

Remote access — implementation notes (v1 WireGuard)

WireGuard v1 — implementation and deploy

Table of contents

App (developer settings)

Android VPN consent (WireGuard)

Production deploy (no new nginx HTTP paths)

1. Database
2. config.php — required keys
3. BE packages and permissions
4. Cron (session cleanup)
5. Verify
6. Browser

WireGuard data plane (UDP, not nginx)

WireGuard third-party (Android)

Tests

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

See the full proposal: 20260602_REVERSE_SSH_proposals_summary.md.

App (developer settings)

- Dropdown: **Disabled** | **WireGuard** (lab) | **RSSH** (alpha target — selectable when implemented)
- Pref key: dev_remote_access_mode (AppPreferences)
- **Disabled**: notifies BE (status:disable), tears down VPN, stops RemoteAccessService
- **WireGuard**: foreground service; jittered poll **1–7 min** → heartbeat.php with type: ra
- Heartbeat URL derived from crash upload URL (.../api/heartbeat.php)
- Session credentials persisted until expires_at; expiry tears down tunnel locally
- Optional full userspace WG via third-party/wireguard-android (:tunnel); otherwise RemoteAccessVpnService TUN fallback

Android VPN consent (WireGuard)

WireGuard on Android **must** use vpnService (See DeveloperSettingsActivity → vpnService.prepare()). This is **not avoidable** on stock devices:

UX	WireGuard (v1, lab)	RSSH (alpha essential)
One-time system dialog	Yes — “Allow VPN?”	No VPN permission
Status-bar key icon	Yes while tunnel up	No
Full-device routing	TUN interface (split routes in config, still a VPN)	No L3 tunnel
Outbound from phone	UDP to BE :51820	HTTPS poll + outbound SSH to bastion

There is no supported “hidden VPN” on non-root Android: any TUN creator is classified as a VPN app. **RSSH** (foreground service + existing HTTPS poll + outbound SSH reverse tunnel) is the **alpha deliverable** agreed by project owners — device initiates, operator reaches a forwarded port on the bastion, without VPN consent or status-bar key. See Rev. 3 Q&A and ROADMAP.md § Remote access. Uses the **existing crashes vhost** — no new nginx location blocks for HTTP.

Piece	Path
Migration	sql/migrations/007_remote_access.sql
Repository	src/RemoteAccessRepository.php
WG peers	src/WireGuardPeerProvisioner.php (wg set wg0 peer ...)
Device API	public/api/heartbeat.php (type: ra)
Admin API	public/api/remote_access.php
UI	?view=remote_access (nav + hub card)
RBAC	remote_access_view, remote_access_operate, remote_access_admin
RBAC admin UI	?view=rbac — global roles (root), company roles, privilege sets
Cron	scripts/purge_remote_access.php (hourly recommended)

RBAC rules (enforced in API)

Permission	Scope
remote_access_view	List devices/sessions/events in allowed companies
remote_access_operate	Open session; close own sessions
remote_access_admin	Whitelist devices; close any session in company
Global admin / root	All companies; root edits global roles via ?view=rbac

Privilege sets (remote_access_user, remote_access_admin) on company_memberships.permissions_json grant extra actions to member rows — see crash console README § RBAC.

Flow

1. Device polls with {type:ra, status:enable, capabilities:[wireguard], tunnel_mode:wireguard}.
2. BE upserts remote_access_devices; returns wait until device is **whitelisted** and operator **opens session**.
3. Next eligible poll returns action:connect + ephemeral WG credentials; BE runs wg set when provision_peers=true.
4. Disable from app → status:disable → closes sessions, removes peers.

URLs (production)

- Dashboard: https://apps.f0xx.org/app/androidcast_project/crashes/?view=remote_access
- Heartbeat: .../crashes/api/heartbeat.php
- Admin API: .../crashes/api/remote_access.php

Production deploy (no new nginx HTTP paths)

After syncing PHP/JS to the BE tree under
.../androidcast_project/android_cast/examples/crash_reporter/backend/:

1. Database

Migration **007** (already applied on prod per deploy notes). Tables auto-create on SQLite dev; MariaDB should use the migration file once as root.

2. config.php — required keys

```
'remote_access' => [
```

```

'wg_endpoint' => 'ra.apps.f0xx.org:51820',    // public UDP hostname:port (FE DNAT → BE)
'wg_server_public_key' => '...',              // wg show wg0 public-key on BE — REQUIRED
'wg_interface' => 'wg0',
'provision_peers' => true,                    // wg set on connect/close
'require_wg_tools' => true,                  // reject connect if wg genkey missing
'session_ttl_s' => 3600,
'min_poll_interval_s' => 45,                 // poll rate limit per device
],

```

Critical: wg_server_public_key must match the live wg0 interface. Empty value makes connect credentials unusable on real devices.

3. BE packages and permissions

```
apk add wireguard-tools wireguard-tools-wg
```

```
sudo -u www-data wg show wg0
```

4. Cron (session cleanup)

```
0 * * * * cd /var/www/.../backend && php81 scripts/purge_remote_access.php --hours=24
```

Use php81 when Alpine default php is 8.5+ without the session module; FPM is php-fpm81.

5. Verify

```

cd examples/crash_reporter/backend
chmod +x scripts/ra_.sh scripts/verify_remote_access_prod.sh scripts/test_remote_access_api.sh
./scripts/verify_remote_access_prod.sh
BASE=https://apps.f0xx.org/app/androidcast_project/crashes ./scripts/test_remote_access_api.sh
BASE=https://apps.f0xx.org/app/androidcast_project/crashes ./scripts/ra_e2e_cli.sh

```

Linux CLI device simulator (same heartbeat payloads as the app): REMOTE_ACCESS_VALIDATION.md.

6. Browser

Hard-refresh after deploying public/assets/js/remote_access.js.

WireGuard data plane (UDP, not nginx)

HTTP control plane stays on FE→BE :80. **WireGuard traffic is UDP** — configure at the **FE firewall/DNAT**, not in nginx:

Internet UDP :51820 → FE DNAT → BE wg0 :51820

Hostname ra.apps.f0xx.org (or similar) should resolve to the FE public IP. Devices use wg_endpoint from config/connect payload.

WireGuard third-party (Android)

```

bash scripts/init-wireguard-submodule.sh
./gradlew :app:assembleDebug # includes :tunnel when submodule present

```

Without the submodule, WireGuardTunnelBridge uses RemoteAccessVpnService (TUN only — sufficient for lab; prefer :tunnel for production tunnels).

Tests

```

./gradlew :app:testDebugUnitTest --tests 'com.foxx.androidcast.remoteaccess.*'
bash examples/crash_reporter/backend/scripts/test_rbac_api.sh
bash examples/crash_reporter/backend/scripts/test_remote_access_api.sh
BASE=https://apps.f0xx.org/app/androidcast_project/crashes bash examples/crash_reporter/backend/scripts/test_remote_access_api.sh
BASE=https://apps.f0xx.org/app/androidcast_project/crashes bash examples/crash_reporter/backend/scripts/ra_e2e_cli.sh
CLI validation guide: REMOTE_ACCESS_VALIDATION.md.

```