@push.rocks/smartnetwork

A toolkit for network diagnostics including speed tests, port availability checks, and more.

- readme.md for @push.rocks/smartnetwork
- changelog.md for @push.rocks/smartnetwork

readme.md for @push.rocks/smartnetwork

Comprehensive network diagnostics and IP intelligence for Node.js — speed tests, port scanning, ICMP ping, traceroute, DNS, RDAP, ASN lookups, and geolocation in a single, promise-based toolkit.

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.

□□ Install

```
pnpm install @push.rocks/smartnetwork --save
```

□□ Overview

@push.rocks/smartnetwork is your Swiss Army knife for network diagnostics in Node.js. Whether you're building monitoring dashboards, investigating IP ownership, or debugging connectivity — this library has you covered with a clean, async API and zero-config setup.

Under the hood, system-level operations (ICMP ping, traceroute, raw-socket port checks, gateway detection) are powered by a bundled **Rust binary** for maximum performance and cross-platform reliability. Everything else — speed tests, DNS, public IP discovery, IP intelligence, HTTP health checks — runs in pure TypeScript.

□ Key Features

Category	Capabilities
□□ **Speed Testing**	Download/upload benchmarks via Cloudflare's global infrastructure
□□ **Port Management**	Local/remote port checks, find free ports (sequential or random), exclusion lists
□□ **Connectivity**	ICMP ping with stats, hop-by-hop traceroute, HTTP/HTTPS health checks
□□ **DNS**	A, AAAA, MX resolution with system-first + DoH fallback strategy
□□ **IP Intelligence**	ASN, organization, geolocation, RDAP registration — all from free public sources
□□ **Network Discovery**	Interfaces, default gateway, public IPv4/IPv6
⚡ **Caching**	Built-in TTL cache for expensive lookups
□□ **Extensible**	Plugin architecture for custom functionality
□□ **TypeScript**	Full type definitions, ESM-native

□□ Usage

Basic Setup

```
import { SmartNetwork } from '@push.rocks/smartnetwork';

const network = new SmartNetwork();

// Start the Rust bridge (auto-started on first use, but explicit start is recommended)
await network.start();

// ... use the network instance ...

// Clean up when done
await network.stop();
```

Enable caching for repeated lookups:

```
const network = new SmartNetwork({ cacheTtl: 60000 }); // 60s TTL
```

□□ IP Intelligence

Get ASN, organization, geolocation, and RDAP registration data for any IPv4 address. Combines three free public data sources in parallel:

- **RDAP** — direct queries to RIRs (RIPE, ARIN, APNIC, LACNIC, AFRINIC) for authoritative registration data
- **Team Cymru DNS** — fast ASN resolution via DNS TXT records
- **MaxMind GeoLite2** — in-memory MMDB databases (auto-downloaded from CDN, periodically refreshed)

```
const intel = await network.getIpIntelligence('8.8.8.8');

console.log(intel);
// {
//   asn: 15169,
//   asnOrg: 'Google LLC',
//   registrantOrg: 'Google LLC',
//   registrantCountry: 'United States',
//   networkRange: '8.8.8.0/24',
//   abuseContact: null,
//   country: null,
//   countryCode: 'US',
//   city: null,
//   latitude: 37.751,
//   longitude: -97.822,
//   accuracyRadius: null,
//   timezone: 'America/Chicago'
// }
```

Works great for your own IP too:

```
const publicIps = await network.getPublicIps();
if (publicIps.v4) {
  const myIntel = await network.getIpIntelligence(publicIps.v4);
  console.log(`You're on AS${myIntel.asn} (${myIntel.asnOrg}) in ${myIntel.city},
${myIntel.countryCode}`);
}
```

The `IIpIntelligenceResult` interface:

```
interface IIpIntelligenceResult {
  // ASN (Team Cymru primary, MaxMind fallback)
  asn: number | null;
  asnOrg: string | null;

  // Registration (RDAP)
  registrantOrg: string | null;
  registrantCountry: string | null;
  networkRange: string | null;       // CIDR or range
  abuseContact: string | null;       // abuse email from RDAP

  // Geolocation (MaxMind GeoLite2)
  country: string | null;
  countryCode: string | null;        // ISO 3166-1 alpha-2
  city: string | null;
  latitude: number | null;
  longitude: number | null;
  accuracyRadius: number | null;     // km
  timezone: string | null;           // IANA timezone
}
```

> □□ The GeoLite2 databases are fetched into memory from jsDelivr CDN on first use (~32 MB total). They auto-refresh in the background every 7 days (configurable via `IpIntelligence` options). No disk I/O, no API keys, no rate limits.

□□ Speed Testing

Measure network performance via Cloudflare's global infrastructure:

```
const result = await network.getSpeed();
console.log(`⬇□  Download: ${result.downloadSpeed} Mbps`);
console.log(`⬆□  Upload: ${result.uploadSpeed} Mbps`);

// Advanced: parallel streams + fixed duration
const advanced = await network.getSpeed({
  parallelStreams: 3,
```

```
  duration: 5,
});
```

□□ Port Management

Check Local Port Availability

Checks both IPv4 and IPv6:

```
const isUnused = await network.isLocalPortUnused(8080);
console.log(isUnused ? '□ Port 8080 is free' : '□ Port 8080 is in use');
```

Find Free Port in Range

```
// First available
const port = await network.findFreePort(3000, 3100);

// Random pick (avoids clustering)
const randomPort = await network.findFreePort(3000, 3100, { randomize: true });

// With exclusions
const port2 = await network.findFreePort(3000, 3100, {
  randomize: true,
  exclude: [3000, 3001, 3005],
});
```

Check Remote Port

```
// Simple
const isOpen = await network.isRemotePortAvailable('example.com', 443);

// With retries and timeout
const isOpen2 = await network.isRemotePortAvailable('example.com', {
  port: 443,
  retries: 3,
  timeout: 5000,
});
```

□□ Ping & Traceroute

```
// Simple ping
const ping = await network.ping('google.com');
console.log(`Alive: ${ping.alive}, RTT: ${ping.time} ms`);

// Multi-ping with statistics
const stats = await network.ping('google.com', { count: 10, timeout: 2000 });
console.log(`□□min=${stats.min} avg=${stats.avg.toFixed(1)} max=${stats.max}
loss=${stats.packetLoss}%`);

// Traceroute
const hops = await network.traceroute('google.com', { maxHops: 20 });
hops.forEach(hop => {
  const rtt = hop.rtt === null ? '*' : `${hop.rtt} ms`;
  console.log(`  ${hop.ttl}\t${hop.ip}\t${rtt}`);
});
```

> ⚠ ICMP ping requires `CAP_NET_RAW` or appropriate `ping_group_range` sysctl on Linux.

□□ DNS Resolution

Uses `@push.rocks/smartdns` with a system-first strategy and automatic DoH (DNS-over-HTTPS) fallback:

```
const dns = await network.resolveDns('example.com');
console.log('A records:', dns.A);       // ['93.184.216.34']
console.log('AAAA records:', dns.AAAA); // ['2606:2800:220:1:248:1893:25c8:1946']
dns.MX.forEach(mx => {
  console.log(`□□${mx.exchange} (priority ${mx.priority})`);
});
```

□□ HTTP/HTTPS Health Checks

```
const health = await network.checkEndpoint('https://api.example.com/health', {
  timeout: 5000,
  rejectUnauthorized: true,
});
console.log(`Status: ${health.status}, RTT: ${health.rtt.toFixed(0)} ms`);
```

□□ Network Interfaces & Public IPs

```
// All interfaces
const gateways = await network.getGateways();
Object.entries(gateways).forEach(([name, ifaces]) => {
  console.log(`□□${name}:`);
  ifaces.forEach(i => console.log(`  ${i.family}: ${i.address}`));
});

// Default gateway
const gw = await network.getDefaultGateway();
if (gw) {
  console.log(`□□Gateway IPv4: ${gw.ipv4.address}`);
}

// Public IPs (multiple fallback services: ipify, ident.me, seeip, icanhazip)
const publicIps = await network.getPublicIps();
console.log(`□□IPv4: ${publicIps.v4 || 'N/A'}`);
console.log(`□□IPv6: ${publicIps.v6 || 'N/A'}`);
```

⚡ Caching

Caching applies to `getGateways()`, `getPublicIps()`, and `getIpIntelligence()`:

```
const network = new SmartNetwork({ cacheTtl: 60000 }); // 60s

const ips1 = await network.getPublicIps();  // fetches
const ips2 = await network.getPublicIps();  // cache hit ⚡
```

□□ Error Handling

```
import { SmartNetwork, NetworkError, TimeoutError } from '@push.rocks/smartnetwork';

try {
  await network.isRemotePortAvailable('example.com', { protocol: 'udp' });
} catch (error) {
  if (error instanceof NetworkError) {
    console.error(`${error.message} (code: ${error.code})`); // ENOTSUP
  }
}
```

Error codes: `EINVAL` (invalid argument), `ENOTSUP` (not supported), `ETIMEOUT` (timeout).

□□ Plugin Architecture

```
class MyPlugin {
  constructor(private network: SmartNetwork) {}
  async doStuff() { /* ... */ }
}

SmartNetwork.registerPlugin('myPlugin', MyPlugin);

const PluginClass = SmartNetwork.pluginsRegistry.get('myPlugin');
const plugin = new PluginClass(network);
await plugin.doStuff();

SmartNetwork.unregisterPlugin('myPlugin');
```

□□ Custom Logging

Replace the default `console` logger:

```
import { setLogger } from '@push.rocks/smartnetwork';

setLogger({
```

```
  debug: (msg) => myLogger.debug(msg),
  info: (msg) => myLogger.info(msg),
  warn: (msg) => myLogger.warn(msg),
  error: (msg) => myLogger.error(msg),
});
```

□□ Advanced Example: Network Monitor

```
const monitor = async () => {
  const network = new SmartNetwork({ cacheTtl: 30000 });

  // Check critical services
  const services = [
    { name: 'Web', host: 'example.com', port: 443 },
    { name: 'DB', host: 'db.internal', port: 5432 },
  ];

  for (const svc of services) {
    const up = await network.isRemotePortAvailable(svc.host, svc.port);
    console.log(`${svc.name}: ${up ? '□ UP' : '□ DOWN'}`);
  }

  // Internet connectivity + latency
  const ping = await network.ping('8.8.8.8');
  console.log(`Internet: ${ping.alive ? '□' : '□'} (${ping.time} ms)`);

  // Speed
  const speed = await network.getSpeed();
  console.log(`Speed: ⬇□ ${speed.downloadSpeed} / ⬆□ ${speed.uploadSpeed} Mbps`);

  // Who am I?
  const ips = await network.getPublicIps();
  if (ips.v4) {
    const intel = await network.getIpIntelligence(ips.v4);
```

```
    console.log(`AS${intel.asn} (${intel.asnOrg}) — ${intel.city || intel.countryCode}`);
  }

  await network.stop();
};
```

□□ Full API Reference

Method	Description	Requires Rust
`start()` / `stop()`	Start/stop the Rust binary bridge	—
`getSpeed(opts?)`	Cloudflare speed test (download + upload)	No
`ping(host, opts?)`	ICMP ping with optional multi-ping stats	Yes
`traceroute(host, opts?)`	Hop-by-hop network path analysis	Yes
`isLocalPortUnused(port)`	Check if local port is free (IPv4+IPv6)	Yes
`findFreePort(start, end, opts?)`	Find available port in range	Yes
`isRemotePortAvailable(target, opts?)`	TCP port check on remote host	Yes
`getGateways()`	List all network interfaces	No
`getDefaultGateway()`	Get default gateway info	Yes
`getPublicIps()`	Discover public IPv4/IPv6 (4 fallback services)	No
`resolveDns(host)`	Resolve A, AAAA, MX records	No
`checkEndpoint(url, opts?)`	HTTP/HTTPS health check with RTT	No
`getIpIntelligence(ip)`	ASN + org + geo + RDAP registration	No

License and Legal Information

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 @push.rocks/smartnetwork

2026-03-26 - 4.5.2 - fix(docs)

refresh README content and align license copyright holder

- Rewrite the README to better present network diagnostics, IP intelligence capabilities, usage examples, and full API reference
- Add issue reporting guidance and clarify legal and trademark wording in project documentation
- Update the license copyright holder to Task Venture Capital GmbH

2026-03-26 - 4.5.1 - fix(ipintelligence)

handle flat geolocation MMDB schema and clean up DNS client lifecycle

- update IP intelligence lookups to read geolocation fields from the flat @ip-location-db schema instead of MaxMind nested city records
- destroy the Team Cymru DNS client after queries to avoid leaking resources
- adjust IP intelligence tests to validate countryCode-based geolocation and RDAP fields against current data sources

2026-03-26 - 4.5.0 - feat(smartnetwork)

add Rust-powered network diagnostics bridge and IP intelligence lookups

- replace system-dependent ping, traceroute, port checks, and gateway detection with a Rust IPC binary integrated through smartrust
- add IP intelligence support combining RDAP, ASN, and MaxMind geolocation data
- update build configuration to compile and package Rust binaries for Linux amd64 and arm64
- refresh tests and docs for explicit SmartNetwork start/stop lifecycle and new functionality

2025-09-12 - 4.4.0 - feat(smartnetwork)

Add exclude option to findFreePort and skip excluded ports during search

- Add an 'exclude' array to IFindFreePortOptions so callers can specify ports to ignore when searching for a free port.
- Respect excluded ports in findFreePort for both random (randomize=true) and sequential searches so excluded ports are never returned.
- Add .claude/settings.local.json to include local permissions used for development/CI helpers.

2025-09-12 - 4.3.0 - feat(smartnetwork)

Add randomizable port search and switch DNS resolution to @push.rocks/smartdns; export smartdns and update docs

- findFreePort: added options.randomize (IFindFreePortOptions) to return a random available port within a range; retains default sequential search behavior
- resolveDns: switched to @push.rocks/smartdns client with 'prefer-system' strategy and DoH fallback; now returns parsed A, AAAA and MX records
- Exported smartdns from the internal plugins module and added @push.rocks/smartdns dependency
- Documentation updates: readme clarifies random port selection, DNS resolution strategy and adds examples for random findFreePort usage
- Bumped @git.zone/tstest dev dependency to ^2.3.7

2025-09-12 - 4.2.0 - feat(PublicIp)

Add PublicIp service and refactor SmartNetwork to use it; remove public-ip dependency; update exports, docs and dependencies

- Add PublicIp class (ts/smartnetwork.classes.publicip.ts) implementing public IPv4/IPv6 lookup with multiple fallback services, timeouts and validation
- Refactor SmartNetwork.getPublicIps to use the new PublicIp class and preserve caching behavior
- Export PublicIp from package entry (ts/index.ts)
- Remove public-ip from plugins/exports and stop using the public-ip package
- Bump devDependencies and runtime dependency versions in package.json (@git.zone/tsbuild, @git.zone/tstest, @push.rocks/smartenv, systeminformation)
- Improve README: expanded usage, examples, formatting and added emojis for clarity
- Add project local settings file (.claude/settings.local.json) for CI/permissions configuration

2025-07-31 - 4.1.0 - feat(port-management)

Add findFreePort method for automatic port discovery within a range

- Added new `findFreePort` method to SmartNetwork class that finds the first available port in a specified range
- Added comprehensive tests for the new port finding functionality
- Updated README documentation with usage examples for the new feature
- Improved port management capabilities for dynamic port allocation scenarios

2025-05-19 - 4.0.2 - fix(tests)

Update dev dependencies and refactor test assertions for improved clarity

- Bumped @git.zone/tsbuild version to ^2.5.1
- Bumped @git.zone/tstest version to ^1.9.0
- Updated npm test script to include the verbose flag
- Replaced expectAsync assertions with resolves based assertions in test files

2025-05-03 - 4.0.1 - fix(formatting)

Fix minor formatting issues and newline consistency across project files

- Ensure newline at end of package.json, errors.ts, logging.ts, and test files
- Refine code block formatting in readme.md
- Adjust whitespace and code style in smartnetwork classes and cloudflarespeed module
- Minor commitinfo data format update

2025-04-28 - 4.0.0 - BREAKING CHANGE(smartnetwork)

Enhance documentation and add configurable speed test options with plugin architecture improvements

- Expanded README with detailed examples for traceroute, speed test, ping, and remote port checks
- Added optional parameters for getSpeed (parallelStreams and duration) to allow configurable testing modes
- Included plugin architecture usage examples to show runtime registration and unregistration of plugins
- Updated test suite to cover DNS resolution, endpoint health-check, caching behavior, and various network diagnostics
- Removed legacy planning documentation from readme.plan.md

2025-04-28 - 3.0.5 - fix(core)

Improve logging and error handling by introducing custom error classes and a global logging interface while refactoring network diagnostics methods.

- Added custom error classes (NetworkError, TimeoutError) for network operations.
- Introduced a global logging interface to replace direct console logging.
- Updated CloudflareSpeed and SmartNetwork classes to use getLogger for improved error reporting.
- Disabled connection pooling in HTTP requests to prevent listener accumulation.

2025-04-28 - 3.0.4 - fix(ci/config)

Improve CI workflows, update project configuration, and clean up code formatting

- Added new Gitea workflow files (default_nottags.yaml and default_tags.yaml) to replace GitLab CI
- Updated package.json with new buildDocs script, revised homepage URL, bug tracking info, and pnpm overrides
- Refined code formatting in TypeScript files, including improved error handling in Cloudflare speed tests and consistent callback structure
- Enhanced tsconfig.json by adding baseUrl and paths for better module resolution
- Introduced readme.plan.md outlining future improvements and feature enhancements

2025-04-28 - 3.0.3 - fix(deps)

Update dependency namespaces and bump package versions in CI configuration and source imports

- Renamed dependency imports from '@pushrocks/...' to '@push.rocks/...' in package.json, test files, and source files
- Updated devDependencies versions and package references (e.g., tsbuild, tsrun, tstest, smartenv, tapbundle, smartpromise, smartstring, public-ip, and @types packages)
- Fixed CI command to install '@git.zone/tsdoc' instead of the misspelled '@gitzone/tsdoc'
- Updated commit info file to reflect the correct package namespace

2024-05-29 - 3.0.2 - misc

Various documentation, configuration, and organizational updates.

- Updated project description.
- Updated tsconfig configuration.
- Updated npmextra.json (githost configuration) across multiple commits.
- Switched to a new organizational scheme.

2022-10-22 - 3.0.1 - core

Core fixes and maintenance applied.

- Fixed core issues.

2022-10-21 - 3.0.0 - core

Core updates and fixes.

- Fixed core issues.

2022-03-24 - 2.0.14 - core

Breaking changes introduced.

- BREAKING CHANGE: Switched core to an ECMAScript Modules (ESM) approach.

2022-02-16 - 2.0.13 - core

Core maintenance update.

- Fixed core issues.

2022-02-16 - 2.0.12 - core

Core maintenance update.

- Fixed core issues.

2022-02-16 - 2.0.11 - core

Core maintenance update.

- Fixed core issues.

2022-02-16 - 2.0.10 - core

Core maintenance update.

- Fixed core issues.

2021-04-29 - 2.0.9 - core

Core maintenance update.

- Fixed core issues.

2021-04-28 - 2.0.8 - core

Core maintenance update.

- Fixed core issues.

2021-04-28 - 2.0.7 - core

Core maintenance update.

- Fixed core issues.

2021-04-28 - 2.0.6 - core

Core maintenance update.

- Fixed core issues.

2021-04-28 - 2.0.5 - core

Core maintenance update.

- Fixed core issues.

2021-04-27 - 2.0.4 - core

Core maintenance update.

- Fixed core issues.

2021-04-13 - 2.0.3 - core

Core maintenance update.

- Fixed core issues.

2021-04-13 - 2.0.2 - core

Core maintenance update.

- Fixed core issues.

2021-04-13 - 2.0.1 - core

Core maintenance update.

- Fixed core issues.

2021-04-13 - 2.0.0 - core

Core maintenance update.

- Fixed core issues.

2021-04-13 - 1.1.22 - core

Breaking update in core functionality.

- BREAKING CHANGE: Updated core functionality.

2020-08-13 - 1.1.21 - core

Core maintenance update.

- Fixed core issues.

2020-08-13 - 1.1.20 - core

Core maintenance update.

- Fixed core issues.

2020-08-12 - 1.1.19 - core

Core maintenance update.

- Fixed core issues.

2020-08-12 - 1.1.18 - core

Core maintenance update.

- Fixed core issues.

2019-11-23 - 1.1.17 - minor

Release with no notable changes.

- No significant changes.

2019-11-23 - 1.1.16 - core

Core maintenance update.

- Fixed core issues.

2019-11-19 - 1.1.15 - security

Security enhancement.

- Added Snyk configuration for improved security.

2019-11-19 - 1.1.14 - dependencies

Dependency update.

- Updated dependencies.

2019-09-08 - 1.1.13 - core

Core maintenance update.

- Fixed core issues.

2019-09-08 - 1.1.12 - core

Core maintenance update.

- Fixed core issues.

2019-09-08 - 1.1.11 - core

Core maintenance update.

- Fixed core issues.

2019-09-08 - 1.1.10 - core

Core maintenance update.

- Fixed core issues.

2019-09-08 - 1.1.9 - core

Core maintenance update.

- Fixed core issues.

2019-09-06 - 1.1.8 - core

Core maintenance update.

- Fixed core issues.

2019-09-06 - 1.1.7 - core

Core maintenance update.

- Fixed core issues.

2019-09-06 - 1.1.6 - core

Core maintenance update.

- Fixed core issues.

2019-04-17 - 1.1.5 - core

Core maintenance update.

- Fixed core issues.

2019-04-17 - 1.1.4 - core

Core maintenance update.

- Fixed core issues.

2019-04-17 - 1.1.3 - core

Core maintenance update.

- Fixed core issues.

2019-04-17 - 1.1.2 - core

Core maintenance update.

- Fixed core issues.

2019-04-16 - 1.1.1 - core

Core maintenance update.

- Fixed core issues.

2019-04-16 - 1.1.0 - core

Core maintenance update.

- Fixed core issues.

2018-07-17 - 1.0.2 - feature

New feature added.

- Added port checking functionality.

2017-12-12 - 1.0.1 - initial

Initial commit.

- Initial project setup.