@serve.zone/dcrouter

a traffic router intended to be gating your datacenter.

- readme.md for @serve.zone/dcrouter
- changelog.md for @serve.zone/dcrouter

readme.md for @serve.zone/dcrouter

dcrouter: The all-in-one gateway for your datacenter.

A comprehensive traffic routing solution that provides unified gateway capabilities for HTTP/HTTPS, TCP/SNI, email (SMTP), DNS, RADIUS, VPN, and remote edge ingress — all from a single process. Designed for enterprises requiring robust traffic management, automatic TLS certificate provisioning, VPN-based access control, distributed edge networking, and enterprise-grade email infrastructure.

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

Table of Contents

Features

□□ Universal Traffic Router

- **HTTP/HTTPS routing** with domain matching, path-based forwarding, and automatic TLS
- **HTTP/3 (QUIC) enabled by default** — qualifying HTTPS routes automatically get QUIC/H3 support with zero configuration
- **TCP/SNI proxy** for any protocol with TLS termination or passthrough
- **DNS server** (Rust-powered via SmartDNS) with authoritative zones, dynamic record management, and DNS-over-HTTPS
- **Multi-protocol support** on the same infrastructure via SmartProxy

□□ Complete Email Infrastructure (powered by smartmta)

- **Multi-domain SMTP server** on standard ports (25, 587, 465)
- **Pattern-based email routing** with four action types: forward, process, deliver, reject

- **DKIM signing & verification**, SPF, DMARC authentication stack
- **Enterprise deliverability** with IP warmup schedules and sender reputation tracking
- **Bounce handling** with automatic suppression lists
- **Hierarchical rate limiting** — global, per-domain, per-sender

□□ Enterprise Security

- **Automatic TLS certificates** via ACME (smartacme v9) with Cloudflare DNS-01 challenges
- **Smart certificate scheduling** — per-domain deduplication, controlled parallelism, and account rate limiting handled automatically
- **Per-domain exponential backoff** — failed provisioning attempts are tracked and backed off to avoid hammering ACME servers
- **IP reputation checking** with caching and configurable thresholds
- **Content scanning** for spam, viruses, and malicious attachments
- **Security event logging** with structured audit trails

□□ RADIUS Server

- **MAC Authentication Bypass (MAB)** for network device authentication
- **VLAN assignment** based on exact MAC, OUI prefix, or wildcard patterns
- **RADIUS accounting** for session tracking, traffic metering, and billing
- **Real-time management** via OpsServer API

□□ Remote Ingress (powered by remoteingress)

- **Distributed edge networking** — accept traffic at remote edge nodes and tunnel it to the hub
- **Edge registration CRUD** with secret-based authentication
- **Auto-derived ports** — edges automatically pick up ports from routes tagged with `remoteIngress.enabled`
- **Connection tokens** — generate a single opaque base64url token containing hubHost, hubPort, edgeId, and secret for easy edge provisioning
- **Real-time status monitoring** — connected/disconnected state, public IP, active tunnels, heartbeat tracking
- **OpsServer dashboard** with enable/disable, edit, secret regeneration, token copy, and delete actions

□□ VPN Access Control (powered by smartvpn)

- **WireGuard + native transports** — standard WireGuard clients (iOS, Android, macOS, Windows, Linux) plus custom WebSocket/QUIC tunnels
- **Route-level VPN gating** — mark any route with `vpn: { enabled: true }` to restrict access to VPN clients only, or `vpn: { enabled: true, mandatory: false }` to add VPN clients alongside existing access rules
- **Tag-based access control** — assign `serverDefinedClientTags` to clients and restrict routes with `allowedServerDefinedClientTags`
- **Constructor-defined clients** — pre-define VPN clients with tags in config for declarative, code-driven setup
- **Rootless operation** — uses userspace NAT (smoltcp) with no root required
- **Destination policy** — configurable `forceTarget`, `block`, or `allow` with allowList/blockList for granular traffic control
- **Client management** — create, enable, disable, rotate keys, export WireGuard/SmartVPN configs via OpsServer API and dashboard
- **IP-based enforcement** — VPN clients get IPs from a configurable subnet; SmartProxy enforces `ipAllowList` per route
- **PROXY protocol v2** — the NAT engine sends PP v2 on outbound connections to preserve VPN client identity

⚡ High Performance

- **Rust-powered proxy engine** via SmartProxy for maximum throughput
- **Rust-powered MTA engine** via smartmta (TypeScript + Rust hybrid) for reliable email delivery
- **Rust-powered DNS engine** via SmartDNS for high-performance UDP and DNS-over-HTTPS
- **Connection pooling** for outbound SMTP and backend services
- **Socket-handler mode** — direct socket passing eliminates internal port hops
- **Real-time metrics** via SmartMetrics (CPU, memory, connections, throughput)

□□ Persistent Storage & Caching

- **Multiple storage backends**: filesystem, custom functions, or in-memory
- **Embedded cache database** via smartdata + smartdb (MongoDB-compatible)
- **Automatic TTL-based cleanup** for cached emails and IP reputation data

□□ OpsServer Dashboard

- **Web-based management interface** with real-time monitoring
- **JWT authentication** with session persistence
- **Live views** for connections, email queues, DNS queries, RADIUS sessions, certificates, remote ingress edges, VPN clients, and security events
- **Domain-centric certificate overview** with backoff status and one-click reprovisioning
- **Remote ingress management** with connection token generation and one-click copy
- **Read-only configuration display** — DcRouter is configured through code
- **Smart tab visibility handling** — auto-pauses all polling, WebSocket connections, and chart updates when the browser tab is hidden, preventing resource waste and tab freezing

□□ Programmatic API Client

- **Object-oriented API** — resource classes (`Route`, `Certificate`, `ApiToken`, `RemoteIngress`, `Email`) with instance methods
- **Builder pattern** — fluent `.setName().setMatch().save()` chains for creating routes, tokens, and edges
- **Auto-injected auth** — JWT identity and API tokens included automatically in every request
- **Dual auth modes** — login with credentials (JWT) or pass an API token for programmatic access
- **Full coverage** — wraps every OpsServer endpoint with typed request/response pairs

Installation

```
pnpm add @serve.zone/dcrouter
# or
npm install @serve.zone/dcrouter
```

Prerequisites

- **Node.js 20+** with ES module support
- Valid domain with DNS control (for ACME certificate automation)
- Cloudflare API token (for DNS-01 challenges) — optional

Quick Start

Basic HTTP/HTTPS Router

```
import { DcRouter } from '@serve.zone/dcrouter';

const router = new DcRouter({
  smartProxyConfig: {
    routes: [
      {
        name: 'web-app',
        match: { domains: ['example.com', 'www.example.com'], ports: [443] },
        action: {
          type: 'forward',
          targets: [{ host: '192.168.1.10', port: 8080 }],
          tls: { mode: 'terminate', certificate: 'auto' }
        }
      }
    ],
    acme: {
      email: 'admin@example.com',
      enabled: true,
      useProduction: true
    }
  }
});

await router.start();
```

Basic Email Server

```
import { DcRouter } from '@serve.zone/dcrouter';

const router = new DcRouter({
  emailConfig: {
    ports: [25, 587, 465],
```

```
    hostname: 'mail.example.com',
    domains: [
      {
        domain: 'example.com',
        dnsMode: 'external-dns'
      }
    ],
    routes: [
      {
        name: 'process-all',
        match: { recipients: '*@example.com' },
        action: {
          type: 'process',
          process: { scan: true, dkim: true, queue: 'normal' }
        }
      }
    ]
  }
});

await router.start();
```

Full Stack with Dashboard

```
import { DcRouter } from '@serve.zone/dcrouter';

const router = new DcRouter({
  // HTTP/HTTPS routing
  smartProxyConfig: {
    routes: [
      {
        name: 'website',
        match: { domains: ['example.com'], ports: [443] },
        action: {
          type: 'forward',
          targets: [{ host: '192.168.1.10', port: 80 }],
          tls: { mode: 'terminate', certificate: 'auto' }
        }
```

```
    }
  ],
  acme: { email: 'ssl@example.com', enabled: true, useProduction: true }
},

// Email system (powered by smartmta)
emailConfig: {
  ports: [25, 587, 465],
  hostname: 'mail.example.com',
  domains: [{ domain: 'example.com', dnsMode: 'external-dns' }],
  routes: [
    {
      name: 'inbound-mail',
      match: { recipients: '*@example.com' },
      action: { type: 'process', process: { scan: true, dkim: true, queue: 'normal' } }
    }
  ]
},

// Authoritative DNS
dnsNsDomains: ['ns1.example.com', 'ns2.example.com'],
dnsScopes: ['example.com'],
publicIp: '203.0.113.1',
dnsRecords: [
  { name: 'example.com', type: 'A', value: '203.0.113.1' },
  { name: 'www.example.com', type: 'CNAME', value: 'example.com' }
],

// RADIUS authentication
radiusConfig: {
  authPort: 1812,
  acctPort: 1813,
  clients: [
    { name: 'switch-1', ipRange: '192.168.1.0/24', secret: 'radius-secret', enabled: true }
  ],
  vlanAssignment: {
    defaultVlan: 100,
    allowUnknownMacs: true,
    mappings: [
```

```
        { mac: 'aa:bb:cc:dd:ee:ff', vlan: 10, enabled: true },
        { mac: 'aa:bb:cc', vlan: 20, enabled: true }  // OUI prefix
      ]
    },
    accounting: { enabled: true, retentionDays: 30 }
  },

  // Remote Ingress — edge nodes tunnel traffic to this hub
  remoteIngressConfig: {
    enabled: true,
    tunnelPort: 8443,
    hubDomain: 'hub.example.com',
  },

  // VPN — restrict sensitive routes to VPN clients
  vpnConfig: {
    enabled: true,
    serverEndpoint: 'vpn.example.com',
    clients: [
      { clientId: 'dev-laptop', serverDefinedClientTags: ['engineering'] },
    ],
  },

  // Persistent storage
  storage: { fsPath: '/var/lib/dcrouter/data' },

  // Cache database
  cacheConfig: { enabled: true, storagePath: '~/.serve.zone/dcrouter/tsmdb' },

  // TLS & ACME
  tls: { contactEmail: 'admin@example.com' },
  dnsChallenge: { cloudflareApiKey: process.env.CLOUDFLARE_API_KEY }
});

await router.start();
// OpsServer dashboard available at http://localhost:3000
```

Architecture

System Overview

```
graph TB
    subgraph "External Traffic"
        HTTP[HTTP/HTTPS Clients]
        SMTP[SMTP Clients]
        TCP[TCP Clients]
        DNS[DNS Queries]
        RAD[RADIUS Clients]
        EDGE[Edge Nodes]
        VPN[VPN Clients]
    end

    subgraph "DcRouter Core"
        DC[DcRouter Orchestrator]
        SP[SmartProxy Engine<br/><i>Rust-powered</i>]
        ES[smartmta Email Server<br/><i>TypeScript + Rust</i>]
        DS[SmartDNS Server<br/><i>Rust-powered</i>]
        RS[SmartRadius Server]
        RI[RemoteIngress Hub<br/><i>Rust data plane</i>]
        VS[SmartVPN Server<br/><i>Rust data plane</i>]
        CM[Certificate Manager<br/><i>smartacme v9</i>]
        OS[OpsServer Dashboard]
        MM[Metrics Manager]
        SM[Storage Manager]
        CD[Cache Database]
    end

    subgraph "Backend Services"
        WEB[Web Services]
        MAIL[Mail Servers]
        DB[Databases]
        API[Internal APIs]
    end

    HTTP --> SP
    TCP --> SP
    SMTP --> ES
    DNS --> DS
```

```
    RAD --> RS
    EDGE --> RI
    VPN --> VS

    DC --> SP
    DC --> ES
    DC --> DS
    DC --> RS
    DC --> RI
    DC --> VS
    DC --> CM
    DC --> OS
    DC --> MM
    DC --> SM
    DC --> CD

    SP --> WEB
    SP --> API
    ES --> MAIL
    ES --> DB
    RI --> SP

    CM -.-> SP
    CM -.-> ES
```

Core Components

Component	Package	Description
DcRouter	`@serve.zone/dcrouter`	Central orchestrator — starts, stops, and coordinates all services
SmartProxy	`@push.rocks/smartproxy`	High-performance HTTP/HTTPS and TCP/SNI proxy with route-based config (Rust engine)
UnifiedEmailServer	`@push.rocks/smartmta`	Full SMTP server with pattern-based routing, DKIM, queue management (TypeScript + Rust)
DNS Server	`@push.rocks/smartdns`	Authoritative DNS with dynamic records and DKIM TXT auto-generation (Rust engine)

Component	Package	Description
SmartAcme	`@push.rocks/smartacme`	ACME certificate management with per-domain dedup, concurrency control, and rate limiting
RADIUS Server	`@push.rocks/smartradius`	Network authentication with MAB, VLAN assignment, and accounting
RemoteIngress	`@serve.zone/remoteingress`	Distributed edge tunneling with Rust data plane and TS management
OpsServer	`@api.global/typedserver`	Web dashboard + TypedRequest API for monitoring and management
MetricsManager	`@push.rocks/smartmetrics`	Real-time metrics collection (CPU, memory, email, DNS, security)
StorageManager	built-in	Pluggable key-value storage (filesystem, custom, or in-memory)
CacheDb	`@push.rocks/smartdb`	Embedded MongoDB-compatible database (LocalSmartDb) for persistent caching

How It Works

DcRouter acts purely as an **orchestrator** — it doesn't implement protocols itself. Instead, it wires together best-in-class packages for each protocol:

1. **On `start()`**: DcRouter initializes OpsServer (default port 3000, configurable via `opsServerPort`), then spins up SmartProxy, smartmta, SmartDNS, SmartRadius, RemoteIngress, and SmartVPN based on which configs are provided. Services start in dependency order via `ServiceManager`.
2. **During operation**: Each service handles its own protocol independently. SmartProxy uses a Rust-powered engine for maximum throughput. smartmta uses a hybrid TypeScript + Rust architecture for reliable email delivery. RemoteIngress runs a Rust data plane for edge tunnel networking. SmartVPN runs a Rust data plane for WireGuard and custom transports. SmartAcme v9 handles all certificate operations with built-in concurrency control and rate limiting.
3. **On `stop()`**: All services are gracefully shut down in parallel, including cleanup of HTTP agents and DNS clients.

Rust-Powered Architecture

DcRouter itself is a pure TypeScript orchestrator, but several of its core sub-components ship with **compiled Rust binaries** for performance-critical paths. At runtime each package detects the platform, unpacks the correct binary, and communicates with TypeScript over IPC/FFI — so you get the ergonomics of TypeScript with the throughput of native code.

Component	Rust Binary	What It Handles
SmartProxy	`smartproxy-bin`	All TCP/TLS/HTTP proxy networking, NFTables integration, connection metrics
smartmta	`mailer-bin`	SMTP server + client, DKIM/SPF/DMARC, content scanning, IP reputation
SmartDNS	`smartdns-bin`	DNS server (UDP + DNS-over-HTTPS), DNSSEC, DNS client resolution
RemoteIngress	`remoteingress-bin`	Edge tunnel data plane, multiplexed streams, heartbeat management
SmartVPN	`smartvpn_daemon`	WireGuard (boringtun), Noise IK handshake, QUIC/WS transports, userspace NAT (smoltcp)
SmartRadius	—	Pure TypeScript (no Rust component)

Configuration Reference

`IDcRouterOptions`

```
interface IDcRouterOptions {
  // ── Base ──────────────────────────────────────────────────
  /** Base directory for all dcrouter data. Defaults to ~/.serve.zone/dcrouter */
  baseDir?: string;

  // ── Traffic Routing ──────────────────────────────────────
  /** SmartProxy config for HTTP/HTTPS and TCP/SNI routing */
  smartProxyConfig?: ISmartProxyOptions;

  // ── Email ────────────────────────────────────────────────
  /** Unified email server configuration (smartmta) */
  emailConfig?: IUnifiedEmailServerOptions;

  /** Custom email port mapping overrides */
  emailPortConfig?: {
    portMapping?: Record<number, number>;
    portSettings?: Record<number, any>;
```

```
  receivedEmailsPath?: string;
};

// ── DNS ──────────────────────────────────────────────────────────
/** Nameserver domains — get A records automatically */
dnsNsDomains?: string[];
/** Domains this server is authoritative for */
dnsScopes?: string[];
/** Public IP for NS A records */
publicIp?: string;
/** Ingress proxy IPs (hides real server IP) */
proxyIps?: string[];
/** Custom DNS records */
dnsRecords?: Array<{
  name: string;
  type: 'A' | 'AAAA' | 'CNAME' | 'MX' | 'TXT' | 'NS' | 'SOA';
  value: string;
  ttl?: number;
  useIngressProxy?: boolean;
}>;

// ── RADIUS ──────────────────────────────────────────────────────
/** RADIUS server for network authentication */
radiusConfig?: {
  authPort?: number;                  // default: 1812
  acctPort?: number;                  // default: 1813
  clients: IRadiusClient[];
  vlanAssignment?: IVlanManagerConfig;
  accounting?: { enabled: boolean; retentionDays?: number };
};

// ── Remote Ingress ──────────────────────────────────────────────
/** Remote Ingress hub for edge tunnel connections */
remoteIngressConfig?: {
  enabled?: boolean;                  // default: false
  tunnelPort?: number;                // default: 8443
  hubDomain?: string;                 // External hostname for connection tokens
  tls?: {
    certPath?: string;
```

```
    keyPath?: string;
  };
};

// ── VPN ──────────────────────────────────────────────────────────
/** VPN server for route-level access control */
vpnConfig?: {
  enabled?: boolean;                  // default: false
  subnet?: string;                    // default: '10.8.0.0/24'
  wgListenPort?: number;              // default: 51820
  dns?: string[];                     // DNS servers pushed to VPN clients
  serverEndpoint?: string;            // Hostname in generated client configs
  clients?: Array<{                   // Pre-defined VPN clients
    clientId: string;
    serverDefinedClientTags?: string[];
    description?: string;
  }>;
  destinationPolicy?: {               // Traffic routing policy
    default: 'forceTarget' | 'block' | 'allow';
    target?: string;                  // IP for forceTarget (default: '127.0.0.1')
    allowList?: string[];             // Pass through directly
    blockList?: string[];             // Always block (overrides allowList)
  };
};

// ── HTTP/3 (QUIC) ────────────────────────────────────────────────
/** HTTP/3 config — enabled by default on qualifying HTTPS routes */
http3?: {
  enabled?: boolean;                  // default: true
  quicSettings?: {
    maxIdleTimeout?: number;          // default: 30000ms
    maxConcurrentBidiStreams?: number;  // default: 100
    maxConcurrentUniStreams?: number;   // default: 100
    initialCongestionWindow?: number;
  };
  altSvc?: {
    port?: number;                    // default: listening port
    maxAge?: number;                  // default: 86400s
  };
```

```
  udpSettings?: {
    sessionTimeout?: number;           // default: 60000ms
    maxSessionsPerIP?: number;         // default: 1000
    maxDatagramSize?: number;          // default: 65535
  };
};

// ── OpsServer ──────────────────────────────────────────────
/** Port for the OpsServer web dashboard (default: 3000) */
opsServerPort?: number;

// ── TLS & Certificates ─────────────────────────────────────
tls?: {
  contactEmail: string;
  domain?: string;
  certPath?: string;
  keyPath?: string;
};
dnsChallenge?: { cloudflareApiKey?: string };

// ── Storage & Caching ──────────────────────────────────────
storage?: {
  fsPath?: string;
  readFunction?: (key: string) => Promise<string>;
  writeFunction?: (key: string, value: string) => Promise<void>;
};
cacheConfig?: {
  enabled?: boolean;                   // default: true
  storagePath?: string;                // default: '~/.serve.zone/dcrouter/tsmdb'
  dbName?: string;                     // default: 'dcrouter'
  cleanupIntervalHours?: number;       // default: 1
  ttlConfig?: {
    emails?: number;                   // default: 30 days
    ipReputation?: number;             // default: 1 day
    bounces?: number;                  // default: 30 days
    dkimKeys?: number;                 // default: 90 days
    suppression?: number;              // default: 30 days
  };
};
```

```
}
```

HTTP/HTTPS & TCP/SNI Routing

DcRouter uses SmartProxy for all HTTP/HTTPS and TCP/SNI routing. Routes are pattern-matched by domain, port, or both.

HTTPS with Auto-TLS

```
{
  name: 'api-gateway',
  match: { domains: ['api.example.com'], ports: [443] },
  action: {
    type: 'forward',
    targets: [{ host: '192.168.1.20', port: 8080 }],
    tls: { mode: 'terminate', certificate: 'auto' }
  }
}
```

TLS Passthrough (SNI Routing)

```
{
  name: 'secure-backend',
  match: { domains: ['secure.example.com'], ports: [8443] },
  action: {
    type: 'forward',
    targets: [{ host: '192.168.1.40', port: 8443 }],
    tls: { mode: 'passthrough' }
  }
}
```

TCP Port Range Forwarding

```
{
  name: 'database-cluster',
```

```
  match: { ports: [{ from: 5432, to: 5439 }] },
  action: {
    type: 'forward',
    targets: [{ host: '192.168.1.30', port: 'preserve' }],
    security: { ipAllowList: ['192.168.1.0/24'] }
  }
}
```

HTTP Redirect

```
{
  name: 'http-to-https',
  match: { ports: [80] },
  action: { type: 'redirect', redirect: { to: 'https://{domain}{path}' } }
}
```

HTTP/3 (QUIC) Support

DcRouter ships with **HTTP/3 enabled by default** □□. All qualifying HTTPS routes on port 443 are automatically augmented with QUIC/H3 configuration — no extra setup needed. Under the hood, SmartProxy's native HTTP/3 support (via `IRouteQuic`) handles QUIC transport, Alt-Svc advertisement, and HTTP/3 negotiation.

How It Works

When DcRouter assembles routes in `setupSmartProxy()`, it automatically augments qualifying routes with:

- `match.transport: 'all'` — listen on both TCP (HTTP/1.1 + HTTP/2) and UDP (QUIC/HTTP/3) on the same port
- `action.udp.quic` — QUIC configuration with `enableHttp3: true` and `altSvcMaxAge: 86400`

Browsers that support HTTP/3 will discover it via the `Alt-Svc` header on initial TCP responses, then upgrade to QUIC for subsequent requests.

What Gets Augmented

A route qualifies for HTTP/3 augmentation when **all** of these are true:

- Port includes **443** (single number, array, or range)
- Action type is `forward` (not `socket-handler`)
- **TLS is enabled** (passthrough, terminate, or terminate-and-reencrypt)
- Route is **not** an email route (ports 25/587/465)
- Route doesn't already have `transport: 'all'` or existing `udp.quic` config

Zero-Config (Default Behavior)

```
// HTTP/3 is ON by default — this route automatically gets QUIC/H3:
const router = new DcRouter({
  smartProxyConfig: {
    routes: [{
      name: 'web-app',
      match: { domains: ['example.com'], ports: [443] },
      action: {
        type: 'forward',
        targets: [{ host: '192.168.1.10', port: 8080 }],
        tls: { mode: 'terminate', certificate: 'auto' }
      }
    }]
  }
});
```

Per-Route Opt-Out

Disable HTTP/3 on a specific route using `action.options.http3`:

```
{
  name: 'legacy-app',
  match: { domains: ['legacy.example.com'], ports: [443] },
  action: {
    type: 'forward',
    targets: [{ host: '192.168.1.50', port: 8080 }],
    tls: { mode: 'terminate', certificate: 'auto' },
    options: { http3: false }  // ← This route stays TCP-only
  }
}
```

Global Opt-Out

Disable HTTP/3 across all routes:

```
const router = new DcRouter({
  http3: { enabled: false },
  smartProxyConfig: { routes: [/* ... */] }
});
```

Custom QUIC Settings

Fine-tune QUIC parameters globally:

```
const router = new DcRouter({
  http3: {
    quicSettings: {
      maxIdleTimeout: 60000,          // 60s idle timeout
      maxConcurrentBidiStreams: 200,   // More parallel streams
      maxConcurrentUniStreams: 50,
    },
    altSvc: {
      maxAge: 3600,                   // 1 hour Alt-Svc cache
    },
    udpSettings: {
      sessionTimeout: 120000,         // 2 min UDP session timeout
      maxSessionsPerIP: 500,
    }
  },
  smartProxyConfig: { routes: [/* ... */] }
});
```

Programmatic Routes

Routes added at runtime via the Route Management API also get HTTP/3 augmentation automatically — the `RouteConfigManager` applies the same augmentation logic when merging programmatic routes.

Email System

The email system is powered by `@push.rocks/smartmta`, a TypeScript + Rust hybrid MTA. DcRouter configures and orchestrates smartmta's **UnifiedEmailServer**, which handles SMTP sessions, route matching, delivery queuing, DKIM signing, and all email processing.

Email Domain Configuration

Domains define *infrastructure* — how DNS and DKIM are handled for each domain:

Forward Mode

Simple forwarding without local DNS management:

```
{
  domain: 'forwarded.com',
  dnsMode: 'forward',
  dns: { forward: { skipDnsValidation: true, targetDomain: 'mail.target.com' } }
}
```

Internal DNS Mode

Uses DcRouter's built-in DNS server (requires `dnsNsDomains` + `dnsScopes`):

```
{
  domain: 'mail.example.com',
  dnsMode: 'internal-dns',
  dns: { internal: { mxPriority: 10, ttl: 3600 } },
  dkim: { selector: 'mail2024', keySize: 2048, rotateKeys: true, rotationInterval: 90 }
}
```

External DNS Mode

Uses existing DNS infrastructure with validation:

```
{
  domain: 'mail.external.com',
  dnsMode: 'external-dns',
  dns: { external: { requiredRecords: ['MX', 'SPF', 'DKIM', 'DMARC'] } },
```

```
  rateLimits: {
    inbound: { messagesPerMinute: 100, connectionsPerIp: 10 },
    outbound: { messagesPerMinute: 200 }
  }
}
```

Email Route Actions

Routes define *behavior* — what happens when an email matches:

Forward □□

Routes emails to an external SMTP server:

```
{
  name: 'forward-to-internal',
  match: { recipients: '*@company.com' },
  action: {
    type: 'forward',
    forward: {
      host: 'internal-mail.company.com',
      port: 25,
      auth: { user: 'relay-user', pass: 'relay-pass' },
      addHeaders: { 'X-Forwarded-By': 'dcrouter' }
    }
  }
}
```

Process ⚙

Full MTA processing with content scanning and delivery queues:

```
{
  name: 'process-notifications',
  match: { recipients: '*@notifications.company.com' },
  action: {
    type: 'process',
    process: { scan: true, dkim: true, queue: 'priority' }
  }
}
```

Deliver □□

Local mailbox delivery:

```
{
  name: 'deliver-local',
  match: { recipients: '*@local.company.com' },
  action: { type: 'deliver' }
}
```

Reject □□

Reject with custom SMTP response code:

```
{
  name: 'reject-spam-domain',
  match: { senders: '*@spam-domain.com', sizeRange: { min: 1000000 } },
  action: {
    type: 'reject',
    reject: { code: 550, message: 'Message rejected due to policy' }
  }
}
```

Route Matching

Routes support powerful matching criteria:

```
// Recipient patterns
match: { recipients: '*@example.com' }           // All addresses at domain
match: { recipients: 'admin@*' }                 // "admin" at any domain
match: { senders: ['*@trusted.com', '*@vip.com'] } // Multiple sender patterns

// IP-based matching (CIDR)
match: { clientIp: '192.168.0.0/16' }
match: { clientIp: ['10.0.0.0/8', '172.16.0.0/12'] }

// Authentication state
match: { authenticated: true }
```

```
// Header matching
match: { headers: { 'X-Priority': 'high', 'Subject': /urgent|emergency/i } }

// Size and content
match: { sizeRange: { min: 1000, max: 5000000 }, hasAttachments: true }
match: { subject: /invoice|receipt/i }
```

Email Security Stack

- **DKIM** — Automatic key generation, signing, and rotation for all domains
- **SPF** — Sender Policy Framework verification on inbound mail
- **DMARC** — Domain-based Message Authentication verification
- **IP Reputation** — Real-time IP reputation checking with caching
- **Content Scanning** — Spam, virus, and attachment scanning
- **Rate Limiting** — Hierarchical limits (global → domain → sender)
- **Bounce Management** — Automatic bounce detection and suppression lists

Email Deliverability

- **IP Warmup Manager** — Multi-stage warmup schedules for new IPs
- **Sender Reputation Monitor** — Per-domain reputation tracking and scoring
- **Connection Pooling** — Pooled outbound SMTP connections per destination

DNS Server

DcRouter includes an authoritative DNS server built on smartdns. It handles standard UDP DNS on port 53 and DNS-over-HTTPS via SmartProxy socket handler.

Enabling DNS

DNS is activated when both `dnsNsDomains` and `dnsScopes` are configured:

```
const router = new DcRouter({
  dnsNsDomains: ['ns1.example.com', 'ns2.example.com'],
  dnsScopes: ['example.com'],
  publicIp: '203.0.113.1',
  dnsRecords: [
```

```
    { name: 'example.com', type: 'A', value: '203.0.113.1' },
    { name: 'www.example.com', type: 'CNAME', value: 'example.com' },
    { name: 'example.com', type: 'MX', value: '10:mail.example.com' },
    { name: 'example.com', type: 'TXT', value: 'v=spf1 a mx ~all' }
  ]
});
```

Automatic DNS Records

DcRouter auto-generates:

- **NS records** for all domains in `dnsScopes`
- **SOA records** for authoritative zones
- **A records** for nameserver domains (`dnsNsDomains`)
- **MX, SPF, DKIM, DMARC records** for email domains with `internal-dns` mode
- **ACME challenge records** for certificate provisioning

Ingress Proxy Support

When `proxyIps` is configured, A records with `useIngressProxy: true` (default) will use the proxy IP instead of the real server IP — hiding your origin:

```
{
  proxyIps: ['198.51.100.1', '198.51.100.2'],
  dnsRecords: [
    { name: 'example.com', type: 'A', value: '203.0.113.1' },  // Will resolve to 198.51.100.1
    { name: 'ns1.example.com', type: 'A', value: '203.0.113.1', useIngressProxy: false }  //
Stays real IP
  ]
}
```

RADIUS Server

DcRouter includes a RADIUS server for network access control, built on smartradius.

Configuration

```
const router = new DcRouter({
  radiusConfig: {
    authPort: 1812,
    acctPort: 1813,
    clients: [
      {
        name: 'core-switch',
        ipRange: '192.168.1.0/24',
        secret: 'shared-secret',
        enabled: true
      }
    ],
    vlanAssignment: {
      defaultVlan: 100,
      allowUnknownMacs: true,
      mappings: [
        { mac: 'aa:bb:cc:dd:ee:ff', vlan: 10, enabled: true },   // Exact MAC
        { mac: 'aa:bb:cc', vlan: 20, enabled: true },             // OUI prefix
      ]
    },
    accounting: {
      enabled: true,
      retentionDays: 30
    }
  }
});
```

Components

Component	Purpose
RadiusServer	Main RADIUS server handling auth + accounting requests
VlanManager	MAC-to-VLAN mapping with exact, OUI, and wildcard patterns
AccountingManager	Session tracking, traffic metering, start/stop/interim updates

OpsServer API

RADIUS is fully manageable at runtime via the OpsServer API:

- Client management (add/remove/list NAS devices)
- VLAN mapping CRUD operations
- Session monitoring and forced disconnects
- Accounting summaries and statistics

Remote Ingress

DcRouter can act as a **hub** for distributed edge nodes using `@serve.zone/remoteingress`. Edge nodes accept incoming traffic at remote locations and tunnel it back to the hub over a single multiplexed connection. This is ideal for scenarios where you need to accept traffic at multiple geographic locations but process it centrally.

Enabling Remote Ingress

```
const router = new DcRouter({
  remoteIngressConfig: {
    enabled: true,
    tunnelPort: 8443,
    hubDomain: 'hub.example.com',  // Embedded in connection tokens
  },
  // Routes tagged with remoteIngress are auto-derived to edge listen ports
  smartProxyConfig: {
    routes: [
      {
        name: 'web-via-edge',
        match: { domains: ['app.example.com'], ports: [443] },
        action: {
          type: 'forward',
          targets: [{ host: '192.168.1.10', port: 8080 }],
          tls: { mode: 'terminate', certificate: 'auto' }
        },
        remoteIngress: { enabled: true }  // Edges will listen on port 443
      }
    ]
  }
});
```

```
await router.start();
```

Edge Registration

Edges are registered via the OpsServer API (or dashboard UI). Each edge gets a unique ID and secret:

```
// Via TypedRequest API
const createReq = new TypedRequest<IReq_CreateRemoteIngress>(
  'https://hub:3000/typedrequest', 'createRemoteIngress'
);
const { edge } = await createReq.fire({
  identity,
  name: 'edge-nyc-01',
  autoDerivePorts: true,
  tags: ['us-east'],
});
// edge.secret is returned only on creation — save it!
```

Connection Tokens □□

Instead of configuring edges with four separate values (hubHost, hubPort, edgeId, secret), DcRouter can generate a single **connection token** — an opaque base64url string that encodes everything:

```
// Via TypedRequest API
const tokenReq = new TypedRequest<IReq_GetRemoteIngressConnectionToken>(
  'https://hub:3000/typedrequest', 'getRemoteIngressConnectionToken'
);
const { token } = await tokenReq.fire({ identity, edgeId: 'edge-uuid' });
// token = "eyJoIjoiaHViLmV4YW1wbGUuY29tIiwicCI6ODQ0MywiZSI6I..."

// On the edge side, just pass the token:
const edge = new RemoteIngressEdge({ token });
await edge.start();
```

The token is generated using `remoteingress.encodeConnectionToken()` and contains `{ hubHost, hubPort, edgeId, secret }`. The `hubHost` comes from `remoteIngressConfig.hubDomain` (or can be

overridden per-request).

In the OpsServer dashboard, click **"Copy Token"** on any edge row to copy the connection token to your clipboard.

Auto-Derived Ports

When routes have `remoteIngress: { enabled: true }`, edges with `autoDerivePorts: true` (default) automatically pick up those routes' ports. You can also use `edgeFilter` to restrict which edges get which ports:

```
{
  name: 'web-route',
  match: { ports: [443] },
  action: { /* ... */ },
  remoteIngress: {
    enabled: true,
    edgeFilter: ['us-east', 'edge-uuid-123']  // Only edges with matching id or tags
  }
}
```

Dashboard Actions

The OpsServer Remote Ingress view provides:

Action	Description
Create Edge Node	Register a new edge with name, ports, tags
Enable / Disable	Toggle an edge on or off
Edit	Modify name, manual ports, auto-derive setting, tags
Regenerate Secret	Issue a new secret (invalidates the old one)
Copy Token	Generate and copy a base64url connection token to clipboard
Delete	Remove the edge registration

VPN Access Control

DcRouter integrates `@push.rocks/smartvpn` to provide VPN-based route access control. VPN clients connect via standard WireGuard or native WebSocket/QUIC transports, receive an IP from a configurable subnet, and can then access routes that are restricted to VPN-only traffic.

How It Works

1. **SmartVPN daemon** runs inside dcrouter with a Rust data plane (WireGuard via `boringtun`, custom protocol via Noise IK)
2. Clients connect and get assigned an IP from the VPN subnet (e.g. `10.8.0.0/24`)
3. **Smart split tunnel** — generated WireGuard configs auto-include the VPN subnet plus DNS-resolved IPs of VPN-gated domains. Domains from routes with `vpn.enabled` are resolved at config generation time, so clients route only the necessary traffic through the tunnel
4. Routes with `vpn: { enabled: true }` get `security.ipAllowList` dynamically injected (re-computed on every client change). With `mandatory: true` (default), the allowlist is replaced; with `mandatory: false`, VPN IPs are appended to existing rules
5. When `allowedServerDefinedClientTags` is set, only matching client IPs are injected (not the whole subnet)
6. SmartProxy enforces the allowlist — only authorized VPN clients can access protected routes
7. All VPN traffic is forced through SmartProxy via userspace NAT with PROXY protocol v2 — no root required

Destination Policy

By default, VPN client traffic is redirected to localhost (SmartProxy) via `forceTarget`. You can customize this with a destination policy:

```
// Default: all traffic → SmartProxy
destinationPolicy: { default: 'forceTarget', target: '127.0.0.1' }

// Allow direct access to a backend subnet
destinationPolicy: {
  default: 'forceTarget',
  target: '127.0.0.1',
  allowList: ['192.168.190.*'],     // direct access to this subnet
  blockList: ['192.168.190.1'],     // except the gateway
}

// Block everything except specific IPs
destinationPolicy: {
```

```
  default: 'block',
  allowList: ['10.0.0.*', '192.168.1.*'],
}
```

Configuration

```
const router = new DcRouter({
  vpnConfig: {
    enabled: true,
    subnet: '10.8.0.0/24',            // VPN client IP pool (default)
    wgListenPort: 51820,               // WireGuard UDP port (default)
    serverEndpoint: 'vpn.example.com', // Hostname in generated client configs
    dns: ['1.1.1.1', '8.8.8.8'],      // DNS servers pushed to clients

    // Pre-define VPN clients with server-defined tags
    clients: [
      { clientId: 'alice-laptop', serverDefinedClientTags: ['engineering'], description: 'Dev
laptop' },
      { clientId: 'bob-phone', serverDefinedClientTags: ['engineering', 'mobile'] },
      { clientId: 'carol-desktop', serverDefinedClientTags: ['finance'] },
    ],

    // Optional: customize destination policy (default: forceTarget → localhost)
    // destinationPolicy: { default: 'forceTarget', target: '127.0.0.1', allowList:
['192.168.1.*'] },
  },
  smartProxyConfig: {
    routes: [
      // □□VPN-only: any VPN client can access
      {
        name: 'internal-app',
        match: { domains: ['internal.example.com'], ports: [443] },
        action: {
          type: 'forward',
          targets: [{ host: '192.168.1.50', port: 8080 }],
          tls: { mode: 'terminate', certificate: 'auto' },
        },
        vpn: { enabled: true },
```

```
      },
      // □□VPN + tag-restricted: only 'engineering' tagged clients
      {
        name: 'eng-dashboard',
        match: { domains: ['eng.example.com'], ports: [443] },
        action: {
          type: 'forward',
          targets: [{ host: '192.168.1.51', port: 8080 }],
          tls: { mode: 'terminate', certificate: 'auto' },
        },
        vpn: { enabled: true, allowedServerDefinedClientTags: ['engineering'] },
        // → alice + bob can access, carol cannot
      },
      // □□Public: no VPN
      {
        name: 'public-site',
        match: { domains: ['example.com'], ports: [443] },
        action: {
          type: 'forward',
          targets: [{ host: '192.168.1.10', port: 80 }],
          tls: { mode: 'terminate', certificate: 'auto' },
        },
      },
    ],
  },
});
```

Client Tags

SmartVPN distinguishes between two types of client tags:

Tag Type	Set By	Purpose
`serverDefinedClientTags`	Admin (via config or API)	**Trusted** — used for route access control
`clientDefinedClientTags`	Connecting client	**Informational** — displayed in dashboard, never used for security

Routes with `allowedServerDefinedClientTags` only permit VPN clients whose admin-assigned tags match. Clients cannot influence their own server-defined tags.

Client Management via OpsServer

The OpsServer dashboard and API provide full VPN client lifecycle management:

- **Create client** — generates WireGuard keypairs, assigns IP, returns a ready-to-use `.conf` file
- **QR code** — scan with the WireGuard mobile app (iOS/Android) for instant setup
- **Enable / Disable** — toggle client access without deleting
- **Rotate keys** — generate fresh keypairs (invalidates old ones)
- **Export config** — download in WireGuard (`.conf`), SmartVPN (`.json`), or scan as QR code
- **Telemetry** — per-client bytes sent/received, keepalives, rate limiting
- **Delete** — remove a client and revoke access

Standard WireGuard clients on any platform (iOS, Android, macOS, Windows, Linux) can connect using the generated `.conf` file or by scanning the QR code — no custom VPN software needed.

Certificate Management

DcRouter uses `@push.rocks/smartacme` v9 for ACME certificate provisioning. smartacme v9 brings significant improvements over previous versions:

How It Works

When a `dnsChallenge` is configured (e.g. with a Cloudflare API key), DcRouter creates a SmartAcme instance that handles DNS-01 challenges for automatic certificate provisioning. SmartProxy calls the `certProvisionFunction` whenever a route needs a TLS certificate, and SmartAcme takes care of the rest.

```
const router = new DcRouter({
  smartProxyConfig: {
    routes: [
      {
        name: 'secure-app',
        match: { domains: ['app.example.com'], ports: [443] },
        action: {
          type: 'forward',
          targets: [{ host: '192.168.1.10', port: 8080 }],
          tls: { mode: 'terminate', certificate: 'auto' }  // ← triggers ACME provisioning
        }
```

```
      }
    ],
    acme: { email: 'admin@example.com', enabled: true, useProduction: true }
  },
  tls: { contactEmail: 'admin@example.com' },
  dnsChallenge: { cloudflareApiKey: process.env.CLOUDFLARE_API_KEY }
});
```

smartacme v9 Features

Feature	Description
Per-domain deduplication	Concurrent requests for the same domain share a single ACME operation
Global concurrency cap	Default 5 parallel ACME operations to prevent overload
Account rate limiting	Sliding window (250 orders / 3 hours) to stay within ACME provider limits
Structured errors	`AcmeError` with `isRetryable`, `isRateLimited`, `retryAfter` fields
Clean shutdown	`stop()` properly destroys HTTP agents and DNS clients

Per-Domain Backoff

DcRouter's `CertProvisionScheduler` adds **per-domain exponential backoff** on top of smartacme's built-in protections. If a DNS-01 challenge fails for a domain:

1. The failure is recorded (persisted to storage)
2. The domain enters backoff: `min(failures² × 1 hour, 24 hours)`
3. Subsequent requests for that domain are rejected until the backoff expires
4. On success, the backoff is cleared

This prevents hammering ACME servers for domains with persistent issues (e.g. missing DNS delegation).

Fallback to HTTP-01

If DNS-01 fails, the `certProvisionFunction` returns `'http01'` to tell SmartProxy to fall back to HTTP-01 challenge validation. This provides a safety net for domains where DNS-01 isn't viable.

Certificate Storage

Certificates are persisted via the `StorageBackedCertManager` which uses DcRouter's `StorageManager`. This means certs survive restarts and don't need to be re-provisioned unless they expire.

Dashboard

The OpsServer includes a **Certificates** view showing:

- All domains with their certificate status (valid, expiring, expired, failed)
- Certificate source (ACME, provision function, static)
- Expiry dates and issuer information
- Backoff status for failed domains
- One-click reprovisioning per domain
- Certificate import and export

Storage & Caching

StorageManager

Provides a unified key-value interface with three backends:

```
// Filesystem backend
storage: { fsPath: '/var/lib/dcrouter/data' }

// Custom backend (Redis, S3, etc.)
storage: {
  readFunction: async (key) => await redis.get(key),
  writeFunction: async (key, value) => await redis.set(key, value)
}

// In-memory (development only — data lost on restart)
// Simply omit the storage config
```

Used for: TLS certificates, DKIM keys, email routes, bounce/suppression lists, IP reputation data, domain configs, cert backoff state, remote ingress edge registrations.

Cache Database

An embedded MongoDB-compatible database (via smartdata + smartdb) for persistent caching with automatic TTL cleanup:

```
cacheConfig: {
  enabled: true,
  storagePath: '~/.serve.zone/dcrouter/tsmdb',
  dbName: 'dcrouter',
  cleanupIntervalHours: 1,
  ttlConfig: {
    emails: 30,         // days
    ipReputation: 1,    // days
    bounces: 30,        // days
    dkimKeys: 90,       // days
    suppression: 30     // days
  }
}
```

Cached document types: `CachedEmail`, `CachedIPReputation`.

Security Features

IP Reputation Checking

Automatic IP reputation checks on inbound connections with configurable caching:

```
// IP reputation is checked automatically for inbound SMTP connections.
// Results are cached according to cacheConfig.ttlConfig.ipReputation.
```

Rate Limiting

Hierarchical rate limits with three levels of specificity:

```
// Global defaults (via emailConfig.defaults.rateLimits)
defaults: {
  rateLimits: {
```

```
    inbound: { messagesPerMinute: 50, connectionsPerIp: 5, recipientsPerMessage: 50 },
    outbound: { messagesPerMinute: 100 }
  }
}

// Per-domain overrides (in domain config)
{
  domain: 'high-volume.com',
  rateLimits: {
    outbound: { messagesPerMinute: 500 }  // Override for this domain
  }
}
```

Precedence: Domain-specific > Pattern-specific > Global

Content Scanning

```
action: {
  type: 'process',
  options: {
    contentScanning: true,
    scanners: [
      { type: 'spam', threshold: 5.0, action: 'tag' },
      { type: 'virus', action: 'reject' },
      { type: 'attachment', blockedExtensions: ['.exe', '.bat', '.scr'], action: 'reject' }
    ]
  }
}
```

OpsServer Dashboard

The OpsServer provides a web-based management interface served on port 3000 by default (configurable via `opsServerPort`). It's built with modern web components using @design.estate/dees-catalog.

Dashboard Views

View	Description
◫ **Overview**	Real-time server stats, CPU/memory, connection counts, email throughput
◫ **Network**	Active connections, top IPs, throughput rates, SmartProxy metrics
◫ **Email**	Queue monitoring (queued/sent/failed), bounce records, security incidents
◫ **Routes**	Merged route list (hardcoded + programmatic), create/edit/toggle/override routes
◫ **API Tokens**	Token management with scopes, create/revoke/roll/toggle
◫ **Certificates**	Domain-centric certificate overview, status, backoff info, reprovisioning, import/export
◫ **RemoteIngress**	Edge node management, connection status, token generation, enable/disable
◫ **VPN**	VPN client management, server status, create/toggle/export/rotate/delete clients
◫ **RADIUS**	NAS client management, VLAN mappings, session monitoring, accounting
◫ **Logs**	Real-time log viewer with level filtering and search
⚙ **Configuration**	Read-only view of current system configuration
◫ **Security**	IP reputation, rate limit status, blocked connections

API Endpoints

All management is done via TypedRequest over HTTP POST to `/typedrequest`:

```
// Authentication
'adminLoginWithUsernameAndPassword'   // Login with credentials → returns JWT identity
'verifyIdentity'                       // Verify JWT token validity
'adminLogout'                          // End admin session

// Statistics & Health
'getServerStatistics'                  // Uptime, CPU, memory, connections
'getHealthStatus'                      // System health check
'getCombinedMetrics'                   // All metrics in one call

// Email Operations
'getAllEmails'                          // List all emails (queued/sent/failed)
```

```
'getEmailDetail'                      // Full detail for a specific email
'resendEmail'                         // Re-queue a failed email

// Certificates
'getCertificateOverview'              // Domain-centric certificate status
'reprovisionCertificate'              // Reprovision by route name (legacy)
'reprovisionCertificateDomain'        // Reprovision by domain (preferred)
'importCertificate'                   // Import a certificate
'exportCertificate'                   // Export a certificate
'deleteCertificate'                   // Delete a certificate

// Remote Ingress
'getRemoteIngresses'                  // List all edge registrations
'createRemoteIngress'                 // Register a new edge
'updateRemoteIngress'                 // Update edge settings
'deleteRemoteIngress'                 // Remove an edge
'regenerateRemoteIngressSecret'       // Issue a new secret
'getRemoteIngressStatus'              // Runtime status of all edges
'getRemoteIngressConnectionToken'     // Generate a connection token for an edge

// Route Management (JWT or API token auth)
'getMergedRoutes'                     // List all routes (hardcoded + programmatic)
'createRoute'                         // Create a new programmatic route
'updateRoute'                         // Update a programmatic route
'deleteRoute'                         // Delete a programmatic route
'toggleRoute'                         // Enable/disable a programmatic route
'setRouteOverride'                    // Override a hardcoded route
'removeRouteOverride'                 // Remove a hardcoded route override

// API Token Management (admin JWT only)
'createApiToken'                      // Create API token → returns raw value once
'listApiTokens'                       // List all tokens (without secrets)
'revokeApiToken'                      // Delete an API token
'rollApiToken'                        // Regenerate token secret
'toggleApiToken'                      // Enable/disable a token

// Configuration (read-only)
'getConfiguration'                    // Current system config
```

```
// Logs
'getRecentLogs'                    // Retrieve system logs with filtering
'getLogStream'                     // Stream live logs

// VPN
'getVpnClients'                    // List all registered VPN clients
'getVpnStatus'                     // VPN server status (running, subnet, port, keys)
'createVpnClient'                  // Create client → returns WireGuard config (shown
once)
'deleteVpnClient'                  // Remove a VPN client
'enableVpnClient'                  // Enable a disabled client
'disableVpnClient'                 // Disable a client
'rotateVpnClientKey'               // Generate new keys (invalidates old ones)
'exportVpnClientConfig'            // Export WireGuard (.conf) or SmartVPN (.json) config
'getVpnClientTelemetry'            // Per-client bytes sent/received, keepalives

// RADIUS
'getRadiusSessions'                // Active RADIUS sessions
'getRadiusClients'                 // List NAS clients
'getRadiusStatistics'              // RADIUS stats
'setRadiusClient'                  // Add/update NAS client
'removeRadiusClient'               // Remove NAS client
'getVlanMappings'                  // List VLAN mappings
'setVlanMapping'                   // Add/update VLAN mapping
'removeVlanMapping'                // Remove VLAN mapping
'testVlanAssignment'               // Test what VLAN a MAC gets
```

API Client

DcRouter ships with a typed, object-oriented API client for programmatic management of a running instance. Install it separately or import from the main package:

```
pnpm add @serve.zone/dcrouter-apiclient
# or import from the main package:
# import { DcRouterApiClient } from '@serve.zone/dcrouter/apiclient';
```

Quick Example

```
import { DcRouterApiClient } from '@serve.zone/dcrouter/apiclient';

const client = new DcRouterApiClient({ baseUrl: 'https://dcrouter.example.com' });
await client.login('admin', 'password');

// OO resource instances with methods
const { routes } = await client.routes.list();
await routes[0].toggle(false);

// Builder pattern for creation
const newRoute = await client.routes.build()
  .setName('api-gateway')
  .setMatch({ ports: 443, domains: ['api.example.com'] })
  .setAction({ type: 'forward', targets: [{ host: 'backend', port: 8080 }] })
  .setTls({ mode: 'terminate', certificate: 'auto' })
  .save();

// Manage certificates
const { certificates, summary } = await client.certificates.list();
await certificates[0].reprovision();

// Create API tokens with builder
const token = await client.apiTokens.build()
  .setName('ci-token')
  .setScopes(['routes:read', 'routes:write'])
  .setExpiresInDays(90)
  .save();
console.log(token.tokenValue); // only available at creation

// Remote ingress edges
const edge = await client.remoteIngress.build()
  .setName('edge-nyc-01')
  .setListenPorts([80, 443])
  .save();
const connToken = await edge.getConnectionToken();

// Read-only managers
const health = await client.stats.getHealth();
const config = await client.config.get();
```

```
const { logs } = await client.logs.getRecent({ level: 'error', limit: 50 });
```

Resource Managers

Manager	Operations
`client.routes`	`list()`, `create()`, `build()` → Route: `update()`, `delete()`, `toggle()`, `setOverride()`, `removeOverride()`
`client.certificates`	`list()`, `import()` → Certificate: `reprovision()`, `delete()`, `export()`
`client.apiTokens`	`list()`, `create()`, `build()` → ApiToken: `revoke()`, `roll()`, `toggle()`
`client.remoteIngress`	`list()`, `getStatuses()`, `create()`, `build()` → RemoteIngress: `update()`, `delete()`, `regenerateSecret()`, `getConnectionToken()`
`client.stats`	`getServer()`, `getEmail()`, `getDns()`, `getSecurity()`, `getConnections()`, `getQueues()`, `getHealth()`, `getNetwork()`, `getCombined()`
`client.config`	`get(section?)`
`client.logs`	`getRecent()`, `getStream()`
`client.emails`	`list()` → Email: `getDetail()`, `resend()`
`client.radius`	`.clients`, `.vlans`, `.sessions` sub-managers + `getStatistics()`, `getAccountingSummary()`

See the full API client documentation for detailed usage of every manager, builder, and resource class.

API Reference

DcRouter Class

```
import { DcRouter } from '@serve.zone/dcrouter';

const router = new DcRouter(options: IDcRouterOptions);
```

Methods

Method	Description

`start(): Promise<void>`	Start all configured services
`stop(): Promise<void>`	Gracefully stop all services
`updateSmartProxyConfig(config): Promise<void>`	Hot-update SmartProxy routes
`updateEmailConfig(config): Promise<void>`	Hot-update email configuration
`updateEmailRoutes(routes): Promise<void>`	Update email routing rules at runtime
`updateRadiusConfig(config): Promise<void>`	Hot-update RADIUS configuration
`getStats(): any`	Get real-time statistics from all services

Properties

Property	Type	Description
`options`	`IDcRouterOptions`	Current configuration
`smartProxy`	`SmartProxy`	SmartProxy instance
`smartAcme`	`SmartAcme`	SmartAcme v9 certificate manager instance
`emailServer`	`UnifiedEmailServer`	Email server instance (from smartmta)
`dnsServer`	`DnsServer`	DNS server instance
`radiusServer`	`RadiusServer`	RADIUS server instance
`remoteIngressManager`	`RemoteIngressManager`	Edge registration CRUD manager
`tunnelManager`	`TunnelManager`	Tunnel lifecycle and status manager
`vpnManager`	`VpnManager`	VPN server lifecycle and client CRUD manager
`storageManager`	`StorageManager`	Storage backend
`opsServer`	`OpsServer`	OpsServer/dashboard instance
`metricsManager`	`MetricsManager`	Metrics collector
`cacheDb`	`CacheDb`	Cache database instance
`certProvisionScheduler`	`CertProvisionScheduler`	Per-domain backoff scheduler for cert provisioning
`certificateStatusMap`	`Map<string, ...>`	Domain-keyed certificate status from SmartProxy events

Re-exported Types

DcRouter re-exports key types for convenience:

```
import {
  DcRouter,
  IDcRouterOptions,
  UnifiedEmailServer,
  type IUnifiedEmailServerOptions,
  type IEmailRoute,
  type IEmailDomainConfig,
  type IHttp3Config,
} from '@serve.zone/dcrouter';
```

Sub-Modules

DcRouter is published as a monorepo with separately-installable interface and web packages:

Package	Description	Install
`@serve.zone/dcrouter`	Main package — the full router	`pnpm add @serve.zone/dcrouter`
`@serve.zone/dcrouter-interfaces`	TypedRequest interfaces for the OpsServer API	`pnpm add @serve.zone/dcrouter-interfaces`
`@serve.zone/dcrouter-apiclient`	OO API client with builder pattern	`pnpm add @serve.zone/dcrouter-apiclient`
`@serve.zone/dcrouter-web`	Web dashboard components	`pnpm add @serve.zone/dcrouter-web`

You can also import directly from the main package:

```
import { data, requests } from '@serve.zone/dcrouter/interfaces';
import { DcRouterApiClient } from '@serve.zone/dcrouter/apiclient';
```

Testing

DcRouter includes a comprehensive test suite covering all system components:

```
# Run all tests
pnpm test

# Run a specific test file
tstest test/test.jwt-auth.ts --verbose
```

```
# Run with extended timeout
tstest test/test.opsserver-api.ts --verbose --timeout 60
```

Test Coverage

Test File	Area	Tests
`test.apiclient.ts`	API client instantiation, builders, resource hydration, exports	18
`test.contentscanner.ts`	Content scanning (spam, phishing, malware, attachments)	13
`test.dcrouter.email.ts`	Email config, domain and route setup	4
`test.dns-server-config.ts`	DNS record parsing, grouping, extraction	5
`test.dns-socket-handler.ts`	DNS socket handler and route generation	6
`test.errors.ts`	Error classes, handler, retry utilities	5
`test.http3-augmentation.ts`	HTTP/3 route augmentation, qualification, opt-in/out, QUIC settings	20
`test.ipreputationchecker.ts`	IP reputation, DNSBL, caching, risk classification	10
`test.jwt-auth.ts`	JWT login, verification, logout, invalid credentials	8
`test.opsserver-api.ts`	Health, statistics, configuration, log APIs	8
`test.protected-endpoint.ts`	Admin auth, identity verification, public endpoints	8
`test.storagemanager.ts`	Memory, filesystem, custom backends, concurrency	8

Docker / OCI Container Deployment

DcRouter ships with a production-ready `Dockerfile` and supports environment-variable-driven configuration for OCI container deployments. The container image includes tini as PID 1 (via the base image), proper health checks, and configurable resource limits. When

`DCROUTER_MODE=OCI_CONTAINER` is set, DcRouter automatically reads configuration from environment variables (and optionally from a JSON config file).

Running with Docker

```
docker run -d \
  --ulimit nofile=65536:65536 \
  -e DCROUTER_TLS_EMAIL=admin@example.com \
  -e DCROUTER_PUBLIC_IP=203.0.113.1 \
  -e DCROUTER_DNS_NS_DOMAINS=ns1.example.com,ns2.example.com \
  -e DCROUTER_DNS_SCOPES=example.com \
  -p 80:80 -p 443:443 -p 25:25 -p 587:587 -p 465:465 \
  -p 53:53/udp -p 3000:3000 -p 8443:8443 \
  code.foss.global/serve.zone/dcrouter:latest
```

> ⚡ **Production tip:** Always set `--ulimit nofile=65536:65536` for production deployments. DcRouter will log a warning at startup if the file descriptor limit is below 65536.

Environment Variables

Variable	Description	Default	Example
`DCROUTER_MODE`	Container mode (set automatically in image)	`OCI_CONTAINER`	—
`DCROUTER_CONFIG_PATH`	Path to JSON config file (env vars override)	—	`/config/dcrouter.json`
`DCROUTER_BASE_DIR`	Base data directory	`~/.serve.zone/dcrouter`	`/data/dcrouter`
`DCROUTER_TLS_EMAIL`	ACME contact email	—	`admin@example.com`
`DCROUTER_TLS_DOMAIN`	Primary TLS domain	—	`example.com`
`DCROUTER_PUBLIC_IP`	Public IP for DNS records	—	`203.0.113.1`
`DCROUTER_PROXY_IPS`	Comma-separated ingress proxy IPs	—	`198.51.100.1,198.51.100.2`
`DCROUTER_DNS_NS_DOMAINS`	Comma-separated nameserver domains	—	`ns1.example.com,ns2.example.com`
`DCROUTER_DNS_SCOPES`	Comma-separated authoritative domains	—	`example.com,other.com`

Variable	Description	Default	Example
`DCROUTER_EMAIL_HOSTNAME`	SMTP server hostname	—	`mail.example.com`
`DCROUTER_EMAIL_PORTS`	Comma-separated email ports	—	`25,587,465`
`DCROUTER_CACHE_ENABLED`	Enable/disable cache database	`true`	`false`
`DCROUTER_HEAP_SIZE`	Node.js V8 heap size in MB	`512`	`1024`
`DCROUTER_MAX_CONNECTIONS`	Global max concurrent connections	`50000`	`100000`
`DCROUTER_MAX_CONNECTIONS_PER_IP`	Max connections per source IP	`100`	`200`
`DCROUTER_CONNECTION_RATE_LIMIT`	Max new connections/min per IP	`600`	`1200`

Exposed Ports

The container exposes all service ports:

Port(s)	Protocol	Service
80, 443	TCP	HTTP/HTTPS (SmartProxy)
25, 587, 465	TCP	SMTP, Submission, SMTPS
53	TCP/UDP	DNS
1812, 1813	UDP	RADIUS auth/acct
3000	TCP	OpsServer dashboard
8443	TCP	Remote ingress tunnels
51820	UDP	WireGuard VPN
29000–30000	TCP	Dynamic port range

Building the Image

```
pnpm run build:docker    # Build the container image
pnpm run release:docker  # Push to registry
```

The Docker build supports multi-platform (`linux/amd64`, `linux/arm64`) via tsdocker.

License and Legal Information

This repository contains open-source code licensed under the MIT License. A copy of the license can be found in the license file.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH or third parties, and are not included within the scope of the MIT license granted herein.

Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines or the guidelines of the respective third-party owners, and any usage must be approved in writing. Third-party trademarks used herein are the property of their respective owners and used only in a descriptive manner, e.g. for an implementation of an API or similar.

Company Information

Task Venture Capital GmbH
Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

changelog.md for @serve.zone/dcrouter

2026-03-31 - 11.23.5 - fix(config)

correct VPN mandatory flag default handling in route config manager

- Changes the VPN mandatory check so it only applies when explicitly set to true, matching the updated default behavior of false.
- Prevents routes from being treated as VPN-mandatory when the setting is omitted.

2026-03-31 - 11.23.4 - fix(deps)

bump @push.rocks/smartvpn to 1.17.1

- Updates the @push.rocks/smartvpn dependency from 1.16.5 to 1.17.1.

2026-03-31 - 11.23.3 - fix(ts_web)

update appstate to import interfaces from source TypeScript module path

- Replaces the appstate interfaces import from ../dist_ts_interfaces/index.js with ../ts_interfaces/index.js.
- Aligns the web app state module with the source interface location instead of the built distribution path.

2026-03-31 - 11.23.2 - fix(repo)

no changes to commit

2026-03-31 - 11.23.1 - fix(repo)

no changes to commit

2026-03-31 - 11.23.0 - feat(vpn)

support optional non-mandatory VPN route access and align route config with enabled semantics

- rename route VPN configuration from `required` to `enabled` across code, docs, and examples
- add `vpn.mandatory` to control whether VPN allowlists replace or extend existing `security.ipAllowList` rules
- improve VPN client status matching in the ops view by falling back to assigned IP when client IDs differ

2026-03-31 - 11.22.0 - feat(vpn)

add VPN client editing and connected client visibility in ops server

- Adds API support to list currently connected VPN clients and update client metadata without rotating keys
- Updates the web VPN view to show live connection status, client detail telemetry, and separate enable/disable actions
- Refreshes documentation for smart split tunnel behavior, QR code setup/export, and storage architecture
- Bumps @push.rocks/smartvpn from 1.16.4 to 1.16.5

2026-03-31 - 11.21.5 - fix(routing)

apply VPN route allowlists dynamically after VPN clients load

- Moves VPN security injection for hardcoded and programmatic routes into RouteConfigManager.applyRoutes() so allowlists are generated from current VPN client state.
- Re-applies routes after starting the VPN manager to ensure tag-based ipAllowLists are available once VPN clients are loaded.

- Avoids caching constructor routes with stale VPN security baked in while preserving HTTP/3 route augmentation.

2026-03-31 - 11.21.4 - fix(deps)

bump @push.rocks/smartvpn to 1.16.4

- Updates the @push.rocks/smartvpn dependency from 1.16.3 to 1.16.4 in package.json.

2026-03-31 - 11.21.3 - fix(deps)

bump @push.rocks/smartvpn to 1.16.3

- Updates the @push.rocks/smartvpn dependency from 1.16.2 to 1.16.3.

2026-03-31 - 11.21.2 - fix(deps)

bump @push.rocks/smartvpn to 1.16.2

- Updates the @push.rocks/smartvpn dependency from 1.16.1 to 1.16.2 in package.json.

2026-03-31 - 11.21.1 - fix(vpn)

resolve VPN-gated route domains into per-client AllowedIPs with cached DNS lookups

- Derive WireGuard AllowedIPs from DNS A records of matched vpn.required route domains instead of only configured public proxy IPs.
- Cache resolved domain IPs for 5 minutes and fall back to stale results on DNS lookup failures.
- Make per-client AllowedIPs generation asynchronous throughout VPN config export and regeneration flows.

2026-03-31 - 11.21.0 - feat(vpn)

add tag-aware WireGuard AllowedIPs for VPN-gated routes

- compute per-client WireGuard AllowedIPs from server-defined client tags and VPN-required proxy routes
- include the server public IP in AllowedIPs when a client can access VPN-gated domains so routed traffic reaches the proxy
- preserve and inject WireGuard private keys in generated and exported client configs for valid exports

2026-03-31 - 11.20.1 - fix(vpn-manager)

persist WireGuard private keys for valid client exports and QR codes

- Store each client's WireGuard private key when creating and rotating keys.
- Inject the stored private key into exported WireGuard configs so generated configs are complete and scannable.

2026-03-30 - 11.20.0 - feat(vpn-ui)

add QR code export for WireGuard client configurations

- adds a QR code action for newly created WireGuard configs in the VPN operations view
- adds a QR code export option for existing VPN clients alongside file downloads
- introduces qrcode and @types/qrcode dependencies and exposes the plugin for web UI use

2026-03-30 - 11.19.1 - fix(vpn)

configure SmartVPN client exports with explicit server endpoint and split-tunnel allowed IPs

- Pass the configured WireGuard server endpoint directly to SmartVPN instead of rewriting generated client configs in dcrouter.
- Set client allowed IPs to the VPN subnet so generated WireGuard configs default to split-tunnel routing.
- Update documentation to reflect SmartVPN startup, dashboard/API coverage, and the new split-tunnel behavior.
- Bump @push.rocks/smartvpn from 1.14.0 to 1.16.1 to support the updated VPN configuration flow.

2026-03-30 - 11.19.0 - feat(vpn)

document tag-based VPN access control, declarative clients, and destination policy options

- Adds documentation for restricting VPN-protected routes with allowedServerDefinedClientTags.
- Documents pre-defined VPN clients in configuration via vpnConfig.clients.
- Describes destinationPolicy behavior for forceTarget, allow, and block traffic handling.
- Updates interface docs to reflect serverDefinedClientTags and revised VPN server status fields.

2026-03-30 - 11.18.0 - feat(vpn-ui)

add format selection for VPN client config exports

- Show an export modal that lets operators choose between WireGuard (.conf) and SmartVPN (.json) client configs.
- Update VPN client row actions to read the selected item from actionData for toggle, export, rotate keys, and delete handlers.

2026-03-30 - 11.17.0 - feat(vpn)

expand VPN operations view with client management and config export actions

- adds predefined VPN clients to the dev server configuration for local testing
- adds table actions to create clients, export WireGuard configs, rotate client keys, toggle access, and delete clients
- updates the VPN view layout and stats grid binding to match the current component API

2026-03-30 - 11.16.0 - feat(vpn)

add destination-based VPN routing policy and standardize socket proxy forwarding

- replace configurable VPN forwarding mode with socket-based forwarding and always enable proxy protocol support to SmartProxy from localhost
- add destinationPolicy configuration for controlling default VPN traffic handling, including forceTarget, allow, and block rules

- remove forwarding mode reporting from VPN status APIs, logs, and ops UI to reflect the simplified VPN runtime model
- update @push.rocks/smartvpn to 1.14.0 to support the new VPN routing behavior

2026-03-30 - 11.15.0 - feat(vpn)

add tag-based VPN route access control and support configured initial VPN clients

- allow VPN-protected routes to restrict access to clients with matching server-defined tags instead of always permitting the full VPN subnet
- create configured VPN clients automatically on startup and re-apply routes when VPN clients change
- rename VPN client tag fields to serverDefinedClientTags across APIs, interfaces, handlers, and UI with legacy tag migration on load
- upgrade @push.rocks/smartvpn from 1.12.0 to 1.13.0

2026-03-30 - 11.14.0 - feat(docs)

document VPN access control and add OpsServer VPN navigation

- Adds comprehensive README documentation for VPN access control, configuration, operating modes, and client management
- Updates TypeScript interface documentation with VPN-related route, client, status, telemetry, and API request types
- Extends web dashboard documentation and router view list to include VPN management

2026-03-30 - 11.13.0 - feat(vpn)

add VPN server management and route-based VPN access control

- introduces a VPN manager backed by @push.rocks/smartvpn with persisted server keys and client registrations
- adds ops API handlers and typed request interfaces for VPN client lifecycle, status, config export, and telemetry
- adds ops dashboard VPN view and application state for managing VPN clients from the web UI
- supports vpn.required on routes by injecting VPN subnet allowlists into static and programmatic SmartProxy routes

- configures SmartProxy to accept proxy protocol in VPN socket forwarding mode to preserve client tunnel IPs

2026-03-27 - 11.12.4 - fix(acme)

use X509 certificate expiry when reporting ACME certificate validity

- Parse the actual X509 validTo value from the PEM public certificate and fall back to SmartAcme's stored expiry if parsing fails
- Update reported certificate expiry data and event communication timestamps to use the verified validity date
- Bump @push.rocks/smartacme to ^9.3.1 and @push.rocks/smartproxy to ^27.1.0

2026-03-27 - 11.12.3 - fix(dcrouter)

re-trigger auto certificate provisioning after SmartAcme becomes ready

- clear certificate provisioning scheduler state before retrying startup-affected routes
- use route updates to re-run certificate provisioning for all current auto-cert routes
- remove the unused single-route domain lookup helper

2026-03-27 - 11.12.2 - fix(dcrouter)

guard auto certificate reprovisioning against unnamed routes

- Only re-triggers certificate provisioning for auto-cert routes when a route name is present.
- Prevents reprovision attempts from running with an undefined route name and reduces warning noise.

2026-03-27 - 11.12.1 - fix(dcrouter)

retry auto certificate provisioning after SmartAcme becomes ready

- detects certificates that failed during startup before the DNS-01 provider was available
- clears provisioning backoff and failed status for affected domains before retrying
- re-triggers auto certificate provisioning for SmartProxy routes once SmartAcme is ready

2026-03-27 - 11.12.0 - feat(web-ui)

pause dashboard polling, sockets, and chart updates when the tab is hidden

- replace interval-based auto-refresh with scheduled actions using visibility-aware auto-pause
- disconnect and reconnect the TypedSocket on tab visibility changes to avoid background log buildup
- batch pushed log entries per animation frame and add an in-flight refresh guard to reduce unnecessary re-renders and overlapping requests
- update state subscriptions to use select() and document the new tab visibility optimization behavior
- bump smartdb, smartproxy, smartstate, remoteingress, dees-element, and tstest dependencies

2026-03-26 - 11.11.0 - feat(docker,cache,proxy)

improve container runtime defaults and add configurable connection limits

- replace the embedded cache backend integration from smartmongo LocalTsmDb to smartdb LocalSmartDb
- add OCI container settings for heap size, threadpool size, expanded exposed ports, image metadata, and a direct node startup command
- introduce startup checks for file descriptor limits and warn when container nofile limits are too low for production

- set gateway-oriented SmartProxy default limits and allow max connections, per-IP connections, and rate limits to be configured through OCI environment variables

2026-03-26 - 11.10.7 - fix(sms)

update sms service to use async ProjectInfo initialization

- Replace direct ProjectInfo construction with the async create() factory in the SMS service startup flow
- Bump related dependencies including @push.rocks/projectinfo, @push.rocks/smartdata, @push.rocks/smartmongo, @serve.zone/remoteingress, and @git.zone/tstest

2026-03-26 - 11.10.6 - fix(typescript)

tighten TypeScript null safety and error handling across backend and ops UI

- add explicit unknown error typing and safe message access in logging and handler code
- mark deferred-initialized class properties with definite assignment assertions to satisfy stricter TypeScript checks
- harden ops web state access and action return types with non-null assertions and explicit Promise state typing
- update storage reads to allow missing values and align license file references with the lowercase license filename

2026-03-26 - 11.10.5 - fix(build)

rename smart tooling config to .smartconfig.json and update package references

- Moves the shared tool configuration from npmextra.json to .smartconfig.json.
- Updates package.json published files and documentation to reference the new config file.
- Refreshes several development and runtime dependency versions alongside the config migration.

2026-03-24 - 11.10.4 - fix(monitoring)

handle multiple protocol cache entries per backend in metrics output

- Group detected protocol cache entries by backend host and port so multiple domain-specific records are preserved.
- Emit one backend metrics row per cached domain and avoid dropping unmatched protocol cache entries by tracking seen entries with a composite host:port:domain key.
- Use cached protocol values when available while keeping backend-only rows for metrics without protocol cache data.

2026-03-23 - 11.10.3 - fix(deps)

bump tstest, smartmetrics, and taskbuffer to latest patch releases

- update @git.zone/tstest from ^3.5.0 to ^3.5.1
- update @push.rocks/smartmetrics from ^3.0.2 to ^3.0.3
- update @push.rocks/taskbuffer from ^8.0.0 to ^8.0.2

2026-03-23 - 11.10.2 - fix(deps)

bump @api.global/typedserver to ^8.4.6 and @push.rocks/smartproxy to ^26.2.1

- Updates @api.global/typedserver from ^8.4.2 to ^8.4.6
- Updates @push.rocks/smartproxy from ^26.2.0 to ^26.2.1

2026-03-23 - 11.10.1 - fix(deps)

bump @push.rocks/smartproxy to ^26.2.0

- Updates the @push.rocks/smartproxy dependency from ^26.1.0 to ^26.2.0 in package.json.

2026-03-23 - 11.10.0 - feat(monitoring)

add backend protocol metrics to network stats and ops dashboard

- Expose backend protocol, connection, error, and suppression metrics in stats responses.
- Add typed backend info interfaces and app state support for backend metrics.
- Render a new backend protocols table in the ops network view with detail modal and suppression badges.
- Update smartproxy and lik dependencies to support backend protocol metrics collection.

2026-03-21 - 11.9.1 - fix(lifecycle)

clean up service subscriptions, proxy retries, and stale runtime state on shutdown

- unsubscribe from ServiceManager event streams and use one-time signal handlers to avoid duplicate shutdown execution
- reset existing SmartProxy instances before retry setup and prune expired certificate backoff cache entries
- add periodic sweeping and shutdown cleanup for stale RADIUS accounting sessions

2026-03-20 - 11.9.0 - feat(dcrouter)

add service manager lifecycle orchestration and health-based ops status reporting

- register dcrouter components with a taskbuffer ServiceManager using dependencies, retries, and critical/optional service roles
- update ops stats health output to reflect aggregated service manager state and per-service error or retry details
- add @push.rocks/taskbuffer to shared plugins and project dependencies for service lifecycle management

2026-03-20 - 11.8.11 - fix(deps)

bump @push.rocks/smartproxy to ^25.17.10

- Updates the @push.rocks/smartproxy dependency from ^25.17.9 to ^25.17.10 in package.json

2026-03-20 - 11.8.10 - fix(deps)

bump @push.rocks/smartproxy to ^25.17.9

- Updates @push.rocks/smartproxy from ^25.17.8 to ^25.17.9 in package.json

2026-03-20 - 11.8.9 - fix(deps)

bump @push.rocks/smartproxy to ^25.17.8

- Updates the @push.rocks/smartproxy dependency from ^25.17.7 to ^25.17.8.

2026-03-20 - 11.8.8 - fix(deps)

bump @push.rocks/smartproxy to ^25.17.7

- Updates the @push.rocks/smartproxy dependency from ^25.17.4 to ^25.17.7 in package.json.

2026-03-20 - 11.8.7 - fix(deps)

bump @push.rocks/smartproxy to ^25.17.4

- updates @push.rocks/smartproxy from ^25.17.3 to ^25.17.4 in package.json

2026-03-20 - 11.8.6 - fix(deps)

bump @push.rocks/smartproxy to ^25.17.3

- updates @push.rocks/smartproxy from ^25.17.1 to ^25.17.3 in package.json

2026-03-20 - 11.8.5 - fix(deps)

bump @push.rocks/smartproxy to ^25.17.1

- Updates the @push.rocks/smartproxy dependency from ^25.17.0 to ^25.17.1.

2026-03-20 - 11.8.4 - fix(deps)

bump @serve.zone/remoteingress to ^4.14.0

- Updates the @serve.zone/remoteingress dependency from ^4.13.2 to ^4.14.0 in package.json.

2026-03-20 - 11.8.3 - fix(deps)

bump @serve.zone/remoteingress to ^4.13.2

- Updates the @serve.zone/remoteingress dependency from ^4.13.1 to ^4.13.2.

2026-03-19 - 11.8.2 - fix(deps)

bump smartproxy and remoteingress dependencies

- updates @push.rocks/smartproxy from ^25.16.3 to ^25.17.0
- updates @serve.zone/remoteingress from ^4.13.0 to ^4.13.1

2026-03-19 - 11.8.1 - fix(dcrouter)

use constructor routes for remote ingress setup and bump smartproxy dependency

- Switch remote ingress initialization to use constructorRoutes instead of smartProxyConfig routes so derived edge ports are based on the active route set.
- Update @push.rocks/smartproxy from ^25.16.2 to ^25.16.3.

2026-03-19 - 11.8.0 - feat(remoteingress)

add UDP listen port derivation and edge configuration support

- derive UDP ports from remote ingress routes using transport 'udp' or 'all'
- expose effective UDP listen ports in allowed edge payloads and remote ingress interfaces
- update @push.rocks/smartproxy to ^25.16.2

2026-03-19 - 11.7.1 - fix(deps)

bump @push.rocks/smartproxy to ^25.16.0

- updates the smartproxy dependency from ^25.15.0 to ^25.16.0

2026-03-19 - 11.7.0 - feat(readme)

document HTTP/3 QUIC support and configuration options

- Add a dedicated README section explaining default HTTP/3 route augmentation, qualification rules, and opt-out behavior.
- Document the new global `http3` configuration shape and re-exported `IHttp3Config` type.
- Update TypeScript module documentation to include the built-in HTTP/3 augmentation module and exports.

2026-03-19 - 11.6.0 - feat(http3)

add automatic HTTP/3 route augmentation for qualifying HTTPS routes

- introduce configurable HTTP/3 augmentation utilities for eligible SmartProxy routes on port 443
- apply HTTP/3 settings to both constructor-defined and stored programmatic routes, with global and per-route opt-out support
- export the HTTP/3 config type and add test coverage for qualification, augmentation behavior, and defaults

- bump @push.rocks/smartproxy to ^25.15.0 for HTTP/3-related support

2026-03-19 - 11.5.1 - fix(project)

no changes to commit

2026-03-19 - 11.5.0 - feat(opsserver)

add configurable OpsServer port and update related tests and documentation

- introduces an optional `opsServerPort` configuration that overrides the default OpsServer port 3000
- updates OpsServer startup logic to use the configured port
- adjusts integration tests to run against dedicated OpsServer ports to avoid conflicts
- documents the new OpsServer port option in the README and TypeScript docs
- includes dependency updates and a remote ingress port range type refinement

2026-03-19 - 11.4.0 - feat(docs)

document OCI container deployment and enable verbose docker build scripts

- adds a new README section covering Docker/OCI container deployment, environment variables, and image build/push commands
- updates docker build and release npm scripts to pass the --verbose flag for more detailed output

2026-03-18 - 11.3.0 - feat(docker)

add OCI container startup configuration and migrate Docker release pipeline to tsdocker

- adds OCI container mode startup that reads DcRouter options from environment variables and an optional JSON config file
- simplifies the Docker image to a two-stage build with production dependencies only and Alpine runtime compatibility packages

- updates Gitea workflows and npm scripts to use tsdocker for image build and release

2026-03-18 - 11.2.56 - fix(deps)

bump @serve.zone/remoteingress to ^4.9.0

- Updates @serve.zone/remoteingress from ^4.8.18 to ^4.9.0 in package.json

2026-03-17 - 11.2.55 - fix(deps)

bump @serve.zone/catalog to ^2.7.0 and @serve.zone/remoteingress to ^4.8.18

- updates @serve.zone/catalog from ^2.6.2 to ^2.7.0
- updates @serve.zone/remoteingress from ^4.8.16 to ^4.8.18

2026-03-17 - 11.2.54 - fix(deps)

bump @serve.zone/remoteingress to ^4.8.16

- Updates @serve.zone/remoteingress from ^4.8.14 to ^4.8.16 in package.json.

2026-03-17 - 11.2.53 - fix(deps)

bump @push.rocks/smartproxy and @serve.zone/remoteingress patch versions

- update @push.rocks/smartproxy from ^25.11.23 to ^25.11.24
- update @serve.zone/remoteingress from ^4.8.13 to ^4.8.14

2026-03-17 - 11.2.52 - fix(deps)

bump @serve.zone/remoteingress to ^4.8.13

- Updates the @serve.zone/remoteingress dependency from ^4.8.12 to ^4.8.13.

2026-03-17 - 11.2.51 - fix(deps)

bump @serve.zone/remoteingress to ^4.8.12

- Updates @serve.zone/remoteingress from ^4.8.11 to ^4.8.12 in package.json

2026-03-17 - 11.2.50 - fix(deps)

bump @serve.zone/remoteingress to ^4.8.11

- updates @serve.zone/remoteingress from ^4.8.10 to ^4.8.11

2026-03-17 - 11.2.49 - fix(deps)

bump @serve.zone/remoteingress to ^4.8.10

- Updates @serve.zone/remoteingress from ^4.8.9 to ^4.8.10 in package.json

2026-03-17 - 11.2.48 - fix(deps)

bump @serve.zone/remoteingress to ^4.8.9

- Updates @serve.zone/remoteingress from ^4.8.7 to ^4.8.9 in package.json

2026-03-17 - 11.2.47 - fix(deps)

bump @push.rocks/smartproxy to ^25.11.23

- Updates the @push.rocks/smartproxy dependency from ^25.11.22 to ^25.11.23 in package.json

2026-03-17 - 11.2.46 - fix(deps)

bump @push.rocks/smartproxy to ^25.11.22

- Updates the @push.rocks/smartproxy dependency from ^25.11.21 to ^25.11.22 in package.json.

2026-03-17 - 11.2.45 - fix(deps)

bump @push.rocks/smartproxy and @serve.zone/remoteingress dependencies

- update @push.rocks/smartproxy from ^25.11.20 to ^25.11.21
- update @serve.zone/remoteingress from ^4.8.3 to ^4.8.7

2026-03-17 - 11.2.44 - fix(deps)

bump @serve.zone/remoteingress to ^4.8.3

- Updates @serve.zone/remoteingress from ^4.8.2 to ^4.8.3 in package.json

2026-03-17 - 11.2.43 - fix(deps)

bump @serve.zone/remoteingress to ^4.8.2

- Updates the @serve.zone/remoteingress dependency from ^4.8.1 to ^4.8.2.

2026-03-17 - 11.2.42 - fix(deps)

bump @serve.zone/remoteingress to ^4.8.1

- Updates @serve.zone/remoteingress from ^4.8.0 to ^4.8.1 in package.json

2026-03-17 - 11.2.41 - fix(deps)

bump @push.rocks/smartproxy to ^25.11.20

- Updates the @push.rocks/smartproxy dependency from ^25.11.19 to ^25.11.20 in package.json.

2026-03-17 - 11.2.40 - fix(deps)

bump @serve.zone/remoteingress to ^4.8.0

- Updates the @serve.zone/remoteingress dependency from ^4.7.2 to ^4.8.0 in package.json.

2026-03-17 - 11.2.39 - fix(repository)

no changes to commit

2026-03-17 - 11.2.38 - fix(deps)

bump @serve.zone/remoteingress to ^4.7.2

- Updates @serve.zone/remoteingress from ^4.7.0 to ^4.7.2 in package.json

2026-03-16 - 11.2.37 - fix(deps)

bump @serve.zone/remoteingress to ^4.7.0

- updates the @serve.zone/remoteingress dependency from ^4.6.0 to ^4.7.0

2026-03-16 - 11.2.36 - fix(deps)

bump @push.rocks/smartproxy to ^25.11.19

- Updates the @push.rocks/smartproxy dependency from ^25.11.18 to ^25.11.19 in package.json

2026-03-16 - 11.2.35 - fix(deps)

bump @push.rocks/smartproxy, @serve.zone/catalog, and @serve.zone/remoteingress dependencies

- updates @push.rocks/smartproxy from ^25.11.17 to ^25.11.18
- updates @serve.zone/catalog from ^2.6.1 to ^2.6.2
- updates @serve.zone/remoteingress from ^4.5.11 to ^4.6.0

2026-03-16 - 11.2.34 - fix(deps)

bump @push.rocks/smartproxy and @serve.zone/catalog patch versions

- updates @push.rocks/smartproxy from ^25.11.16 to ^25.11.17
- updates @serve.zone/catalog from ^2.6.0 to ^2.6.1

2026-03-16 - 11.2.33 - fix(deps)

bump smartproxy and remoteingress dependencies

- update @push.rocks/smartproxy from ^25.11.14 to ^25.11.16
- update @serve.zone/remoteingress from ^4.5.10 to ^4.5.11

2026-03-16 - 11.2.32 - fix(deps)

bump smartproxy and remoteingress dependencies

- update @push.rocks/smartproxy from ^25.11.11 to ^25.11.14
- update @serve.zone/remoteingress from ^4.5.9 to ^4.5.10

2026-03-16 - 11.2.31 - fix(deps)

bump @push.rocks/smartproxy to ^25.11.11

- Updates the @push.rocks/smartproxy dependency from ^25.11.10 to ^25.11.11 in package.json.

2026-03-16 - 11.2.30 - fix(deps)

bump @push.rocks/smartproxy and @serve.zone/catalog dependencies

- update @push.rocks/smartproxy from ^25.11.9 to ^25.11.10
- update @serve.zone/catalog from ^2.5.0 to ^2.6.0

2026-03-16 - 11.2.29 - fix(deps)

bump @serve.zone/remoteingress to ^4.5.9

- Updates @serve.zone/remoteingress from ^4.5.8 to ^4.5.9 in package.json

2026-03-16 - 11.2.28 - fix(deps)

bump @serve.zone/remoteingress to ^4.5.8

- Updates the @serve.zone/remoteingress dependency from ^4.5.7 to ^4.5.8.

2026-03-16 - 11.2.27 - fix(deps)

bump smartproxy and remoteingress dependencies

- update @push.rocks/smartproxy from ^25.11.8 to ^25.11.9
- update @serve.zone/remoteingress from ^4.5.5 to ^4.5.7

2026-03-16 - 11.2.26 - fix(deps)

bump @serve.zone/remoteingress to ^4.5.5

- Updates the @serve.zone/remoteingress dependency from ^4.5.4 to ^4.5.5.

2026-03-16 - 11.2.25 - fix(deps)

bump @push.rocks/smartproxy to ^25.11.8

- updates the smartproxy dependency from ^25.11.7 to ^25.11.8

2026-03-16 - 11.2.24 - fix(deps)

bump @push.rocks/smartproxy to ^25.11.7

- Updates the @push.rocks/smartproxy dependency from ^25.11.6 to ^25.11.7.

2026-03-16 - 11.2.23 - fix(deps)

bump @push.rocks/smartproxy to ^25.11.6

- Updates the @push.rocks/smartproxy dependency from ^25.11.5 to ^25.11.6.

2026-03-16 - 11.2.22 - fix(deps)

bump @push.rocks/smartproxy to ^25.11.5

- updates the @push.rocks/smartproxy dependency from ^25.11.4 to ^25.11.5

2026-03-15 - 11.2.21 - fix(deps)

bump @push.rocks/smartproxy to ^25.11.4

- updates the @push.rocks/smartproxy dependency from ^25.11.3 to ^25.11.4

2026-03-15 - 11.2.20 - fix(deps)

bump @serve.zone/remoteingress to ^4.5.4

- Updates @serve.zone/remoteingress from ^4.5.3 to ^4.5.4 in package.json

2026-03-15 - 11.2.19 - fix(deps)

bump @serve.zone/remoteingress to ^4.5.3

- Updates the @serve.zone/remoteingress dependency from ^4.5.2 to ^4.5.3.

2026-03-15 - 11.2.18 - fix(deps)

bump @serve.zone/remoteingress to ^4.5.2

- Updates @serve.zone/remoteingress from ^4.5.1 to ^4.5.2 in package.json

2026-03-15 - 11.2.17 - fix(repo)

no changes to commit

2026-03-15 - 11.2.16 - fix(deps)

bump @serve.zone/remoteingress to ^4.5.1

- Updates @serve.zone/remoteingress from ^4.5.0 to ^4.5.1 in package dependencies.

2026-03-15 - 11.2.15 - fix(deps)

bump @serve.zone/remoteingress to ^4.5.0

- Updates the @serve.zone/remoteingress dependency from ^4.4.1 to ^4.5.0.

2026-03-15 - 11.2.14 - fix(deps)

bump smartproxy and remoteingress patch dependencies

- update @push.rocks/smartproxy from ^25.11.1 to ^25.11.3
- update @serve.zone/remoteingress from ^4.4.0 to ^4.4.1

2026-03-15 - 11.2.13 - fix(deps)

bump runtime dependencies to latest compatible patch and minor versions

- update @design.estate/dees-catalog from ^3.48.2 to ^3.48.5
- update @push.rocks/smartproxy from ^25.10.7 to ^25.11.1
- update @tsclass/tsclass from ^9.3.0 to ^9.4.0
- update lru-cache from ^11.2.6 to ^11.2.7

2026-03-12 - 11.2.12 - fix(deps)

bump @push.rocks/smartproxy to ^25.10.7

- Updates the @push.rocks/smartproxy dependency from ^25.10.6 to ^25.10.7 in package.json.

2026-03-12 - 11.2.11 - fix(deps)

bump @push.rocks/smartproxy to ^25.10.6

- Updates the @push.rocks/smartproxy dependency from ^25.10.5 to ^25.10.6 in package.json

2026-03-12 - 11.2.10 - fix(deps)

bump @push.rocks/smartproxy to ^25.10.5

- Updates the @push.rocks/smartproxy dependency from ^25.10.4 to ^25.10.5 in package.json.

2026-03-12 - 11.2.9 - fix(deps)

bump @push.rocks/smartproxy to ^25.10.4

- Updates the @push.rocks/smartproxy dependency from ^25.10.3 to ^25.10.4.

2026-03-12 - 11.2.8 - fix(deps)

bump @design.estate/dees-element and @push.rocks/smartproxy patch versions

- update @design.estate/dees-element from ^2.2.2 to ^2.2.3
- update @push.rocks/smartproxy from ^25.10.2 to ^25.10.3

2026-03-12 - 11.2.7 - fix(deps)

bump @design.estate/dees-catalog and @push.rocks/smartproxy patch versions

- update @design.estate/dees-catalog from ^3.48.1 to ^3.48.2
- update @push.rocks/smartproxy from ^25.10.1 to ^25.10.2

2026-03-12 - 11.2.6 - fix(deps)

bump @design.estate/dees-catalog and @push.rocks/smartproxy patch versions

- update @design.estate/dees-catalog from ^3.48.0 to ^3.48.1
- update @push.rocks/smartproxy from ^25.10.0 to ^25.10.1

2026-03-12 - 11.2.5 - fix(repo)

no changes to commit

2026-03-12 - 11.2.4 - fix(repo)

no changes to commit

2026-03-12 - 11.2.3 - fix(deps)

bump package dependencies to latest compatible patch and minor releases

- update @types/node to ^25.5.0
- upgrade @design.estate/dees-catalog to ^3.48.0 and @design.estate/dees-element to ^2.2.2
- bump @push.rocks/smartproxy to ^25.10.0

2026-03-11 - 11.2.2 - fix(deps)

update dependencies and devDependencies to newer patch/minor versions

- devDependency @git.zone/tstest: ^3.3.0 -> ^3.3.2
- devDependency @git.zone/tswatch: ^3.2.5 -> ^3.3.0
- devDependency @types/node: ^25.3.5 -> ^25.4.0
- dependency @design.estate/dees-catalog: ^3.43.3 -> ^3.47.0
- dependency @design.estate/dees-element: ^2.1.6 -> ^2.2.1
- dependency @push.rocks/smartproxy: ^25.9.2 -> ^25.9.3

2026-03-08 - 11.2.1 - fix(deps)

bump devDependency @git.zone/tstest to ^3.3.0 and dependency @push.rocks/smartproxy to ^25.9.2

- Bumped devDependency @git.zone/tstest: ^3.2.0 -> ^3.3.0
- Bumped dependency @push.rocks/smartproxy: ^25.9.1 -> ^25.9.2
- Current package version: 11.2.0 — recommended patch release to 11.2.1

2026-03-06 - 11.2.0 - feat(apiclient)

add typed, object-oriented API client documentation and interfaces; document builders, resource managers, and new programmatic endpoints

- Add new @serve.zone/dcrouter-apiclient documentation (ts_apiclient/readme.md) and publish ordering (ts_apiclient/tspublish.json).
- Document OO resource classes, fluent builders, auth modes, examples, and API surface for routes, certificates, apiTokens, remoteIngress, stats, config, logs, emails, and radius.
- Update main readme: add API Client section, list new client methods, add package entry for @serve.zone/dcrouter-apiclient, and add apiclient test coverage entry.
- Update interfaces readme: add Route Management and API Token Management request interfaces and email method changes (getAllEmails, getEmailDetail).
- API reference changes: consolidate email endpoints (getAllEmails/getEmailDetail), add route and api token management methods, rename getLogs to getRecentLogs and add getLogStream.
- Update web docs to include route & API token management pages and ops view (ops-view-routes)

2026-03-06 - 11.1.0 - feat(apiclient)

add TypeScript API client (ts_apiclient) with resource managers and package exports

- Add new ts_apiclient module providing DcRouterApiClient and resource managers: routes, certificates, api tokens, remote ingress, emails, stats, config, logs, and radius (with sub-managers).
- Add resource classes and builders (Route, RemoteIngress, ApiToken, Certificate, Email) and convenience manager APIs for common operations.
- Export apiclient in package.json (exports and files) and add ts_apiclient index and plugins wrapper for @api.global/typedrequest.
- Add comprehensive tests for the API client (test/test.apiclient.ts).
- Bump devDependencies: @git.zone/tsbuild -> ^4.3.0 and @types/node -> ^25.3.5

2026-03-05 - 11.0.51 - fix(build)

include HTML files in tsbundle output and bump tsbuild/tsbundle devDependencies

- Add includeFiles: ["./html/**/*.html"] to bundler config in npmextra.json so HTML assets are included in the bundle
- Bump devDependencies: @git.zone/tsbuild ^4.2.4 -> ^4.2.6, @git.zone/tsbundle ^2.9.0 -> ^2.9.1 (non-breaking tooling updates)

2026-03-05 - 11.0.50 - fix(devDependencies)

bump @git.zone/tsbuild to ^4.2.4

- updated devDependency @git.zone/tsbuild from ^4.2.3 to ^4.2.4
- no other package changes

2026-03-05 - 11.0.49 - fix(dcrouter)

no changes detected

- No files changed in this commit
- Working tree unchanged; no version bump required

2026-03-05 - 11.0.48 - fix(deps)

bump @git.zone/tsbuild to ^4.2.3

- package.json: updated devDependency @git.zone/tsbuild from ^4.2.2 to ^4.2.3

2026-03-05 - 11.0.47 - fix(dcrouter)

no code changes; nothing to release

- No files changed in this commit (git diff is empty)
- No version bump required

2026-03-05 - 11.0.46 - fix(none)

no changes detected

- Git diff reported no changes
- No files were modified; no version bump required

2026-03-05 - 11.0.45 - fix(deps)

bump @git.zone/tsbuild to ^4.2.2

- Updated @git.zone/tsbuild from ^4.2.1 to ^4.2.2 in package.json

2026-03-05 - 11.0.44 - fix(dev-deps)

bump @git.zone/tsbuild devDependency to ^4.2.1

- Updated package.json devDependency @git.zone/tsbuild from ^4.2.0 to ^4.2.1
- Non-breaking patch update for build tool dependency

2026-03-05 - 11.0.43 - fix(dcrouter)

no changes detected; nothing to release

- Git diff reported no changes
- No files were modified, so no version bump is recommended

2026-03-05 - 11.0.42 - fix(dcrouter)

empty commit — no changes

- No files were modified in this commit
- No version bump required

2026-03-05 - 11.0.41 - fix(deps)

bump devDependency @git.zone/tsbuild to ^4.2.0

- Updated @git.zone/tsbuild from ^4.1.26 to ^4.2.0
- Change made in package.json under devDependencies
- No source code changes — dev tooling dependency bump

2026-03-05 - 11.0.40 - fix(deps)

bump @git.zone/tsbuild devDependency to ^4.1.26

- Updated devDependency @git.zone/tsbuild: ^4.1.25 → ^4.1.26 in package.json
- Build tooling/dev dependency bump only; no runtime or API changes

2026-03-05 - 11.0.39 - fix(devDependencies)

bump @git.zone/tsbuild devDependency to ^4.1.25

- Updated devDependency @git.zone/tsbuild from ^4.1.24 to ^4.1.25 in package.json
- Only a devDependency was changed; no runtime dependencies or source files modified
- Current package version is 11.0.38; recommend a patch release

2026-03-05 - 11.0.38 - fix(deps)

bump @git.zone/tsbuild to ^4.1.24

- Updated @git.zone/tsbuild in devDependencies from ^4.1.23 to ^4.1.24
- Dev tooling dependency bump; no runtime or API changes expected

2026-03-05 - 11.0.37 - fix(dcrouter)

bump patch version (no changes detected)

- No files changed in the provided diff
- Current package version is 11.0.36 (package.json)
- Recommend a patch bump to record a new release if desired

2026-03-05 - 11.0.36 - fix(repo)

no changes detected; no release necessary

- Diff contains no changes
- No files were modified — skip version bump

2026-03-05 - 11.0.35 - fix(dev-deps)

bump @git.zone/tsbuild devDependency to ^4.1.23

- Updated devDependency @git.zone/tsbuild from ^4.1.22 to ^4.1.23 in package.json

2026-03-05 - 11.0.34 - fix(dcrouter)

empty diff — no changes detected; no version bump suggested

- No file changes in the provided git diff
- Current package.json version is 11.0.33 — keep unchanged

2026-03-05 - 11.0.33 - fix(build)

bump @git.zone/tsbuild to ^4.1.22

- Updated devDependency @git.zone/tsbuild from ^4.1.21 to ^4.1.22
- Change affects build tooling only (devDependencies) — no runtime or API changes expected

2026-03-05 - 11.0.32 - fix(dev-deps)

bump @git.zone/tsbuild devDependency to ^4.1.21

- Updated package.json devDependency @git.zone/tsbuild from ^4.1.20 to ^4.1.21
- Change affects development tooling only (no runtime/source changes)
- Bump package patch version from 11.0.31 to 11.0.32 recommended

2026-03-05 - 11.0.31 - fix(deps)

bump @git.zone/tsbuild devDependency to ^4.1.20

- Updated devDependency @git.zone/tsbuild from ^4.1.19 to ^4.1.20

2026-03-05 - 11.0.30 - fix(devDependencies)

bump @git.zone/tsbuild devDependency to ^4.1.19

- Updated @git.zone/tsbuild from ^4.1.18 to ^4.1.19 in package.json
- Change is limited to devDependencies (build toolchain) and should not affect runtime behavior

2026-03-05 - 11.0.29 - fix(build)

bump @git.zone/tsbuild devDependency to ^4.1.18

- Updated @git.zone/tsbuild from ^4.1.17 to ^4.1.18
- Change is a devDependency update only; no runtime behavior expected to change
- Recommend patch version bump

2026-03-05 - 11.0.28 - fix(devDependencies)

bump @git.zone/tsbuild devDependency to ^4.1.17

- package.json: updated @git.zone/tsbuild from ^4.1.16 to ^4.1.17 (devDependency)

2026-03-05 - 11.0.27 - fix(deps)

bump @git.zone/tsbuild to ^4.1.16

- Updated devDependency @git.zone/tsbuild from ^4.1.15 to ^4.1.16 in package.json
- No runtime code or dependency changes; only a dev/build tool bump

2026-03-05 - 11.0.26 - fix(devDependencies)

bump @git.zone/tsbuild devDependency to ^4.1.15

- Updated devDependency @git.zone/tsbuild from ^4.1.14 to ^4.1.15 in package.json
- No runtime changes; development tooling update only

2026-03-05 - 11.0.25 - fix(logger)

remove build verification comment from logger export

- Removed parenthetical '(build verification)' from export comment in ts/logger.ts
- No functional changes — this is a comment-only cleanup

2026-03-05 - 11.0.24 - fix(dcrouter)

no changes detected — no release necessary

- No files changed in the provided diff; no code, docs, or dependency updates to release.

2026-03-05 - 11.0.23 - fix(deps)

bump @git.zone/tsbuild devDependency to ^4.1.14

- Updated devDependency @git.zone/tsbuild from ^4.1.13 to ^4.1.14 in package.json
- Change affects build tooling only (devDependencies); no production code changes

2026-03-05 - 11.0.22 - fix(deps)

bump @git.zone/tsbuild devDependency to ^4.1.13

- Updated @git.zone/tsbuild from ^4.1.9 to ^4.1.13 in devDependencies
- No runtime code changes; build/dev dependency update only

2026-03-05 - 11.0.21 - fix()

no changes detected

- No files changed in this diff; no release required.

2026-03-05 - 11.0.20 - fix(logger)

annotate singleton logger export comment for build verification

- Changed comment in ts/logger.ts to add '(build verification)'
- No functional code changes; only a comment update
- Intended to mark the export for build verification purposes

2026-03-05 - 11.0.19 - fix(dcrouter)

no changes

- No files changed in this commit.
- Package version remains 11.0.18.

2026-03-05 - 11.0.18 - fix(dcrouter)

no changes detected; no version bump required

- Git diff contains no changes — nothing to release

2026-03-05 - 11.0.17 - fix(dcrouter)

no changes detected in diff; no code or documentation updates

- No files changed in this diff
- No code, tests, or documentation modified; no release required

2026-03-05 - 11.0.16 - fix(dcrouter)

noop commit: no changes detected

- No files changed in this diff.
- No code or configuration modifications detected.

2026-03-05 - 11.0.15 - fix()

no changes detected; no version bump necessary

- Diff contains no changes; no files were modified

2026-03-05 - 11.0.14 - fix(dcrouter)

no changes detected

- Provided git diff contains no changes; nothing to release or bump
- Create a commit only if an empty/placeholder commit is intentionally required

2026-03-05 - 11.0.13 - fix()

no code changes

- No files were changed in this commit.

2026-03-05 - 11.0.12 - fix(dcrouter)

no changes detected — nothing to commit

- Diff reported: No changes
- No files were modified or staged; no functional or documentation changes to release

2026-03-05 - 11.0.11 - fix(deps)

bump @git.zone/tsbuild devDependency to ^4.1.9

- Updated @git.zone/tsbuild from ^4.1.4 to ^4.1.9 in package.json

2026-03-05 - 11.0.10 - fix(playwright-mcp)

remove committed Playwright artifacts and add .playwright-mcp/ to .gitignore

- Added .playwright-mcp/ to .gitignore to avoid committing transient Playwright outputs
- Removed many Playwright-generated logs, screenshots and console dumps under .playwright-mcp/ to reduce repository noise/size
- Prevents accidental check-in of large test artifacts generated by Playwright runs

2026-03-05 - 11.0.9 - fix(devDependencies)

bump @git.zone/tsbuild devDependency to ^4.1.4

- package.json: Updated @git.zone/tsbuild from ^4.1.3 to ^4.1.4

2026-03-05 - 11.0.8 - fix()

no changes detected

- No files changed in this commit
- No version bump recommended

2026-03-05 - 11.0.7 - fix(deps)

bump @git.zone/tsbuild to ^4.1.3 and @push.rocks/lik to ^6.3.1

- Updated devDependency @git.zone/tsbuild from ^4.1.2 to ^4.1.3 in package.json
- Updated dependency @push.rocks/lik from ^6.2.2 to ^6.3.1 in package.json
- Changes are non-breaking dependency bumps; no source code changes

2026-03-04 - 11.0.5 - fix(none)

no changes detected; nothing to release

- Diff contained no changes
- No files modified; no version bump required

2026-03-04 - 11.0.4 - fix()

no changes

- No files changed in the provided diff; no release or version bump required.

2026-03-04 - 11.0.3 - fix()

no changes detected

- Diff shows no file changes; no code changes to release.

2026-03-04 - 11.0.2 - fix(dcrouter)

no changes detected; no files were modified

- diff was empty
- no source or package changes detected

2026-03-04 - 11.0.1 - fix(auth)

treat expired JWTs as no identity, improve logout and token verification flow, and bump deps

- App: getActionContext now treats expired JWTs as null to avoid using stale identities for requests.
- Logout action always clears local login state; server-side adminLogout is attempted only when a valid identity exists.
- Dashboard: verify persisted JWT with server (verifyIdentity) on startup; if verification fails, clear state and show login.
- Auto-refresh: on combined refresh failure, detect auth-related errors (invalid/unauthorized/401), dispatch logout and reload to force re-login.
- Deps: bumped devDependencies @git.zone/tstest (^3.2.0) and @git.zone/tswatch (^3.2.5); added runtime dependency @push.rocks/lik (^6.2.2).
- Tests/artifacts: added Playwright console logs and page screenshots (test artifacts) to the commit.

2026-03-03 - 11.0.0 - BREAKING CHANGE(opsserver)

Require authentication for OpsServer endpoints, split handlers into authenticated view/admin routers, and make identity required on many TypedRequest interfaces

- Added viewRouter and adminRouter to OpsServer and wired middleware to enforce identity/admin checks (requireValidIdentity, requireAdminIdentity).
- Moved handlers to appropriate routers (viewRouter for read endpoints, adminRouter for write/admin endpoints) instead of registering on the unauthenticated main typedrouter.
- Made identity a required field on numerous ts_interfaces request types (breaking change to request typings).
- Refactored ApiTokenHandler to register directly on adminRouter and use dataArg.identity.userId (no per-handler admin checks needed thanks to middleware).
- Updated tests: added admin login to obtain identity, adjusted protected endpoint tests to expect rejection when unauthenticated, and adapted other tests to pass identity where required.
- Added IReq_GetNetworkStats request/response typings to ts_interfaces/requests/stats.ts.
- Bumped dependencies: @api.global/typedrequest ^3.3.0 and @api.global/typedserver ^8.4.2.

2026-03-03 - 10.1.9 - fix(deps)

bump @push.rocks/smartproxy to ^25.9.1

- Updated package.json dependency @push.rocks/smartproxy from ^25.9.0 to ^25.9.1
- No other code changes; current package version is 10.1.8, recommend a patch release

2026-03-03 - 10.1.8 - fix(deps)

bump dependencies: @push.rocks/smartmetrics to ^3.0.2, @push.rocks/smartproxy to ^25.9.0, @serve.zone/remoteingress to ^4.4.0

- @push.rocks/smartmetrics: 3.0.1 -> 3.0.2 (patch)
- @push.rocks/smartproxy: 25.8.5 -> 25.9.0 (minor)
- @serve.zone/remoteingress: 4.3.0 -> 4.4.0 (minor)

2026-03-03 - 10.1.7 - fix(ops-view-apitokens)

use correct lucide icon name for roll/rotate actions in API tokens view

- Updated iconName from 'lucide:rotate-cw' to 'lucide:rotateCw' in ts_web/elements/ops-view-apitokens.ts (two occurrences) to match lucide icon naming and ensure icons render correctly
- Non-functional UI fix; no API or behavior changes

2026-03-02 - 10.1.6 - fix(ts_web)

use actionContext for dispatches in web state actions and bump @push.rocks/smartstate to ^2.2.0

- Action handlers in ts_web/appstate.ts now accept an actionContext parameter and call await actionContext.dispatch(...) instead of using statePartArg.dispatchAction(...).
- Handlers return the awaited dispatch result (ensuring callers receive refreshed state) instead of returning the previous statePartArg.getState().
- Dependency bumped in package.json: @push.rocks/smartstate from ^2.1.1 to ^2.2.0.
- Playwright artifacts (logs and page screenshots) were added under .playwright-mcp.

2026-03-02 - 10.1.5 - fix(monitoring)

use a per-second ring buffer for DNS query metrics, improve DNS logging rate limiting and security event aggregation, and bump smartmta dependency

- Replace unbounded query timestamp array with a fixed-size per-second Int32Array ring buffer (300s) to calculate queries-per-second with O(1) updates and bounded memory
- Add incrementQueryRing and getQueryRingSum helpers to correctly zero stale slots and sum recent seconds
- Change metrics cache interval from 200ms to 1000ms to better match dashboard polling and reduce update frequency
- Refactor DNS adaptive logging to use per-second counters (dnsLogWindowSecond / dnsLogWindowCount) instead of timestamp arrays to avoid per-query array filtering and improve rate limiting accuracy; reset counters on flush
- Security logger: avoid mutating source when sorting/filtering, and implement single-pass aggregation with optional time-window filtering for byLevel/byType/top lists
- Bump dependency @push.rocks/smartmta from ^5.3.0 to ^5.3.1

2026-03-02 - 10.1.4 - fix(no-changes)

no changes detected; no version bump required

- package version is 10.1.3
- git diff contains no changes

2026-03-02 - 10.1.3 - fix(deps)

bump @api.global/typedrequest to ^3.2.7

- Updated @api.global/typedrequest from ^3.2.6 to ^3.2.7 in package.json
- Dependency patch bump only — no source code changes detected
- Current package version 10.1.2 -> recommended next version 10.1.3 (patch)

2026-03-01 - 10.1.2 - fix(core)

improve shutdown cleanup, socket/stream robustness, and memory/cache handling

- Reset security singletons and CacheDb on shutdown to allow GC (SecurityLogger, ContentScanner, IPReputationChecker, CacheDb).
- Add DNS socket 'error' handler and only destroy socket when not already destroyed to avoid uncaught exceptions.
- Move pruning of dnsMetrics.queryTimestamps to a periodic interval to avoid O(n) work on every query.
- Debounce IPReputationChecker cache saves (save timer + reset on instance reset) to reduce IO and prevent duplicate saves.
- Fix virtualStream send timeout handling by keeping/clearing a timeout handle to avoid leaks and hung promises.
- Add memory store eviction in StorageManager to cap entries (MAX_MEMORY_ENTRIES) and evict oldest entries when exceeded.
- Add terminal-ready timeout in ops-view-logs to avoid blocking UI initialization if xterm CDN fails to initialize.
- Bump dev dependency @types/node and push.rocks/smartstate versions.

2026-02-27 - 10.1.1 - fix(ops-view-apitokens)

replace lucide:refresh-cw with lucide:rotate-cw for Roll action icon

- Updated ts_web/elements/ops-view-apitokens.ts: changed iconName in two locations to 'lucide:rotate-cw' for the Roll/Roll Token actions.
- UI-only change — no functional or API behavior modified.
- Current package version is 10.1.0; recommended patch bump to 10.1.1.

2026-02-27 - 10.1.0 - feat(api-tokens)

add ability to roll (regenerate) API token secrets and UI to display the newly generated token once

- Server: added ApiTokenManager.rollToken(id) to regenerate a token secret, update its hash, persist it and log the action.
- Server: added opsserver handler 'rollApiToken' which requires admin identity and returns the new raw token value (shown once) or error messages.
- API: added typed request interface IReq_RollApiToken for the rollApiToken RPC.
- Web: added appstate.rollApiToken wrapper to call the new typed request.
- UI: ops-view-apitokens updated with a 'Roll' action and a modal flow to confirm rolling, call the API, refresh token list, and present the new token value to copy (token value is shown only once).
- Security: operation is admin-only and the raw token is returned only once after rolling.

2026-02-27 - 10.0.0 - BREAKING CHANGE(remote-ingress)

replace tlsConfigured boolean with tlsMode ('custom' | 'acme' | 'self-signed') and compute TLS mode server-side

- Server: compute remoteIngress.tlsMode = 'custom' when custom certPath/keyPath provided; else attempt to detect ACME by checking stored certs for hubDomain; default to

'self-signed' as fallback.
- API: replaced remoteIngress.tlsConfigured:boolean with tlsMode:'custom'|'acme'|'self-signed' — this is a breaking change for consumers of the config API.
- UI: ops view updated to display TLS Mode as a badge instead of a boolean "TLS Configured" field.
- Action required: update clients and integrations to read remoteIngress.tlsMode instead of tlsConfigured.

2026-02-26 - 9.3.0 - feat(remoteingress)

add TLS certificate resolution and passthrough for RemoteIngress tunnel

- Resolve TLS certs for the RemoteIngress tunnel with priority: explicit certPath/keyPath files → stored ACME cert for hubDomain → fallback to self-signed
- Expose tls option on ITunnelManagerConfig and forward certPem/keyPem into hub.start so the hub can use the provided TLS materials
- Add logging for cert selection and file read failures
- Bump dependency @serve.zone/remoteingress from ^4.2.0 to ^4.3.0

2026-02-26 - 9.2.0 - feat(remoteingress)

expose connected edge IPs and detected public IP; resolve proxy IPs from SmartProxy and improve ops UI

- Add detectedPublicIp to DC Router and populate it when a configured or auto-discovered public IP is chosen
- Use dcRouter.detectedPublicIp as a fallback for system.publicIp in the config handler
- Resolve proxy IPs from SmartProxy runtime settings when opts.proxyIps is not provided
- TunnelManager: capture peerAddr on edgeConnected and from Rust heartbeats, store per-edge publicIp, and add getConnectedEdgeIps()
- Expose connectedEdgeIps in the config API and return it in remoteIngress config
- Ops UI: show Connected Edge IPs, annotate 127.0.0.1 proxy IP as 'Remote Ingress' when applicable, and refresh remote ingress data during combined refresh when viewing remoteingress
- Bump dependency @serve.zone/remoteingress to ^4.2.0

2026-02-26 - 9.1.10 - fix(deps)

bump @push.rocks/smartproxy to ^25.8.5

- package.json: @push.rocks/smartproxy version updated from ^25.8.4 to ^25.8.5
- No other files changed

2026-02-26 - 9.1.9 - fix(deps(smartmta))

bump @push.rocks/smartmta to ^5.3.0

- Updated @push.rocks/smartmta from ^5.2.6 to ^5.3.0 in package.json
- Patch release recommended (no source code changes)

2026-02-26 - 9.1.8 - fix(deps)

bump @serve.zone/remoteingress to ^4.1.0

- Updated dependency @serve.zone/remoteingress from ^4.0.1 to ^4.1.0 in package.json
- Non-breaking dependency update; recommend patch version bump

2026-02-26 - 9.1.7 - fix(dcrouter)

bump @push.rocks/smartproxy to ^25.8.4 and remove custom smartProxy timeout/connection lifetime settings from dcrouter

- Bumped dependency @push.rocks/smartproxy from ^25.8.3 to ^25.8.4 in package.json
- Removed explicit smartProxy options: socketTimeout, inactivityTimeout, keepAliveInactivityMultiplier, extendedKeepAliveLifetime, and maxConnectionLifetime from ts/classes.dcrouter.ts

2026-02-26 - 9.1.6 - fix(cleanup)

prevent event listener and log stream leaks, tighten smartProxy connection timeouts, and improve graceful shutdown behavior

- Tightened smartProxy connection timeouts and lifetimes (5m socketTimeout, 10m inactivityTimeout, keep-alive multiplier, 1h extendedKeepAliveLifetime, 4h maxConnectionLifetime).
- Remove event listeners before stopping services to avoid leaks (smartProxy, emailServer, dnsServer, remote ingress hub).
- OpsServer.stop now invokes logsHandler.cleanup to tear down active log streams and avoid duplicate push destinations.
- LogsHandler rewritten to use a module-level singleton push destination, track active stream stop callbacks, add cleanup(), guard against hung VirtualStream.sendData with a 10s timeout, and ensure intervals are cleared on stop.
- updateSmartProxyConfig removes listeners on the old instance before stopping it.
- Dependency bumps: @api.global/typedsocket ^4.1.2, @push.rocks/smartdata ^7.1.0, @push.rocks/smartmta ^5.2.6, @push.rocks/smartproxy ^25.8.3.

2026-02-26 - 9.1.5 - fix(remoteingress)

Reconcile tunnel manager edge statuses with authoritative Rust hub periodically; update active tunnel counts and heartbeats, add missed edges, remove stale entries, and clear reconcile interval on stop

- Add reconcile() to sync TS-side edgeStatuses with hub.getStatus and overwrite activeTunnels with the authoritative activeStreams.
- Start a periodic reconcile (setInterval every 15s) and store the interval handle on the tunnel manager.
- Clear the reconcile interval in stop() to avoid background timers; remove edgeStatuses entries that are no longer connected in Rust.
- Bump dependency @serve.zone/remoteingress from ^4.0.0 to ^4.0.1.

2026-02-25 - 9.1.4 - fix(deps)

bump @push.rocks/smartproxy to ^25.8.1

- Updated package.json dependency @push.rocks/smartproxy from ^25.8.0 to ^25.8.1

2026-02-24 - 9.1.3 - fix(deps)

bump @api.global/typedserver to ^8.4.0 and @push.rocks/smartproxy to ^25.8.0

- Updated @api.global/typedserver from ^8.3.1 to ^8.4.0
- Updated @push.rocks/smartproxy from ^25.7.9 to ^25.8.0

2026-02-24 - 9.1.2 - fix(deps)

bump dependency versions for build and runtime packages

- @git.zone/tsbundle: ^2.8.3 -> ^2.9.0
- @git.zone/tswatch: ^3.1.0 -> ^3.2.0
- @api.global/typedserver: ^8.3.0 -> ^8.3.1
- @design.estate/dees-catalog: ^3.43.2 -> ^3.43.3

2026-02-23 - 9.1.1 - fix(dcrouter)

no changes detected — no files modified, no release necessary

- Git diff contained no changes
- No files added, modified, or deleted

2026-02-23 - 9.1.0 - feat(ops-dashboard)

add lucide icons to Ops dashboard view tabs

- Added iconName property to 10 view tabs in ts_web/elements/ops-dashboard.ts to enable icons in the UI
- Icon mappings: Overview -> lucide:layoutDashboard, Configuration -> lucide:settings, Network -> lucide:network, Emails -> lucide:mail, Logs -> lucide:scrollText, Routes -> lucide:route, ApiTokens -> lucide:key, Security -> lucide:shield, Certificates -> lucide:badgeCheck, RemoteIngress -> lucide:globe
- Improves visual clarity of dashboard navigation

2026-02-23 - 9.0.0 - BREAKING CHANGE(opsserver)

Return structured configuration (IConfigData) from opsserver and update UI to render detailed config sections

- Introduce IConfigData interface with typed sections: system, smartProxy, email, dns, tls, cache, radius, remoteIngress.
- Replace ConfigHandler.getConfiguration implementation to assemble and return IConfigData (changes API response shape for getConfiguration).
- Refactor frontend: update appstate types and ops-view-config to render the new config sections, use @serve.zone/catalog IConfigField/IConfigSectionAction, add uptime formatting and remote ingress UI.
- Fix ops-view-apitokens form handling to correctly read dees-input-tags values.
- Update tests to expect new configuration fields.
- Bump dependency @serve.zone/catalog to ^2.5.0.

2026-02-23 - 8.1.0 - feat(route-management)

add programmatic route management API with API tokens and admin UI

- Introduce RouteConfigManager to persist and manage programmatic routes and hardcoded-route overrides
- Add ApiTokenManager to create, validate, list, toggle and revoke API tokens (stored hashed)
- New OpsServer TypedRequest handlers: RouteManagementHandler (getMergedRoutes, create/update/delete/toggle routes, set/remove overrides) and ApiTokenHandler (create/list/revoke/toggle tokens)
- DcRouter integration: initialize routeConfigManager and apiTokenManager, expose getConstructorRoutes and re-apply programmatic routes after SmartProxy restarts
- Front-end additions: new 'Routes' and 'ApiTokens' views and UI components (ops-view-routes, ops-view-apitokens), router and appstate actions to fetch/manage routes and tokens
- New TS interfaces and request types for route-management and API tokens, plus storage schemas for persisted routes, overrides and tokens
- Bump dependency @serve.zone/catalog to ^2.3.0

2026-02-22 - 8.0.0 - BREAKING CHANGE(email-ops)

migrate email operations to catalog-compatible email model and simplify UI/router

- Add @serve.zone/catalog dependency and import (szCatalog) in web plugins
- Replace queue-based typedrequest methods with catalog APIs: getQueuedEmails / getSentEmails / getFailedEmails => getAllEmails and getEmailDetail (request/response shapes changed)
- Update TypeScript interfaces: IEmailQueueItem/IBounceRecord/ISecurityIncident etc. replaced by IEmail, IEmailDetail, ISmtpLogEntry, IConnectionInfo, IAuthenticationResults (breaking type changes)
- Frontend state and actions consolidated: emailOps state now holds emails array; multiple fetch actions removed and replaced by fetchAllEmailsAction and getEmailDetail usage
- UI components updated: ops-view-emails switched to list/detail view and now requests email detail via new API; router no longer exposes email folder routes and email-folder navigation removed
- Ops server handler refactored to return catalog-style emails and email detail; added status mapping and size formatting helpers

2026-02-21 - 7.4.3 - fix(logging)

add adaptive rate-limited DNS query logging, flush pending DNS logs on shutdown, and enhance email delivery logging

- Introduce adaptive DNS logging: allow up to 2 individual DNS query logs per second, then aggregate further queries and emit a batched summary (dnsLogWindow, dnsBatchCount, dnsBatchTimer) with a 5s flush.
- Flush pending DNS batch on stop() and log final DNS batch count during shutdown.
- Enhance email observability by logging deliveryStart, deliverySuccess, deliveryFailed and bounceProcessed events alongside existing MetricsManager tracking.
- Dependency bump: @design.estate/dees-catalog updated from ^3.43.1 to ^3.43.2.
- Non-breaking change; intended as a patch release.

2026-02-21 - 7.4.2 - fix(monitoring,remoteingress,web)

Prune old metrics buckets periodically, clear metrics caches on shutdown, simplify edge disconnect handling, and optimize network view data updates

- Call pruneOldBuckets() each minute to proactively remove stale time-series buckets in MetricsManager
- Clear metricsCache, emailMinuteBuckets and dnsMinuteBuckets when MetricsManager stops to avoid stale state on shutdown
- On edgeDisconnected remove the edgeStatuses entry instead of mutating an existing record (more explicit cleanup)
- Remove unused traffic-timer variables and move requestsPerSec history updates from render() into updateNetworkData() to avoid unnecessary re-renders
- Optimize traffic data array updates by shifting in-place then reassigning arrays to preserve Lit reactivity and reduce intermediate allocations

2026-02-21 - 7.4.1 - fix(dcrouter)

replace console logging with structured logger, improve metrics logging, add terminal-ready wait in ops UI, bump dees-catalog patch

- Replace console.log/console.error calls in classes.dcrouter.ts with structured logger.log (info/debug/error) including contextual data and stringified errors
- MetricsManager: create a dedicated Smartlog instance (metricsLogger) for SmartMetrics and use shared logger for lifecycle events (start/stop)
- SmartProxy/ACME: convert startup/stop/cert events and error logging to structured logs; include generated route and cert metadata where relevant
- Shutdown/startup flows: unify service start/stop/error messages through logger to provide consistent, structured output
- UI change: ops-view-logs now waits for xterm terminalReady before pushing initial logs to avoid race conditions
- Bump dependency @design.estate/dees-catalog from 3.43.0 to 3.43.1

2026-02-21 - 7.4.0 - feat(opsserver)

add real-time log push to ops dashboard and recent DNS query tracking

- Export baseLogger and add a log destination that pushes log entries to connected ops_dashboard TypedSocket clients (ts/opsserver/handlers/logs.handler.ts, ts/logger.ts).

- Introduce a new TypedRequest (pushLogEntry) interface for server→client log pushes (ts_interfaces/requests/logs.ts) and wire client handling in the web UI (ts_web/appstate.ts, ts_web/plugins.ts).
- Add TypedSocket client connection lifecycle to the web app, stream pushed log entries into app state and update log views incrementally (ts_web/appstate.ts, ts_web/elements/ops-view-logs.ts).
- MetricsManager now records recent DNS queries (timestamp, domain, type, answered, responseTimeMs) and exposes them via stats endpoints for display in the UI (ts/monitoring/classes.metricsmanager.ts, ts/opsserver/handlers/stats.handler.ts, ts_interfaces/data/stats.ts).
- UI overview now displays DNS query entries and uses answered flag to set log level (ts_web/elements/ops-view-overview.ts).
- Add import/export for typedsocket in web plugins to enable real-time push (ts_web/plugins.ts).
- Bump dependency @push.rocks/smartproxy patch version ^25.7.8 → ^25.7.9 (package.json).

2026-02-20 - 7.3.0 - feat(dcrouter)

Wire DNS server 'query' events to MetricsManager for time-series tracking and bump @push.rocks/smartdns to ^7.9.0

- Add dnsServer 'query' event listener that iterates event.questions and calls metricsManager.trackDnsQuery(question.type, question.name, false, event.responseTimeMs).
- Listener is guarded by a metricsManager existence check to avoid runtime errors when metrics are not configured.
- Bump dependency @push.rocks/smartdns from ^7.8.1 to ^7.9.0 in package.json.

2026-02-20 - 7.2.0 - feat(logs)

replace custom logs list with dees-chart-log component and push logs to chart, add log mapping and lifecycle sync, and bump smartlog dependency

- Replaced the legacy in-component log list and styling with a dees-chart-log element to render application logs.
- Added updated() lifecycle handler to push new logs to the chart and new helper methods pushLogsToChart() and getMappedLogEntries() to map log entries to the chart's expected format.
- Removed the streaming toggle, getActiveFilters(), legacy CSS for the log list, and the old per-entry rendering markup.

- Added explicit typing for dropdown @selectedOption handlers (e: any).
- Bumped dependency @push.rocks/smartlog from ^3.2.0 to ^3.2.1 in package.json.

2026-02-19 - 7.1.0 - feat(ops/monitoring)

add in-memory log buffer, metrics time-series and ops UI integration

- bump @push.rocks/smartlog to ^3.2.0
- introduce SmartlogDestinationBuffer (logBuffer) and wire it into the base logger to provide an in-memory log store for the Ops UI
- implement minute-resolution time-series buckets in MetricsManager with increment/prune helpers and new APIs getEmailTimeSeries and getDnsTimeSeries
- sync security counters from SecurityLogger and expose recent security events via StatsHandler
- wire email delivery lifecycle events and bounce processing to MetricsManager for tracking sent/received/failed metrics
- LogsHandler now queries the in-memory log buffer, maps smartlog levels/categories, supports search/level/time-range filtering and pagination
- UI updates: ops-view-overview, ops-view-logs and ops-view-security consume time-series and recent events to render charts, tables and filters

2026-02-19 - 7.0.1 - fix(monitoring)

Use smartMetrics cpuPercentage for cpuUsage.user and update smartmetrics and smartproxy dependencies

- Switch cpuUsage.user from parseFloat(smartMetricsData.cpuUsageText) to smartMetricsData.cpuPercentage to align with smartmetrics v3 API
- Bump @push.rocks/smartmetrics from ^2.0.10 to ^3.0.1
- Bump @push.rocks/smartproxy from ^25.7.6 to ^25.7.8

2026-02-19 - 7.0.0 - BREAKING CHANGE(deps)

bump dependencies: @serve.zone/remoteingress to ^4.0.0 (breaking), @push.rocks/smartproxy to ^25.7.6, @types/node to ^25.3.0

- Updated @serve.zone/remoteingress from ^3.3.0 to ^4.0.0 — major breaking change; may require code changes to adapt to new API.
- Updated @push.rocks/smartproxy from ^25.7.3 to ^25.7.6 — patch update (non-breaking).
- Updated @types/node from ^25.2.3 to ^25.3.0 — patch update (non-breaking).
- Current package version is 6.13.2; recommend bumping to 7.0.0 due to the breaking dependency upgrade.

2026-02-19 - 6.13.2 - fix(runtime)

prevent memory leaks and improve shutdown/stream handling across services

- Add CertProvisionScheduler.clear() to reset in-memory backoff cache and call it during DcRouter shutdown
- Stop any existing SmartAcme instance before creating a new one (await stop and log errors) to avoid duplicate running instances
- Null out many DcRouter service references and clear certificateStatusMap on shutdown to allow GC of stopped services
- Cap emailMetrics.recipients map size and trim to ~80% of MAX_TOP_DOMAINS to prevent unbounded growth
- Await virtualStream.sendData in logs follow handler and clear the interval if the stream errors/closes to avoid interval leaks
- Limit normalizedMacCache size and evict oldest entries when it exceeds 10000 to prevent unbounded cache growth

2026-02-18 - 6.13.1 - fix(dcrouter)

enable PROXY protocol v1 handling for SmartProxy when remoteIngress is enabled to preserve client IPs

- Set smartProxyConfig.acceptProxyProtocol = true when options.remoteIngressConfig.enabled
- Whitelist loopback address by setting smartProxyConfig.proxyIPs = ['127.0.0.1']
- Only applies when remoteIngress is enabled; used to accept tunneled connections forwarded by the hub to preserve original client IPs

2026-02-18 - 6.13.0 - feat(remoteingress)

include listenPorts for allowed edges sent to the Rust hub and always resync allowed edges when edge properties change

- getAllowedEdges now returns listenPorts for each allowed edge (uses getEffectiveListenPorts)
- remoteingress handler now calls tunnelManager.syncAllowedEdges() whenever tunnelManager exists so ports/tags/enabled changes are propagated
- Improves Rust hub routing by providing per-edge listening ports and ensuring allowed-edge list is kept up-to-date

2026-02-18 - 6.12.0 - feat(remote-ingress)

add Remote Ingress hub integration, OpsServer UI, APIs, and docs

- Integrates RemoteIngress (hub/tunnel) into DcRouter: runtime manager, tunnel manager and Rust data plane references added
- Bumps dependency @serve.zone/remoteingress to ^3.3.0
- Adds configuration defaults and IDcRouterOptions.remoteIngressConfig with tunnelPort/hubDomain/tls fields
- Introduces OpsServer API endpoints and TypedRequest methods for remote ingress: getRemoteIngresses, createRemoteIngress, updateRemoteIngress, deleteRemoteIngress, regenerateRemoteIngressSecret, getRemoteIngressStatus, getRemoteIngressConnectionToken
- UI updates: new Remote Ingress dashboard view, connection token generation & copy (clipboard API + fallback), auto-derived ports display, and toast notifications
- State/API rename: newEdgeSecret -> newEdgeId and clearNewEdgeIdAction; appstate fetchConnectionToken usage

- Documentation: README, ts/ and ts_web readmes, and ts_interfaces updated with interfaces and examples for Remote Ingress
- Minor UI icon updates (search -> fa:magnifyingGlass, clipboard icon casing) and other doc/README improvements

2026-02-18 - 6.11.0 - feat(remoteingress)

add ability to generate remote ingress connection tokens and UI copy action; add hubDomain config option; update remoteingress dependency to ^3.1.1

- Add server typed handler 'getRemoteIngressConnectionToken' to generate an encoded connection token containing hubHost, hubPort, edgeId and secret.
- Add request interface IReq_GetRemoteIngressConnectionToken for typed requests.
- Add fetchConnectionToken helper in web appstate and a 'Copy Token' action in ops-view-remoteingress to copy tokens to the clipboard with toast feedback.
- Add hubDomain option to remoteIngressConfig in dcrouter options so an external hostname can be embedded in connection tokens.
- Bump dependency @serve.zone/remoteingress from ^3.0.4 to ^3.1.1 in package.json.

2026-02-17 - 6.10.0 - feat(ops-view-certificates)

Make Export and Delete actions available inline (inRow) as well as in the context menu; bump @design.estate/dees-catalog to ^3.43.0

- Added 'inRow' to action types for 'Export' and 'Delete' in ts_web/elements/ops-view-certificates.ts to expose actions inline in the row
- Updated dependency @design.estate/dees-catalog from ^3.42.2 to ^3.43.0 in package.json

2026-02-17 - 6.9.0 - feat(certificates)

add certificate import, export, and deletion support (server handlers, request types, and UI)

- Add typed request handlers in opsserver: deleteCertificate, exportCertificate, importCertificate (ts/opsserver/handlers/certificate.handler.ts)
- Implement deleteCertificate/exportCertificate/importCertificate functions handling storage paths, in-memory status map updates, backoff clearing, validation, and SmartAcme-compatible /certs/ and /proxy-certs/ formats
- Add request interfaces IReq_DeleteCertificate, IReq_ExportCertificate, IReq_ImportCertificate (ts_interfaces/requests/certificate.ts)
- Add web app actions deleteCertificateAction, importCertificateAction and fetchCertificateExport to call new typed requests (ts_web/appstate.ts)
- Update certificates UI to support Import, Export, and Delete actions and add downloadJsonFile helper (ts_web/elements/ops-view-certificates.ts)

2026-02-17 - 6.8.0 - feat(remote-ingress)

support auto-deriving ports for remote ingress edges and expose manual/derived port breakdown in API and UI

- Add autoDerivePorts flag to IRemoteIngress with default true and migration to set existing stored edges to autoDerivePorts = true
- RemoteIngressManager: getEffectiveListenPorts now returns the union of manual + derived ports when autoDerivePorts is enabled; added getPortBreakdown to return manual vs derived lists
- API handlers updated: create/update requests accept autoDerivePorts; responses now include effectiveListenPorts, manualPorts, and derivedPorts (secrets still masked)
- Web UI updated: create and edit dialogs include an Auto-derive checkbox; port badges now visually distinguish manual vs derived ports; added updateRemoteIngressAction
- Non-breaking change: new field defaults to true so existing behavior is preserved

2026-02-17 - 6.7.0 - feat(remote-ingress)

Support auto-derived effective listen ports, make listenPorts optional, add toggle action and refine remote ingress creation/management UI

- Add effectiveListenPorts?: number[] to IRemoteIngress interface (present in API responses)
- Make createRemoteIngressAction.listenPorts optional and update creation modal to allow empty ports (auto-derived)
- Add toggleRemoteIngressAction to enable/disable remote ingress edges and wire up Enable/Disable row/context-menu actions
- Update getPortsHtml to prefer manual listenPorts, fall back to effectiveListenPorts, show '(auto)' when derived and 'none' when no ports
- Standardize UI actions to use inRow/contextmenu and actionFunc signatures; update create modal to use explicit Cancel/Create menu options and collect form data programmatically

2026-02-17 - 6.6.1 - fix(icons)

standardize icon identifiers to lucide-prefixed names across operational views

- Replaced legacy/ambiguous icon names with 'lucide:...' identifiers in four UI modules: ts_web/elements/ops-view-certificates.ts, ops-view-network.ts, ops-view-overview.ts, and ops-view-security.ts.
- Updated common action/menu icons (e.g. arrowsRotate -> lucide:RefreshCw, magnifyingGlass -> lucide:Search, copy -> lucide:Copy, fileExport -> lucide:FileOutput).
- Mapped dashboard/tile icons to lucide equivalents (e.g. server -> lucide:Server, networkWired/sitemap -> lucide:Network, download/upload -> lucide:Download/Upload, microchip/memory -> lucide:Cpu/MemoryStick).
- Normalized alert and status icons to lucide names (e.g. triangleExclamation -> lucide:TriangleAlert, shield/userShield -> lucide:Shield/ShieldCheck, clock/clockRotateLeft -> lucide:Clock/History).

2026-02-17 - 6.6.0 - feat(remoteingress)

derive effective remote ingress listen ports from route configs and expose them via ops API

- Derive listen ports from SmartProxy route configs with remoteIngress.enabled; supports optional edgeFilter to target edges by id or tags.
- Add RemoteIngressManager.setRoutes(), derivePortsForEdge(), and getEffectiveListenPorts() which falls back to manual listenPorts when present.
- dcrouter now supplies route configs to RemoteIngressManager during initialization and when updating SmartProxy configuration to keep derived ports in sync.

- Ops API now returns effectiveListenPorts for edges; createRemoteIngress.listenPorts is optional and createEdge defaults listenPorts to an empty array.
- Bump dependency @serve.zone/remoteingress to ^3.0.4 to align types/behavior.

2026-02-16 - 6.5.0 - feat(ops-view-remoteingress)

add 'Create Edge Node' header action to remote ingress table and remove duplicate createNewAction

- Add a 'Create Edge Node' header action in dataActions that opens DeesModal to collect name, listenPorts and tags
- Parse comma-separated listenPorts into integer array and normalize optional tags
- Dispatch appstate.createRemoteIngressAction with the collected payload
- Remove the previously duplicated createNewAction prop from the dees-table

2026-02-16 - 6.4.5 - fix(remoteingress)

mark remote ingress data actions as row actions and bump @design.estate/dees-catalog dependency

- Add type:['row'] to 'Regenerate Secret' and 'Delete' dataActions in ts_web/elements/ops-view-remoteingress.ts to ensure they are treated as row actions in the UI
- Bump @design.estate/dees-catalog from ^3.42.0 to ^3.42.2 in package.json

2026-02-16 - 6.4.4 - fix(deps)

bump @push.rocks/smartproxy to ^25.7.3

- Updated @push.rocks/smartproxy from ^25.7.2 to ^25.7.3 in package.json

2026-02-16 - 6.4.3 - fix(deps)

bump @push.rocks/smartproxy to ^25.7.2

- Updated package.json: @push.rocks/smartproxy ^25.7.1 -> ^25.7.2 (patch dependency update)

2026-02-16 - 6.4.2 - fix(smartproxy)

bump @push.rocks/smartproxy to ^25.7.1

- Updated dependency @push.rocks/smartproxy from ^25.7.0 to ^25.7.1 in package.json
- No other source changes; dependency patch bump only

2026-02-16 - 6.4.1 - fix(deps)

bump dependencies: @push.rocks/smartproxy to ^25.7.0 and @serve.zone/remoteingress to ^3.0.2

- Bumped @push.rocks/smartproxy from ^25.5.0 to ^25.7.0
- Bumped @serve.zone/remoteingress from ^3.0.1 to ^3.0.2
- Package current version is 6.4.0 — recommended patch release

2026-02-16 - 6.4.0 - feat(remoteingress)

add Remote Ingress hub and management for edge tunnel nodes, including backend managers, tunnel hub integration, opsserver handlers, typedrequest APIs, and web UI

- Introduce RemoteIngressManager for CRUD and persistent storage of edge registrations
- Introduce TunnelManager to run the RemoteIngressHub, track connected edge statuses, and sync allowed edges to the hub
- Integrate remote ingress into DcRouter (options.remoteIngressConfig, setupRemoteIngress, startup/shutdown handling, and startup summary)
- Add OpsServer RemoteIngressHandler exposing typedrequest APIs (create/update/delete/regenerate/get/status)

- Add web UI: Remote Ingress view, app state parts, actions and components to manage edges and display runtime statuses
- Add typedrequest and data interfaces for remoteingress and export the remoteingress module; add @serve.zone/remoteingress dependency in package.json

2026-02-16 - 6.3.0 - feat(dcrouter)

add configurable baseDir and centralized path resolution; use resolved data paths for storage, cache and DNS

- Introduce IDcRouterOptions.baseDir to allow configuring base directory for dcrouter data (defaults to ~/.serve.zone/dcrouter).
- Add DcRouter.resolvedPaths and resolvePaths(baseDir) in ts/paths.ts to centralize computation of dcrouterHomeDir, dataDir, defaultTsmDbPath, defaultStoragePath and dnsRecordsDir.
- Use resolvedPaths throughout DcRouter: default filesystem storage fsPath, CacheDb storagePath, and DNS records loading now reference resolved paths.
- Replace ensureDirectories() behavior with ensureDataDirectories(resolvedPaths) to only create data-related directories; keep legacy ensureDirectories wrapper delegating to the new function.
- Simplify paths module by removing unused legacy path constants and adding a focused API for path resolution and directory creation.
- Remove an unused import (paths) in contentscanner, cleaning up imports.

2026-02-16 - 6.2.4 - fix(deps)

bump @push.rocks/smartproxy to ^25.5.0

- Updated @push.rocks/smartproxy from ^25.4.0 to ^25.5.0 in package.json

2026-02-16 - 6.2.3 - fix(dcrouter)

persist proxy certificate validity dates and improve certificate status initialization

- Bump @push.rocks/smartacme dependency from ^9.0.0 to ^9.1.3
- Store validFrom and validUntil alongside proxy cert entries (/proxy-certs) when saving, extracting values by parsing PEM where possible
- Use stored cert entries (domain, publicKey, validUntil, validFrom) to populate certificateStatusMap at startup

- Fallback to SmartAcme /certs/ metadata and finally to parsing X.509 from stored PEM to determine expiry/issuedAt when initializing status
- Update opsserver certificate handler to parse publicKey PEM from cert-store and set expiry/issuedAt and issuer accordingly
- Adjust variable names and logging to reflect stored cert entry usage

2026-02-16 - 6.2.2 - fix(certs)

Populate certificate status for cert-store-loaded certificates after SmartProxy startup and check proxy-certs in opsserver certificate handler

- Track domains loaded from storageManager '/proxy-certs/' and populate certificateStatusMap with status, routeNames, expiryDate and issuedAt (when available) after SmartProxy starts
- Opsserver certificate handler now falls back to '/proxy-certs/{domain}' if '/certs/{cleanDomain}' is missing and marks cert-store-only entries as valid with issuer 'cert-store'
- Bump @push.rocks/smartproxy dependency from ^25.3.1 to ^25.4.0

2026-02-16 - 6.2.1 - fix(smartacme,storage)

Respect wildcard domain requests when retrieving certificates and treat empty/whitespace storage values as null in getJSON

- Pass includeWildcard flag to smartAcme.getCertificateForDomain to avoid incorrectly including/excluding wildcard certificates based on whether the requested domain itself is a wildcard
- Detect wildcard domains via domain.startsWith('*.') and set includeWildcard to false for wildcard requests
- Treat empty or whitespace-only stored values as null in StorageManager.getJSON to avoid parsing empty strings as JSON and potential errors

2026-02-16 - 6.2.0 - feat(ts_web)

add Certificate Management documentation and ops-view-certificates reference

- Adds a new 'Certificate Management' section to ts_web/readme.md describing domain-centric overview, certificate sources (ACME/provision/static), expiry monitoring, per-domain backoff, and one-click reprovisioning
- Adds ops-view-certificates.ts entry to the ops UI file list
- Documents new route mapping '/certificates' in the readme navigation

2026-02-16 - 6.1.0 - feat(certs)

integrate smartacme v9 for ACME certificate provisioning and add certificate management features, docs, dashboard views, API endpoints, and per-domain backoff scheduler

- Bump dependency: @push.rocks/smartacme -> ^9.0.0
- Add Certificate Management documentation, examples, and a new Certificates view in the OpsServer dashboard (status, source, expiry, backoff, one-click reprovision)
- Integrate smartacme v9 features: per-domain deduplication, global concurrency control, account rate limiting, structured errors, and clean shutdown behavior
- Introduce per-domain exponential backoff persisted via StorageManager (CertProvisionScheduler) and remove the older serial stagger queue (smartacme v9 handles concurrency/deduping)
- Expose new typedrequest API methods: getCertificateOverview, reprovisionCertificate (legacy), reprovisionCertificateDomain (preferred)
- DcRouter now surfaces smartAcme, certProvisionScheduler, and certificateStatusMap; cert provisioning paths call smartAcme directly and clear backoff on success
- Docs updated to note parallel shutdown/cleanup of HTTP agents and DNS clients

2026-02-15 - 6.0.0 - BREAKING CHANGE(certs)

Introduce domain-centric certificate provisioning with per-domain exponential backoff and a staggered serial scheduler; add domain-based reprovision API and UI backoff display; change certificate overview API to be domain-first and include backoff info; bump related deps.

- Add CertProvisionScheduler: persistent per-domain exponential backoff, retry calculation, and an in-memory serial stagger queue.
- Integrate scheduler with SmartAcme certProvisionFunction: enqueue provisions, clear backoff on success, record failures to drive backoff.
- Switch certificate event tracking to be keyed by domain (certificateStatusMap now keyed by domain) and add findRouteNamesForDomain helper.

- BREAKING: ICertificateInfo shape changed — replaced routeName/domains with domain and routeNames; added optional backoffInfo (failures, retryAfter, lastError).
- Add domain-based reprovision endpoint (reprovisionCertificateDomain) while retaining legacy route-based reprovision for backward compatibility (internal rename to reprovisionCertificateByRoute).
- Web UI updated to domain-centric certificate overview, displays route pills, backoff indicator and retry timing, and uses domain-based reprovision action.
- Dependency bumps: @push.rocks/smartlog -> ^3.1.11, @push.rocks/smartproxy -> ^25.3.1.

2026-02-14 - 5.5.0 - feat(certs)

persist ACME certificates in StorageManager, add storage-backed cert manager, default storage to filesystem, and improve certificate status reporting

- Add StorageBackedCertManager to persist SmartAcme certificates under /certs/ via StorageManager
- Default storage to filesystem path (dcrouterHomeDir/storage) when options.storage is not provided
- Wire SmartAcme to use StorageBackedCertManager and provide SmartProxy certStore handlers that load/save/remove certs under /proxy-certs/
- Ops server certificate handler reads persisted cert data to report expiry/issued dates and treats acme/provision-function routes with no cert data as provisioning
- Bump @push.rocks/smartproxy dependency to ^25.3.0

2026-02-14 - 5.4.6 - fix(deps)

bump @push.rocks/smartproxy dependency to ^25.2.2

- Updated dependency @push.rocks/smartproxy: ^25.2.0 → ^25.2.2
- Change is a dependency-only patch update, no source code modifications
- Current package version is 5.4.5; recommend a patch release

2026-02-14 - 5.4.5 - fix(dcrouter)

bump patch for release pipeline consistency - no code changes

- current version: 5.4.4 (from package.json)
- git diff: no changes detected

- recommend patch bump to trigger release artifacts if required

2026-02-14 - 5.4.4 - fix(deps)

bump @push.rocks/smartproxy to ^25.2.0

- Updated @push.rocks/smartproxy from ^25.1.0 to ^25.2.0 (patch, non-breaking).
- Current package version is 5.4.3; recommend a patch release to 5.4.4.

2026-02-14 - 5.4.3 - fix(dependencies)

bump @push.rocks/smartproxy to ^25.1.0

- Updated @push.rocks/smartproxy from ^25.0.0 to ^25.1.0 in package.json

2026-02-13 - 5.4.2 - fix(dcrouter)

improve domain pattern matching to support routing-glob and wildcard patterns and use matching logic when resolving routes

- Support routing-glob patterns beginning with '*' (e.g. *example.com) to match base domain, wildcard form, and subdomains
- Treat standard wildcard patterns ('*.example.com') as matching both the base domain (example.com) and its subdomains
- Use isDomainMatch when resolving routes instead of exact array includes to allow pattern matching
- Normalize domain and pattern to lowercase and simplify equality checks

2026-02-13 - 5.4.1 - fix(network,dcrouter)

Always register SmartProxy certificate event handlers and include total bytes + improved connection metrics in network stats/UI

- Always register SmartProxy 'certificate-issued', 'certificate-renewed', and 'certificate-failed' handlers (previously only registered when acmeConfig was present) so certificate events are processed regardless of provisioning path.
- Add totalBytes (in/out) to network stats and propagate it through ts_interfaces and app state so total data transferred is available to the UI.
- Combine metricsManager.getNetworkStats with collectServerStats to compute activeConnections and adjust connectionDetails/TopEndpoints handling.
- Update ops UI to display totalBytes in throughput cards and remove a redundant network-specific auto-refresh fetch.
- Type and state updates: ts_interfaces/data/stats.ts and ts_web/appstate.ts updated with totalBytes and initialization/default mapping adjusted.

2026-02-13 - 5.4.0 - feat(certificates)

include certificate source/issuer and Rust-side status checks; pass eventComms into certProvisionFunction and record expiry information

- bump @push.rocks/smartproxy dependency to ^25.0.0
- add optional 'source' field to certificate status and propagate event.source when certificates are issued, renewed, or failed
- change smartProxy.certProvisionFunction signature to accept eventComms; use it to log attempts, set source and expiryDate, and fall back to http-01 on DNS-01 failure
- make buildCertificateOverview async and query smartProxy.getCertificateStatus for a route when event-based status is unknown
- improve logging to include certificate source and more contextual messages

2026-02-13 - 5.3.0 - feat(certificates)

add certificate overview and reprovisioning in ops UI and API; track SmartProxy certificate events

- Add CertificateHandler with typedrequest endpoints: getCertificateOverview and reprovisionCertificate
- Introduce ICertificateInfo and request/response interfaces for certificate operations
- Frontend: add certificate state part, actions (fetchCertificateOverview, reprovisionCertificate), router view, and ops-view-certificates component

- DcRouter: add certificateStatusMap, listen to SmartProxy certificate-issued/renewed/failed events, and add findRouteNameForDomain helper
- Bump dependency @push.rocks/smartproxy to ^24.0.0

2026-02-13 - 5.2.0 - feat(monitoring)

add throughput metrics and expose them in ops UI

- MetricsManager now reports bytesInPerSecond and bytesOutPerSecond as part of throughput
- Extended IServerStats with requestsPerSecond and throughput {bytesIn, bytesOut, bytesInPerSecond, bytesOutPerSecond}
- Stats handler updated to include requestsPerSecond and throughput; fallback stats initialize throughput fields to zero
- Web UI ops overview displays Throughput In/Out (bits/s) and total bytes with new formatting helper
- Bumped dependency @push.rocks/smartproxy to ^23.1.6

2026-02-13 - 5.1.0 - feat(acme)

Integrate SmartAcme DNS-01 handling and add certificate provisioning for SmartProxy

- Add smartAcme property and lifecycle management (start/stop) in DcRouter
- Create SmartAcme instance when DNS challenge handlers are present and wire certProvisionFunction to SmartProxy to return certificates for domains
- Fall back to http-01 provisioning on SmartAcme errors for a domain
- Stop SmartAcme during shutdown sequence to clean up resources
- Bump dependency @push.rocks/smartproxy to ^23.1.5

2026-02-13 - 5.0.7 - fix(deps)

bump @push.rocks/smartdns to ^7.8.1 and @push.rocks/smartmta to ^5.2.2

- package.json: updated @push.rocks/smartdns from ^7.8.0 to ^7.8.1 (patch)
- package.json: updated @push.rocks/smartmta from ^5.2.1 to ^5.2.2 (patch)

2026-02-12 - 5.0.6 - fix(deps)

bump @push.rocks/smartproxy to ^23.1.4

- package.json: @push.rocks/smartproxy ^23.1.2 → ^23.1.4
- Dependency-only version bump, no source code changes

2026-02-12 - 5.0.5 - fix(dcrouter)

remove legacy handling of emailConfig.routes that added domain-based routes

- Removed loop that added domain-based email routes from emailConfig.routes into emailRoutes
- Previously created match.domains by extracting the recipient domain (split on '@') and defaulted forward target port to 25
- Removed creation of TLS passthrough configuration for those forwarded routes
- This prevents duplicate or incorrect domain-based routes being appended during email route construction

2026-02-12 - 5.0.4 - fix(cache)

use user-writable ~/.serve.zone/dcrouter for TsmDB and centralize data path logic

- Default TsmDB storage changed from /etc/dcrouter/tsmdb to ~/.serve.zone/dcrouter/tsmdb
- Introduced dcrouterHomeDir, dataDir, and defaultTsmDbPath in ts/paths.ts
- CacheDb now defaults to defaultTsmDbPath when no storagePath is provided
- DcRouter initialization updated to use paths.defaultTsmDbPath; README and readme.hints updated to document the new defaults
- Avoids /etc permission issues and prevents starting a real MongoDB process in tests by using a user-writable default path

2026-02-12 - 5.0.3 - fix(packaging)

add files whitelist to package.json and remove Playwright-generated screenshots

- Add a "files" array to package.json to control published package contents (includes ts/, ts_web/, dist/, dist_*/**, dist_ts/, dist_ts_web/, assets/, cli.js, npmextra.json, readme.md).
- Remove multiple .playwright-mcp/*.png screenshot files (clean up Playwright test artifacts and reduce repository noise/size).

2026-02-12 - 5.0.2 - fix(docs)

update documentation and packaging configuration: document smartmta/smartdns integrations, adjust API method names, and add release registry info

- README: document SmartDNS as Rust-powered DNS engine and smartmta as TypeScript+Rust MTA; add Rust-powered architecture section and component package table
- README: update Node.js requirement from 18+ to 20+; replace embedded cache DB TsmDb with LocalTsmDb and reduce listed cached document types
- README & ts_interfaces: rename typedrequest API adminLogin -> adminLoginWithUsernameAndPassword and add/clarify several API methods (logout, suppression management, RADIUS client/VLAN helpers)
- README: update test instructions, change test file references and add a test coverage table
- npmextra.json: re-key package configs (@git.zone/cli, @ship.zone/szci), tidy watch array formatting, and add release.registries and accessLevel for publishing

2026-02-11 - 5.0.1 - fix(deps/tests)

bump two dependencies and disable cache in tests

- Bumped @api.global/typedrequest from ^3.2.5 to ^3.2.6
- Bumped @push.rocks/smartradius from ^1.1.0 to ^1.1.1
- Disabled cache in tests by adding cacheConfig: { enabled: false } to DcRouter instantiation in test/test.jwt-auth.ts, test/test.opsserver-api.ts, and test/test.protected-endpoint.ts

2026-02-11 - 5.0.0 - BREAKING CHANGE(mta)

migrate internal MTA to @push.rocks/smartmta and remove legacy mail/deliverability implementation

- Replace ~27k LOC custom MTA (ts/mail/, ts/deliverability/) with @push.rocks/smartmta v5.2.1 (TypeScript+Rust hybrid)
- Remove many SMTP client/server test suites and test helpers; testing approach and fixtures changed/removed
- Upgrade dependencies: @push.rocks/smartproxy -> 23.1.2, @push.rocks/smartdns -> 7.8.0, add @push.rocks/smartmta@5.2.1; bump other minor deps
- API differences: updateEmailRoutes() replaces updateRoutes(); UnifiedEmailServer exposes dkimCreator publicly; bounce/suppression APIs moved to emailServer.* helpers; Email class and IAttachment types moved into @push.rocks/smartmta exports
- SmartProxy route validation stricter: forward actions must use targets (array) instead of target (singular) — tests/configs updated accordingly
- DKIM generation/serving moved to smartmta (dcrouter no longer manages DKIM keys directly)

2026-02-10 - 4.1.1 - fix(smartproxy)

upgrade @push.rocks/smartproxy to ^23.1.0 and adapt code/tests for its async getStatistics() API

- Bumped dependency @push.rocks/smartproxy 22.4.2 → 23.1.0 in package.json
- Changed ts/monitoring/classes.metricsmanager.ts to await smartProxy.getStatistics() (was synchronous)
- Updated multiple tests to set cacheConfig: { enabled: false } and added socketTimeouts where appropriate
- Improved SMTP test servers: handle multi-line input, drop data for packet-loss simulation, and ignore socket errors to make tests more robust
- Added migration notes to readme.hints.md documenting SmartProxy v23.1.0 changes (async getStatistics, Rust proxy behavior)

2026-02-10 - 4.1.0 - feat(cache)

add persistent smartdata-backed cache with LocalTsmDb, cache cleaner, and DcRouter integration

- Introduce CacheDb and CacheCleaner using @push.rocks/smartdata and @push.rocks/smartmongo (LocalTsmDb) for persistent caching

- Integrate cache initialization, console summary, and graceful shutdown into DcRouter (options.cacheConfig and setupCacheDb())
- Require svDb() decorators on concrete cache document classes; remove decorators from the abstract CachedDocument base class
- Switch CacheCleaner to smartdata getInstances() + per-document delete() instead of deleteMany
- Adapt to LocalTsmDb API changes (folderPath option and start() returning connectionUri) and initialize SmartdataDb with mongoDbUrl/mongoDbName
- Remove experimentalDecorators and emitDecoratorMetadata from tsconfig to use TC39 Stage 3 decorators (smartdata v7+ compatibility)
- Add package.json exports mapping (remove main/typings entries) to expose dist entry points
- Add README documentation for the Smartdata Cache System and configuration/usage examples

2026-02-03 - 4.0.0 - BREAKING CHANGE(config)

convert configuration management to read-only; remove updateConfiguration endpoint and client-side editing

- Removed server-side 'updateConfiguration' TypedHandler and the private updateConfiguration() method; getConfiguration remains as a read-only handler.
- Removed IReq_UpdateConfiguration interface from request typings; IReq_GetConfiguration marked as read-only.
- Removed client-side editing functionality: ops-view-config editing state and methods, Edit/Save/Cancel buttons, and updateConfigurationAction; ops-view-config enhanced to display read-only configuration (badges for booleans, array pills, icons, formatted numbers/bytes, empty states, etc.).
- Tests updated: replaced configuration update tests with verifyIdentity tests and added a read-only configuration access test.
- Documentation updated to reflect configuration is read-only (readme.md, ts_web/readme.md, ts_interfaces/readme.md, readme.hints.md).
- Dependencies adjusted: bumped @push.rocks/smartdata to ^7.0.15 and added @push.rocks/smartmongo ^5.1.0; ts/plugins updated to import/export smartmongo.

2026-02-02 - 3.1.0 - feat(web)

determine initial UI view from URL and wire selected view to appdash; add interface and web README files; bump various dependencies

- UI: derive initial active view from window.location.pathname so the dashboard supports deep linking and bookmarks (ts_web/appstate.ts)
- UI: pass selectedView to dees-simple-appdash by adding a currentViewTab getter in ops-dashboard (ts_web/elements/ops-dashboard.ts)
- Docs: add TypeScript interfaces README for @serve.zone/dcrouter-interfaces (ts_interfaces/readme.md)
- Docs: add/update web module README detailing features, routing, and build instructions (ts_web/readme.md) and expand main project README
- Deps: bump multiple dependencies in package.json (notable bumps: @api.global/typedrequest -> ^3.2.5, @design.estate/dees-catalog -> ^3.42.0, @design.estate/dees-element -> ^2.1.6, @push.rocks/projectinfo -> ^5.0.2, @push.rocks/smartdata -> ^5.16.7, @push.rocks/smartpromise -> ^4.2.3, @push.rocks/smartradius -> ^1.1.0, @push.rocks/smartstate -> ^2.0.30, mailauth -> ^4.12.1)

2026-02-01 - 3.0.0 - BREAKING CHANGE(deps)

upgrade major dependencies, migrate action.target to action.targets (array), adapt to SmartRequest API changes, and add RADIUS server support

- Bumped many major dependencies: @api.global/typedserver 3.x → 8.3.0, @api.global/typedsocket 3.x → 4.1.0, @apiclient.xyz/cloudflare 6.x → 7.1.0, @design.estate/dees-catalog 1.x → 3.41.4, @push.rocks/smartpath 5.x → 6.x, @push.rocks/smartproxy 19.x → 22.x, @push.rocks/smartrequest 2.x → 5.x, uuid 11.x → 13.x, @types/node 25.1.0 → 25.2.0

2026-02-01 - 2.13.0 - feat(radius)

add RADIUS server with MAC authentication (MAB), VLAN assignment, accounting and OpsServer API handlers

- Introduce full RADIUS module under ts/radius: classes.radius.server, classes.vlan.manager, classes.accounting.manager (authentication, VLAN mapping, OUI patterns, accounting, persistence).
- Integrate RADIUS into DcRouter: add radiusConfig option, setupRadiusServer(), updateRadiusConfig(), start/stop lifecycle handling and startup summary output.
- Add OpsServer RadiusHandler (ts/opsserver/handlers/radius.handler.ts) exposing TypedRequest endpoints for client management, VLAN mappings, accounting reports and

statistics.
- Add typed request interfaces for RADIUS under ts_interfaces/requests/radius.ts and export them from the requests index.
- Wire smartradius into plugins (ts/plugins.ts) and export the new module; export RADIUS from ts/index.ts and re-export RADIUS types from classes.dcrouter.
- Update package.json & npmextra.json: add tswatch script and dev watcher configuration, add @push.rocks/smartradius dependency and a test_watch/devserver.ts dev server entrypoint.
- Refactor several web UI components (ops-dashboard, ops-view-*) to use 'accessor' for @state properties (small UI state API adjustments).
- Documentation: update readme.hints.md with RADIUS integration notes and examples.

2026-02-01 - 2.12.6 - fix(tests)

update tests and test helpers to current email/DNS APIs, use non-privileged ports, and improve robustness and resilience

- Email tests: switch to IEmailConfig properties (domains, routes), use router.emailServer (not unifiedEmailServer), change to non-privileged ports (e.g. 2525) and use fs.rmSync for cleanup.
- SMTP client helper: add pool and domain options; adjust tests to use STARTTLS (secure: false) and tolerate TLS/cipher negotiation failures with try/catch fallbacks.
- DNS tests: replace dnsDomain with dnsNsDomains and dnsScopes; test route generation without starting services, verify route names/domains, and create socket handlers without binding privileged ports.
- Socket-handler tests: use high non-standard ports for route/handler tests, verify route naming (email-port--route), ensure handlers are functions and handle errors gracefully without starting full routers.
- Integration/storage/rate-limit tests: add waits for async persistence, create/cleanup test directories, return and manage test server instances, relax strict assertions (memory threshold, rate-limiting enforcement) and make tests tolerant of implementation differences.
- Misc: use getAvailablePort in perf test setup, export tap.start() where appropriate, and generally make tests less brittle by adding try/catch, fallbacks and clearer logs for expected non-deterministic behavior.

2026-02-01 - 2.12.5 - fix(mail)

migrate filesystem helpers to fsUtils, update DKIM and mail APIs, harden SMTP client, and bump dependencies

- Introduce plugins.fsUtils compatibility layer and replace usages of plugins.smartfile.* with plugins.fsUtils.* across storage, routing, deliverability, and paths to support newer smartfile behaviour
- Update DKIM signing/verifying to new mailauth API: use signingDomain/selector/privateKey and read keys from dkimCreator before signing; adjust verifier fields to use signingDomain
- Harden SMTP client CommandHandler: add MAX_BUFFER_SIZE, socket close/error handlers, robust cleanup, clear response buffer, and adjust command/data timeouts; reduce default SOCKET_TIMEOUT to 45s
- Use SmartFileFactory for creating SmartFile attachments and update saving/loading to use fsUtils async/sync helpers
- Switch test runners to export default tap.start(), relax some memory-test thresholds, and add test helper methods (recordAuthFailure, recordError)
- Update package.json: simplify bundle script and bump multiple devDependencies/dependencies to compatible versions

2025-01-29 - 2.13.0 - feat(socket-handler)

Implement socket-handler mode for DNS and email services, enabling direct socket passing from SmartProxy

- Add `dnsDomain` configuration option that automatically sets up DNS server with DNS-over-HTTPS (DoH) support
- Implement socket-handler mode for email services with `useSocketHandler` flag in email configuration
- Update SmartProxy route generation to create socket-handler actions instead of port forwarding
- Add automatic route creation for DNS paths `/dns-query` and `/resolve` when dnsDomain is configured
- Enhance UnifiedEmailServer with `handleSocket` method for direct socket processing
- Configure DnsServer with `manualHttpsMode: true` to prevent HTTPS port binding while enabling DoH
- Improve performance by eliminating internal port forwarding overhead
- Update documentation with socket-handler mode configuration and benefits

2025-05-16 - 2.12.0 - feat(smartproxy)

Update documentation and configuration guides to adopt new route-based SmartProxy architecture

- Revise SmartProxy implementation hints in readme.hints.md to describe route-based configuration with glob pattern matching
- Add migration examples showing transition from old direct configuration to new route-based style
- Update DcRouter and SMTP port configuration to generate SmartProxy routes for email handling (ports 25, 587, 465 mapped to internal services)
- Enhance integration documentation with examples for HTTP and email services using the new SmartProxy routes

2025-05-16 - 2.11.2 - fix(dependencies)

Update dependency versions and adjust test imports to use new packages

- Upgraded @git.zone/tsbuild from ^2.3.2 to ^2.5.1
- Upgraded @git.zone/tstest/tapbundle from ^1.0.88 to ^1.9.0 and replaced @push.rocks/tapbundle imports in tests
- Upgraded @push.rocks/smartlog from ^3.0.3 to ^3.1.2
- Upgraded @push.rocks/smartproxy from ^10.2.0 to ^18.1.0
- Upgraded mailauth from ^4.8.4 to ^4.8.5

2025-05-08 - 2.11.1 - fix(platform)

Update commit info with no functional changes; regenerated commit information.

2025-05-08 - 2.11.0 - feat(platformservice)

Expose DcRouter and update package visibility. Changed package.json 'private' flag from true to false to allow public publication, and added export of DcRouter in ts/index.ts for improved API accessibility.

- Changed package.json: set 'private' to false

- Added export for DcRouter in ts/index.ts

2025-05-08 - 2.10.0 - feat(config): Implement standardized configuration system

Create a comprehensive configuration system with validation, defaults, and documentation

- Added consistent configuration interfaces across all services
- Implemented validation for all configuration objects with detailed error reporting
- Added default values for optional configuration parameters
- Created an extensive documentation system for configuration options
- Added migration helpers for managing configuration format changes
- Enhanced platform service to load configuration from multiple sources (file, environment, code)
- Updated email and SMS services to use the new configuration system

2025-05-08 - 2.9.0 - feat(errors): Implement comprehensive error handling system

Enhance error handling with structured errors, consistent patterns, and improved logging

- Added domain-specific error classes for better error categorization and handling
- Created comprehensive error codes for all service types (email, MTA, security, etc.)
- Implemented detailed error context with severity, category, and recoverability classification
- Added utilities for error conversion, formatting, and handling with automatic retry mechanisms
- Enhanced logging with correlation tracking, context support, and structured data
- Created middleware for handling errors in HTTP requests with proper status code mapping
- Added retry with exponential backoff for transient failures

2025-05-08 - 2.8.9 - fix(types)

Fix TypeScript build errors and improve API type safety across platformservice interfaces

- Fixed interface placement in EmailService and MtaConnector classes
- Aligned DeliveryStatus enum and updated ApiManager handlers with proper type-safe signatures
- Added comprehensive TypeScript interfaces for ISendEmailOptions, ITemplateContext, IValidateEmailOptions, IValidationResult, and IEmailServiceStats
- Removed circular dependencies in type definitions and added proper type assertions
- Improved test stability by handling race conditions in SenderReputationMonitor and IPWarmupManager; external DNS lookups are disabled under test environment

2025-05-08 - 2.8.8 - fix(types): Fix TypeScript build errors and improve API interfaces

Fix TypeScript build errors caused by interface placement and improve API type alignment

- Fixed interface placement in EmailService and MtaConnector classes
- Aligned DeliveryStatus enum with EmailSendJob implementation
- Added proper method signatures for API endpoint handlers in ApiManager class
- Updated getStats and checkEmailStatus methods to conform to API contracts
- Implemented type-safe return values for all API methods
- Fixed circular dependencies in type definitions
- Added proper type assertion where needed to satisfy TypeScript compiler

2025-05-08 - 2.8.7 - feat(types): Add comprehensive TypeScript interfaces for API types

Improve type safety across the platform by adding detailed TypeScript interfaces for APIs

- Added ISendEmailOptions interface with complete documentation for email sending options
- Created ITemplateContext interface for email template rendering with full type safety
- Added IValidateEmailOptions and IValidationResult interfaces for email validation
- Improved IEmailServiceStats interface with detailed statistics types
- Added IEmailStatusResponse and IEmailStatusDetails interfaces for MTA status checking
- Updated sendEmail and other methods to use these new interfaces instead of 'any'
- Removed need for type assertions in various components

2025-05-08 - 2.8.6 - fix(tests)

fix: Improve test stability by handling race conditions in SenderReputationMonitor and IPWarmupManager. Disable filesystem operations and external DNS lookups during tests by checking NODE_ENV, add proper cleanup of singleton instances and active timeouts to ensure consistent test environment.

- Bumped version from 2.8.4 to 2.8.5 in package.json and changelog.md
- Improved SenderReputationMonitor to skip filesystem operations and DNS record loading when NODE_ENV is set to test
- Added cleanup of singleton instances and active timeouts in test files
- Updated readme.plan.md with roadmap items for test stability

2025-05-08 - 2.8.5 - fix(tests): Improve test stability by fixing race conditions

Enhance the SenderReputationMonitor tests to prevent race conditions and make tests more reliable

- Modified SenderReputationMonitor to detect test environment and disable filesystem operations
- Added proper cleanup of singleton instances and timeouts between tests
- Disabled DNS lookups during tests to prevent external dependencies
- Set a consistent test environment using NODE_ENV=test
- Made all tests independent of each other to prevent shared state issues

2025-05-08 - 2.8.4 - fix(mail)

refactor(mail): Remove Mailgun references from PlatformService. Update keywords, error messages, and documentation to use MTA exclusively.

- Removed Mailgun integration from keywords in package.json and npmextra.json
- Updated EmailService to remove Mailgun API key usage and reference MTA instead
- Updated changelog.md and readme.md to reflect removal of Mailgun and update examples
- Revised error messages to mention 'MTA not configured' instead of generic provider errors
- Updated readme.plan.md to document Mailgun removal

2025-05-08 - 2.8.3 - refactor(mail): Remove Mailgun references

Remove all Mailgun references from the codebase since it's no longer used as an email provider

- Removed "mailgun integration" from keywords in package.json and npmextra.json
- Updated comments and documentation in EmailService to remove Mailgun mentions
- Updated error messages to reference MTA instead of generic email providers
- Updated the readme email example to use PlatformService reference instead of Mailgun API key

2025-05-08 - 2.8.2 - fix(tests)

Fix outdated import paths in test files for dcrouter and ratelimiter modules

- Updated dcrouter import from '../ts/dcrouter/index.js' to '../ts/classes.dcrouter.js'
- Updated ratelimiter import from '../ts/mta/classes.ratelimiter.js' to '../ts/mail/delivery/classes.ratelimiter.js'

2025-05-08 - 2.8.1 - fix(readme)

Update readme with consolidated email system improvements and modular directory structure

Clarify that the platform now organizes email functionality into distinct directories (mail/core, mail/delivery, mail/routing, mail/security, mail/services) and update the diagram and key features list accordingly. Adjust code examples to reflect explicit module imports and the use of SzPlatformService.

- Changed description of consolidated email configuration to include 'streamlined directory structure'.
- Updated mermaid diagram to show 'Mail System Structure' with separate components for core, delivery, routing, security, and services.
- Modified key features list to document modular directory structure.
- Revised code sample imports to use explicit paths and SzPlatformService.

2025-05-08 - 2.8.0 - feat(docs)

Update documentation to include consolidated email handling and pattern-based routing details

- Extended MTA section to describe the new unified email processing system with forward, MTA, and process modes
- Updated system diagram to reflect DcRouter integration with UnifiedEmailServer, DeliveryQueue, DeliverySystem, and RateLimiter
- Revised readme.plan.md checklists to mark completed features in core architecture, multi-modal processing, unified queue, and DcRouter integration

2025-05-08 - 2.7.0 - feat(dcrouter)

Implement unified email configuration with pattern-based routing and consolidated email processing. Migrate SMTP forwarding and store-and-forward into a single, configuration-driven system that supports glob pattern matching in domain rules.

- Introduced IEmailConfig interface to consolidate MTA, forwarding, and processing settings.
- Added pattern-based domain routing with glob patterns (e.g., '*@example.com', '@*.example.net').
- Reworked DcRouter integration to expose unified email handling and updated readme.plan.md and changelog.md accordingly.
- Removed deprecated SMTP forwarding components in favor of the consolidated approach.

2025-05-08 - 2.7.0 - feat(dcrouter)

Implement consolidated email configuration with pattern-based routing

- Added new pattern-based email routing with glob patterns (e.g., `*@task.vc`, `*@*.example.net`)
- Consolidated all email functionality (MTA, forwarding, processing) under a unified `emailConfig` interface
- Implemented domain router with pattern specificity calculation for most accurate matching
- Removed deprecated components (SMTP forwarding, Store-and-Forward) in favor of the unified approach
- Updated DcRouter tests to use the new consolidated email configuration pattern
- Enhanced inline documentation with detailed interface definitions and configuration examples
- Updated implementation plan with comprehensive component designs for the unified email system

2025-05-07 - 2.6.0 - feat(dcrouter)

Implement integrated DcRouter with comprehensive SmartProxy configuration, enhanced SMTP processing, and robust store-and-forward email routing

- Marked completion of tasks in readme.plan.md with [x] flags for SMTP server setup, email processing pipeline, queue management, and delivery system.
- Reworked DcRouter to use direct SmartProxy configuration, separating smtpConfig and smtpForwarding approaches.
- Added new components for delivery queue and delivery system with persistent storage support.
- Improved SMTP server implementation with TLS support, event handlers for connection, authentication, sender/recipient validation, and data processing.
- Refined domain-based routing and transformation logic in EmailProcessor with metrics and logging.
- Updated exported modules in dcrouter index to include SMTP store-and-forward components.
- Enhanced inline documentation and code comments for configuration interfaces and integration details.

2025-05-07 - 2.5.0 - feat(dcrouter)

Enhance DcRouter configuration and update documentation

- Added new implementation hints (readme.hints.md) and planning documentation (readme.plan.md) outlining removal of SzPlatformService dependency and improvements in SMTP forwarding, domain routing, and certificate management.

- Introduced new interfaces: ISmtpForwardingConfig and IDomainRoutingConfig for precise SMTP and HTTP domain routing configuration.
- Refactored DcRouter classes to support direct integration with SmartProxy and enhanced MTA functionality, including SMTP port configuration and improved TLS handling.
- Updated supporting modules such as SmtpPortConfig and EmailDomainRouter to provide better routing and security options.
- Enhanced test coverage across dcrouter, rate limiter, IP warmup manager, and email authentication, ensuring backward compatibility and improved quality.

2025-05-07 - 2.4.2 - fix(tests)

Update test assertions and singleton instance references in DMARC, integration, and IP warmup manager tests

- In test.emailauth.ts, update expected DMARC policy from 'none' to 'reject' and verify actualPolicy and action accordingly
- In test.integration.ts, remove deprecated casting and adjust dedicated policy naming (use 'dedicated' instead of 'dedicatedDomain')
- In test.ipwarmupmanager.ts and test.reputationmonitor.ts, replace singleton reset from '_instance' to 'instance' for proper instance access
- Update round robin allocation tests to verify IP cycle returns one of the available IPs
- Enhance daily limit tests by verifying getBestIPForSending returns null when limit is reached
- General refactoring across tests for improved clarity and consistency

2025-05-07 - 2.4.1 - fix(tests)

Update test assertions and refine service interfaces

- Converted outdated chai assertions to use tap's toBeTruthy, toEqual, and toBeGreaterThan methods in multiple test files
- Appended tap.stopForcefully() tests to ensure proper cleanup in test suites
- Added stop() method to PlatformService for graceful shutdown
- Exposed certificate property in MtaService from private to public
- Refactored dcrouter smartProxy configuration to better handle MTA service integration and certificate provisioning

2025-05-07 - 2.4.0 - feat(email)

Enhance email integration by updating @push.rocks/smartmail to ^2.1.0 and improving the entire email stack including validation, DKIM verification, templating, MIME conversion, and attachment handling.

- Updated smartmail dependency from ^2.0.1 to ^2.1.0 in package.json
- Enhanced EmailValidator with comprehensive checks (syntax, MX, disposable and role validations)
- Refactored TemplateManager to support dynamic variable substitution and loading templates from directory
- Improved conversion between internal Email and smartmail.Smartmail, streamlining MIME handling and attachment mapping
- Augmented DKIM verification with caching and custom header injection for improved security reporting
- Updated readme.plan.md with detailed roadmap for further performance, security, analytics, and deliverability enhancements
- Expanded test suite to cover smartmail integration, validation, templating, and conversion between formats

2025-05-04 - 1.0.10 to 1.0.8 - core

Applied core fixes across several versions on this day.

- Fixed core issues in versions 1.0.10, 1.0.9, and 1.0.8

2024-04-01 - 1.0.7 - core

Applied a core fix.

- Fixed core functionality for version 1.0.7

2024-03-19 - 1.0.6 - core

Applied a core fix.

- Fixed core functionality for version 1.0.6

2024-02-16 - 1.0.5 to 1.0.2 - core

Applied multiple core fixes in a contiguous range of versions.

- Fixed core functionality for versions 1.0.5, 1.0.4, 1.0.3, and 1.0.2

2024-02-15 - 1.0.1 - core

Applied a core fix.

- Fixed core functionality for version 1.0.1

–––––––––––––––––––––– Note: Versions that only contained version bumps (for example, 1.0.11 and the plain "1.0.x" commits) have been omitted from individual entries and are implicitly included in the version ranges above.