

changelog.md for @serve.zone/remotingress

2026-03-27 - 4.15.3 - fix(core)

harden UDP session handling, QUIC control message validation, and bridge process cleanup

- cap UDP session creation and drop excess datagrams with warnings to prevent unbounded session growth
- periodically prune closed datagram sessions on the hub and reject oversized QUIC control messages to avoid resource exhaustion
- clean up spawned edge and hub bridge processes on startup failure, remove listeners on stop, and avoid restarting after shutdown during backoff

2026-03-26 - 4.15.2 - fix(readme)

adjust tunnel diagram alignment in the README

- Improves formatting consistency in the Hub/Edge topology diagram.

2026-03-26 - 4.15.1 - fix(readme)

clarify unified runtime configuration and firewall update behavior

- Updates the architecture and feature descriptions to reflect that ports, firewall rules, and rate limits are pushed together in a single config update
- Clarifies that firewall configuration is delivered via FRAME_CONFIG on handshake and subsequent updates, with atomic full-rule replacement at the edge
- Simplifies and reorganizes README wording around edge and hub responsibilities without changing implementation behavior

2026-03-26 - 4.15.0 - feat(edge, hub)

add hub-controlled nftables firewall configuration for remote ingress edges

- add firewallConfig support to allowed edge definitions, handshake responses, and runtime config updates
- emit firewallConfigUpdated events from the Rust bridge and edge runtime when firewall settings change
- initialize SmartNftables on edges, apply blocked IPs, rate limits, and custom rules, and clean up nftables rules on stop
- document centralized firewall management, root requirements, and new edge events in the README

2026-03-26 - 4.14.3 - fix(docs)

refresh project metadata and README to reflect current ingress tunnel capabilities

- update package metadata description and keywords to better describe edge ingress, TLS/QUIC transport, and SmartProxy integration
- revise README terminology, API docs, and feature list to document crash recovery, bindAddress support, and current event names
- improve README formatting and examples for architecture, wire protocol, QoS, and token usage

2026-03-26 - 4.14.2 - fix(hub-core)

improve stream shutdown handling and connection cleanup in hub and edge

- Cancel edge upload loops immediately when the hub closes a stream instead of waiting for the window stall timeout.
- Reduce stalled stream timeouts from 120s to 55s to detect broken connections faster.
- Allow hub writer tasks to shut down gracefully before aborting to avoid unnecessary TCP resets.
- Enable TCP keepalive on hub upstream connections to detect silent SmartProxy failures.
- Remove leaked QUIC UDP session entries when setup fails or sessions end.
- Rename npxmextra.json to .smartconfig.json and update package packaging references.

2026-03-21 - 4.14.1 - fix(remoteingress edge/hub crash recovery)

prevent duplicate crash recovery listeners and reset saved runtime state on shutdown

- Removes existing exit listeners before re-registering crash recovery handlers for edge and hub processes.
- Clears saved edge and hub configuration on stop to avoid stale restart state.
- Resets orphaned edge status intervals and restarts periodic status logging after successful crash recovery.

2026-03-20 - 4.14.0 - feat(quic)

add QUIC stability test coverage and bridge logging for hub and edge

- adds a long-running QUIC stability test with periodic echo probes and disconnect detection
- enables prefixed bridge logging for RemoteIngressHub and RemoteIngressEdge to improve runtime diagnostics

2026-03-20 - 4.13.2 - fix(remoteingress-core)

preserve reconnected edge entries during disconnect cleanup

- Guard edge removal so disconnect handlers only delete entries whose cancel token is already cancelled
- Prevents stale TCP and QUIC disconnect paths from removing a newer connection after an edge reconnects

2026-03-19 - 4.13.1 - fix(remoteingress-core)

default edge transport mode to QUIC with fallback

- Changes the default transport mode in edge connections from TCP/TLS to QUIC with fallback when no transport mode is explicitly configured.

2026-03-19 - 4.13.0 - feat(docs)

document TCP and UDP tunneling over TLS and QUIC

- update package description to reflect TCP and UDP support and TLS or QUIC transports
- refresh README architecture, features, and usage examples for UDP forwarding, QUIC transport, and PROXY protocol v1/v2 support

2026-03-19 - 4.12.1 - fix(remoteingress-core)

send PROXY v2 headers for UDP upstream sessions and expire idle UDP sessions

- Adds periodic idle UDP session expiry in edge tunnel and QUIC loops, including UDP close signaling for expired tunnel sessions.
- Sends the PROXY v2 header as the first datagram for UDP upstream connections in both standard and QUIC hub paths.
- Updates the UDP node test server to ignore the initial PROXY v2 datagram per source before echoing payload traffic.

2026-03-19 - 4.12.0 - feat(remoteingress-core)

add UDP tunneling over QUIC datagrams and expand transport-specific test coverage

- Implement QUIC datagram-based UDP forwarding on both edge and hub, including session setup, payload routing, and listener cleanup
- Enable QUIC datagram receive buffers in client and server transport configuration
- Add UDP-over-QUIC tests and clarify existing test names to distinguish TCP/TLS, UDP/TLS, and QUIC scenarios

2026-03-19 - 4.11.0 - feat(remoteingress-core)

add UDP tunneling support between edge and hub

- extend edge and hub handshake/config updates with UDP listen ports
- add UDP tunnel frame types and PROXY protocol v2 header helpers in the protocol crate
- introduce UDP session management on the edge and upstream UDP forwarding on the hub
- add Node.js integration tests covering UDP echo and concurrent datagrams

- expose UDP listen port configuration in the TypeScript hub API

2026-03-19 - 4.10.0 - feat(core,edge,hub,transport)

add QUIC tunnel transport support with optional edge transport selection

- adds a shared transport module with QUIC configuration helpers, control message framing, and PROXY header handling
- enables the hub to accept QUIC connections on the tunnel port alongside existing TCP/TLS support
- adds edge transportMode configuration with quic and quicWithFallback options and propagates it through restarts
- includes end-to-end QUIC transport tests covering large payloads and concurrent streams

2026-03-18 - 4.9.1 - fix(readme)

document QoS tiers, heartbeat frames, and adaptive flow control in the protocol overview

- Adds PING, PONG, WINDOW_UPDATE, and WINDOW_UPDATE_BACK frame types to the protocol documentation
- Describes the 3-tier priority queues for control, normal data, and sustained traffic
- Explains sustained stream classification and adaptive per-stream window sizing

2026-03-18 - 4.9.0 - feat(protocol)

add sustained-stream tunnel scheduling to isolate high-throughput traffic

- Introduce a third low-priority sustained queue in Tunnello with a forced drain budget to prevent long-lived high-bandwidth streams from starving control and normal data frames.
- Classify upload and download streams as sustained after exceeding the throughput threshold for the minimum duration, and route their DATA and CLOSE frames through the sustained channel.
- Wire the new sustained channel through edge and hub stream handling so sustained traffic is scheduled consistently on both sides of the tunnel.

2026-03-18 - 4.8.19 - fix(remoteingress-protocol)

reduce per-stream flow control windows and increase control channel buffering

- Lower the initial and maximum per-stream window from 16MB to 4MB and scale adaptive windows against a 200MB total budget with a 1MB minimum.
- Increase edge and hub control frame channel capacity from 256 to 512 to better handle prioritized control traffic.
- Update flow-control tests and comments to reflect the new window sizing and budget behavior.

2026-03-17 - 4.8.18 - fix(rust-protocol)

switch tunnel frame buffers from Vec to Bytes to reduce copying and memory overhead

- Add the bytes crate to core and protocol crates
- Update frame encoding, reader payloads, channel queues, and stream backchannels to use Bytes
- Adjust edge and hub data/control paths to send framed payloads as Bytes

2026-03-17 - 4.8.17 - fix(protocol)

increase per-stream flow control windows and remove adaptive read caps

- Raise the initial per-stream window from 4MB to 16MB and expand the adaptive window budget to 800MB with a 4MB floor
- Stop limiting edge and hub reads by the adaptive per-stream target window, keeping reads capped only by the current window and 32KB chunk size
- Update protocol tests to match the new adaptive window scaling and budget boundaries

2026-03-17 - 4.8.16 - fix(release)

bump package version to 4.8.15

- Updates the package.json version field from 4.8.13 to 4.8.15.

2026-03-17 - 4.8.13 - fix(remoteingress-protocol)

require a flush after each written frame to bound TLS buffer growth

- Remove the unflushed byte threshold and stop queueing additional writes while a flush is pending
- Simplify write and flush error logging after dropping unflushed byte tracking
- Update tunnel I/O comments to reflect the stricter flush behavior that avoids OOM and connection resets

2026-03-17 - 4.8.12 - fix(tunnel)

prevent tunnel backpressure buffering from exhausting memory and cancel stream handlers before TLS shutdown

- stop self-waking and writing new frames while a flush is pending to avoid unbounded TLS session buffer growth under load
- reorder edge and hub shutdown cleanup so stream cancellation happens before TLS close_notify, preventing handlers from blocking on dead channels
- add load tests covering sustained large transfers, burst traffic, and rapid stream churn to verify tunnel stability

2026-03-17 - 4.8.11 - fix(remoteingress-core)

stop data frame send loops promptly when stream cancellation is triggered

- Use cancellation-aware tokio::select! around data channel sends in both edge and hub stream forwarding paths
- Prevent stalled or noisy shutdown behavior when stream or client cancellation happens while awaiting frame delivery

2026-03-17 - 4.8.10 - fix(remoteingress-core)

guard tunnel frame sends with cancellation to prevent async send deadlocks

- Wrap OPEN, CLOSE, CLOSE_BACK, WINDOW_UPDATE, and cleanup channel sends in cancellation-aware tokio::select! blocks.
- Avoid indefinite blocking when tunnel, stream, or writer tasks are cancelled while awaiting channel capacity.
- Improve shutdown reliability for edge and hub stream handling under tunnel failure conditions.

2026-03-17 - 4.8.9 - fix(repo)

no changes to commit

2026-03-17 - 4.8.8 - fix(remoteingress-core)

cancel stale edge connections when an edge reconnects

- Remove any existing edge entry before registering a reconnected edge
- Trigger the previous connection's cancellation token so stale sessions shut down immediately instead of waiting for TCP keepalive

2026-03-17 - 4.8.7 - fix(remoteingress-core)

perform graceful TLS shutdown on edge and hub tunnel streams

- Send TLS close_notify before cleanup to avoid peer disconnect warnings on both tunnel endpoints
- Wrap stream shutdown in a 2 second timeout so connection teardown does not block cleanup

2026-03-17 - 4.8.6 - fix(remoteingress-core)

initialize disconnect reason only when set in hub loop break paths

- Replace the default "unknown" disconnect reason with an explicitly assigned string and document that all hub loop exits set it before use
- Add an allow attribute for unused assignments to avoid warnings around the deferred initialization pattern

2026-03-17 - 4.8.5 - fix(repo)

no changes to commit

2026-03-17 - 4.8.4 - fix(remoteingress-core)

prevent stream stalls by guaranteeing flow-control updates and avoiding bounded per-stream channel overflows

- Replace bounded per-stream data channels with unbounded channels on edge and hub, relying on existing WINDOW_UPDATE flow control to limit bytes in flight
- Use awaited sends for FRAME_WINDOW_UPDATE and FRAME_WINDOW_UPDATE_BACK so updates are not dropped and streams do not deadlock under backpressure
- Clean up stream state when channel receivers have already exited instead of closing active streams because a bounded queue filled

2026-03-17 - 4.8.3 - fix(protocol,edge)

optimize tunnel frame handling and zero-copy uploads in edge I/O

- extract hub frame processing into a shared edge handler to remove duplicated tunnel logic
- add zero-copy frame header encoding and read payloads directly into framed buffers for client-to-hub uploads
- refactor Tunnelo read/write state to avoid unsafe queue access and reduce buffer churn with incremental parsing

2026-03-17 - 4.8.2 - fix(rust-edge)

refactor tunnel I/O to preserve TLS state and prioritize control frames

- replace split TLS handling with a single-owner Tunnello to avoid handshake and buffered read corruption
- prioritize control frames over data frames to prevent WINDOW_UPDATE starvation and flow-control deadlocks
- improve tunnel reliability with incremental frame parsing, liveness/error events, and corrupt frame header logging

2026-03-17 - 4.8.1 - fix(remoteingress-core)

remove tunnel writer timeouts from edge and hub buffered writes

- Drops the 30 second timeout wrapper around writer.write_all and writer.flush in both edge and hub tunnel writers.
- Updates error logging to report write failures without referring to stalled writes.

2026-03-17 - 4.8.0 - feat(events)

include disconnect reasons in edge and hub management events

- Add reason fields to tunnelDisconnected and edgeDisconnected events emitted from the Rust core and binary bridge
- Propagate specific disconnect causes such as EOF, liveness timeout, writer failure, handshake failure, and hub cancellation
- Update TypeScript edge and hub classes to log and forward disconnect reason data
- Extend serialization tests to cover the new reason fields

2026-03-17 - 4.7.2 - fix(remoteingress-core)

add tunnel write timeouts and scale initial stream windows by active stream count

- Wrap tunnel frame writes and flushes in a 30-second timeout on both edge and hub to detect stalled writers and trigger faster reconnect or cleanup.
- Compute each stream's initial send window from the current active stream count instead of using a fixed window to keep total in-flight data within the 32MB budget.

2026-03-17 - 4.7.1 - fix(remoteingress-core)

improve tunnel failure detection and reconnect handling

- Enable TCP keepalive on edge and hub connections to detect silent network failures sooner
- Trigger immediate reconnect or disconnect when tunnel writer tasks fail instead of waiting for liveness timeouts
- Prevent active stream counter underflow during concurrent connection cleanup

2026-03-16 - 4.7.0 - feat(edge,protocol,test)

add configurable edge bind address and expand flow-control test coverage

- adds an optional bindAddress configuration for edge TCP listeners, defaulting to 0.0.0.0 when not provided
- passes bindAddress through the TypeScript edge client and Rust edge runtime so local test setups can bind to localhost
- adds protocol unit tests for adaptive stream window sizing and window update frame encoding/decoding
- introduces end-to-end flow-control tests and updates the test script to build before running tests

2026-03-16 - 4.6.1 - fix(remoteingress-core)

avoid spurious tunnel disconnect events and increase control channel capacity

- Emit TunnelDisconnected only after an established connection is actually lost, preventing false disconnect events during failed reconnect attempts.
- Increase edge and hub control-channel buffer sizes from 64 to 256 to better prioritize control frames under load.

2026-03-16 - 4.6.0 - feat(remoteingress-core)

add adaptive per-stream flow control based on active stream counts

- Track active stream counts on edge and hub connections to size per-stream flow control windows dynamically.
- Cap WINDOW_UPDATE increments and read sizes to the adaptive window so bandwidth is shared more evenly across concurrent streams.
- Apply the adaptive logic to both upload and download paths on edge and hub stream handlers.

2026-03-16 - 4.5.12 - fix(remoteingress-core)

improve tunnel liveness handling and enable TCP keepalive for accepted client sockets

- Avoid disconnecting edges when PING or PONG frames cannot be queued because the control channel is temporarily full.
- Enable TCP_NODELAY and TCP keepalive on accepted client connections to help detect stale or dropped clients.

2026-03-16 - 4.5.11 - fix(repo)

no changes to commit

2026-03-16 - 4.5.10 - fix(remoteingress-core)

guard zero-window reads to avoid false EOF handling on stalled streams

- Prevent upload and download loops from calling read on an empty buffer when flow-control window remains at 0 after stall timeout
- Log a warning and close the affected stream instead of misinterpreting Ok(0) as end-of-file

2026-03-16 - 4.5.9 - fix(remoteingress-core)

delay stream close until downstream response draining finishes to prevent truncated transfers

- Waits for the hub-to-client download task to finish before sending the stream CLOSE frame
- Prevents upstream reads from being cancelled mid-response during asymmetric transfers such as git fetch
- Retains the existing timeout so stalled downloads still clean up safely

2026-03-16 - 4.5.8 - fix(remoteingress-core)

ensure upstream writes cancel promptly and reliably deliver CLOSE_BACK frames

- listen for stream cancellation while waiting on upstream write timeouts so FRAME_CLOSE does not block for up to 60 seconds
- replace try_send with send().await when emitting CLOSE_BACK frames to avoid silently dropping close notifications when the data channel is full

2026-03-16 - 4.5.7 - fix(remoteingress-core)

improve tunnel reconnect and frame write efficiency

- Reuse the TLS connector across edge reconnections to preserve session resumption state and reduce reconnect latency.
- Buffer hub and edge frame writes to coalesce small control and data frames into fewer TLS records and syscalls while still flushing each frame promptly.

2026-03-16 - 4.5.6 - fix(remoteingress-core)

disable Nagle's algorithm on edge, hub, and upstream TCP sockets to reduce control-frame latency

- Enable TCP_NODELAY on the edge connection to the hub for faster PING/PONG and WINDOW_UPDATE delivery
- Apply TCP_NODELAY on accepted hub streams before TLS handling
- Enable TCP_NODELAY on SmartProxy upstream connections before sending the PROXY header

2026-03-16 - 4.5.5 - fix(remoteingress-core)

wait for hub-to-client draining before cleanup and reliably send close frames

- switch CLOSE frame delivery on the data channel from try_send to send().await to avoid dropping it when the channel is full
- delay stream cleanup until the hub-to-client task finishes or times out so large downstream responses continue after upload EOF
- add a bounded 5-minute wait for download draining to prevent premature termination of asymmetric transfers such as git fetch

2026-03-15 - 4.5.4 - fix(remoteingress-core)

preserve stream close ordering and add flow-control stall timeouts

- Send CLOSE and CLOSE_BACK frames on the data channel so they arrive after the final stream data frames.
- Log and abort stalled upload and download paths when flow-control windows stay empty for 120 seconds.
- Apply a 60-second timeout when writing buffered stream data to the upstream connection to prevent hung streams.

2026-03-15 - 4.5.3 - fix(remoteingress-core)

prioritize control frames over data in edge and hub tunnel writers

- Split tunnel/frame writers into separate control and data channels in edge and hub

- Use biased select loops so PING, PONG, WINDOW_UPDATE, OPEN, and CLOSE frames are sent before data frames
- Route stream data through dedicated data channels while keeping OPEN, CLOSE, and flow-control updates on control channels to prevent keepalive starvation under load

2026-03-15 - 4.5.2 - fix(remoteingress-core)

improve stream flow control retries and increase channel buffer capacity

- increase per-stream mpsc channel capacity from 128 to 256 on both edge and hub paths
- only reset accumulated window update bytes after a successful try_send to avoid dropping flow-control credits when the update channel is busy

2026-03-15 - 4.5.1 - fix(protocol)

increase per-stream flow control window and channel buffers to improve high-RTT throughput

- raise the initial stream window from 256 KB to 4 MB to allow more in-flight data per stream
- increase edge and hub mpsc channel capacities from 16 to 128 to better absorb throughput under flow control

2026-03-15 - 4.5.0 - feat(remoteingress-core)

add per-stream flow control for edge and hub tunnel data transfer

- introduce WINDOW_UPDATE frame types and protocol helpers for per-stream flow control
- track per-stream send windows on both edge and hub to limit reads based on available capacity
- send window updates after downstream writes to reduce channel pressure during large transfers

2026-03-15 - 4.4.1 - fix(remoteingress-core)

prevent stream data loss by applying backpressure and closing saturated channels

- replace non-blocking frame writes with awaited sends in per-stream tasks so large transfers respect backpressure instead of dropping data
- close and remove streams when back-channel or data channels fill up to avoid TCP stream corruption from silently dropped frames

2026-03-03 - 4.4.0 - feat(remoteingress)

add heartbeat PING/PONG and liveness timeouts; implement fast-reconnect/backoff reset and JS crash-recovery auto-restart

- protocol: add FRAME_PING and FRAME_PONG and unit tests for ping/pong frames
- edge (Rust): reset backoff after successful connection, respond to PING with PONG, track liveness via deadline and reconnect on timeout, use Duration/Instant helpers
- hub (Rust): send periodic PING to edges, handle PONGs, enforce liveness timeout and disconnect inactive edges, use tokio interval and time utilities
- ts: RemoteIngressEdge and RemoteIngressHub: add crash-recovery auto-restart with exponential backoff and max attempts, save/restore config and allowed edges, register/remove exit handlers, ensure stop() marks stopping and cleans up listeners
- minor API/typing: introduce TAllowedEdge alias and persist allowed edges for restart recovery

2026-02-26 - 4.3.0 - feat(hub)

add optional TLS certificate/key support to hub start config and bridge

- TypeScript: add tls.certPem and tls.keyPem to IHubConfig and include tlsCertPem/tlsKeyPem in startHub bridge command when both are provided
- TypeScript: extend startHub params with tlsCertPem and tlsKeyPem and conditionally send them
- Rust: change HubConfig serde attributes for tls_cert_pem and tls_key_pem from skip to default so absent PEM fields deserialize as None
- Enables optional provisioning of TLS certificate and key to the hub when provided from the JS side

2026-02-26 - 4.2.0 - feat(core)

expose edge peer address in hub events and migrate writers to channel-based, non-blocking framing with stream limits and timeouts

- Add `peerAddr` to `ConnectedEdgeStatus` and `HubEvent::EdgeConnected` and surface it to the TS frontend event (`management:edgeConnected`).
- Replace `Arc<Mutex<WriteHalf>>` writers with dedicated mpsc channel writer tasks in both hub and edge crates to serialize writes off the main tasks.
- Use non-blocking `try_send` for data frames to avoid head-of-line blocking and drop frames with warnings when channels are full.
- Introduce `MAX_STREAMS_PER_EDGE` semaphore to limit concurrent streams per edge and reject excess opens with a `CLOSE_BACK` frame.
- Add a 10s timeout when connecting to `SmartProxy` to avoid hanging connections.
- Ensure writer tasks are aborted on shutdown/cleanup and propagate cancellation tokens appropriately.

2026-02-26 - 4.1.0 - feat(remoteingress-bin)

use `mimalloc` as the global allocator to reduce memory overhead and improve allocation performance

- added `mimalloc = "0.1"` dependency to `rust/crates/remoteingress-bin/Cargo.toml`
- registered `mimalloc` as the `#[global_allocator]` in `rust/crates/remoteingress-bin/src/main.rs`
- updated `Cargo.lock` with `libmimalloc-sys` and `mimalloc` package entries

2026-02-26 - 4.0.1 - fix(hub)

cancel per-stream tokens on stream close and avoid duplicate `StreamClosed` events; bump `@types/node` devDependency to `^25.3.0`

- Add `CancellationToken` to per-stream entries so each stream can be cancelled independently.
- Ensure `StreamClosed` event is only emitted when a stream was actually present (guards against duplicate events).
- Cancel the stream-specific token on `FRAME_CLOSE` to stop associated tasks and free resources.
- DevDependency bump: `@types/node` updated from `^25.2.3` to `^25.3.0`.

2026-02-19 - 4.0.0 - BREAKING CHANGE(remoteingress-core)

add cancellation tokens and cooperative shutdown; switch event channels to bounded mpsc and improve cleanup

- Introduce `tokio-util::sync::CancellationToken` for hub/edge and per-connection/stream cancellation, enabling cooperative shutdown of spawned tasks.
- Replace unbounded mpsc channels with bounded `mpsc::channel(1024)` and switch from `UnboundedSender/Receiver` to `Sender/Receiver`; use `try_send` where non-blocking sends are appropriate.
- Wire cancellation tokens through edge and hub codepaths: child tokens per connection, per-port, per-stream; cancel tokens in `stop()` and `Drop` impls to ensure deterministic task termination and cleanup.
- Reset stream id counters and clear listener state on reconnect; improved error handling around `accept/read` loops using `tokio::select!` and cancellation checks.
- Update `Cargo.toml` and `Cargo.lock` to add `tokio-util` (and related futures entries) as dependencies.
- BREAKING: public API/types changed — `take_event_rx` return types and `event_tx/event_rx` fields now use bounded `mpsc::Sender/mpsc::Receiver` instead of the unbounded variants; callers must adapt to the new types and bounded behavior.

2026-02-18 - 3.3.0 - feat(readme)

document dynamic port assignment and runtime port updates; clarify TLS multiplexing, frame format, and handshake sequence

- Adds documentation for dynamic port configuration: hub-assigned listen ports, hot-reloadable via `FRAME_CONFIG` frames
- Introduces new `FRAME` type `CONFIG (0x06)` and describes payload as JSON; notes immediate push of port changes to connected edges
- Clarifies that the tunnel is a single encrypted TLS multiplexed connection to the hub (preserves `PROXY v1` behavior)
- Specifies frame integer fields are big-endian and that stream IDs are 32-bit unsigned integers
- Adds new events: `portsAssigned` and `portsUpdated`, and updates examples showing `updateAllowedEdges` usage and live port changes

2026-02-18 - 3.2.1 - fix(tests)

add comprehensive unit and async tests across Rust crates and TypeScript runtime

- Added IPC serialization tests in remoteingress-bin (IPC request/response/event)
- Added serde and async tests for Edge and Handshake configs and EdgeEvent/EdgeStatus in remoteingress-core (edge.rs)
- Added extensive Hub tests: constant_time_eq, PROXY header port parsing, serde/camelCase checks, Hub events and async TunnelHub behavior (hub.rs)
- Added STUN parser unit tests including XOR_MAPPED_ADDRESS, MAPPED_ADDRESS fallback, truncated attribute handling and other edge cases (stun.rs)
- Added protocol frame encoding and FrameReader tests covering all frame types, payload limits and EOF conditions (remoteingress-protocol)
- Added TypeScript Node tests for token encode/decode edge cases and RemoteIngressHub/RemoteIngressEdge class basics (test/*.node.ts)

2026-02-18 - 3.2.0 - feat(remoteingress (edge/hub/protocol))

add dynamic port configuration: handshake, FRAME_CONFIG frames, and hot-reloadable listeners

- Introduce a JSON handshake from hub -> edge with initial listen ports and stun interval so edges can configure listeners at connect time.
- Add FRAME_CONFIG (0x06) to the protocol and implement runtime config updates pushed from hub to connected edges.
- Edge now applies initial ports and supports hot-reloading: spawn/abort listeners when ports change, and emit PortsAssigned / PortsUpdated events.
- Hub now stores allowed edge metadata (listen_ports, stun_interval_secs), sends handshake responses on auth, and forwards config updates to connected edges.
- TypeScript bridge/client updated to emit new port events and periodically log status; updateAllowedEdges API accepts listenPorts and stunIntervalSecs.
- Stun interval handling moved to use handshake-provided/stored value instead of config.listen_ports being static.

2026-02-18 - 3.1.1 - fix(readme)

update README: add issue reporting/security section, document connection tokens and token utilities, clarify architecture/API and improve examples/formatting

- Added an 'Issue Reporting and Security' section linking to community.foss.global for bug/security reports and contributor onboarding.
- Documented connection tokens: encodeConnectionToken/decodeConnectionToken utilities, token format (base64url), and examples for hub and edge provisioning.

- Clarified Hub/Edge usage and examples: condensed event handlers, added token-based start() example, and provided explicit config alternative.
- Improved README formatting: added emojis, rephrased architecture descriptions, fixed wording and license path capitalization, and expanded example scenarios and interfaces.

2026-02-17 - 3.1.0 - feat(edge)

support connection tokens when starting an edge and add token encode/decode utilities

- Add classes.token.ts with encodeConnectionToken/decodeConnectionToken using a base64url compact JSON format
- Export token utilities from ts/index.ts
- RemoteIngressEdge.start now accepts a { token } option and decodes it to an IEdgeConfig before starting
- Add tests covering export availability, encode→decode roundtrip, malformed token, and missing fields
- Non-breaking change — recommend a minor version bump

2026-02-17 - 3.0.4 - fix(build)

bump dev dependencies, update build script, and refresh README docs

- Bumped devDependencies: @git.zone/tsbuild ^2.1.25 → ^4.1.2, @git.zone/tsbundle ^2.0.5 → ^2.8.3, @git.zone/tsrun ^1.2.46 → ^2.0.1, @git.zone/tstest ^1.0.44 → ^3.1.8, @push.rocks/tapbundle ^5.0.15 → ^6.0.3, @types/node ^20.8.7 → ^25.2.3
- Bumped runtime dependency: @push.rocks/qenv ^6.0.5 → ^6.1.3
- Changed build script: replaced "tsbuild --web --allowimplicitany" with "tsbuild tsfolders --allowimplicitany" (kept tsrust invocation)
- README updates: added RustBridge notes (localPaths must be full file paths), production binary naming conventions, rust core uses ring as rustls provider; removed emoji from example console output; clarified stunIntervalSecs is optional; renamed example status variable to edgeStatus; minor wire-protocol formatting and wording/legal text tweaks

2026-02-17 - 3.0.3 - fix(rust,ts)

initialize rustls ring CryptoProvider at startup; add rustls dependency and features; make native binary lookup platform-aware

- Install rustls::crypto::ring default_provider at startup to ensure ring-based crypto is available before any TLS usage.

- Add rustls dependency to remoteingress-bin and update remoteingress-core rustls configuration (disable default-features; enable ring, logging, std, tls12).
- Adjust TS classes to prefer platform-suffixed production binaries, add exact fallback names, and include explicit cargo output paths for release/debug.
- Cargo.lock updated to include rustls entry.

2026-02-16 - 3.0.2 - fix(readme)

Document Hub/Edge architecture and new RemoteIngressHub/RemoteIngressEdge API; add Rust core binary, protocol and usage details; note removal of ConnectorPublic/ConnectorPrivate (breaking change)

- Adds comprehensive README describing v3 Hub↔Edge topology and usage examples
- Introduces Rust core binary (remoteingress-bin) and RustBridge IPC via @push.rocks/smartrust
- Documents custom multiplexed binary frame protocol over TLS and PROXY protocol v1 for client IP preservation
- Notes STUN-based public IP discovery and cross-compiled linux/amd64 and linux/arm64 binaries
- Calls out removal/rename of ConnectorPublic/ConnectorPrivate to RemoteIngressHub/RemoteIngressEdge (breaking API change)
- Updates install instruction to use pnpm and expands API reference, events, and examples

2026-02-16 - 3.0.1 - fix(remoteingress)

no changes detected in diff; no code modifications to release

- No files changed in the provided diff.
- No release or version bump required based on current changes.

2026-02-16 - 3.0.0 - BREAKING CHANGE(remoteingress)

migrate core to Rust, add RemoteIngressHub/RemoteIngressEdge JS bridge, and bump package to v2.0.0

- Added Rust workspace and crates: remoteingress-protocol, remoteingress-core, remoteingress-bin (IPC management mode via JSON over stdin/stdout).

- Implemented protocol framing, PROXY v1 header builder, and async FrameReader in remoteingress-protocol.
- Implemented hub and edge tunnel logic in Rust including TLS handling, PROXY parsing, and STUN public IP discovery.
- Added TypeScript runtime bridge classes RemoteIngressHub and RemoteIngressEdge that use @push.rocks/smartrust to spawn/manage the Rust binary.
- Removed legacy connector public/private TS files and simplified ts/index exports to expose hub/edge classes.
- Updated package.json: bumped version to 2.0.0, adjusted description, added tsrust build step, new dependency @push.rocks/smartrust and keywords, and included dist_rust in files/glob.
- Added rust build config for cross-target linkers and new Cargo.toml manifests for the workspace.

2024-04-14 - 1.0.2 - 1.0.4 - releases

Version-only tag commits (no code changes) for recent releases.

- 1.0.2 (2024-03-24) — release tag / version bump only
- 1.0.3 (2024-04-14) — release tag / version bump only
- 1.0.4 (2024-04-14) — release tag / version bump only

2024-04-14 - 1.0.3 - core

Core updates and fixes.

- fix(core): update

2024-04-14 - 1.0.2 - core

Core updates and fixes.

- fix(core): update

2024-03-24 - 1.0.1 - core

Core updates and fixes.

- fix(core): update
-

Revision #2

Created 2026-03-28 13:11:44 UTC by foss.global Team

Updated 2026-03-29 16:54:13 UTC by foss.global Team