

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

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.

Revision #3

Created 2026-03-28 11:11:33 UTC by foss.global Team

Updated 2026-03-28 12:18:24 UTC by foss.global Team