

RSSH routed egress (dev-only) — design review (DR)

From DRs/20260616_rssh_routed_egress.md

Field	Value
Author	Anton Afanasyeu
Revision	R0
Creation date	2026-06-16
Last modification date	2026-06-16
Co-authored	Cursor Agent (project assistant)
Severity	low
State	postponed
Document type	DR
Pre-requisite to	Finish WireGuard remote-access E2E (REMOTE_ACCESS_VALIDATION.md, VPN_DEMO_GAPS_20260613.md)

Table of contents

1. Executive summary
 2. Problem statement
 3. Vocabulary and scope
 4. Current state
 5. Goals and non-goals
 - Goals (if implemented later)
 - Non-goals
 6. Architecture options
 - 6.1 Option A — WireGuard full-tunnel (existing)
 - 6.2 Option B — RSSH + SSH -D (SOCKS) + VpnService + tun2socks
 - 6.3 Option C — SSH layer-3 tun (ssh -w)
 - 6.4 Option D — Hybrid (proposal doc)
 - 6.5 Comparison matrix
 7. DR recommendation
 8. Reference design (if pursued later)
 - 8.1 Control plane
 - 8.2 Data plane
 - 8.3 Routing semantics
 - 8.4 Failure modes
 9. Developer settings UX
 10. Backend and bastion requirements
 11. App components (sketch)
 12. Third-party and licenses
 13. Risks
 14. Effort estimate (order of magnitude)
 15. Task dependency graph
 16. Open questions (PO)
 17. Changelog
- Related docs

Source: `20260616_rssh_routed_egress.md` — section links work in PDF viewers that support internal anchors.

Document type: DR (Design Review)

Source draft: docs/drafts/20260616_rssh_routed_egress.txt

PDF: 20260616_rssh_routed_egress.pdf · Regenerate: `bash scripts/build-all-docs-pdf.sh`

Status: Draft for PO review — **no implementation** until WireGuard track is closed

Scope: Developer settings only; optional future path; does **not** replace alpha RSSH (reverse tunnel, no VPN)

Related: REMOTE_ACCESS_IMPL.md · 20260602_REVERSE_SSH_proposals_summary.md · ROADMAP.md (RSSH alpha)

1. Executive summary

Question: Can we add a developer-only, system-wide VPN-style path that routes part or all device traffic through the **RSSH** (reverse SSH) session?

Short answer: Yes, technically — but it requires a **new data plane** (VpnService + IP-to-SOCKS or TUN-over-SSH), not an extension of today's RSSH -R forward. It **reintroduces VPN permission and the status-bar key**, and overlaps **WireGuard full-tunnel**, which already implements route/app scope in the app.

DR decision (R0):

Item	Decision
Timing	Postponed until WireGuard remote-access work is finished and validated E2E
Alpha RSSH	Unchanged — outbound reverse SSH for operator → device; no device routing
Near-term routing	Use WireGuard + RemoteAccessVpnRouting (hub-only or full-tunnel)
If revived	Dev-only sub-mode “ RSSH + route traffic (experimental) ” via SSH dynamic forward (-D) + tun2socks + existing route/app prefs
SPEC	Do not write implementation SPEC until PO confirms after WG closure

2. Problem statement

Developers may want:

1. **Operator access** to the device (shell/SFTP) — **RSSH today**.
2. **Device traffic** to egress via the hub (e.g. Minsk NAT) or reach hub-private IPs — **WireGuard today** (split or full tunnel).
3. In some networks, **UDP WireGuard** may be blocked while **TCP 443 SSH** works — motivation to explore routing **over RSSH**.

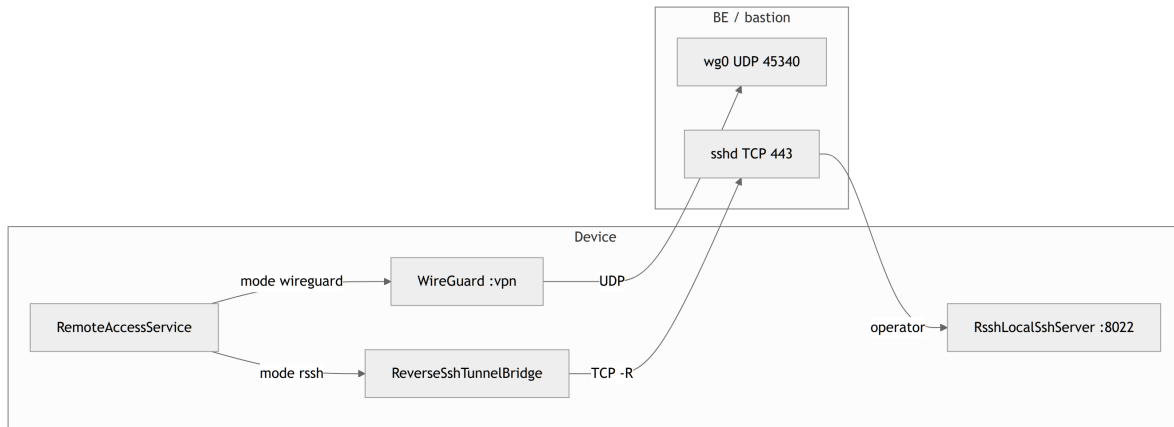
Without a clear DR, it is easy to assume vpn-route-scope / vpn-app-scope apply to RSSH; they currently apply to **WireGuard only** (REMOTE_ACCESS_IMPL.md).

3. Vocabulary and scope

Term	Meaning here
RSSH	Device-initiated reverse SSH; -R bastion:port → device:8022 for operator access
Routed egress	Device IP traffic (some or all) sent through bastion/hub NAT to the Internet or private nets
VpnService	Android API for app-provided VPN; required for system-wide or per-app routing from our app
Route scope	Hub-only (172.200.2.1/32) vs full-tunnel (0.0.0.0/0) — RemoteAccessVpnRouting
App scope	All apps vs this app only — IncludedApplications / addAllowedApplication

Not in scope for this DR: Commercial VPN product (F4), always-on VPN in Android Settings, routing for non-developer builds.

4. Current state



Mode	Transport	Device routing	VPN icon
WireGuard	UDP → wg0	Yes (hub-only or full)	Yes
RSSH	TCP → sshd	No	No

Existing prefs (dev_remote_access_vpn_route_scope, dev_remote_access_vpn_app_scope) are sent on heartbeat for WG; RSSH ignores them.

5. Goals and non-goals

Goals (if implemented later)

ID	Goal
G1	Optional dev-only routing of device traffic via TCP 443 SSH path
G2	Reuse route scope and app scope semantics from RemoteAccessVpnRouting
G3	Network self-test shows bastion, bind ports, egress expectation (like WG vpn-ip)
G4	Clear separation from alpha RSSH (reverse access only)

Non-goals

ID	Non-goal
NG1	Replace WireGuard for production/lab routing
NG2	Alpha release blocker
NG3	Avoid VpnService.prepare() — routed mode will need it
NG4	SSH PermitTunnel / ssh -w as v1 (too heavy for Alpine bastion v1)
NG5	Third-party "VPN provider" in Android Settings (no such API; we use VpnService)

6. Architecture options

6.1 Option A — WireGuard full-tunnel (existing)

Use when: Hub egress or full WAN via Minsk NAT is needed **now**.

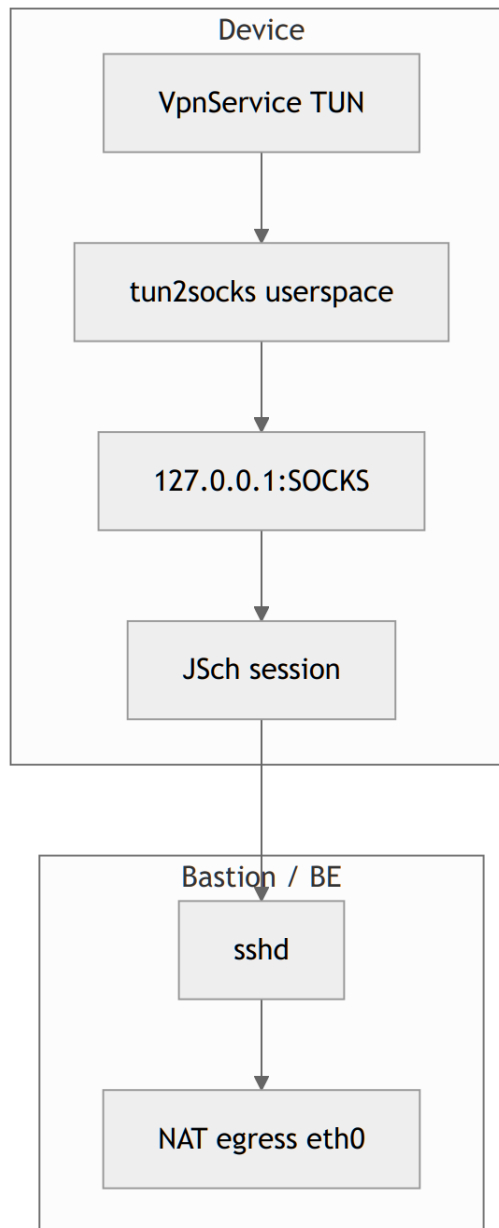
- Dev settings → WireGuard → route scope **All traffic** → reconnect session.
- BE wg0 PostUp MASQUERADE already documented.

Pros	Cons
Implemented	UDP DNAT + VPN consent
Kernel/userspace WG	Not RSSH transport

Verdict: Default for routing until WG track is done.

6.2 Option B — RSSH + SSH -D (SOCKS) + VpnService + tun2socks

Use when: PO explicitly needs **TCP-only** routing and accepts VPN icon + extra stack.



1. Extend ReverseSshTunnelBridge: keep -R for operator; add -D 127.0.0.1:<socks_port> on same session.
2. New **RsshRoutedVpnService** (or branch in :vpn) establishes TUN with routes from RemoteAccessVpnRouting.
3. **tun2socks** forwards IP packets → local SOCKS → SSH stream → bastion → Internet.

Pros	Cons
TCP 443 friendly	VPN permission + icon
Reuses SSH session	tun2socks dep + licenses
Reuses route/app prefs	TCP-over-TCP, MTU, battery
	Bastion egress policy + capacity
	JSch dynamic forwarding QA

Verdict: Reference design if PO revives routed RSSH.

6.3 Option C — SSH layer-3 tun (ssh -w)

Both ends need TUN, PermitTunnel, root/capabilities on bastion, custom Android client.

Verdict: Reject for dev-only v1 (operational cost).

6.4 Option D — Hybrid (proposal doc)

WG for packets + RSSH (or SSH over WG) for operator files.

Verdict: Valid long-term; does not require RSSH alone to carry IP routing.

6.5 Comparison matrix

Criterion	A WG	B RSSH+SOCKS	C ssh -w	D hybrid
-----------	------	--------------	----------	----------

Ready today	Yes	No	No	Partial
TCP 443 only	No	Yes	Yes	Mixed
VPN icon	Yes	Yes	Yes	Yes
Alpha RSSH purity	N/A	Weakens if default	Weakens	OK
Impl effort	Done	High	Very high	Medium

7. DR recommendation

1. **Close WireGuard E2E** (peer reconcile, self-test vpn-ip, hub→phone diagnostics, full-tunnel egress check) — **current priority**.
2. **Keep alpha RSSH** as reverse tunnel only (ROADMAP.md).
3. **Park Option B** in this DR; PO reviews §16 and decides **go / no-go** after WG.
4. If **go**: spawn SPEC rssh_routed_egress (dev-only feature flag), not a change to core RSSH connect payload for alpha.

8. Reference design (if pursued later)

8.1 Control plane

Unchanged: heartbeat.php type: ra, tunnel: ssh_reverse, credentials for bastion + -R port.

Optional dev flag in heartbeat meta: rssh_route_experimental: true (telemetry only; no BE requirement for v0).

8.2 Data plane

Step	Component
1	ReverseSshTunnelBridge connects; -R as today
2	Same session: session.setPortForwardingD("127.0.0.1", socksPort) (JSch)
3	RsshRoutedVpnEngine starts TUN via VpnService
4	tun2socks reads TUN, connects to 127.0.0.1:socksPort
5	Bastion sshd forwards; iptables MASQUERADE on bastion for egress

8.3 Routing semantics

Reuse RemoteAccessVpnRouting.resolveAllowedIps():

Route scope	TUN routes
HUB_ONLY	172.200.2.1/32 (or configured hub CIDR)
FULL_TUNNEL	0.0.0.0/0, ::/0

Reuse appendInterfaceExtras() for **THIS_APP_ONLY**.

Note: Hub reachability over RSSH may need **explicit route to hub IP via TUN** or SOCKS; validate that hub-only does not assume WG interface addresses.

8.4 Failure modes

Symptom	Likely cause
SOCKS up, no Internet	Bastion NAT / forwarding disabled
Tunnel up, wrong egress IP	Route scope hub-only (expected)
Intermittent stalls	TCP-over-TCP congestion
VPN works, operator SFTP dead	-R forward dropped; session limits

9. Developer settings UX

Proposed (not implemented):

Remote access mode: [Disabled | WireGuard | RSSH]

– When WireGuard –
 VPN route scope: Hub only | All traffic
 VPN app scope: All apps | This app only

– When RSSH –
 [] Route device traffic via bastion (experimental)
 (shows route/app scope only when checked)
 Hint: Reverse tunnel for operator access is always on in RSSH mode.
 Routing requires VPN permission (same as WireGuard).

Network self-test (RSSH + experimental): rssh-bastion, rssh-remote-bind, rssh-socks, wan-egress-via-vpn, rssh-connected.

10. Backend and bastion requirements

Requirement	Notes
-------------	-------

AllowTcpForwarding yes	Already for -R
Dynamic forwarding	Confirm sshd GatewayPorts / no restrictive Match block for -D
Egress NAT	iptables/nft MASQUERADE on bastion (similar to wg0 PostUp)
Capacity	Routed dev traffic adds bandwidth; separate from §12.1 RSSH SSH counts in scaling DR
Audit	Log experimental flag; no change to alpha session schema required

No new nginx HTTP paths; optional stream unchanged.

11. App components (sketch)

Component	Process	Role
ReverseSshTunnelBridge	main	SSH session; -R + -D
RsshRoutedVpnService	:vpn	TUN + lifecycle
RsshTun2SocksEngine	:vpn	Packet relay
RemoteAccessVpnRouting	shared	Route/app scope
DevVpnStatusProbe	main	Diagnostics

Do **not** fold into wireGuardVpnEngine; separate engine behind common vpnServiceClient interface if refactored later.

12. Third-party and licenses

Candidate	License	Action if adopted
badvpn / tun2socks	Various	Evaluate; add to third-party/, mobile + BE license files
JSch	BSD	Already in app; verify -D support
Apache MINA SSHD	Apache-2.0	Local server only today

Update COMMERCIAL.md and app/src/main/assets/licenses/ before merge.

13. Risks

Risk	Mitigation
Duplicates WG	PO go/no-go; default to WG
Alpha scope creep	Dev-only flag; off in release if needed
VPN consent UX regression	Document in FR; separate from "no VPN" RSSH story
Bastion abuse (open egress)	Dev whitelist + session TTL + rate limits
Maintenance burden	Two routing stacks

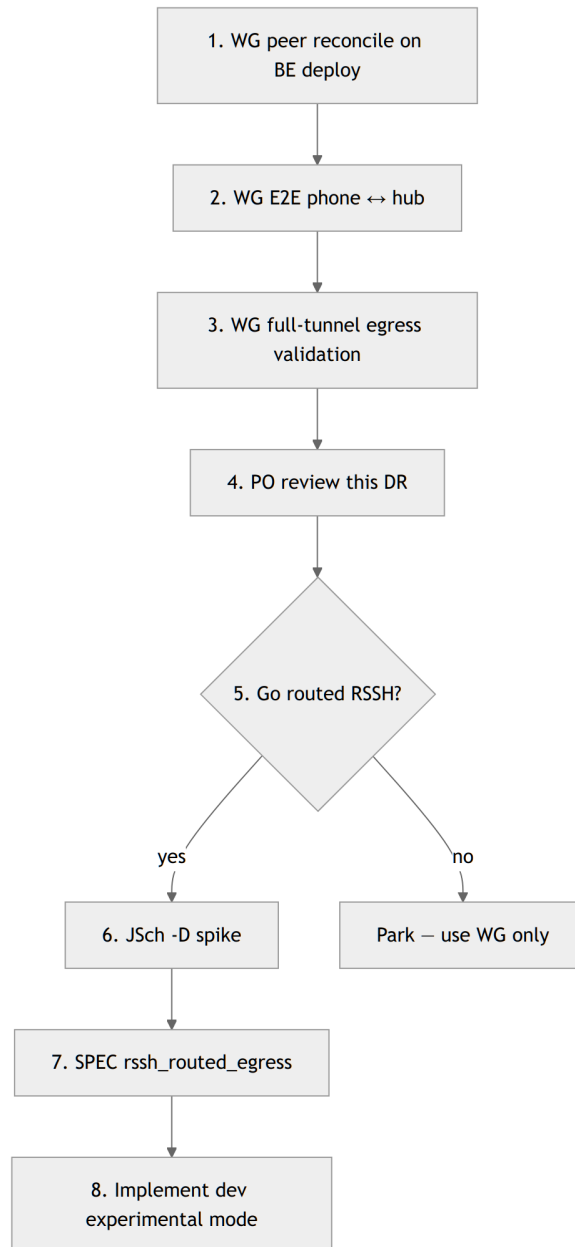
14. Effort estimate (order of magnitude)

Phase	Work	Duration (indicative)
P0	WG E2E closure	Current sprint (PO priority)
P1	Spike: JSch -D + manual SOCKS browser test	2–3 d
P2	tun2socks + VpnService + route prefs	5–8 d
P3	Bastion NAT + self-test + docs	2–3 d
P4	Unit/integration tests, soak	3–5 d

Total if pursued: ~3–4 weeks **after** WG done — not parallelized with WG closure unless PO overrides.

15. Task dependency graph

Legend: **blocked by** upstream task.



Priority	Task	Owner	State
1	Finish WG (peer sync, ping/handshake, self-test, full-tunnel)	Dev/PO	In progress
2	Review DR R0	PO	Pending
3	RSSH routed egress	—	Postponed

16. Open questions (PO)

#	Question	Default if silent
Q1	After WG E2E, is TCP-only routing still a requirement?	No — stay WG-only for routing
Q2	Accept VPN icon for an experimental RSSH routing mode?	Required for Option B
Q3	Should hub-only over RSSH target 172.200.2.1 without WG?	Clarify use case
Q4	Bastion egress NAT for dev traffic — allowed on prod be-vpn?	Lab-only
Q5	Single SSH session for -R + -D or two sessions?	Prefer one session

17. Changelog

Rev	Date	Change
R0	2026-06-16	Initial DR from PO question; postponed pending WG closure

Related docs

- REMOTE_ACCESS_IMPL.md — WG vs RSSH modes
- REMOTE_ACCESS_VALIDATION.md — CLI/mobile WG checks
- VPN_DEMO_GAPS_20260613.md — WG demo gaps
- 20260602_REVERSE_SSH_proposals_summary.md — original transport comparison
- specs/20100612_1_scaling.md — be-vpn, RSSH capacity