Custom HTTPS file API — PHP endpoints; no shell; minimal attack surface; more app code for file ops.”

Tooling alternatives — full survey (rev. 1)

Option	Maturity	Mobile fit	BE fit	Pros	Cons / risks
WireGuard (outbound from device)	High	Excellent (userspace WG)	Single UDP listener / per-session peers	Fast, modern crypto, small code	UDP blocked on some carriers; peer lifecycle automation
Reverse SSH (autossh -R)	High	Good (embedded dropbear client)	sshd on bastion	Familiar, SFTP, shell	Key mgmt; battery if misconfigured
SSH via WebSocket (chisel, frp)	Medium	Good	One HTTPS port	Traverses strict firewalls	Extra binary; audit frp history
OpenVPN / IKEv2 (on-demand)	High	Heavy	openvpn-access	Full tunnel	Overkill; battery; not file-focused

tinc	Medium	Poor (daemon, mesh)	mesh coordinator	P2P mesh	Persistent mesh ≠ on-demand; key distribution painful
Tailscale / Headscale	High	App + VPN	control server	Easy UX	Persistent identity; fleet-wide trust
MeshCentral / RustDesk	High	Agent app	relay server	Remote desktop	Different UX; screen control not primary need
ADB over TLS tunnel	Medium	Dev-only	reverse port	Exact Android debug	Dev options; security sensitive
Custom HTTPS file API	Low (build)	Native	PHP endpoints	No shell; minimal attack surface	No arbitrary shell; more app code

Tooling summary — infra-focused (rev. 2)

Option	Embed?	FE→BE fit	SCP-like	License	Verdict
Reverse SSH + SFTP	Yes	Excellent (stream)	Native	BSD	v1 pick (rev. 2)
WireGuard outbound	Yes	Fair (UDP DNAT)	Via SFTP over WG	MIT/GPL	v1 pick (rev. 1) / v2 if UDP path ready
Custom file API	Yes	Excellent (HTTP)	Similar UX	Yours	Phase-1 / narrow
frp/chisel	Yes	Good	Possible	MIT/Apache	Fallback (both revs)
tinc / Tailscale	VPN app	Poor on-demand	Varies	GPL/BSL	Not recommended

Comparison matrix — initial scoring (rev. 1)

Scoring: 1 = low, 5 = high. Lower total was interpreted as stronger general fit in rev. 1 table orientation.

Criterion	WG outbound	Reverse SSH	frp/chisel	tinc	Tailscale	Custom file API
Security	5	4	3	3	4	5
Simplicity (ops)	4	4	3	2	3	4
On-demand fit	5	5	5	1	2	5
File access (app dir)	4	5	4	3	4	5
Shell/debug	4	5	4	3	4	1
Fits LAMP BE	4	5	4	3	3	5
Battery / data	4	3	3	2	2	5
Weighted total	31	30	26	17	22	30

Rev. 1 tie note: WireGuard, reverse SSH, and Custom file API scored highest. Rev. 1 suggested WireGuard + SFTP/SSH bastion if shell needed; Custom file API if only codecs .json + logs.

Comparison matrix — FE proxy focus (rev. 2)

Criterion	Rev SSH	WG outbound	Custom file API	frp/chisel
Proxyable (FE→BE)	5	2	5	4
Security	4	5	5	3
On-demand fit	5	5	5	5
SCP-like files	5	3*	4	3
BE integration ease	5	3	5	3
Embed without 3rd-party app	5	5	5	4
Total	29	23	29	22

*WG files row assumes SSH/SFTP layered on tunnel.

WireGuard vs SSH — decision lens (rev. 2)

Criterion	WireGuard	Reverse SSH + SFTP
One-word headline	Lightweight	Proxyable
Embed in app	Yes (VpnService)	Yes (libssh2/MINA/dropbear)
FE nginx HTTP proxy	No (UDP)	Yes (stream TCP)
Gentoo → Alpine path	Firewall UDP DNAT	Same as HTTP ops model
SCP-like files	Needs SSH/SFTP on top	Built-in (SFTP)
Carrier networks	UDP sometimes blocked	TCP 443 usually works

Recommendations

Initial (rev. 1)

1. **Phase 0** — Design + threat model (1–2 days): consent copy, port allocation, key ceremony.
 2. **Phase 1** — Control plane (1–2 weeks): heartbeat ra, whitelist, admin UI, audit, cron; dry-run wait/connect.
 3. **Phase 2** — Tunnel (1–2 weeks): **WireGuard outbound** from app OR **reverse SSH to bastion**; operator connects via existing VPN/SSH to BE network.
 4. **Phase 3** — Hardening: rate limits, mTLS on heartbeat, device identity signing, reconnect, pen-test.
- Primary pick (rev. 1):** WireGuard outbound + SSH/SFTP on bastion for operator.
Fallback (rev. 1): chisel/frp over HTTPS on same ra vhost.

Refined (rev. 2 — Gentoo FE → Alpine BE)

- Primary v1: reverse SSH (+ SFTP)** — path is Mobile → Gentoo FE (nginx) → Alpine BE; **Proxyable** TCP; SFTP for codecs.json.
Secondary: WireGuard when dedicated **UDP forwarding** on FE is acceptable — **Lightweight** tunnel.
Optional parallel: Custom file API for whitelist-only file push/pull without internet-facing sshd.
- Phases (merged):**
1. Phase 0 — FE/BE port map + threat model (1–2 d).
 2. Phase 1 — heartbeat ra + admin UI + optional Custom file API (1–2 w).
 3. Phase 2 — reverse SSH/SFTP + Gentoo stream → Alpine sshd **or** WG track if chosen (1–2 w).
 4. Phase 3 — MFA, reconnect, pen-test.

RBAC roles (BE)

Role	Permissions
remote_access_user	Open/close own sessions for whitelisted devices; view active/inactive list (scoped); SFTP / file API.
admin	Same as user + add/remove whitelist entries + assign remote_access_user role.
root	Full role management + force-disconnect any session + view all audit logs.

Integrate with existing users / Rbac; new permissions: remote_access_view, remote_access_operate, remote_access_admin.

BE admin UI (crashes look-and-feel)

- New nav: **Remote access** under /app/androidcast_project/crashes/?view=remote_access (or /remote/).
 - Reuse layout.php, themes, locale picker, session auth.
 - Screens (rev. 1):
1. **Devices** — whitelist, last seen, RA opt-in status (from last heartbeat).
 2. **Sessions** — active / inactive; device, operator, started, last activity, tunnel type, port, status.
 3. **Audit log** — filter by device, user, action, date.
 4. **Open session** — pick device → pending session → waits for next device poll.
 - Rev. 2 add-on: **file browser** tab when Custom file API enabled (upload/download codecs.json).
 - No new frontend framework; same CSS/JS as tickets/graphs.

MariaDB schema (sketch — rev. 1)

```
-- remote_access_devices (whitelist)
-- remote_access_sessions (session_id, device_id, operator_user_id, status,
-- tunnel, port, keys_ref, expires_at, last_activity)
-- remote_access_events (audit)
-- remote_access_credentials (optional encrypted blob, TTL)
Cron: scripts/purge_remote_access.php — inactive > 24h → closed_timeout; revoke keys; free ports.
```

Android app changes (sketch — rev. 1)

- Settings → Developer → **Allow remote access** (off by default, strong warning).
- RemoteAccessService (foreground while tunnel active): jittered alarm 1–7 min → POST heartbeat.
- On action: connect: start WireGuard/SSH client with supplied creds; expose app files dir only unless Storage permission granted (broader scope needs second consent).
- On disable: stop service, wipe session keys, POST status: disable.
- Persist nothing long-term except optional device identity pubkey for poll authentication.

Estimates (engineering)

Track	Initial delivery	Hardening	Total
-------	------------------	-----------	-------

Control plane only (heartbeat + DB + admin list)	5–8 d	3–5 d	8–13 d
Control plane + Custom file API	6–9 d	3–5 d	9–14 d
+ WireGuard outbound tunnel (rev. 1 pick)	8–12 d	5–8 d	13–20 d (+ FE UDP)
+ Reverse SSH/SFTP bastion (rev. 2 pick)	7–11 d	4–7 d	11–18 d
+ FE nginx stream + firewall docs	+1–2 d	+1 d	(included in SSH track)

Assumptions: one BE bastion; excludes Play Store policy review.

Open questions (merged)

Rev. 1:

- Maximum concurrent RA sessions per BE?
- Store tunnel configs only in RAM vs encrypted MariaDB blob?
- Require operator MFA (TOTP) before open session?
- Play Store disclosure for “remote access” foreground service?

Rev. 2:

- Public TCP port: dedicated 2222 vs share 443 with ssl_preread?
- SFTP-only vs full shell for remote_access_user?
- Store device identity Ed25519 pubkey at whitelist time?
- Rate limit heartbeat ra polls?

Rev. 3:

- Confirm public RA port(s) with FE firewall owner before Phase 2?
- SFTP-only (ForceCommand internal-sftp) vs full shell for v1?

Rev. 3 — WG vs SSH Q&A + bastion deep-dive

Added 2026-06-02 — concrete FE/BE commands and decision support.

Constants used below (from INFRA.md):

Symbol	Value
FE	Gentoo HVM, public apps.f0xx.org (TLS 443 terminates here)
BE	Alpine artc0.intra.raptor.org / 10.7.16.128, app vhost listen 80
FE→BE (existing HTTP)	proxy_pass http://artc0.intra.raptor.org:80
Operator jump (existing)	f0xx.org:222 → Alpine :22

Replace 10.7.16.128 with the live BE IP if it changes.

Q1 — Exact FE/BE rules: WireGuard (DNAT) vs SSH (no DNAT required)

Short answer: WireGuard **must** use **UDP DNAT** (or equivalent firewall forward) on Gentoo FE because nginx proxy_pass is **HTTP/TCP-only** and cannot carry WG packets. SSH **does not need DNAT** on FE if you use nginx stream TCP forwarding (same nginx daemon, different block). You *can* use TCP DNAT for SSH as an alternative, but it is redundant when stream works.

WireGuard track — FE (Gentoo): UDP DNAT + forward allow

Public UDP **51820** on FE → BE **51820**. Example with **nftables** (adjust interface eth0, table/chain names to match your FE):

BE_IP=10.7.16.128

WG_PORT=51820

```
nft add rule inet nat prerouting iifname "eth0" udp dport $WG_PORT \
    dnat ip to ${BE_IP}:${WG_PORT}
nft add rule inet filter forward ip daddr ${BE_IP} udp dport ${WG_PORT} accept
```

iptables equivalent:

BE_IP=10.7.16.128

WG_PORT=51820

```
iptables -t nat -A PREROUTING -i eth0 -p udp --dport $WG_PORT \
    -j DNAT --to-destination ${BE_IP}:${WG_PORT}
iptables -A FORWARD -p udp -d ${BE_IP} --dport $WG_PORT -j ACCEPT
```

Persist: save rules (/etc/nftables.conf or iptables-save → Gentoo iptables init). **Open UDP 51820** on FE edge firewall (in addition to 443/tcp).

Verify from outside:

```
nc -u -v apps.f0xx.org 51820
```

WireGuard track — BE (Alpine): interface + listen

```
apk add wireguard-tools
```

```
modprobe wireguard 2>/dev/null || echo "wireguard module required"
```

```
cat >/etc/wireguard/wg0.conf <<'EOF'
[Interface]
Address = 10.66.66.1/24
ListenPort = 51820
PrivateKey = <BE_WG_PRIVATE_KEY>
EOF
wg-quick up wg0
```



```
rc-update add wg-quick.wg0 default # if using OpenRC
```

Per RA session (PHP or admin script adds ephemeral peer — example):

```
wg set wg0 peer <DEVICE_EPHEMERAL_PUBKEY> allowed-ips 10.66.66.2/32
wg set wg0 peer <DEVICE_EPHEMERAL_PUBKEY> remove
```

Device connects **outbound UDP** to apps.f0xx.org:51820 (DNAT lands on BE). Operator joins same WG net (second peer) or uses SSH over the tunnel (see Q5).

SSH track — FE (Gentoo): nginx stream (TCP proxy, not DNAT)

Preferred: public TCP **443** on ra.apps.f0xx.org (or dedicated **2222**) → BE **22**. No kernel DNAT if nginx terminates the TCP hop.

Add to FE nginx (often /etc/nginx/nginx.conf, **outside** http {}):

```
stream {
    upstream be_sshd {
        server 10.7.16.128:22;
    }
    server {
        listen 443;
        proxy_pass be_sshd;
        proxy_connect_timeout 30s;
    }
}
```

If **443** is already used for HTTPS on the same IP, either:

- use listen 2222 for RA SSH, or
- use ssl_preread on 443 to split HTTPS vs SSH by ClientHello (more complex).

Reload FE nginx:

```
nginx -t && nginx -s reload
```

Verify:

```
ssh -p 443 -o StrictHostKeyChecking=accept-new ra-test@ra.apps.f0xx.org
```

SSH track — optional FE DNAT (parallel to WG, for comparison only)

If you refused nginx stream, SSH could mirror WG with **TCP DNAT**:

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 2222 \
-j DNAT --to-destination 10.7.16.128:22
iptables -A FORWARD -p tcp -d 10.7.16.128 --dport 22 -j ACCEPT
```

This works but **duplicates** what stream already does and bypasses nginx logging/timeouts. **Recommendation:** use stream, not DNAT, for SSH.

SSH track — BE (Alpine): sshd + reverse-forward slot

```
apk add openssh
rc-update add sshd default
rc-service sshd start
```

```
cat >/etc/ssh/sshd_config.d/ra.conf <<'EOF'
Match User ra-*
    AllowTcpForwarding yes
    GatewayPorts no
    X11Forwarding no
    PermitTTY yes
    # v1 file-only: ForceCommand internal-sftp
EOF
rc-service sshd restart
```

Per session — device outbound reverse forward (embedded client runs equivalent):

```
ssh -N -R 127.0.0.1:18022:127.0.0.1:8022 ra-<session_id>@ra.apps.f0xx.org -p 443
```

BE — operator reaches device file area via forwarded port:

```
sftp -P 18022 ra-<session_id>@127.0.0.1
```

Side-by-side: what runs where

Step	WireGuard	Reverse SSH
FE edge open	UDP 51820	TCP 443 or 2222
FE mechanism	nft/iptables DNAT	nginx stream (or optional TCP DNAT)
BE daemon	wg0 / wg set	sshd + ephemeral Match User ra-*
Device direction	Outbound UDP to FE public IP	Outbound TCP to FE → proxied to BE sshd
Operator path	WG peer IP or SSH over WG	SFTP/SSH to BE forwarded port (see Q5)

Q2 — Can we use ports other than :80? (heartbeats and everything else)

Two different “ports” — do not conflate them.

Leg	Port today	Can change?	Notes
Phone → FE	443 (HTTPS)	Yes, but 443 is correct for TLS + carrier-friendly egress	App already targets https://apps.f0xx.org/...; users never hit :80 directly.
FE → BE (heartbeat, crashes, graphs)	80 (HTTP)	Yes	Requires new FE proxy_pass upstream and BE listen block. Not forbidden — just unused today .
RA data plane (WG)	51820/udp (proposed)	Must be non-80	UDP tunnel; unrelated to HTTP vhost.
RA data plane (SSH)	443/tcp or 2222/tcp	Must be non-80	Raw TCP/stream; not HTTP location /.

Why heartbeats stay on the existing :80 FE→BE path (recommended for v1):

1. **Already deployed** — nginx.fe-apps.f0xx.org.snippet and BE listen 80 fragment are the repo source of truth; zero new FE/BE holes for control plane.
2. **Same security zone** as crash upload and graphs — one ops playbook.
3. **No benefit** to moving heartbeat to BE :443 internally — the phone still speaks HTTPS to FE; internal hop is your choice.
4. **Port 8089** on BE is **Janus/other** — do **not** reuse for androidcast (INFRA.md §2).

If you intentionally want heartbeat on BE :443 internally:

That is a **separate migration**; not required for RA.

Summary: Public **443** for all HTTPS (including heartbeat). Internal FE→BE **80** for LAMP today — **can** change, **should not** for v1 unless you have another reason. RA tunnels **must** use **dedicated non-80** ports (UDP 51820 or TCP 443/2222 stream).

Q3 — Why is “Proxyable (FE→BE)” low for WG (2) and high for SSH (5)?

The rev. 2 score measures one thing: “Can we forward this service from Gentoo FE to Alpine BE the same way we already forward crashes/hub HTTP?”

	Reverse SSH	WireGuard
Protocol	TCP	UDP
Existing FE tool	nginx stream → 10.7.16.128:22	Not nginx location / proxy_pass
FE change type	Add stream {} block; reload nginx	Firewall DNAT + forward rules; separate from nginx HTTP config in repo
Ops familiarity	Same team that edits nginx.conf	Often different (iptables/nft, UDP, kernel module)
Debugging	nginx -t, error log, curl/nc TCP	nft list ruleset, UDP blocked on some carriers, wg show on BE
Matches INFRA docs	Extends nginx pattern	New parallel path not in current snippets

Score meaning:

- **5 (SSH):** “Drop in a stream upstream to BE — same deployment rhythm as today’s reverse proxy.”
- **2 (WG):** “Works, but **not** proxyable via HTTP nginx; needs **DNAT** (or a UDP proxy like udp2raw, which is extra complexity). Partial credit — not zero — because FE→BE forwarding is still one hop.”

Important nuance: Low **Proxyable** does **not** mean WG is insecure or bad on the phone — it means **your FE→BE operational model** favours TCP/nginx. That is why rev. 2 refined the v1 pick to SSH despite WG being **Lightweight** on-device.

Custom file API also scores 5 because it reuses existing proxy_pass http://artc0:80 with new PHP locations — no stream, no DNAT.

Rev. 3 — Bastion Q&A

Q4 — How easy to implement? Command checklist (FE + BE)

Rough effort (after control plane PHP/DB exists):

Track	FE ops	BE ops	App embed	Typical first-time
Reverse SSH + SFTP	1 nginx stream block + reload	openssh, sshd_config.d/ra.conf, session keys	libssh2/MINA client, -R forward	~0.5–1 day infra
WireGuard outbound	UDP DNAT + firewall + persist	wireguard-tools, wg0, dynamic peers	wireguard-android + VpnService	~1–2 days infra
Custom file API only	none (HTTP already proxied)	PHP endpoints only	OkHttp upload/download	~0 FE days

SSH bastion — minimal ordered checklist

FE (Gentoo):

```
nginx -t && nginx -s reload
```

```
iptables -A INPUT -p tcp --dport 2222 -j ACCEPT
```

BE (Alpine):

```
apk add openssh
```

```
mkdir -p /etc/ssh/sshd_config.d
```

```
rc-service sshd restart
```

App (Android): foreground service → heartbeat poll → on connect, run embedded SSH client with -R

127.0.0.1:<be_port>:127.0.0.1:8022 → local SFTP server or internal-sftp on device side (design choice).

Operator:

```
ssh -J f0xx.org:222 sftp -P 18022 ra-<session_id>@127.0.0.1
```

WireGuard — minimal ordered checklist

FE (Gentoo):

```
BE_IP=10.7.16.128
```

```
WG_PORT=51820
```

BE (Alpine):

```
apk add wireguard-tools
```

App (Android): VpnService + userspace WG → outbound to apps.f0xx.org:51820.

Operator (files still need a channel):

```
wg-quick up operator-ra-<session_id>
```

```
sftp user@10.66.66.2
```

WG **alone** does not give SFTP; you almost always add **SSH on top** or fall back to **Custom file API** — extra moving parts vs SSH-only track.

Q5 — Bastion: file transfers only, or other operations too?

Capability	Custom file API	SSH/SFTP bastion	WireGuard alone	WG + SSH on tunnel
Edit codecs.json	Yes (scoped paths)	Yes (SFTP)	No	Yes
Pull log files	Yes	Yes	No	Yes
Interactive shell (logcat, run-as, debug cmds)	No	Yes (if not ForceCommand internal-sftp)	No	Yes
Port forwarding (ADB-like / local tools)	No	Yes (-L/-R)	Yes (L3)	Yes
Packet capture / low-level network	No	Limited	Yes (L3 VPN)	Yes

What “bastion” means in this project:

- 1. **Narrow bastion (file-only v1):** ForceCommand internal-sftp, chroot or path jail to app home — enough for **codecs.json + logs** with smallest attack surface. Matches your stated primary need.
- 2. **Full debug bastion:** interactive shell + reverse forwards — for **logcat**, running one-off commands, inspecting /data/data/... (where policy allows), or forwarding a port to a laptop tool. Same SSH track, one config flag difference.
- 3. **WireGuard “bastion”:** really a **VPN endpoint**, not a file bastion — gives IP reachability; you still mount **SFTP/SSH or file API** for codecs.json unless you build a custom agent.

Practical recommendation for your decision:

Your priority	Best fit
Only replace codecs.json + download logs	Custom file API or SFTP-only SSH — bastion = file transfer only
Occasional shell debug on one field device	Reverse SSH bastion with shell allowed (not SFTP-only)
Lowest FE friction + familiar ops	Reverse SSH (stream to BE) — scores Proxiable 5
Minimum bytes on wire / best battery on tunnel	WireGuard — but budget +SSH or file API for files

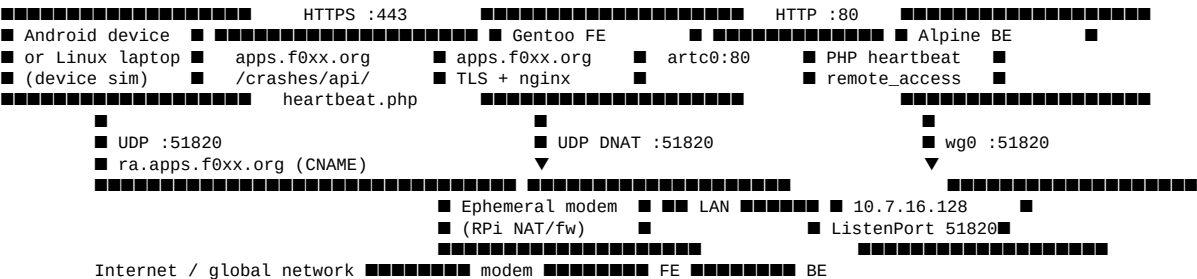
Rev. 3 decision hint: If bastion = **files only**, Custom file API + existing :80 heartbeat is the shortest path (no FE stream, no UDP DNAT). If bastion = **files + shell debug**, **reverse SSH** wins on infra fit. Pure **WG** only wins if you explicitly want L3 VPN and accept FE UDP DNAT + a second protocol for files.

Appendix 1 — WG test configuration (Linux laptop)

References: REMOTE_ACCESS_IMPL.md, REMOTE_ACCESS_VALIDATION.md.

DNS (prod): ra.apps.f0xx.org is a CNAME of apps.f0xx.org / f0xx.org — same public edge IP for HTTPS heartbeats and UDP WireGuard.

Network topology (v1 WireGuard)



Control plane = HTTP only (no new nginx location for RA). Data plane = UDP to ra.apps.f0xx.org:51820 → FE DNAT → BE wg0.

Prerequisites (laptop)

```
sudo emerge -av net-vpn/wireguard-tools # or: apt install wireguard-tools jq curl
```

A. HTTP E2E (no tunnel yet)

```
cd examples/crash_reporter/backend
export CRASHES_BASE=https://apps.f0xx.org/app/androidcast_project/crashes
export ADMIN_USER=admin ADMIN_PASS='...'

./scripts/ra_e2e_cli.sh
```

B. UDP / WireGuard from laptop (after open session)

Option 1 — device sim with tunnel bring-up:

```
export CRASHES_BASE=https://apps.f0xx.org/app/androidcast_project/crashes
export RA_DEVICE_ID=ra-laptop-$(hostname -s)
```

```
./scripts/ra_device_sim.sh poll --until connect --apply-wg
```

Option 2 — manual:

```
source scripts/ra_lib.sh
RESP="$(ra_ra_post enable "$RA_DEVICE_ID" "$(cat /tmp/androidcast-ra-${RA_DEVICE_ID}.random)" cli-sim)"
ra_wg_quick_from_connect "$RESP" | sudo tee /tmp/ra-laptop.conf
sudo wg-quick up /tmp/ra-laptop.conf
sudo wg show
```

```
ping -c3 10.66.66.1          # BE wg0 — OK if routed
sudo wg-quick down /tmp/ra-laptop.conf
```

C. Quick checks before blaming WG

Check	Command	Expect
CNAME	dig +short ra.apps.f0xx.org	same A/AAAA as apps.f0xx.org
UDP port	nc -vzu ra.apps.f0xx.org 51820	open / reachable (not TCP)
HTTP	curl -sS -o /dev/null -w '%{http_code}\n' https://apps.f0xx.org/app/androidcast_project/crashes/	200 or 302
BE wg	on BE: wg show wg0	interface up, peers after connect

If HTTP E2E passes but wg-quick up fails → FE DNAT or firewall on modem/FE, not PHP.

Appendix 2 — WG production configuration

References: REMOTE_ACCESS_IMPL.md § Production deploy, REMOTE_ACCESS_VALIDATION.md.

BE config.php (remote_access)

```
'remote_access' => [
    'wg_endpoint' => 'ra.apps.f0xx.org:51820',
    'wg_server_public_key' => '<output of wg show wg0 public-key on BE>',
    'wg_interface' => 'wg0',
    'provision_peers' => true,
    'require_wg_tools' => true,
    'session_ttl_s' => 3600,
    'min_poll_interval_s' => 45,
],
```

BE (Alpine) — WireGuard interface

```
apk add wireguard-tools wireguard-tools-wg
sudo wg-quick up wg0
sudo wg show wg0 public-key # → paste into config.php
PHP-FPM user (nginx on Alpine) must run wg set when provision_peers=true:
sudo -u nginx wg show wg0
Cron (hourly purge): php81 scripts/purge_remote_access.php --hours=24
```

FE (Gentoo) — UDP DNAT (not nginx)

HTTP stays as today: https://apps.f0xx.org → http://artc0.intra.raptor.org:80.
WireGuard uses **UDP DNAT** on the FE edge (and/or modem) — **not** an nginx location:
BE_IP=10.7.16.128 # Alpine BE on LAN
WG_PORT=51820
Ensure **UDP \$WG_PORT** is allowed on FE (and modem) toward the BE. ra.apps.f0xx.org CNAME ensures devices and laptops target the same edge as apps.f0xx.org.

MariaDB / migrations

Migration 007 (remote_access_* tables). See sql/migrations/007_remote_access.sql.

Production verify (on BE or laptop)

```
cd examples/crash_reporter/backend
BASE=https://apps.f0xx.org/app/androidcast_project/crashes ./scripts/verify_remote_access_prod.sh
RA_E2E=1 BASE="$BASE" ./scripts/verify_remote_access_prod.sh
Verified 2026-06-04: HTTP ra_e2e_cli.sh on prod returned connect with live peer_public_key and endpoint ra.apps.f0xx.org:51820.
UDP handshake is validated separately (Appendix 1 § B).
```

Android (field device)

Developer settings → Remote access → **WireGuard**; polls .../api/heartbeat.php (type: ra). See REMOTE_ACCESS_IMPL.md § App.

Source linkage

- Editable source: docs/20260602_REVERSE_SSH_proposals_summary.md
- PDF: docs/20260602_REVERSE_SSH_proposals_summary.pdf
- Regenerate: bash scripts/build-reverse-ssh-proposals-pdf.sh
- All docs/*.md PDFs: bash scripts/build-all-docs-pdf.sh