@api.global/typedserver

Documentation for @api.global/typedserver

- readme.md for @api.global/typedserver
- changelog.md for @api.global/typedserver

readme.md for @api.global/typedserver

A powerful TypeScript-first web server framework for building modern full-stack applications. Features static file serving, live reload, type-safe API integration, decorator-based routing, service worker support, and edge computing capabilities. Part of the `@api.global` ecosystem.

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.

□ Features

- □□ **Type-Safe API** — Full TypeScript support with `@api.global/typedrequest` and `@api.global/typedsocket`
- □□ **Decorator Routing** — Clean, expressive routing with `@Route`, `@Get`, `@Post` decorators via smartserve
- □□ **Security Headers** — Built-in CSP, HSTS, X-Frame-Options, and comprehensive security configuration
- ⚡ **Live Reload** — Automatic browser refresh on file changes during development
- □□ **Service Worker** — Advanced caching, offline support, and background sync
- ☁ **Edge Workers** — Cloudflare Workers compatible edge computing with domain routing
- □□ **WebSocket** — Real-time bidirectional communication via TypedSocket
- □□ **SEO Tools** — Built-in sitemap, RSS feed, and robots.txt generation
- □□ **SPA Support** — Single-page application fallback routing
- □□ **PWA Ready** — Web App Manifest generation for progressive web apps
- □□ **Compression** — Automatic Brotli + Gzip response compression
- □□ **Bundled Content** — Serve pre-bundled content from memory for zero-filesystem deployments

□□ Installation

```
# Using pnpm (recommended)
pnpm add @api.global/typedserver

# Using npm
npm install @api.global/typedserver
```

□□ Quick Start

Basic Server

```
import { TypedServer } from '@api.global/typedserver';

const server = new TypedServer({
  serveDir: './public',
  cors: true,
  watch: true,           // Enable file watching
  injectReload: true,    // Inject live reload script
});

await server.start();
console.log('Server running on port 3000!');
```

Full Configuration

```
import { TypedServer } from '@api.global/typedserver';

const server = new TypedServer({
  port: 8080,
  serveDir: './dist',
  cors: true,

  // Development
```

```
  watch: true,
  injectReload: true,
  noCache: true,            // Disable browser caching

  // Production
  forceSsl: true,
  spaFallback: true,        // Serve index.html for client-side routes

  // SEO
  sitemap: true,
  feed: true,
  robots: true,
  domain: 'example.com',
  blockWaybackMachine: false,

  // PWA
  appVersion: 'v1.0.0',
  manifest: {
    name: 'My App',
    short_name: 'myapp',
    start_url: '/',
    display: 'standalone',
    background_color: '#ffffff',
    theme_color: '#000000',
  },

  // Compression
  compression: {
    enabled: true,
    algorithms: ['br', 'gzip'],
    threshold: 1024,
  },
});

await server.start();
```

□□ Routing

TypedServer uses a unified routing system powered by `@push.rocks/smartserve`. You can add routes using decorators or the programmatic API.

Decorator-Based Routing

Create clean, expressive controllers using decorators:

```
import * as smartserve from '@push.rocks/smartserve';

@smartserve.Route('/api/users')
class UserController {
  @smartserve.Get('/')
  async listUsers(ctx: smartserve.IRequestContext): Promise<Response> {
    const users = await getUsersFromDb();
    return new Response(JSON.stringify(users), {
      headers: { 'Content-Type': 'application/json' },
    });
  }

  @smartserve.Get('/:id')
  async getUser(ctx: smartserve.IRequestContext): Promise<Response> {
    const userId = ctx.params.id;
    const user = await getUserById(userId);
    return new Response(JSON.stringify(user), {
      headers: { 'Content-Type': 'application/json' },
    });
  }

  @smartserve.Post('/')
  async createUser(ctx: smartserve.IRequestContext): Promise<Response> {
    const userData = await ctx.json();
    const newUser = await createUserInDb(userData);
    return new Response(JSON.stringify(newUser), {
      status: 201,
      headers: { 'Content-Type': 'application/json' },
    });
  }
}
```

```
// Register the controller
smartserve.ControllerRegistry.registerInstance(new UserController());
```

Programmatic Routes with `addRoute()`

Add routes dynamically using the `addRoute()` API:

```
import { TypedServer } from '@api.global/typedserver';

const server = new TypedServer({ serveDir: './public', cors: true });

// Simple route
server.addRoute('/api/health', 'GET', async (ctx) => {
  return new Response(JSON.stringify({ status: 'ok' }), {
    headers: { 'Content-Type': 'application/json' },
  });
});

// Route with parameters (Express-style :param syntax)
server.addRoute('/api/items/:id', 'GET', async (ctx) => {
  const itemId = ctx.params.id;
  return new Response(JSON.stringify({ id: itemId }), {
    headers: { 'Content-Type': 'application/json' },
  });
});

// Wildcard routes
server.addRoute('/files/*path', 'GET', async (ctx) => {
  const filePath = ctx.params.path;
  return new Response(`Requested: ${filePath}`);
});

await server.start();
```

□□ Type-Safe API Integration

Adding TypedRequest Handlers

```
import { TypedServer } from '@api.global/typedserver';
import * as typedrequest from '@api.global/typedrequest';

// Define your typed request interface
interface IGetUser extends typedrequest.implementsTR<IGetUser> {
  method: 'getUser';
  request: { userId: string };
  response: { name: string; email: string };
}

const server = new TypedServer({ serveDir: './public', cors: true });

// Add a typed handler directly to the server's router
server.typedrouter.addTypedHandler<IGetUser>(
  new typedrequest.TypedHandler('getUser', async (data) => {
    return { name: 'John Doe', email: 'john@example.com' };
  })
);

await server.start();
```

Real-Time WebSocket Communication

TypedServer automatically sets up TypedSocket for real-time communication:

```
import { TypedServer } from '@api.global/typedserver';
import * as typedrequest from '@api.global/typedrequest';

interface IChatMessage extends typedrequest.implementsTR<IChatMessage> {
  method: 'sendMessage';
  request: { text: string; room: string };
  response: { messageId: string; timestamp: number };
}

const server = new TypedServer({ serveDir: './public', cors: true });
```

```
// Handle real-time messages
server.typedrouter.addTypedHandler<IChatMessage>(
  new typedrequest.TypedHandler('sendMessage', async (data) => {
    return { messageId: crypto.randomUUID(), timestamp: Date.now() };
  })
);

await server.start();

// Push messages to connected clients
const connections = await server.typedsocket.findAllTargetConnectionsByTag('allClients');
for (const conn of connections) {
  // Push to specific clients via TypedSocket
}
```

☁ Edge Worker (Cloudflare Workers)

Deploy your application to the edge with Cloudflare Workers:

```
import { EdgeWorker, DomainRouter } from '@api.global/typedserver/edgeworker';

const worker = new EdgeWorker();

// Configure domain routing with caching
worker.domainRouter.addDomainInstruction({
  domainPattern: '*.example.com',
  originUrl: 'https://origin.example.com',
  type: 'cache',
  cacheConfig: { maxAge: 3600 },
});

// Pass-through to origin for API routes
worker.domainRouter.addDomainInstruction({
  domainPattern: 'api.example.com',
  originUrl: 'https://api-origin.example.com',
```

```
  type: 'origin',
});

// Cloudflare Worker entry point
export default {
  fetch: worker.fetchFunction.bind(worker),
};
```

□□ Service Worker Client

Manage service workers in your frontend application:

```
import { getServiceworkerClient } from '@api.global/typedserver/web_serviceworker_client';

// Initialize and register service worker
const swClient = await getServiceworkerClient({
  pollInterval: 30000,  // Poll for updates every 30s
});

// The service worker handles:
// - Cache invalidation from server
// - Offline support
// - Background sync
// - Version updates
```

□□ Bundled Content

Serve pre-bundled content directly from memory — useful for single-binary deployments or embedding assets in server-side code:

```
import { TypedServer } from '@api.global/typedserver';

const server = new TypedServer({
  cors: true,
  bundledContent: [
    {
      path: '/index.html',
```

```
      contentBase64: Buffer.from('<html><body>Hello!</body></html>').toString('base64'),
    },
    {
      path: '/app.js',
      contentBase64: Buffer.from('console.log("loaded")').toString('base64'),
    },
  ],
  spaFallback: true,
});

await server.start();
```

Bundled content takes priority over filesystem serving and supports ETag-based conditional requests with immutable caching.

□□ Security Headers

Configure comprehensive security headers including CSP, HSTS, and more:

```
import { TypedServer } from '@api.global/typedserver';

const server = new TypedServer({
  serveDir: './dist',
  cors: true,

  securityHeaders: {
    // Content Security Policy
    csp: {
      defaultSrc: ["'self'"],
      scriptSrc: ["'self'", "'unsafe-inline'", 'https://cdn.example.com'],
      styleSrc: ["'self'", "'unsafe-inline'"],
      imgSrc: ["'self'", 'data:', 'https:'],
      connectSrc: ["'self'", 'wss:', 'https://api.example.com'],
      fontSrc: ["'self'", 'https://fonts.gstatic.com'],
      frameAncestors: ["'none'"],
      upgradeInsecureRequests: true,
    },
```

```
    // HSTS (HTTP Strict Transport Security)
    hstsMaxAge: 31536000,         // 1 year
    hstsIncludeSubDomains: true,
    hstsPreload: true,

    // Other security headers
    xFrameOptions: 'DENY',
    xContentTypeOptions: true,
    xXssProtection: true,
    referrerPolicy: 'strict-origin-when-cross-origin',

    // Cross-Origin policies
    crossOriginOpenerPolicy: 'same-origin',
    crossOriginEmbedderPolicy: 'require-corp',
    crossOriginResourcePolicy: 'same-origin',

    // Permissions Policy
    permissionsPolicy: {
      camera: [],
      microphone: [],
      geolocation: ['self'],
    },
  },
});

await server.start();
```

Security Headers Reference

Header	Option	Description
`Content-Security-Policy`	`csp`	Controls resources the browser can load
`Strict-Transport-Security`	`hstsMaxAge`, `hstsIncludeSubDomains`, `hstsPreload`	Forces HTTPS connections
`X-Frame-Options`	`xFrameOptions`	Prevents clickjacking attacks
`X-Content-Type-Options`	`xContentTypeOptions`	Prevents MIME-sniffing
`X-XSS-Protection`	`xXssProtection`	Legacy XSS filter
`Referrer-Policy`	`referrerPolicy`	Controls referrer information

Header	Option	Description
`Permissions-Policy`	`permissionsPolicy`	Controls browser features
`Cross-Origin-Opener-Policy`	`crossOriginOpenerPolicy`	Isolates browsing context
`Cross-Origin-Embedder-Policy`	`crossOriginEmbedderPolicy`	Controls cross-origin embedding
`Cross-Origin-Resource-Policy`	`crossOriginResourcePolicy`	Controls cross-origin resource sharing

□□ Compression

TypedServer supports automatic response compression using Brotli and Gzip. Compression is powered by smartserve and enabled by default.

Configuration

```
import { TypedServer } from '@api.global/typedserver';

const server = new TypedServer({
  serveDir: './dist',
  cors: true,

  // Enable with defaults (brotli + gzip, threshold: 1024 bytes)
  compression: true,

  // Or disable completely
  // compression: false,

  // Or configure in detail
  // compression: {
  //   enabled: true,
  //   algorithms: ['br', 'gzip'],    // Preferred order
  //   threshold: 1024,               // Min size to compress (bytes)
  //   level: 4,                      // Compression level (1-11 for brotli, 1-9 for gzip)
  //   exclude: ['/api/stream/*'],    // Skip these paths
  // },
});
```

Compression Options Reference

Option	Type	Default	Description
`enabled`	`boolean`	`true`	Enable/disable compression
`algorithms`	`string[]`	`['br', 'gzip']`	Preferred algorithms in order
`threshold`	`number`	`1024`	Minimum response size (bytes) to compress
`level`	`number`	`4`	Compression level (1-11 for brotli, 1-9 for gzip)
`compressibleTypes`	`string[]`	auto	MIME types to compress
`exclude`	`string[]`	`[]`	Path patterns to skip

□□ Configuration Reference

IServerOptions

Option	Type	Default	Description
`serveDir`	`string`	—	Directory to serve static files from
`bundledContent`	`IBundledContentItem[]`	—	Base64-encoded files to serve from memory
`port`	`number | string`	`3000`	Port to listen on
`cors`	`boolean`	`true`	Enable CORS headers
`watch`	`boolean`	`false`	Watch files for changes
`injectReload`	`boolean`	`false`	Inject live reload script into HTML
`noCache`	`boolean`	`false`	Disable browser caching via response headers
`forceSsl`	`boolean`	`false`	Redirect HTTP to HTTPS
`spaFallback`	`boolean`	`false`	Serve index.html for non-file routes
`sitemap`	`boolean`	`false`	Generate sitemap at `/sitemap`

Option	Type	Default	Description
`feed`	`boolean`	`false`	Generate RSS feed at `/feed`
`robots`	`boolean`	`false`	Serve robots.txt
`domain`	`string`	—	Domain name for sitemap/feeds
`appVersion`	`string`	—	Application version string
`manifest`	`object`	—	Web App Manifest configuration
`publicKey`	`string`	—	SSL certificate
`privateKey`	`string`	—	SSL private key
`defaultAnswer`	`function`	—	Custom default response handler
`feedMetadata`	`object`	—	RSS feed metadata options
`blockWaybackMachine`	`boolean`	`false`	Block Wayback Machine archiving
`securityHeaders`	`ISecurityHeaders`	—	Security headers configuration
`compression`	`ICompressionConfig | boolean`	`true`	Response compression configuration

□□ Package Exports

```
@api.global/typedserver
├── .                          – Main server (TypedServer)
├── /backend                   – Alias for main server
├── /infohtml                  – Info HTML page generator
├── /edgeworker                – Cloudflare Workers edge computing
├── /web_inject                – Live reload script injection
├── /web_serviceworker         – Service Worker implementation
└── /web_serviceworker_client  – Service Worker client utilities
```

□□ Utility Servers

Pre-configured server templates with best practices built-in.

UtilityWebsiteServer

Optimized for modern web applications with SPA support, live reload, and caching disabled by default during development:

```
import { utilityservers } from '@api.global/typedserver';

const websiteServer = new utilityservers.UtilityWebsiteServer({
  serveDir: './dist',
  domain: 'example.com',

  // SPA fallback enabled by default
  spaFallback: true,   // default: true

  // Security headers
  securityHeaders: {
    csp: {
      defaultSrc: ["'self'"],
      scriptSrc: ["'self'", "'unsafe-inline'"],
      styleSrc: ["'self'", "'unsafe-inline'"],
    },
    xFrameOptions: 'SAMEORIGIN',
    xContentTypeOptions: true,
  },

  // Compression (enabled by default)
  compression: true,

  // Other options
  cors: true,          // default: true
  forceSsl: false,     // default: false
  appSemVer: '1.0.0',
  port: 3000,          // default: 3000

  // Optional ads.txt entries (only served if configured)
  adsTxt: [
    'google.com, pub-1234567890, DIRECT, f08c47fec0942fa0',
  ],
```

```
  // RSS feed metadata
  feedMetadata: {
    title: 'My Blog',
    description: 'A cool blog',
    link: 'https://example.com',
  },

  // Add custom routes
  addCustomRoutes: async (typedserver) => {
    typedserver.addRoute('/api/custom', 'GET', async () => {
      return new Response('Custom route!');
    });
  },
});

await websiteServer.start();
```

UtilityServiceServer

Optimized for API services with auto-generated info page:

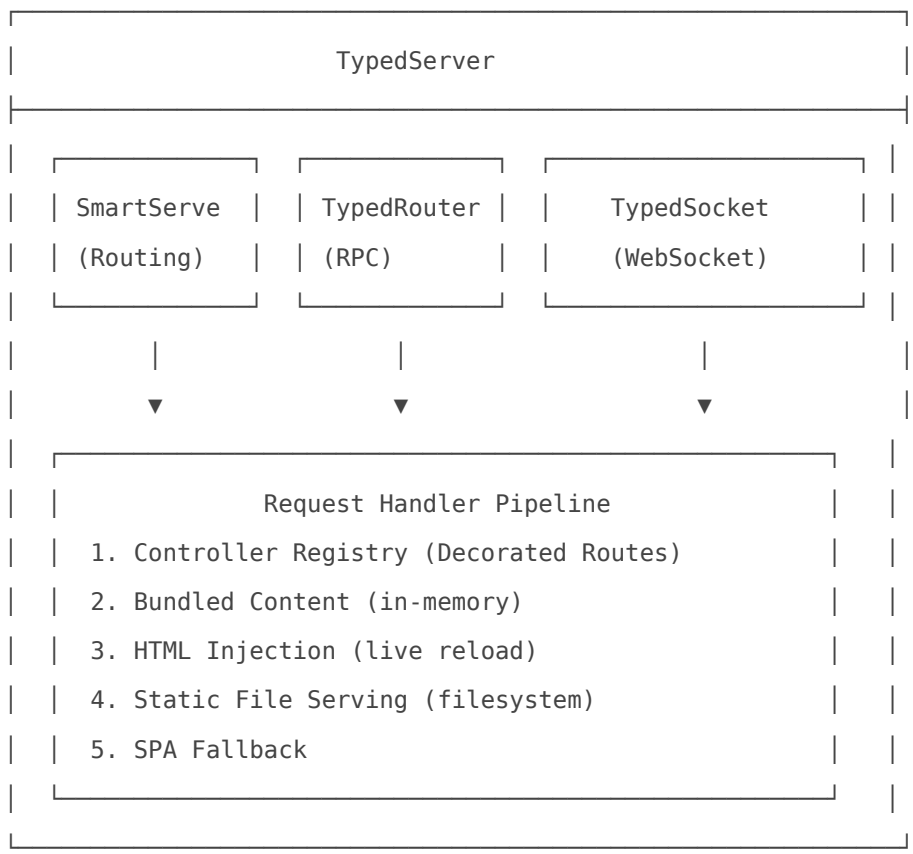
```
import { utilityservers } from '@api.global/typedserver';

const serviceServer = new utilityservers.UtilityServiceServer({
  serviceName: 'My API',
  serviceVersion: '1.0.0',
  serviceDomain: 'api.example.com',
  port: 8080,

  // Add custom routes
  addCustomRoutes: async (typedserver) => {
    typedserver.addRoute('/api/status', 'GET', async () => {
      return new Response(JSON.stringify({ status: 'healthy' }), {
        headers: { 'Content-Type': 'application/json' },
      });
    });
  },
});
```

```
await serviceServer.start();
```

□□ Architecture Overview



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 @api.global/typedserver

2026-03-23 - 8.4.6 - fix(deps)

bump @push.rocks/smartwatch to ^6.4.0

- Updates the @push.rocks/smartwatch dependency from ^6.3.1 to ^6.4.0 in package.json.

2026-03-23 - 8.4.5 - fix(build)

migrate build tooling to tsbuild v4 and tsbundle config while updating sitemap integration

- move tsbundle bundle definitions into npmextra.json and simplify build scripts for tsbuild v4
- replace cross-folder imports from dist_ts_interfaces with ts_interfaces source paths to match automatic path rewriting
- update sitemap generation to the smartsitemap v4 builder API for standard and news sitemaps
- refresh package dependencies and documentation for the new build flow and server capabilities

2026-03-23 - 8.4.4 - fix(utilityservers)

enable file watching whenever a static serve directory is configured

- Decouples directory watching from live-reload injection so watched static assets can update even when reload script injection is disabled.
- Changes UtilityWebsiteServer to set watch based solely on the presence of serveDir.

2026-03-23 - 8.4.3 - fix(websiteserver)

only enable file watching when reload injection is active

- Updates WebsiteServer to disable watch mode when injectReload is turned off, even if serveDir is configured.
- Keeps development watching behavior aligned with the reload feature toggle to avoid unnecessary file watching.

2026-03-03 - 8.4.2 - fix(ts_web_inject)

improve ReloadChecker resilience and TypedSocket handling

- Added retry counters and limits for service worker subscription (MAX_SW_RETRIES) and traffic logging setup (MAX_TRAFFIC_LOGGING_RETRIES) to avoid infinite retry loops
- Made TypedSocket connection non-blocking by connecting in the background (createClient().then(...).catch(...)) so HTTP polling isn't blocked
- Introduced typedsocketConnected flag to track WS state and avoid double-connects; early-return if typedsocket exists
- Adjusted connection lifecycle handling: only trigger reload check when reconnect completes; set backendConnectionLost/state appropriately
- Adaptive polling interval: poll every 5s when WebSocket is not connected, otherwise 120s
- Added safer failure handling for TypedSocket connection with warning logs on errors
- Simplified/changed several logger levels/messages (info -> ok/warn) and removed some noisy info logs

2026-03-03 - 8.4.1 - fix(statuspill)

wait for document.body before appending status pill when script loads before

is parsed; defer via DOMContentLoaded or requestAnimationFrame

- Guard against missing document.body to avoid errors when the script runs before the is parsed
- Retry showing on DOMContentLoaded if the document is still loading

- Fallback to requestAnimationFrame to schedule the show on the next frame if DOM is already parsed

2026-02-24 - 8.4.0 - feat(utilityservers)

add injectReload and noCache options and enable dev features by default

- Adds optional configuration properties 'injectReload' and 'noCache' to the website server options interface.
- Dev features (injectReload and noCache) are no longer only enabled when serveDir is set; they now default to true when not explicitly provided.
- This changes default runtime behavior: live-reload injection and disabled browser caching may be enabled for servers that previously did not have them — consumers should set options explicitly to preserve previous behavior.

2026-02-24 - 8.3.1 - fix(typedserver)

no changes detected — no version bump needed

- No files changed in the diff
- Current package version: 8.3.0

2026-01-23 - 8.3.0 - feat(typedserver)

add noCache option to disable client-side caching and set no-cache headers on responses

- Introduces an optional noCache?: boolean in the server options interface
- applyResponseHeaders now sets Cache-Control, Pragma, and Expires headers when noCache is true
- Existing CORS and security header behavior unchanged when noCache is not set or false

2026-01-23 - 8.2.0 - feat(typedserver)

serve bundled in-memory content with caching and reload injection

- Add IBundledContentItem and IServerOptions.bundledContent to accept tsbundle output (path + contentBase64).
- Initialize in-memory bundled content map with MIME type detection, SHA-256-based ETag, size and combined app hash.
- Serve bundled files with proper Content-Type, ETag, Cache-Control, support HEAD and conditional 304 responses.
- Inject live-reload script into bundled HTML responses when injectReload is enabled; modify SPA fallback to prefer bundled index.html.
- Require serveDir or bundledContent when injectReload is enabled; prefer in-memory bundled content over filesystem requests.
- Add helper methods: initializeBundledContent, getMimeType, serveBundledContent and serveBundledHtmlWithInjection; log initialization.

2025-12-22 - 8.1.0 - feat(types)

export IRequestContext type from @push.rocks/smartserve for consumers to use in route handlers

- Adds a type export: IRequestContext from @push.rocks/smartserve
- Type-only change — no runtime or behavioral changes

2025-12-20 - 8.0.0 - BREAKING CHANGE(typedserver)

migrate route handlers to use IRequestContext and lazy body parsers

- Route handlers now receive plugins.smartserve.IRequestContext instead of Request (breaking API change). addRoute no longer wraps handlers to convert context → Request.
- createContext() is now synchronous and provides lazy body accessors: ctx.json(), ctx.text(), ctx.arrayBuffer(), ctx.formData(); ctx.body property was removed.
- DevToolsController constructor now accepts optional options and supplies no-op defaults so controllers can be auto-instantiated without args.

- TypedRequest controller now reads the request via await ctx.json() and forwards typed requests accordingly.
- Utility website server handlers and other internal callsites updated to use ctx.params and the new context API.
- Tests updated to the new TypedServer API, improved assertions, changed test port and reduced delays, and switched tap runner export to default.
- Bumped dependency @push.rocks/smartserve to ^2.0.1 to match API changes.
- npmextra.json reorganized git.zone/tsdoc entries and added release registries and @ship.zone/szci metadata.

2025-12-08 - 7.11.1 - fix(dependencies)

Upgrade dependencies: bump @design.estate/dees-catalog to v3.1.1 and @push.rocks/smartwatch to v6.0.0; update migration notes in readme.hints.md

- package.json: @design.estate/dees-catalog updated from ^2.0.3 to ^3.1.1 (includes new icons, components and DeesIcon unified icon property; legacy iconFA deprecated)
- package.json: @push.rocks/smartwatch updated from ^5.0.0 to ^6.0.0 (cross-runtime support, native fs.watch, API compatibility maintained: new Smartwatch class methods and events documented)
- readme.hints.md: added migration notes for smartwatch v6 and dees-catalog v3, plus other dependency update summaries

2025-12-08 - 7.11.0 - feat(typedserver)

Add configurable response compression (Brotli + Gzip) with defaults enabled and documentation

- Expose a new compression option on IServerOptions (plugins.smartserve.ICompressionConfig | boolean).
- Pass the compression setting through to SmartServe (smartServeOptions.compression = this.options.compression).
- Add compression option to UtilityWebsiteServer and forward it when creating SmartServe options.
- Update README: new Compression section with global config examples, per-route decorator usage, and options reference.

- Add a small readme.todo.md with service worker wake/reload TODO notes.

2025-12-05 - 7.10.2 - fix(docs)

Update README with routing examples and utility server config; bump @cloudflare/workers-types and @push.rocks/smartserve versions

- Bumped dependency @cloudflare/workers-types to ^4.20251205.0
- Bumped dependency @push.rocks/smartserve to ^1.3.0
- Expanded README: added decorator-based routing examples (Route/Get/Post) using smartserve
- Added programmatic routing examples (addRoute) and SPA/wildcard route samples
- Enhanced UtilityWebsiteServer and UtilityServiceServer docs: default port, ads.txt, feedMetadata, addCustomRoutes example and other config options
- Clarified security headers descriptions and configuration reference
- Updated Quick Start console message to show running port ("Server running on port 3000!")
- Documented EdgeWorker/domain routing caching example and noted service worker version update behavior
- Adjusted TypedSocket example tag to use 'allClients' in README

2025-12-05 - 7.10.1 - fix(typedserver)

Use smartserve ControllerRegistry for custom routes and remove custom route parsing

- addRoute now delegates to plugins.smartserve.ControllerRegistry instead of building its own regex-based matcher
- Backwards compatibility: incoming smartserve IRequestContext is converted to a Request and ctx.params is attached to request.params before invoking the handler
- Removed internal IRegisteredRoute, customRoutes storage, and parseRouteParams helper
- Request handling now uses ControllerRegistry.matchRoute and registered controllers are compiled via ControllerRegistry.compileRoutes()

2025-12-05 - 7.10.0 - feat(website-server)

Add configurable ads.txt support to website server

- Introduce adsTxt?: string[] option to the server options to allow configuring ads.txt entries.
- Serve /ads.txt only when adsTxt is provided; the route is not registered if no entries are configured.
- Replace previous hard-coded Google ads.txt entry with values joined from the provided adsTxt array and served as text/plain.
- Preserves existing behavior when adsTxt is not set (no /ads.txt endpoint will be exposed).

2025-12-05 - 7.9.0 - feat(typedserver)

Add configurable security headers and default SPA behavior

Introduce structured security headers support (CSP, HSTS, X-Frame-Options, COOP/COEP/CORP, Permissions-Policy, Referrer-Policy, X-XSS-Protection, etc.) and apply them to responses and OPTIONS preflight. Expose configuration via the server API and document usage. Also update UtilityWebsiteServer defaults (SPA fallback enabled by default) and related docs.

- Add ISecurityHeaders and IContentSecurityPolicy TypeScript interfaces to configure CSP, HSTS and other security-related headers.
- Implement buildCspHeader to serialize CSP config and applyResponseHeaders to add CORS and all configured security headers to outgoing responses.
- Apply security headers to OPTIONS preflight responses and all other responses by default when securityHeaders option is provided.
- Add securityHeaders option to IServerOptions and wire it through TypedServer and UtilityWebsiteServer constructors.
- Update UtilityWebsiteServer: renamed template to UtilityWebsiteServer, enable SPA fallback by default, expose options (cors, spaFallback, securityHeaders, forceSsl, port, feedMetadata, etc.) and forward them into the TypedServer instance.
- Documentation: add Security Headers section and example usage to readme.md; document the UtilityWebsiteServer defaults and example.
- Ensure CORS headers are only added when cors option is enabled.

2025-12-05 - 7.8.18 - fix(readme)

Update README to reflect new features and updated examples (SPA/PWA/Edge/ServiceWorker) and clarify API usage

- Rewrite project introduction and features list to highlight Service Worker, Edge Workers, SPA support, and PWA readiness
- Replace and expand example sections: Basic Server, Full Configuration, TypedRequest handlers, WebSocket usage, Edge Worker entrypoint, and Service Worker client usage
- Update configuration reference: remove legacy compression flags, add spaFallback, defaultAnswer, feedMetadata, and blockWaybackMachine options
- Document package exports and add examples for utility servers (WebsiteServer, ServiceServer)
- Clarify TypedRequest/TypedSocket usage by showing server.typedrouter and service worker client initializer (getServiceworkerClient)

2025-12-04 - 7.8.11 - fix(web_inject)

Improve logging in web injection (TypedRequest) and update dees-comms dependency

- Add debug logging to ts_web_inject to explicitly filter serviceworker_* methods and avoid infinite loops
- Log incoming TypedRequest methods for better visibility during debugging
- Bump dependency @design.estate/dees-comms from ^1.0.27 to ^1.0.28

2025-12-04 - 7.8.0 - feat(serviceworker)

Add TypedRequest traffic monitoring and SW dashboard 'Requests' panel

- Add TypedRequest traffic monitoring interfaces and shared SW dashboard HTML (SW_DASH_HTML) to ts_interfaces/serviceworker.ts.
- Introduce RequestLogStore (ts_web_serviceworker/classes.requestlogstore.ts) to collect, persist in-memory, and compute stats for TypedRequest traffic (logs, counts, methods, averages).
- Add service worker backend handlers to receive and broadcast TypedRequest logs and to expose endpoints: serviceworker_typedRequestLog, serviceworker_getTypedRequestLogs, serviceworker_getTypedRequestStats, serviceworker_clearTypedRequestLogs.
- Expose HTTP routes and fallback behaviors in the server built-in controller to serve the SW dashboard (GET /sw-dash and /sw-dash/bundle.js) and to return sensible 503 placeholders when the SW is not active.

- Extend the service worker CacheManager and DashboardGenerator to serve TypedRequest-related endpoints (/sw-dash/requests, /sw-dash/requests/stats, /sw-dash/requests/methods) and to integrate RequestLogStore data into the dashboard APIs.
- Add a Lit-based dashboard component sw-dash-requests (ts_swdash/sw-dash-requests.ts) and integrate it into the main sw-dash-app UI to display live TypedRequest traffic with filtering, payload toggles, pagination and clear logs action.
- Enable client-side traffic logging from the injected reload checker (ts_web_inject/index.ts) by setting global TypedRouter hooks that send log entries to the service worker, with safeguards to avoid logging the logging requests themselves.
- Add action manager utilities (ts_web_serviceworker_client/classes.actionmanager.ts) to log TypedRequest entries to the SW, query logs and stats, clear logs, and subscribe to real-time TypedRequest broadcasts.
- Refactor Dashboard HTML generation to use the shared SW_DASH_HTML constant so server and service worker serve the same UI shell.
- Integrate broadcasting of TypedRequest log events from service worker backend to connected clients so the SW dashboard updates in real time.

2025-12-04 - 7.7.1 - fix(web_serviceworker)

Standardize DeesComms message format in service worker backend

- Add createMessage helper to generate consistent TypedRequest-shaped messages (includes messageId and correlation.id/phase).
- Replace inline postMessage payloads with createMessage(...) calls across ServiceworkerBackend (status updates, new-version broadcasts, alerts, event pushes, metrics updates, resource-cached notifications).
- Improves message consistency and enables easier correlation/tracing of DeesComms messages; behavior should remain backward-compatible.

2025-12-04 - 7.7.0 - feat(typedserver)

Add SPA fallback support to TypedServer

- Introduce new IServerOptions.spaFallback boolean to enable SPA routing fallback.

- When enabled, GET requests for paths without a file extension will serve serveDir/index.html.
- Preserves existing HTML injection behavior: injectReload still injects devtools script and typedserver metadata into when enabled.
- Responses from SPA fallback include Cache-Control: no-cache and appHash header for cache-busting; falls through to 404 on errors.
- Non-file routes that contain a dot (.) are not considered for SPA fallback to avoid interfering with asset requests.

2025-12-04 - 7.6.0 - feat(typedserver)

Remove legacy Express-based servertools, drop express deps, and refactor TypedServer to SmartServe + typedrouter with CORS support

- Remove legacy ts/servertools module and many Express-based helpers (classes.server, handler, handlerstatic, handlerproxy, compressor, sitemap, feed, tools.*). The servertools compatibility layer is no longer available.
- Drop express-related dependencies from package.json (@types/express, express, body-parser, cors, express-force-ssl).
- Refactor core API: ts/index.ts no longer exports servertools and ts/plugins.ts no longer re-exports Express middleware — consumers must migrate to SmartServe/typedrequest/typedsocket primitives.
- TypedServer rewritten: integrates with @push.rocks/smartserve ControllerRegistry, adds custom route parsing, CORS header helper and OPTIONS preflight handling, improved static file handling with optional reload injection, file watching, typedrouter and typedsocket integration.
- UtilityWebsiteServer now registers the serviceworker versionInfo handler on typedrouter instead of using the removed servertools.serviceworker helper.
- This is a breaking change — public APIs and dependency surface changed; bump major version.

2025-12-04 - 7.5.0 - feat(serviceworker)

Add real-time service worker push updates and DeesComms integration (metrics, events, resource caching)

- Integrate DeesComms push channel for real-time SW → client communication and export/consume deesComms in relevant plugin modules.
- Add typed push message interfaces for events, metrics snapshots and resource-cached notifications in serviceworker interfaces.
- Implement backend push methods: pushEvent, pushMetricsUpdate (with 500ms throttle) and pushResourceCached in ServiceworkerBackend.
- Trigger push updates from MetricsCollector and PersistentStore so metrics and logged events are broadcast to connected clients.
- Add client-side DeesComms handlers in sw-dash app: receive metrics, event logs and resource notifications; add heartbeat and initial HTTP seed to maintain SW health state.
- Add event push listener and cleanup in sw-dash-events component to prepend incoming events and avoid leaks.
- Expose getServiceWorkerBackend() from SW init for internal modules to call push methods.
- Misc: implement request deduplication and various robustness improvements (throttling, heartbeat, safer polling, removed noisy debug logs).

2025-12-04 - 7.4.1 - fix(web_serviceworker)

Improve service worker persistence, metrics and caching robustness

- Ensure persistent store is initialized before use in dashboard handlers and service worker activation/handlers (calls to persistentStore.init())
- Make serveCumulativeMetrics async and align fetchEvent.respondWith usage (remove unnecessary Promise.resolve)
- Change persistent WebStore database name to 'losslessServiceworkerPersistent' to separate durable store from runtime store
- Make PersistentStore.init() more resilient: add detailed logging, avoid throwing on init failure, mark initialized to prevent retry loops, and start periodic save only after load
- Ensure logEvent awaits initialization and adds defensive logging around reading/writing the event log
- Add request deduplication logic and improved cache handling in CacheManager (fetchWithDeduplication usage and safer respondWith)

2025-12-04 - 7.4.0 - feat(serviceworker)

Add persistent event store, cumulative metrics and dashboard events UI for service worker observability

- Add PersistentStore (ts_web_serviceworker/classes.persistentstore.ts) to persist event log and cumulative metrics with retention policy and periodic saving.
- Introduce persistent event types and interfaces for event log and cumulative metrics (ts_interfaces/serviceworker.ts).
- Log lifecycle, update, network and speedtest events to the persistent store (install, activate, update available/applied/error, network online/offline, speedtest started/completed/failed, cache invalidation).
- Expose persistent-store APIs via typed handlers in the service worker backend: serviceworker_getEventLog, serviceworker_getCumulativeMetrics, serviceworker_clearEventLog, serviceworker_getEventCount.
- Serve new dashboard endpoints from the service worker: /sw-dash/events (GET), /sw-dash/events/count (GET), /sw-dash/cumulative-metrics (GET) and DELETE /sw-dash/events to clear the log (handled in classes.cachemanager and classes.dashboard).
- Add sw-dash events panel component (ts_swdash/sw-dash-events.ts) and integrate an Events tab into the dashboard UI (ts_swdash/sw-dash-app.ts, sw-dash-overview.ts shows 1h event count).
- Reset cumulative metrics on cache invalidation and increment swRestartCount on PersistentStore.init().
- Record speedtest lifecycle events (started/completed/failed) and include details in the event log.

2025-12-04 - 7.3.0 - feat(serviceworker)

Modernize SW dashboard UI and improve service worker backend and server tooling

- Revamped sw-dash UI: new header/logo, uptime badge, live auto-refresh indicator, reorganized panels and improved speedtest UI and controls
- Shared styles overhaul: new theming variables, spacing scale, badges, refined progress/pulse animations and cleaner typography
- Dashboard internals: metrics endpoint and SPA shell updated; Lit bundle loading and table sort icon changed to ↑/↓
- Service worker: added request deduplication (in-flight request coalescing), safer caching logic, consistent CORS/caching headers, and cache revalidation
- Metrics: richer MetricsCollector with per-resource tracking, domain/content-type stats, speedtest metrics and better summary/stat helpers
- Update & network managers: rate-limited update checks, debounced update/revalidation tasks, online/offline checks and improved retry/backoff logic

- TypedServer & tooling: addRoute API for custom routes, improved HTML reload script injection, TypedSocket integration and a backend speedtest handler
- servertools: improved static/proxy handlers (more robust path extraction, compression handling) and deprecation notice for addTypedSocket()

2025-12-04 - 7.2.0 - feat(serviceworker)

Add service worker status updates, EventBus and UI status pill for realtime observability

- Introduce a status update protocol for service worker <-> clients (IStatusUpdate, IMessage_Serviceworker_StatusUpdate, IRequest_Serviceworker_GetStatus).
- Add typedserver-statuspill Lit component to display backend/serviceworker/network status in the UI, with expand/collapse details and persistent/error states.
- Wire ReloadChecker to use the new status pill: show network/backend/serviceworker status, handle online/offline events, and subscribe to service worker status broadcasts.
- Extend ActionManager (client) with subscribeToStatusUpdates and getServiceWorkerStatus helpers; forward serviceworker_statusUpdate broadcasts to registered callbacks.
- Serviceworker backend: add serviceworker_getStatus handler and broadcastStatusUpdate API; subscribe to EventBus lifecycle/network/update events to broadcast status changes to clients.
- Add EventBus for decoupled service worker internal events (ServiceWorkerEvent enum, pub/sub API, history and convenience emitters).
- Ensure proper subscribe/unsubscribe lifecycle (ReloadChecker stops SW subscription on stop).
- Improve cache/connection status reporting integration so status updates include details like cacheHitRate, resourceCount and connected clients.

2025-12-04 - 7.1.0 - feat(swdash)

Add live speedtest progress UI to service worker dashboard

- Introduce reactive speedtest state (phase, progress, elapsed) in sw-dash-overview component
- Start a progress interval to animate overall test progress and estimate phases (latency, download, upload)
- Dispatch 'speedtest-complete' event and show a brief complete state before resetting UI
- Add helper methods for phase labels and elapsed time formatting

- Add CSS for progress bar, shimmer animation and phase pulse to sw-dash-styles

2025-12-04 - 7.0.0 - BREAKING CHANGE(serviceworker)

Move serviceworker speedtest to time-based chunked transfers and update dashboard/server contract

- Change speedtest protocol to time-based chunk transfers: new request types 'download_chunk' and 'upload_chunk' plus 'latency'. Clients should call chunk requests in a loop for the desired test duration.
- IRequest_Serviceworker_Speedtest interface updated: request fields renamed/changed (chunkSizeKB, payload) and response no longer includes durationMs or speedMbps — server now returns bytesTransferred, timestamp, and optional payload.
- TypedServer speedtest handler updated to support 'download_chunk' and 'upload_chunk' semantics and to return bytesTransferred/timestamp/payload only (removed server-side duration/speed calculation).
- Dashboard runSpeedtest now performs time-based tests (TEST_DURATION_MS = 5000, CHUNK_SIZE_KB = 64) by repeatedly requesting chunks and computing throughput on the client side.
- Documentation/comments updated to clarify new speedtest behavior and default chunk sizes.

2025-12-04 - 6.8.1 - fix(web_serviceworker)

Move service worker initialization to init.ts and remove exports from service worker entrypoint to avoid ESM bundle output

- Remove exports from ts_web_serviceworker/index.ts so the service worker entrypoint does not export symbols (prevents tsbundle from producing ESM output).
- Add ts_web_serviceworker/init.ts which initializes the ServiceWorker instance and exports getServiceWorkerInstance() for internal imports.
- Update ts_web_serviceworker/classes.dashboard.ts to import getServiceWorkerInstance from init.ts instead of index.ts.

2025-12-04 - 6.8.0 - feat(swdash)

Add SW-Dash (Lit-based service worker dashboard), bundle & serve it; improve servertools and static handlers

- Add a new sw-dash frontend (ts_swdash) implemented with Lit: sw-dash-app, sw-dash-overview, sw-dash-urls, sw-dash-domains, sw-dash-types, sw-dash-table, shared styles and plugin shims.
- Wire sw-dash into build pipeline and packaging: add ts_swdash bundle to npm build script and include ts_swdash in package files.
- Serve the dashboard bundle: add paths (swdashBundleDir / swdashBundlePath) and a built-in route (/sw-dash/bundle.js) in BuiltInRoutesController.
- Simplify service-worker dashboard HTML output to a minimal shell that mounts and loads the module /sw-dash/bundle.js (reduces inline HTML/CSS/JS duplication).
- Lazy-load service worker bundle and source map in servertools.tools.serviceworker and expose /sw-typedrequest endpoints for SW typed requests (including speedtest handler).
- Enhance compression utilities and static serving: Compressor now caches compressed results, prioritizes preferred compression methods, provides safer zlib calls, and exposes createCompressionStream; HandlerStatic gained improved path resolution, Express 5 wildcard handling and optional compression flow.
- Improve proxy/static handler path handling to be compatible with Express 5 wildcard parameters and more robust fallback logic.
- Deprecate Server.addTypedSocket (no-op) and document recommended SmartServe/TypedServer integration for WebSocket support.
- Various minor packaging/path updates (paths.ts, plugins exports) to support the new dashboard and bundles.

2025-12-04 - 6.7.0 - feat(web_serviceworker)

Add per-resource metrics and request deduplication to service worker cache manager

- Introduce per-resource tracking in metrics: ICachedResource, IDomainStats, IContentTypeStats and a resourceStats map.
- Add MetricsCollector.recordResourceAccess(...) to record hits/misses, content-type and size; provide getters: getCachedResources, getDomainStats, getContentTypeStats and getResourceCount.
- Reset resourceStats when metrics are reset and limit resource entries via cleanupResourceStats to avoid memory bloat.

- Add request deduplication in CacheManager (fetchWithDeduplication) to coalesce identical concurrent fetches and a periodic safety cleanup for in-flight requests.
- Record resource accesses on cache hit and when storing new cache entries (captures content-type and body size).
- Expose a dashboard resources endpoint (/sw-dash/resources) served by the SW dashboard to return detailed resource data for SPA views.

2025-12-04 - 6.6.0 - feat(web_serviceworker)

Enable service worker dashboard speedtests via TypedSocket, expose ServiceWorker instance to dashboard, and add server-side speedtest handler

- Add `serviceworker_speedtest` typed handler in TypedServer to support download/upload/latency tests from service workers
- Export `getServiceWorkerInstance` from the web_serviceworker entrypoint so other modules (dashboard) can access the running ServiceWorker instance
- Make ServiceWorker.typedsocket and ServiceWorker.typedrouter public to allow the dashboard to create and fire TypedSocket requests
- Update dashboard to run latency, download and upload tests over TypedSocket instead of POSTing to /sw-typedrequest
- Deprecate legacy servertools.Server.addTypedSocket (now a no-op) and recommend using TypedServer with SmartServe integration for WebSocket support

2025-12-04 - 6.5.0 - feat(serviceworker)

Add server-driven service worker cache invalidation and TypedSocket integration

- TypedServer: push cache invalidation messages to service worker clients (tagged 'serviceworker') before notifying frontend clients on reload
- Service Worker: connect to TypedServer via TypedSocket; handle 'serviceworker_cacheInvalidate' typed request to clean caches and trigger client reloads
- Web inject: add fallback to clear caches via the Cache API when global service worker helper is not available
- Interfaces: add IRequest_Serviceworker_CacheInvalidate typedrequest interface
- Plugins: export typedsocket in web_serviceworker plugin surface

- Service worker connection: retry logic and improved logging for TypedSocket connection attempts

2025-12-04 - 6.4.0 - feat(serviceworker)

Add speedtest support to service worker and dashboard

- Add serviceworker_speedtest typed request handler to measure download, upload and latency
- Expose dashboard speedtest endpoint (/sw-dash/speedtest) and integrate runSpeedtest flow
- Dashboard UI: add speedtest panel, run button, visual speed bars and online indicator
- Metrics: introduce ISpeedtestMetrics and methods (recordSpeedtest, setOnlineStatus, getSpeedtestMetrics) and include speedtest data in metrics output
- Server/tools: add typedrequest handling for speedtest in sw-typedrequest and route service worker dashboard path in CacheManager

2025-12-04 - 6.3.0 - feat(web_serviceworker)

Add advanced service worker subsystems: cache deduplication, metrics, update & network managers, event bus and dashboard

- CacheManager: request deduplication for concurrent fetches, safer caching (preserve CORS headers), periodic in-flight cleanup and full cache cleaning API
- Fetch handling: improved handling for same-origin vs cross-origin requests, more robust 500 debug responses when upstream fetch fails
- UpdateManager: rate-limited update checks, offline grace period, debounced update and cache revalidation tasks, forceUpdate logic and persisted version/cache timestamps
- NetworkManager: online/offline detection, retry/backoff, request timeouts and more resilient makeRequest implementation
- EventBus: singleton pub/sub with history, once/onMany/onAll helpers and convenience emitters for cache/network/update events
- MetricsCollector: comprehensive metrics for cache, network, updates and connections with helper methods and JSON/HTML dashboard endpoints (/sw-dash, /sw-dash/metrics)

- ErrorHandler & ServiceWorkerError: structured error types, severity, context, history and helper APIs for consistent error reporting
- ServiceWorker & backend: improved install/activate flows, clients.claim(), cache cleaning on activation, backend APIs to purge cache and trigger reloads/notifications
- TypedServer / servertools: addRoute path pattern parsing (named params & wildcards), safer HTML injection for reload script, TypedRequest controller and service worker route helpers
- Various safety and compatibility improvements (response cloning, header normalization, cache-control decisions, and fallback behaviors)

2025-12-04 - 6.2.0 - feat(web_serviceworker)

Add service-worker dashboard and request deduplication; improve caching, metrics and error handling

- Add DashboardGenerator to serve an interactive terminal-style dashboard at /sw-dash and a metrics JSON endpoint at /sw-dash/metrics
- Introduce request deduplication in CacheManager to coalesce concurrent network fetches and avoid duplicate requests
- Add periodic cleanup for in-flight request tracking to prevent unbounded memory growth
- Improve caching flow: preserve response headers (excluding cache-control headers), ensure CORS headers and Cross-Origin-Resource-Policy, and store response bodies as blobs to avoid locked stream issues
- Provide clearer 500 error HTML responses for failed fetches to aid debugging
- Integrate metrics and event emissions for network and cache operations (record request success/failure, cache hits/misses, and emit corresponding events)

2025-12-04 - 6.1.0 - feat(web_serviceworker)

Enhance service worker subsystem: add metrics, event bus, error handling, config and caching/update improvements; make client connection & polling robust

- Introduce MetricsCollector (cache, network, update, connection) for runtime observability and APIs to retrieve metrics

- Add EventBus singleton to emit/subscribe to internal SW events (cache hits/misses, network events, update lifecycle, connection events)
- Add ErrorHandler and ServiceWorkerError types for consistent error classification and tracking
- Add ServiceWorkerConfig with defaults and WebStore persistence to centralize SW settings (cache, update, network, blocked/cacheable domains)
- CacheManager: implement request deduplication (in-flight request coalescing), periodic in-flight cleanup, record cache hit/miss metrics and safer cache storing (headers/body handling)
- UpdateManager: rate-limited and concurrency-safe update checks, improved stale-cache handling, event emissions, debounced update and revalidation tasks, and metrics recording
- NetworkManager: enhanced online/offline detection and robust request retries/timeouts/backoff handling
- ServiceworkerBackend: improved client reload logic and notification handling via DeesComms and clients API
- Serviceworker client-side: ActionManager.waitForServiceWorkerConnection now returns a structured result with timeout/retries/backoff; ServiceworkerClient gains controllable polling (AbortController), visibility-based pause/resume, manual trigger and lifecycle cleanup
- Expose serviceworker bundle routes at both nested and root paths (/serviceworker/*splat and /serviceworker.bundle.js(.map)) in servertools
- Add/extend typed interfaces for serviceworker metrics and connection results

2025-12-04 - 6.0.1 - fix(web_inject)

Use TypedSocket status API in web_inject and bump dependencies

- ts_web_inject: switch from typedsocket.addTag + eventSubject to await typedsocket.setTag + statusSubject; update logging and handle 'reconnecting' status as backend connection loss
- Await setTag call to ensure tag is applied before relying on socket state
- Bump dependencies: @api.global/typedrequest -> ^3.1.11, @api.global/typedsocket -> ^4.1.0, @push.rocks/smartserve -> ^1.1.2

2025-12-03 - 6.0.0 - BREAKING CHANGE(servertools.Server.addTypedSocket)

Deprecate Server.addTypedSocket and upgrade typedsocket to v4; make addTypedSocket a no-op and log a deprecation warning. Bump tsbundle devDependency.

- Upgrade dependency @api.global/typedsocket to ^4.0.0. TypedSocket v4 no longer supports attaching to an existing Express server.
- Deprecate servertools.Server.addTypedSocket(): the method is now a no-op and emits a console.warn directing users to use TypedServer with SmartServe integration for WebSocket support.
- Bump devDependency @git.zone/tsbundle to ^2.6.3.
- Breaking change: any consumer code that relied on addTypedSocket to attach a WebSocket server to an existing Express instance will need to migrate to the new SmartServe/TypedServer integration.

2025-12-02 - 5.0.0 - BREAKING CHANGE(devtools)

Switch /reloadcheck endpoint from GET to POST in DevToolsController

- Updated ts/controllers/controller.devtools.ts: decorator changed from @plugins.smartserve.Get('/reloadcheck') to @plugins.smartserve.Post('/reloadcheck').
- Clients that previously performed GET requests against /reloadcheck must be updated to use POST. This is a breaking API change.
- Bump major version to reflect the change in the public HTTP API.

2025-12-02 - 4.1.1 - fix(classes.typedserver)

Instantiate and register DevToolsController only when injectReload is enabled; compile ControllerRegistry routes after registration

- DevToolsController is now created and registered only if options.injectReload is true to avoid unnecessary/invalid registrations when live reload is disabled.
- ControllerRegistry.compileRoutes() is invoked after registering controllers to precompile decorated routes for faster route matching.

2025-12-02 - 4.1.0 - feat(TypedServer)

Integrate SmartServe controller routing; add built-in routes controller and refactor TypedServer to use controllers and FileServer

- Add BuiltInRoutesController exposing /robots.txt, /manifest.json, /sitemap, /sitemap-news, /feed and /appversion
- Refactor TypedRequestHandler into a SmartServe-decorated TypedRequestController and register it with ControllerRegistry
- Refactor TypedServer to use SmartServe: register controller instances, use ControllerRegistry matching, and delegate WebSocket integration to SmartServe
- Introduce FileServer-based static serving with HTML reload script injection and improved default root handling
- Expand supported HTTP methods to include HEAD and OPTIONS
- Remove legacy FeedHelper and consolidate sitemap/feed handling into controllers and helpers
- Enhance servertools legacy Express utilities: improved HandlerProxy, HandlerStatic, Compressor with caching and preferred compression support
- Service worker subsystem improvements: CacheManager, NetworkManager, UpdateManager and backend enhancements for robust caching, revalidation and client reloads
- Web-inject LitElement properties switched from private fields to accessor syntax (typedserver_web.infoscreen)

2025-12-02 - 4.0.0 - BREAKING CHANGE(typedserver)

Migrate to new push.rocks packages and async smartfs API; replace smartchok with smartwatch; update deps and service worker handling

- Major dependency updates: @push.rocks packages upgraded (smartfile -> v13, smartfs added v1.2.0, smartenv v6, smartrequest v5, smartwatch v5, webrequest v4), Express and dev-tooling bumped.
- Replace sync smartfile APIs with async SmartFs instance (plugins.fsInstance). All file reads now use async smartfs calls.
- smartfile v13 migration: removed sync fs helpers; added plugins.fsInstance (SmartFs + Node provider).

- Renamed file watcher usage: Smartchok -> Smartwatch and property names updated (smartwatchInstance).
- WebRequest class rename handled: WebRequest -> WebrequestClient (webrequest v4).
- Service worker bundle is lazy-loaded (avoid startup sync file reads) and added routes for bundle and source map.
- Service worker & SW-related improvements: more robust caching, enhanced cache headers, offline handling, revalidation, and update forcing logic.
- createServeDirHash now uses smartfs.directory().recursive().treeHash() and truncates hash to 12 chars; fallback hash logic retained.
- HandlerStatic, HandlerProxy and route handling hardened for Express 5 wildcard params (array/various shapes supported).
- Various runtime improvements: safer start/stop handling, improved error logging, and non-blocking initialization of optional features (file watcher, TypedSocket).
- Updated tooling/dev deps: @git.zone/tsbuild/tsbundle/tstest and @types/node updated.

2025-11-19 - 3.0.80 - fix(dependencies)

Bump dependencies and devDependencies to updated versions

- Upgrade @push.rocks/smartfeed: ^1.0.11 → ^1.4.0
- Upgrade @push.rocks/smartfile: ^11.2.5 → ^11.2.7
- Upgrade @push.rocks/smartjson: ^5.0.20 → ^5.2.0
- Upgrade @push.rocks/smartlog: ^3.1.8 → ^3.1.10
- Upgrade @push.rocks/smartsitemap: ^2.0.3 → ^2.0.4
- Upgrade @push.rocks/taskbuffer: ^3.1.7 → ^3.4.0
- Upgrade @tsclass/tsclass: ^9.2.0 → ^9.3.0
- Upgrade @types/express: ^5.0.3 → ^5.0.5
- Dev: Upgrade @git.zone/tsbuild: ^2.6.4 → ^3.1.0
- Dev: Upgrade @git.zone/tsbundle: ^2.5.1 → ^2.5.2
- Dev: Upgrade @git.zone/tsrun: ^1.3.3 → ^2.0.0
- Dev: Upgrade @git.zone/tstest: ^2.3.4 → ^2.8.2

2025-09-03 - 3.0.79 - fix(servertools)

Normalize Express wildcard parameter notation to /{*splat} across server routes and handlers; add local Claude settings

- Replaced route pattern '/*splat' with '/{*splat}' in ts/classes.typedserver.ts and ts/servertools/tools.sslredirect.ts
- Updated Express options route to use '/{*splat}' in ts/servertools/classes.server.ts
- Clarified wildcard handling comments and array-join logic for splat params in ts/servertools/classes.handlerproxy.ts and ts/servertools/classes.handlerstatic.ts
- Added .claude/settings.local.json containing local tool permissions for Claude/dev onboarding
- No functional API changes — routing pattern normalization and comment/handler improvements only

2025-09-03 - 3.0.78 - fix(servertools)

Fix wildcard path extraction for static/proxy handlers, correct serviceworker route, add local settings and test typo fix

- Make HandlerProxy and HandlerStatic robust to Express 5 wildcard param shapes (handle req.params.splat, numeric params, req.baseUrl and root routes) to correctly compute relative paths
- Change serviceworker route registration to use '/serviceworker/*splat' (instead of previous pattern) for consistent wildcard handling
- Fix test wording typo in test/test.server.ts ('exposer' -> 'expose')
- Add .claude/settings.local.json with local tool permissions and add .serena/.gitignore to ignore /cache

2025-08-17 - 3.0.77 - fix(servertools)

Adjust route wildcard patterns and CORS handling; update serviceworker and SSL redirect patterns; bump express dependency; add local Claude settings

- Normalize route wildcard patterns to use splat tokens (e.g. '/someroute/*' -> '/someroute/*splat*', '/' -> '/*splat') for consistent route matching
- Update serviceworker route registration to '/serviceworker{.*}' and adjust related handler
- Change SSL redirect route from '*' to '/*splat'
- Adjust CORS preflight options path to '/*splat' and update tests to reflect new route patterns

- Bump express dependency from ^4.21.2 to ^5.1.0
- Add .claude/settings.local.json for local Claude tool permissions

2025-08-16 - 3.0.76 - fix(handlerproxy)

Use SmartRequest API and improve proxy/asset response handling; update tests and bump dependencies; add local project configuration files

- Replace deprecated smartrequest.request usage with SmartRequest fluent API in ts/servertools/classes.handlerproxy.ts and add explicit handling for GET/POST/PUT/DELETE/PATCH methods.
- Normalize proxied response body handling by using arrayBuffer()/text() and converting to Buffer to avoid body type inconsistencies.
- Switch asset fetching in ts/utilityservers/classes.websiteserver.ts to SmartRequest + arrayBuffer for reliable binary handling.
- Update tests (test/test.server.ts) to use SmartRequest.create() and to read response bodies via response.text(), matching the updated request API.
- Bump dependencies: @push.rocks/smartrequest -> ^4.2.1 and body-parser -> ^2.2.0.
- Add local project configuration files: .claude/settings.local.json and .serena/project.yml.

2025-08-16 - 3.0.75 - fix(deps)

Update dependencies, test tooling and test imports; enhance npm test script

- Bump multiple runtime dependencies to newer patch/minor versions (notable updates: @cloudflare/workers-types, @push.rocks/* packages such as smartchok, smartfile, smartlog, smartpath, smartrx, and others, @tsclass/tsclass, @types/express, lit).
- Upgrade dev tooling versions: @git.zone/tsbuild and @git.zone/tsbundle; update @git.zone/tstest to v2.3.4.
- Improve npm test script by adding --verbose, --logfile and increased --timeout.
- Fix test imports to use @git.zone/tstest/tapbundle in test/test.reload.ts and test/test.server.ts.

2025-04-12 - 3.0.74 - fix(commit-info)

chore: update commit metadata (no source code changes)

- Uncommitted diff shows no changes in source files; the commit updates internal commit info only.

2025-04-11 - 3.0.73 - fix(metadata)

Update repository URLs and metadata to reflect the new organization scope

- Changed gitscope from 'pushrocks' to 'api.global' in npmextra.json
- Updated repository URL, bugs URL, and homepage in package.json to use code.foss.global/api.global/typedserver

2025-04-11 - 3.0.72 - fix(project)

chore: no changes - commit metadata update

2025-04-11 - 3.0.71 - fix(serviceworker)

Improve error handling and logging in service worker backend and network manager; update multiple dependency versions and packageManager settings.

- Upgrade dependency versions in package.json (e.g. @cloudflare/workers-types, @push.rocks/smartfile, @push.rocks/smartpromise, @push.rocks/smartrequest, @tsclass/tsclass, and @types/express)
- Add packageManager field to package.json
- Enhance error handling in ServiceworkerBackend (using try/catch and detailed logging) during client reload, notification display, and alert message sending
- Improve network request handling by clearing timeouts and converting errors reliably in NetworkManager
- Wrap service worker install and activate event handlers with try/catch to log errors appropriately

2025-03-16 - 3.0.70 - fix(TypedServer)

Improve error handling in server startup and response buffering. Validate configuration for reload injections, wrap file watching and TypedSocket initialization in try/catch blocks, enhance client notification and stop procedures, and ensure proper Buffer conversion in the proxy handler.

- Add validation to throw error if reload script is enabled without a serve directory
- Wrap file watching and TypedSocket initialization in try/catch to prevent crashes during startup
- Update the reload function to safely notify clients and handle notification errors
- Enhance the stop procedure to aggregate cleanup tasks with error handling
- Ensure consistent conversion of response bodies to Buffer in HandlerProxy with fallback when undefined
- Include fallback hash generation in createServeDirHash for error resilience

2025-03-16 - 3.0.69 - fix(servertools)

Fix compression stream creation returns, handler proxy buffer conversion, and sitemap URL concatenation

- Return compression stream immediately in createCompressionStream for each case instead of using break statements
- Convert proxied response to a Buffer in handler proxy rather than throwing an error when it isn't a string
- Fix addUrls method in sitemap to correctly concatenate new URLs without duplicating existing entries

2025-02-07 - 3.0.68 - fix(cache-manager)

Simplify cache control headers in cache manager

- Removed unnecessary cache control headers while setting modern Cache-Control.

2025-02-06 - 3.0.67 - fix(serviceworker)

Enhance header security for cached resources in service worker

- Added Cross-Origin-Resource-Policy header management for service worker cached resources.

2025-02-06 - 3.0.66 - fix(serviceworker)

Improve error handling and logging in cache manager and update manager.

- Enhanced error handling and logging in cache management functions.
- Corrected network request handling in update manager.
- Added missing error handling for fetch events.

2025-02-04 - 3.0.65 - fix(readme)

Update documentation with advanced usage and examples

- Added section on advanced usage including service worker and edge worker setup
- Detailed integration examples for type-safe API requests and WebSocket communication
- Expanded configuration options and cache strategies

2025-02-04 - 3.0.64 - fix(serviceworker)

Improve cache handling and response header management in service worker.

- Addressed issue preventing caching of certain responses due to missing CORS headers.
- Added 'Vary: Origin' header to ensure proper response handling.
- Included 'Access-Control-Expose-Headers' for better CORS support.

2025-02-04 - 3.0.63 - fix(core)

Refactored caching strategy for service worker to improve compatibility and performance.

- Removed hard and soft caching distinctions.
- Simplified cache setup process.
- Improved browser caching control headers.

2025-02-04 - 3.0.62 - fix(Service Worker)

Refactor and clean up the cache logic in the Service Worker to improve maintainability and handle Safari-specific cache behavior.

- Refactored logic for determining cached domains, enhancing the readability and maintainability of the code.
- Improved handling of CORS settings in caching requests, notably bypassing caching for soft cached domains in Safari to avoid CORS issues.
- Enhanced error response creation for failed resource fetching, maintaining clarity on why and how certain resources were not fetched or cached.
- Revised the structure of the caching logic to ensure consistent behavior across all supported browsers.

2025-02-04 - 3.0.61 - fix(ServiceWorkerCacheManager)

Fixed caching mechanism to better support Safari's handling of soft-cached domains.

- Added logic to differentiate between hard and soft cached domains.
- Implemented special handling for soft cached domains on Safari by bypassing caching.
- Ensured appropriate CORS headers are present in cached responses.

- Improved error handling with informative 500 error responses.
- Optimized caching logic to prevent redundant caching and potential issues with locked streams on Safari.

2025-02-04 - 3.0.61 - fix(ServiceWorkerCacheManager)

Fixed caching mechanism to better support Safari's handling of soft-cached domains.

- Added logic to differentiate between hard and soft cached domains.
- Implemented special handling for soft cached domains on Safari by bypassing caching.
- Ensured appropriate CORS headers are present in cached responses.
- Improved error handling with informative 500 error responses.
- Optimized caching logic to prevent redundant caching and potential issues with locked streams on Safari.

2025-02-04 - 3.0.61 - fix(ServiceWorkerCacheManager)

Fixed caching mechanism to better support Safari's handling of soft-cached domains.

- Added logic to differentiate between hard and soft cached domains.
- Implemented special handling for soft cached domains on Safari by bypassing caching.
- Ensured appropriate CORS headers are present in cached responses.
- Improved error handling with informative 500 error responses.
- Optimized caching logic to prevent redundant caching and potential issues with locked streams on Safari.

2025-02-04 - 3.0.60 - fix(cachemanager)

Improve cache management and error handling

- Updated comments for clarity and consistency.
- Enhanced error handling in `fetch` event listener.
- Optimized cache key management and cleanup process.
- Ensured CORS headers are set for cached responses.
- Improved logging for caching operations.

2025-02-03 - 3.0.59 - fix(serviceworker)

Fixed CORS and Cache Control handling for Service Worker

- Improved handling of CORS settings for external requests.
- Preserved important headers while excluding caching headers.
- Ensured the presence of CORS headers in cached responses.
- Adjusted Cache-Control headers to prevent browser caching but allow service worker caching.

2025-02-03 - 3.0.58 - fix(network-manager)

Refined network management logic for better offline handling.

- Improved logic to handle missing connections more gracefully.
- Added detailed online/offline connection status logging.
- Implemented a check for stale cache with a grace period for offline scenarios.
- Network requests now use optimized retries and timeouts.

2025-02-03 - 3.0.57 - fix(updateManager)

Refine cache management for service worker updates.

- Ensured cache is forcibly updated if older than defined maximum age.

- Implemented interval checks and forced updates for cache staleness.
- Updated version information and cache timestamps upon forced updates or validations.

2025-02-03 - 3.0.56 - fix(cachemanager)

Adjust cache control headers and fix redundant code

- Remove duplicate assetbroker URLs in the cache evaluation logic.
- Update cache control headers to improve caching behavior.
- Increase the timeout for fetch operations to improve compatibility.

2025-01-28 - 3.0.55 - fix(server)

Fix response content manipulation for HTML files with injectReload

- Moved fileString declaration inside HTML file handling block to prevent unnecessary string conversion for non-HTML files.
- Corrected responseContent assignment to ensure modified HTML strings are converted back to Buffer format.

2025-01-28 - 3.0.54 - fix(servertools)

Fixed an issue with compression results handling in HandlerStatic where content was always being written even if not compressed.

- Corrected the double writing of response in HandlerStatic.
- Ensured that file buffers are only conditionally written based on compression availability.

2024-12-26 - 3.0.53 - fix(infohtml)

Remove Sentry script and logo from HTML template

- Removed Sentry script from the HTML template.
- Removed Lossless GmbH logo and contact info.
- Updated footer link to point to foss.global.

2024-12-25 - 3.0.52 - fix(dependencies)

Bump package versions in dependencies and exports.

- Updated package dependencies to their latest versions.
- Added './infohtml' in package exports.

2024-08-27 - 3.0.51 - fix(core)

Update dependencies and fix service worker cache manager and task manager functionalities

- Updated dependencies in package.json to their latest versions
- Enhanced service worker cache manager to include additional scoped URLs
- Fixed task manager to start the task manager and added update task functionality
- Removed .gitlab-ci.yml from the repository as part of the cleanup

2024-05-25 - 3.0.43 to 3.0.50 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.43 to 3.0.50

2024-05-23 - 3.0.37 to 3.0.42 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.37 to 3.0.42

2024-05-17 - 3.0.37 - Core

Routine update and bug fix

- Updated core functionalities

2024-05-14 - 3.0.33 to 3.0.36 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.33 to 3.0.36

2024-05-13 - 3.0.31 to 3.0.32 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.31 to 3.0.32

2024-05-11 - 3.0.29 to 3.0.31 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.29 to 3.0.31

2024-04-19 - 3.0.27 to 3.0.28 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.27 to 3.0.28

2024-04-14 - 3.0.27 - Documentation

Updated Documentation

- Improved and updated documentation

2024-03-01 - 3.0.25 to 3.0.26 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.25 to 3.0.26

2024-02-21 - 3.0.20 to 3.0.24 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.20 to 3.0.24

2024-01-19 - 3.0.19 to 3.0.20 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.19 to 3.0.20

2024-01-09 - 3.0.14 to 3.0.18 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.14 to 3.0.18

2024-01-08 - 3.0.11 to 3.0.13 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.11 to 3.0.13

2024-01-07 - 3.0.9 to 3.0.10 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.9 to 3.0.10

2023-11-06 - 3.0.8 to 3.0.9 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.8 to 3.0.9

2023-10-23 - 3.0.6 to 3.0.7 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.6 to 3.0.7

2023-10-20 - 3.0.5 to 3.0.6 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.5 to 3.0.6

2023-09-21 - 3.0.4 to 3.0.5 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.4 to 3.0.5

2023-08-06 - 3.0.2 to 3.0.3 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.2 to 3.0.3

2023-08-03 - 3.0.1 to 3.0.0 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 3.0.1 to 3.0.0

2023-08-03 - 2.0.65 - Core

Breaking change in core update

- Introduced breaking changes updating core functionalities

2023-07-02 - 2.0.59 to 2.0.64 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 2.0.59 to 2.0.64

2023-07-01 - 2.0.54 to 2.0.58 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 2.0.54 to 2.0.58

2023-06-12 - 2.0.53 - Core

Routine update and bug fix

- Updated core functionalities

2023-04-10 - 2.0.52 to 2.0.53 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 2.0.52 to 2.0.53

2023-04-04 - 2.0.49 to 2.0.51 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 2.0.49 to 2.0.51

2023-03-31 - 2.0.45 to 2.0.48 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 2.0.45 to 2.0.48

2023-03-30 - 2.0.37 to 2.0.44 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 2.0.37 to 2.0.44

2023-03-29 - 2.0.33 to 2.0.36 - Core

Routine updates and bug fixes

- Updated core functionalities for better performance and stability in versions 2.0.33 to 2.0.36