changelog.md for @push.rocks/smartrequest

2025-11-17 - 5.0.1 - fix(test)

Enable --logfile in test script and bump @git.zone/tstest to ^2.8.2

- Update npm script: add --logfile flag to the test command to produce test logs
- Bump devDependency @git.zone/tstest from ^2.8.1 to ^2.8.2

2025-11-17 - 5.0.0 - BREAKING CHANGE(client/streaming)

Unify streaming APIs: remove raw()/streamNode() and standardize on web ReadableStream across runtimes

- Removed SmartRequest.raw() and RawStreamFunction type. The raw streaming function API is gone — use .stream() with a web ReadableStream for request body streaming.
- Removed response.streamNode() from all runtimes. Responses now expose only response.stream() (ReadableStream<Uint8Array>). Node.js consumers must convert using Readable.fromWeb() if a Node.js stream is required.
- Node implementation now uses Readable.toWeb() to convert native Node streams into web ReadableStream for a single cross-platform streaming API.
- Client request.stream() still accepts Node.js streams but they are converted internally to web streams; temporary internal properties for raw streaming were removed.
- Updated tests and documentation (readme) with migration guidance and examples for converting between web and Node.js streams.
- Bumped devDependencies (@git.zone/tsbuild, tsrun, tstest) and upgraded form-data to a newer patch release.

2025-11-16 - 4.4.2 - fix(core_base/request)

Strip 'unix:' prefix when parsing unix socket URLs so socketPath is a clean filesystem path

- CoreRequest.parseUnixSocketUrl now removes a leading 'unix:' prefix and returns socketPath as a filesystem path (e.g., /var/run/docker.sock)
- Updated tests for Bun, Deno and Node to expect socketPath without the 'unix:' prefix
- Adjusted comments/documentation in core_base/request.ts to clarify returned socketPath format

2025-11-16 - 4.4.1 - fix(core_node)

Fix unix socket URL parsing and handling in CoreRequest

- CoreRequest.parseUnixSocketUrl now strips http:// and https:// prefixes so it correctly parses both full URLs (e.g. http://unix:/path/to/socket:/route) and already-stripped unix: paths.
- Node.js CoreRequest now passes the original request URL to parseUnixSocketUrl instead of options.path, preventing incorrect socketPath/path extraction.
- Fixes connection failures when using unix socket URLs (for example when targeting Docker via http://unix:/var/run/docker.sock:/v1.24/...).

2025-11-16 - 4.4.0 - feat(core)

Add Bun and Deno runtime support, unify core loader, unix-socket support and cross-runtime streaming/tests

- package.json: expose ./core_bun and ./core_deno in exports and add runtime-related keywords
- Core dynamic loader (ts/core/index.ts): detect Bun and Deno at runtime and load corresponding implementations
- New runtime modules: added ts/core_bun/* and ts/core_deno/* (response, types, index) to provide Bun and Deno CoreResponse/CoreRequest wrappers
- Client streaming: SmartRequest no longer immediately deletes temporary __nodeStream and __rawStreamFunc props — CoreRequest implementations handle them; temporary properties are cleaned up after CoreRequest is created

- Node.js request: core_node/request.ts converts web ReadableStream to Node.js Readable via stream.Readable.fromWeb and pipes it; also supports passing requestDataFunc for raw streaming
- core_node/plugins: export stream helper and rework third-party exports (agentkeepalive, form-data) for Node implementation
- CoreResponse for Bun/Deno: new implementations wrap native fetch Response and expose raw(), stream(), and streamNode() behavior (streamNode() throws in Bun/Deno with guidance to use web streams)
- Tests: added unified cross-runtime streaming tests and separate unix-socket tests for Node/Bun/Deno with Docker-socket availability checks; removed old node-only streaming test
- Docs/readme: updated to describe Node, Bun, Deno, and browser support, unix socket behavior per runtime, and new test conventions

2025-11-16 - 4.3.8 - fix(core)

Ensure correct ArrayBuffer return, fix fetch body typing, reorganize node-only tests, and bump tsbuild devDependency

- core_node: Fix arrayBuffer() to ensure an ArrayBuffer is returned (avoid returning SharedArrayBuffer) to improve interoperability when consuming binary responses.
- core_fetch: Cast request body to BodyInit when assigning to fetch options and preserve duplex = 'half' for ReadableStream bodies to satisfy typings and streaming behavior.
- tests: Reorganize tests into Node-only variants (rename/remove multi-platform test files to test.*.node.ts) to separate platform-specific test coverage.
- chore: Bump devDependency @git.zone/tsbuild from ^2.6.8 to ^2.7.1.

2025-11-01 - 4.3.7 - fix(ci)

Update dependencies, add deno.lock, and reorganize tests for browser and Node environments

- Add deno.lock with resolved npm package versions for deterministic Deno/npm usage
- Bump @push.rocks/smartenv dependency to ^6.0.0
- Bump devDependencies: @git.zone/tsbuild -> ^2.6.8, @git.zone/tsrun -> ^1.6.2, @git.zone/tstest -> ^2.7.0
- Reorganize tests: move browser tests to chromium variants and add environment-specific test files for node, bun, deno (streaming, timeout, streamNode, etc.)
- Update package.json dependency ranges to match upgraded lockfile and test tooling

2025-10-26 - 4.3.6 - fix(ci)

Use .npmrc for registry authentication in Gitea workflow and add conditional npmjs publish

- Replace npm config set commands with creating a .npmrc file for Gitea registry authentication in .gitea/workflows/default_tags.yaml
- Add conditional update of .npmrc and publishing to npmjs.org when NPMCI_TOKEN_NPM is provided
- Keep pnpm publish --no-git-checks; improve CI credential handling to be file-based

2025-10-26 - 4.3.5 - fix(workflows)

Remove npmci wrappers from CI workflows and use pnpm/npm CLI directly

- Removed global npmci installation and npmci npm prepare steps from Gitea workflow files
- Use pnpm install/test/build instead of npmci-wrapped commands in test jobs
- Replace npmci command npm config set ... with direct npm config set calls for registry/auth configuration
- Use pnpm publish --no-git-checks for Gitea publishing and use pnpm publish for conditional npmjs publish when token present
- Simplified dependency auditing to run pnpm audit and set registry via npm config set
- Install tsdoc globally and run tsdoc during docs build step (replacing npmci command usage)

2025-10-25 - 4.3.4 - fix(ci)

Fix Gitea workflow publish invocation to run npm publish via npmci command

- Update .gitea/workflows/default_tags.yaml to use 'npmci command npm publish' for the publish step
- Ensures the workflow runs npm publish through the npmci command wrapper to avoid incorrect task invocation

2025-10-25 - 4.3.3 - fix(ci)

Improve Gitea release workflow: install deps, configure Gitea npm registry, and optionally publish to npmjs.org

- Run npm install in the release job to ensure dependencies are available before publishing.
- Configure Gitea/npm registry using GITHUB_SERVER_URL and set auth token for the @ scope.
- Publish to the Gitea npm registry during release.
- If NPMCI_TOKEN_NPM is provided, also publish to the public npmjs.org registry (conditional publish).
- Extract host from GITHUB_SERVER_URL to correctly set the registry auth URL.

2025-10-17 - 4.3.2 - fix(core)

Remove stray console.log from core module

- Removed a stray debug console.log(modulePath) from ts/core/index.ts that printed the module path during Node environment initialization

2025-08-19 - 4.3.1 - fix(core)

Improve streaming support and timeout handling; add browser streaming & timeout tests and README clarifications

- core_fetch: accept Uint8Array and Buffer-like bodies; set fetch duplex for ReadableStream bodies so streaming requests work in environments that require duplex
- core_fetch: implement AbortController-based timeouts and ensure timeouts are cleared on success/error to avoid hanging timers
- core_node: add explicit request timeout handling (request.setTimeout) and hard-data-cutting timeout tracking with proper timeoutId clear on success/error
- client: document that raw(streamFunc) is Node-only (not supported in browsers)
- tests: add browser streaming tests (test/test.streaming.browser.ts) that exercise buffer() and web ReadableStream via stream()
- tests: add timeout tests (test/test.timeout.ts) to validate clearing timers, enforcing timeouts, and preventing timer leaks across multiple requests
- docs: update README streaming section to clarify cross-platform behavior of buffer(), stream(), and raw() methods

2025-08-18 - 4.3.0 - feat(client/smartrequest)

Add streaming and raw buffer support to SmartRequest (buffer, stream, raw); update docs and tests

- Add SmartRequest.buffer(data, contentType?) to send Buffer or Uint8Array bodies with Content-Type header.
- Add SmartRequest.stream(stream, contentType?) to accept Node.js Readable streams or web ReadableStream and set Content-Type when provided.
- Add SmartRequest.raw(streamFunc) to allow custom raw streaming functions (Node.js only) and a RawStreamFunction type.
- Wire Node.js stream handling into CoreRequest by passing a requestDataFunc when creating CoreRequest instances.
- Add comprehensive streaming examples and documentation to README describing buffer/stream/raw usage and streaming methods.
- Add tests for streaming behavior (test/test.streaming.ts) covering buffer, stream, raw, and Uint8Array usage.
- Update client exports and plugins to support streaming features and FormData usage where needed.

2025-08-18 - 4.2.2 - fix(client)

Fix CI configuration, prevent socket hangs with auto-drain, and apply various client/core TypeScript fixes and test updates

- CI/workflow updates: switch container IMAGE to code.foss.global/host.today/ht-docker-node:npmci, adjust NPMCI_COMPUTED_REPOURL, and install @ship.zone/npmci instead of @shipzone/npmci
- Prevent socket hanging by adding automatic draining of unconsumed Node.js response bodies (configurable via options.autoDrain / SmartRequest.autoDrain); added logging when auto-drain runs and updated tests to consume bodies
- Client improvements: fixes and cleanups in SmartRequest (accept header mapping, formData header handling, options(), pagination helpers, handle429Backoff backoff/Retry-After parsing and callbacks, retry logic and small API ergonomics)
- Core fixes: fetch and node implementations corrected (buildUrl, fetch options, request/response constructors, stream conversions to web ReadableStream, proper error messages) and consistent exports
- TypeScript and formatting fixes across many files (consistent trailing commas, object layout, newline fixes, typed function signatures, cleaned up exports and module imports)
- Package metadata and tooling updates: package.json bug/homepage URLs adjusted to code.foss.global, bumped @git.zone/tstest devDependency, added pnpm overrides field; small .gitignore additions

2025-07-29 - 4.2.1 - fix(client)

Fix socket hanging issues and add auto-drain feature

Fixes:

- Fixed socket hanging issues caused by unconsumed response bodies
- Resolved test timeout problems where sockets remained open after tests completed

Features:

- Added automatic response body draining to prevent socket pool exhaustion
- Made auto-drain configurable via `autoDrain()` method (enabled by default)
- Added logging when auto-drain activates for debugging purposes

Improvements:

- Updated all tests to properly consume response bodies
- Enhanced documentation about the importance of consuming response bodies

2025-07-29 - 4.2.0 - feat(client)

Add handle429Backoff method for intelligent rate limit handling

Features:

- Added `handle429Backoff()` method to SmartRequest class for automatic HTTP 429 handling
- Respects `Retry-After` headers with support for both seconds and HTTP date formats
- Configurable exponential backoff when no Retry-After header is present
- Added `RateLimitConfig` interface with customizable retry behavior
- Optional callback for monitoring rate limit events
- Maximum wait time capping to prevent excessive delays

Improvements:

- Updated test endpoints to use more reliable services (jsonplaceholder, echo.zuplo.io)
- Added timeout parameter to test script for better CI/CD compatibility

Documentation:

- Added comprehensive rate limiting section to README with examples

- Documented all configuration options for handle429Backoff

2025-07-29 - 4.1.0 - feat(client)

Add missing options() method to SmartRequest client

Features:

- Added `options()` method to SmartRequest class for setting arbitrary request options
- Enables setting keepAlive and other platform-specific options via fluent API
- Added test coverage for keepAlive functionality

Documentation:

- Updated README with examples of using the `options()` method
- Added specific examples for enabling keepAlive connections
- Corrected all documentation to use `options()` instead of `option()`

2025-07-28 - 4.0.0 - BREAKING CHANGE(core)

Complete architectural overhaul with cross-platform support

Breaking Changes:

- Renamed `SmartRequestClient` to `SmartRequest` for simpler, cleaner API
- Removed legacy API entirely (no more `/legacy` import path)
- Major architectural refactoring:
  - Added abstraction layer with `core_base` containing abstract classes
  - Split implementations into `core_node` (Node.js) and `core_fetch` (browser)
  - Dynamic implementation selection based on environment
- Response streaming API changes:
  - `stream()` now always returns web-style `ReadableStream<Uint8Array>`
  - Added `streamNode()` for Node.js streams (throws error in browser)
- Unified type system with single `ICoreRequestOptions` interface
- Removed all "Abstract" prefixes from type names

Features:

- Full cross-platform support (Node.js and browsers)

- Automatic platform detection using @push.rocks/smartenv
- Consistent API across platforms with platform-specific capabilities
- Web Streams API support in both environments
- Better error messages for unsupported platform features

Documentation:

- Completely rewritten README with platform-specific examples
- Added architecture overview section
- Added migration guide from v2.x and v3.x
- Updated all examples to use the new `SmartRequest` class name

2025-07-27 - 3.0.0 - BREAKING CHANGE(core)

Major architectural refactoring with fetch-like API

Breaking Changes:

- Legacy API functions are now imported from `@push.rocks/smartrequest/legacy` instead of the main export
- Modern API response objects now use fetch-like methods (`.json()`, `.text()`, `.arrayBuffer()`, `.stream()`) instead of direct `.body` access
- Renamed `responseType()` method to `accept()` in modern API
- Removed automatic defaults:
  - No default keepAlive (must be explicitly set)
  - No default timeouts
  - No automatic JSON parsing in core
- Complete internal architecture refactoring:
  - Core module now always returns raw streams
  - Response parsing happens in SmartResponse methods
  - Legacy API is now just an adapter over the core module

Features:

- New fetch-like response API with single-use body consumption
- Better TypeScript support and type safety
- Cleaner separation of concerns between request and response
- More predictable behavior aligned with fetch API standards

Documentation:

- Updated all examples to show correct import paths
- Added comprehensive examples for the new response API
- Enhanced migration guide

2025-04-03 - 2.1.0 - feat(docs)

Enhance documentation and tests with modern API usage examples and migration guide

- Updated README to include detailed examples for the modern fluent API, covering GET, POST, headers, query, timeout, retries, and pagination
- Added a migration guide comparing the legacy API and modern API usage
- Improved installation instructions with npm, pnpm, and yarn examples
- Added and updated test files for both legacy and modern API functionalities
- Minor formatting improvements in the code and documentation examples

2024-11-06 - 2.0.23 - fix(core)

Enhance type safety for response in binary requests

- Updated the dependency versions in package.json to their latest versions.
- Improved type inference for the response body in getBinary method of smartrequest.binaryrest.ts.
- Introduced generic typing to IExtendedIncomingMessage interface for better type safety.

2024-05-29 - 2.0.22 - Documentation

update description

2024-04-01 - 2.0.21 - Configuration

Updated configuration files

- Updated `tsconfig`
- Updated `npmextra.json`: githost

2023-07-10 - 2.0.15 - Structure

Refactored the organization structure

- Switched to a new organization scheme

2022-07-29 - 1.1.57 to 2.0.0 - Major Update

Significant changes and improvements leading to a major version update

- **BREAKING CHANGE**: Switched the core to use ECMAScript modules (ESM)

2018-08-14 - 1.1.12 to 1.1.13 - Functional Enhancements

Enhanced request capabilities and removed unnecessary dependencies

- Fixed request module to allow sending strings
- Removed CI dependencies

2018-07-19 - 1.1.1 to 1.1.11 - Various Fixes and Improvements

Improvements and fixes across various components

- Added formData capability
- Corrected path resolution to use current working directory (CWD)

- Improved formData handling
- Included correct headers
- Updated request ending method

2018-06-19 - 1.0.14 - Structural Fix

Resolved conflicts with file extensions

- Changed `.json.ts` to `.jsonrest.ts` to avoid conflicts

2018-06-13 - 1.0.8 to 1.0.10 - Core Updates

Ensured binary handling compliance

- Enhanced core to uphold latest standards
- Correct binary file handling response
- Fix for handling and returning binary responses

2017-06-09 - 1.0.4 to 1.0.6 - Infrastructure and Type Improvements

Types and infrastructure updates

- Improved types
- Removed need for content type on post requests
- Updated for new infrastructure

Revision #2
Created 2026-03-28 13:09:51 UTC by foss.global Team
Updated 2026-03-29 16:52:18 UTC by foss.global Team