changelog.md for @push.rocks/smartipc

Changelog

2025-08-30 - 2.3.0 - feat(streaming)

Add streaming support: chunked stream transfers, file send/receive, stream events and helpers

- Implement chunked streaming protocol in IpcChannel (init / chunk / end / error / cancel messages)
- Add sendStream, cancelOutgoingStream and cancelIncomingStream methods to IpcChannel
- Expose high-level streaming API on client: sendStream, sendFile, cancelOutgoingStream, cancelIncomingStream
- Expose high-level streaming API on server: sendStreamToClient, sendFileToClient, cancelIncomingStreamFromClient, cancelOutgoingStreamToClient
- Emit 'stream' events from channels/servers/clients with (info, readable) where info includes streamId, meta, headers and clientId
- Add maxConcurrentStreams option (default 32) and enforce concurrent stream limits for incoming/outgoing
- Add SmartIpc.pipeStreamToFile helper to persist incoming streams to disk
- Export stream in smartipc.plugins and update README with streaming usage and examples
- Add comprehensive streaming tests (test/test.streaming.ts) covering large payloads, file transfer, cancellation and concurrency limits

2025-08-29 - 2.2.2 - fix(ipc)

Propagate per-client disconnects, add proper routing for targeted messages, and remove unused node-ipc deps

- Forward per-client 'clientDisconnected' events from transports up through IpcChannel and IpcServer so higher layers can react and clean up state.
- IpcChannel re-emits 'clientDisconnected' and allows registering handlers for it.

- IpcServer now listens for 'clientDisconnected' to cleanup topic subscriptions, remove clients from the map, and emit 'clientDisconnect'.
- sendToClient injects the target clientId into headers so transports can route messages to the correct socket instead of broadcasting.
- broadcast and broadcastTo delegate to sendToClient to ensure messages are routed to intended recipients and errors are attributed to the correct client.
- Transports now emit 'clientDisconnected' with the clientId when known.
- package.json: removed unused node-ipc and @types/node-ipc dependencies (dependency cleanup).

2025-08-29 - 2.2.1 - fix(tests)

Remove redundant manual topic handlers from tests and rely on server built-in pub/sub

- Removed manual server.onMessage('**subscribe**') and server.onMessage('**publish**') handlers from test/test.ts
- Tests now rely on the server's built-in publish/subscribe behavior: clients publish directly and subscribers receive messages
- Test code simplified without changing public API or runtime behavior

2025-08-29 - 2.2.0 - feat(ipcclient)

Add clientOnly mode to prevent clients from auto-starting servers and improve registration/reconnect behavior

- Introduce a clientOnly option on transports and clients, and support SMARTIPC_CLIENT_ONLY=1 env override to prevent a client from auto-starting a server when connect() encounters ECONNREFUSED/ENOENT.
- Update UnixSocketTransport/TcpTransport connect behavior: if clientOnly (or env override) is enabled, reject connect with a descriptive error instead of starting a server (preserves backward compatibility when disabled).
- Make SmartIpc.waitForServer use clientOnly probing to avoid accidental server creation during readiness checks.
- Refactor IpcClient registration flow: extract attemptRegistrationInternal, set didRegisterOnce flag, and automatically re-register on reconnects when previously registered.
- Add and update tests to cover clientOnly behavior, SMARTIPC_CLIENT_ONLY env enforcement, temporary socket paths and automatic cleanup, and other reliability improvements.
- Update README with a new 'Client-Only Mode' section documenting the option, env override, and examples.

2025-08-28 - 2.1.3 - fix(classes.ipcchannel)

Normalize heartbeatThrowOnTimeout option parsing and allow registering 'heartbeatTimeout' via IpcChannel.on

- Normalize heartbeatThrowOnTimeout to boolean (accepts 'true'/'false' strings and other truthy/falsey values) to be defensive for JS consumers
- Expose 'heartbeatTimeout' as a special channel event so handlers registered via IpcChannel.on('heartbeatTimeout', ...) will be called

2025-08-26 - 2.1.2 - fix(core)

Improve heartbeat handling and transport routing; forward heartbeat timeout events; include clientId routing and probe improvements

- IpcChannel: add heartbeatInitialGracePeriod handling — delay heartbeat timeout checks until the grace period elapses and use a minimum check interval (>= 1000ms)
- IpcChannel: add heartbeatGraceTimer and ensure stopHeartbeat clears the grace timer to avoid repeated events
- IpcChannel / Client / Server: forward heartbeatTimeout events instead of only throwing when configured (heartbeatThrowOnTimeout = false) so consumers can handle timeouts via events
- IpcClient: include clientId in registration request headers to enable proper routing on the server/transport side
- UnixSocketTransport: track socket <-> clientId mappings, clean them up on socket close, and update mappings when **register** or messages containing clientId are received
- UnixSocketTransport: route messages to a specific client when headers.clientId is present (fallback to broadcasting when no target is found), and emit both clientMessage and message for parsed client messages
- ts/index.waitForServer: use SmartIpc.createClient for probing, shorten probe register timeout, and use a slightly longer retry delay between probes for stability

2025-08-25 - 2.1.1 - fix(readme)

Update README: expand docs, examples, server readiness, heartbeat, and testing utilities

- Rewrite introduction and overall tone to emphasize zero-dependency, reliability, and TypeScript support

- Replace several Quick Start examples to use socketPath and show autoCleanupSocketFile usage
- Add Server readiness detection docs and SmartIpc.waitForServer example
- Document smart connection retry options (connectRetry) and registerTimeoutMs usage
- Clarify heartbeat configuration and add heartbeatThrowOnTimeout option to emit events instead of throwing
- Add sections for automatic socket cleanup, broadcasting, testing utilities (waitForServer, spawnAndConnect), and metrics
- Various formatting and copy improvements throughout README

2025-08-25 - 2.1.0 - feat(core)

Add heartbeat grace/timeout options, client retry/wait-for-ready, server readiness and socket cleanup, transport socket options, helper utilities, and tests

- IpcChannel: add heartbeatInitialGracePeriodMs and heartbeatThrowOnTimeout; emit 'heartbeatTimeout' event when configured instead of throwing and disconnecting immediately.
- IpcClient: add connectRetry configuration, registerTimeoutMs, waitForReady option and robust connect logic with exponential backoff and total timeout handling.
- IpcServer: add start option readyWhen ('accepting'), isReady/getIsReady API, autoCleanupSocketFile and socketMode support for managing stale socket files and permissions.
- Transports: support autoCleanupSocketFile and socketMode (cleanup stale socket files and set socket permissions where applicable).
- SmartIpc: add waitForServer helper to wait until a server is ready and spawnAndConnect helper to spawn a server process and connect a client.
- Tests: add comprehensive tests (test.improvements.ts and test.reliability.ts) covering readiness, socket cleanup, retries, heartbeat behavior, race conditions, multiple clients, and server restart scenarios.

2025-08-25 - 2.0.3 - fix(ipc)

Patch release prep: bump patch version and release minor fixes

- No changes detected in the provided diff; repository files currently declare version 2.0.2.
- Recommend a patch bump to 2.0.3 to prepare a new release (no breaking changes identified).

2025-08-24 - 2.0.2 - fix(packaging)

Update package metadata: add exports, mark package public; clean up README contributing section

- Add an exports entry in package.json pointing to ./dist_ts/index.js for proper ESM exports resolution
- Mark package as public (private: false) and remove legacy main/typings fields
- Remove the Contributing section and example contributor workflow from README

2025-08-24 - 2.0.1 - fix(npm)

Remove .npmrc to avoid committing npm registry configuration

- Deleted .npmrc which contained a hardcoded registry (https://registry.npmjs.org/).
- Prevents accidental leakage of local npm configuration into the repository and avoids affecting CI/publish behavior.

2025-08-24 - 2.0.0 - BREAKING CHANGE(core)

Refactor core IPC: replace node-ipc with native transports and add IpcChannel / IpcServer / IpcClient with heartbeat, reconnection, request/response and pub/sub. Update tests and documentation.

- Replaced node-ipc with native Node.js transports (net module) and length-prefixed framing
- Added transport abstraction (IpcTransport) and implementations: UnixSocketTransport, NamedPipeTransport, TcpTransport plus createTransport factory
- Introduced IpcChannel with automatic reconnection (exponential backoff), heartbeat, request/response tracking, pending request timeouts and metrics
- Implemented IpcServer and IpcClient classes with client registration, pub/sub (subscribe/publish), broadcast, targeted messaging, client management and idle timeout handling
- Exported factory API via SmartIpc.createServer / createClient / createChannel and updated ts/index accordingly
- Updated and expanded README with usage, examples, advanced features and migration guidance; added readme.plan.md
- Added and updated comprehensive tests (test/test.ts, test/test.simple.ts) to cover TCP transport, messaging patterns, reconnection and metrics

2025-08-23 - 1.0.8 - chore

Metadata and configuration updates; repository/org migration.

- Update package description and general project metadata.
- Update TypeScript configuration (tsconfig).
- Update npmextra.json githost entries (multiple updates).
- Switch to new organization scheme for the repository.
- Miscellaneous minor updates.

2019-04-09 - 1.0.1 - 1.0.7 - core

Initial release and a series of patch fixes to core components.

- 1.0.1: initial release.
- 1.0.2 → 1.0.7: a sequence of small core fixes and maintenance updates (repeated "fix(core): update" commits).

2025-08-29 - 2.1.4 - feat(transports)

Add client-only mode to prevent unintended server auto-start in Unix/NamedPipe transports; safer probing

- Add `clientOnly?: boolean` to transport options; when true (or `SMARTIPC_CLIENT_ONLY=1`), a client will fail fast on `ECONNREFUSED`/`ENOENT` instead of auto-starting a server.
- Update `SmartIpc.waitForServer()` to probe with `clientOnly: true` to avoid races during readiness checks.
- Extend tests to cover option and env override; update core test to use unique socket path and auto-cleanup.
- Docs: add README section for client-only mode.

Revision #1
Created 2026-03-28 13:08:57 UTC by foss.global Team
Updated 2026-03-28 13:08:57 UTC by foss.global Team