changelog.md for @push.rocks/smartvpn

2026-03-31 - 1.17.1 - fix(readme)

document per-transport metrics and handshake-driven WireGuard connection state

- Add README examples for getStatistics() per-transport active client and total connection counters
- Clarify that WireGuard peers are marked connected only after a successful handshake and disconnect after idle timeout
- Refresh API and project structure documentation to reflect newly documented stats fields and source files

2026-03-31 - 1.17.0 - feat(wireguard)

track per-transport server statistics and make WireGuard clients active only after handshake

- add websocket, quic, and wireguard active-client and total-connection counters to server statistics
- register WireGuard peers without marking them active until handshake/data is received, and remove them from active clients on expiration or idle timeout
- sync WireGuard byte counters into aggregate server stats independently of active client presence and expose new statistics fields in TypeScript interfaces

2026-03-31 - 1.16.5 - fix(rust-userspace-nat)

improve TCP session backpressure, buffering, and idle cleanup in userspace NAT

- apply proper bridge-channel backpressure by reserving channel capacity before consuming smoltcp TCP data
- defer bridge sender initialization until the bridge task starts and track TCP session activity timestamps
- cap per-session pending TCP send buffers at 512KB and abort stalled sessions when clients cannot keep up
- add idle TCP session cleanup and switch NAT polling to a dynamic smoltcp-driven delay

2026-03-31 - 1.16.4 - fix(server)

register preloaded WireGuard clients as peers on server startup

- Adds configured clients from the runtime registry to the WireGuard listener when the server starts.
- Ensures clients loaded from config can complete WireGuard handshakes without requiring separate peer registration.
- Logs a warning if automatic peer registration fails for an individual client.

2026-03-31 - 1.16.3 - fix(rust-nat)

defer TCP bridge startup until handshake completion and buffer partial NAT socket writes

- Start TCP bridge tasks only after the smoltcp socket becomes active to prevent server data from arriving before the client handshake completes.
- Buffer pending TCP payloads and flush partial writes so bridge-to-socket data is not silently lost under backpressure.
- Keep closing TCP sessions alive until FIN processing completes and add logging for dropped packets when bridge or route channels are full.

2026-03-31 - 1.16.2 - fix(wireguard)

sync runtime peer management with client registration and derive the correct server public key from the WireGuard private key

- Register, remove, and rotate WireGuard peers in the running listener when clients are added, deleted, or rekeyed.
- Generate client WireGuard configs with the public key derived from the configured WireGuard private key instead of reusing the generic server public key.
- Handle expired WireGuard sessions by re-initiating handshakes and mark client state as handshaking until the tunnel becomes active.
- Improve allowed IP matching and peer VPN IP extraction for runtime packet routing.

2026-03-30 - 1.16.1 - fix(rust/server)

add serde alias for clientAllowedIPs in server config

- Accepts the camelCase clientAllowedIPs field when deserializing server configuration.
- Improves compatibility with existing or external configuration formats without changing runtime behavior.

2026-03-30 - 1.16.0 - feat(server)

add configurable client endpoint and allowed IPs for generated VPN configs

- adds serverEndpoint to generated SmartVPN and WireGuard client configs so remote clients can use a public address instead of the listen address
- adds clientAllowedIPs to generated WireGuard configs to support full-tunnel or split-tunnel routing
- updates TypeScript interfaces to expose the new server configuration options

2026-03-30 - 1.15.0 - feat(vpnserver)

add nftables-backed destination policy enforcement for TUN mode

- add @push.rocks/smartnftables dependency and export it through the plugin layer
- apply destination policy rules via nftables when starting the server in TUN mode
- add periodic nftables health checks and best-effort cleanup on server stop

- update documentation for destination routing policy, socket transport mode, trusted client tags, events, and service generation

2026-03-30 - 1.14.0 - feat(nat)

add destination routing policy support for socket-mode VPN traffic

- introduce configurable destinationPolicy settings in server and TypeScript interfaces
- apply allow, block, and forceTarget routing decisions when creating TCP and UDP NAT sessions
- export ACL IP matching helper for destination policy evaluation

2026-03-30 - 1.13.0 - feat(client-registry)

separate trusted server-defined client tags from client-reported tags with legacy tag compatibility

- Adds distinct serverDefinedClientTags and clientDefinedClientTags fields to client registry and TypeScript interfaces.
- Treats legacy tags values as serverDefinedClientTags during deserialization and server-side create/update flows for backward compatibility.
- Clarifies that only server-defined tags are trusted for access control while client-defined tags are informational only.

2026-03-30 - 1.12.0 - feat(server)

add optional PROXY protocol v2 headers for socket-based userspace NAT forwarding

- introduce a socketForwardProxyProtocol server option in Rust and TypeScript interfaces
- pass the new setting into the userspace NAT engine and TCP bridge tasks
- prepend PROXY protocol v2 headers on outbound TCP connections when socket forwarding is enabled

2026-03-30 - 1.11.0 - feat(server)

unify WireGuard into the shared server transport pipeline

- add integrated WireGuard server support to VpnServer with shared startup, shutdown, status, statistics, and peer management
- introduce transportMode 'all' as the default and add server config support for wgPrivateKey, wgListenPort, and preconfigured peers
- register WireGuard peers in the shared client registry and IP pool so they use the same forwarding engine, routing, and monitoring as WebSocket and QUIC clients
- expose transportType in server client info and update TypeScript interfaces and documentation to reflect unified multi-transport forwarding

2026-03-30 - 1.10.2 - fix(client)

wait for the connection task to shut down cleanly before disconnecting and increase test timeout

- store the spawned client connection task handle and await it during disconnect with a 5 second timeout so the disconnect frame can be sent before closing
- increase the test script timeout from 60 seconds to 90 seconds to reduce flaky test runs

2026-03-29 - 1.10.1 - fix(test, docs, scripts)

correct test command verbosity, shorten load test timings, and document forwarding modes

- Fixes the test script by removing the duplicated verbose flag in package.json.
- Reduces load test delays and burst sizes to keep keepalive and connection tests faster and more stable.
- Updates the README to describe forwardingMode options, userspace NAT support, and related configuration examples.

2026-03-29 - 1.10.0 - feat(rust-server, rust-client, ts-interfaces)

add configurable packet forwarding with TUN and userspace NAT modes

- introduce forwardingMode options for client and server configuration interfaces
- add server-side forwarding engines for kernel TUN, userspace socket NAT, and testing mode
- add a smoltcp-based userspace NAT implementation for packet forwarding without root-only TUN routing
- enable client-side TUN forwarding support with route setup, packet I/O, and cleanup
- centralize raw packet destination IP extraction in tunnel utilities for shared routing logic
- update test command timeout and logging flags

2026-03-29 - 1.9.0 - feat(server)

add PROXY protocol v2 support for real client IP handling and connection ACLs

- add PROXY protocol v2 parsing for WebSocket connections, including IPv4/IPv6 support, LOCAL command handling, and header read timeout protection
- apply server-level connection IP block lists before the Noise handshake and enforce per-client source IP allow/block lists using the resolved remote address
- expose proxy protocol configuration and remote client address fields in Rust and TypeScript interfaces, and document reverse-proxy usage in the README

2026-03-29 - 1.8.0 - feat(auth,client-registry)

add Noise IK client authentication with managed client registry and per-client ACL controls

- switch the native tunnel handshake from Noise NK to Noise IK and require client keypairs in client configuration
- add server-side client registry management APIs for creating, updating, disabling, rotating, listing, and exporting client configs
- enforce client authorization from the registry during handshake and expose authenticated client metadata in server client info
- introduce per-client security policies with source/destination ACLs and per-client rate limit settings
- add Rust ACL matching support for exact IPs, CIDR ranges, wildcards, and IP ranges with test coverage

2026-03-29 - 1.7.0 - feat(rust-tests)

add end-to-end WireGuard UDP integration tests and align TypeScript build configuration

- Add userspace Rust end-to-end tests that validate WireGuard handshake, encryption, peer isolation, and preshared-key data exchange over real UDP sockets.
- Update the TypeScript build setup by removing the allowimplicitany build flag and explicitly including Node types in tsconfig.
- Refresh development toolchain versions to support the updated test and build workflow.

2026-03-29 - 1.6.0 - feat(readme)

document WireGuard transport support, configuration, and usage examples

- Expand the README from dual-transport to triple-transport support by adding WireGuard alongside WebSocket and QUIC
- Add client and server WireGuard examples, including live peer management and .conf generation with WgConfigGenerator
- Document new WireGuard-related API methods, config fields, transport modes, and security model details

2026-03-29 - 1.5.0 - feat(wireguard)

add WireGuard transport support with management APIs and config generation

- add Rust WireGuard module integration using boringtun and route management through client/server management handlers
- extend TypeScript client and server configuration schemas with WireGuard-specific options and validation
- add server-side WireGuard peer management commands including keypair generation, peer add/remove, and peer listing
- export a WireGuard config generator for producing client and server .conf files
- add WireGuard-focused test coverage for config validation and config generation

2026-03-21 - 1.4.1 - fix(readme)

preserve markdown line breaks in feature list

- Adds trailing spaces to the README feature list so each highlighted capability renders on its own line.

2026-03-19 - 1.4.0 - feat(vpn transport)

add QUIC transport support with auto fallback to WebSocket

- introduces a transport abstraction in the Rust daemon so client and server can operate over WebSocket or QUIC
- adds dual-mode server configuration with websocket, quic, and both transport modes plus QUIC idle timeout and listen address options
- adds client transport selection with auto mode that attempts QUIC first and falls back to WebSocket
- adds QUIC certificate hash pinning support and required Rust dependencies for QUIC and TLS
- updates TypeScript interfaces, config validation, tests, and documentation to cover the new transport modes

2026-03-17 - 1.3.0 - feat(tests,client)

add flow control and load test coverage and honor configured keepalive intervals

- Adds end-to-end node tests for client/server flow control, keepalive exchange, connection quality telemetry, rate limiting, concurrent clients, and disconnect tracking.
- Adds load testing with throttled proxy scenarios to validate behavior under constrained bandwidth and repeated client churn.
- Updates the Rust client to pass configured keepaliveIntervalSecs into the adaptive keepalive monitor instead of always using defaults.

2026-03-15 - 1.2.0 - feat(readme)

document QoS, telemetry, MTU, and rate limiting capabilities in the README

- Expand the architecture and feature overview to cover adaptive keepalive, telemetry, QoS, rate limiting, and MTU handling
- Update client and server examples to show new APIs such as getConnectionQuality(), getMtuInfo(), setClientRateLimit(), and getClientTelemetry()
- Add TypeScript interface documentation for connection quality, MTU info, enriched client statistics, and per-client telemetry

2026-03-15 - 1.1.0 - feat(rust-core)

add adaptive keepalive telemetry, MTU handling, and per-client rate limiting APIs

- adds adaptive keepalive monitoring with RTT, jitter, loss, and link health reporting to client statistics and management endpoints
- introduces MTU overhead calculation and oversized-packet handling support, plus client MTU info APIs
- adds token-bucket rate limiting with configurable default limits and server management commands to set, remove, and inspect per-client telemetry
- extends TypeScript client and server interfaces with connection quality, MTU, and client telemetry methods

2026-02-27 - 1.0.3 - fix(build)

add aarch64 linker configuration for cross-compilation

- Added rust/.cargo/config.toml to configure linker for target aarch64-unknown-linux-gnu
- Sets linker to 'aarch64-linux-gnu-gcc' to enable cross-compilation to ARM64

2026-02-27 - 1.0.2 - fix()

no changes detected - no code or content modifications

2026-02-27 - 1.0.1 - fix(release)

bump patch version (no code changes)

- No changes detected in the provided git diff
- Current package.json version is 1.0.0
- Recommend patch bump to 1.0.1 to create a release/trivial update

2026-02-27 - 1.0.0 - initial release

Initial commit creating the project repository and baseline files.

- Initial project scaffold and configuration
- Repository initialized with base files and metadata

Revision #6
Created 2026-03-28 13:11:37 UTC by foss.global Team
Updated 2026-03-31 14:22:36 UTC by foss.global Team