@apiclient.xyz/elasticsearch

Documentation for @apiclient.xyz/elasticsearch

- readme.md for @apiclient.xyz/elasticsearch
- changelog.md for @apiclient.xyz/elasticsearch

readme.md for @apiclient.xyz/elasticsearch

> □□ **Enterprise-grade TypeScript client for Elasticsearch** — Type-safe, observable, and production-ready

A modern, fully-typed Elasticsearch client built for scale. Features fluent configuration, distributed transactions, intelligent bulk operations, advanced logging with Kibana compatibility, and comprehensive observability out of the box.

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

Feature	Description
□□ **Fluent Configuration**	Builder pattern for type-safe, environment-aware configuration
□□ **Document Management**	Full CRUD with sessions, snapshots, and lifecycle management
□□ **Query Builder**	Type-safe DSL for complex queries and aggregations
□□ **Bulk Operations**	High-throughput indexing with backpressure and adaptive batching
□□ **KV Store**	Distributed key-value storage with TTL, caching, and compression
□□ **Transactions**	ACID-like semantics with optimistic concurrency control

Feature	Description
□□ **Schema Management**	Index templates, migrations, and schema validation
□□ **Logging**	Kibana-compatible log destination with ILM policies
□□ **Plugin System**	Extensible architecture with built-in retry, caching, rate limiting
□□ **Observability**	Prometheus metrics, distributed tracing, structured logging
⚡ **Circuit Breaker**	Automatic failure detection and recovery

□□ Installation

```
npm install @apiclient.xyz/elasticsearch
# or
pnpm add @apiclient.xyz/elasticsearch
```

□□ Quick Start

Configuration

```
import { createConfig, ElasticsearchConnectionManager, LogLevel } from
'@apiclient.xyz/elasticsearch';

// Fluent configuration builder
const config = createConfig()
  .fromEnv()  // Load from ELASTICSEARCH_URL, ELASTICSEARCH_USERNAME, etc.
  .nodes('http://localhost:9200')
  .basicAuth('elastic', 'changeme')
  .timeout(30000)
  .retries(3)
  .compression(true)
  .poolSize(10, 2)
  .logLevel(LogLevel.INFO)
  .enableMetrics(true)
  .enableTracing(true, {
    serviceName: 'my-service',
```

```
    serviceVersion: '1.0.0',
  })
  .build();

// Initialize connection with health monitoring
const connection = ElasticsearchConnectionManager.getInstance(config);
await connection.initialize();

console.log(`Health: ${connection.getHealthStatus()}`);
console.log(`Circuit: ${connection.getCircuitState()}`);
```

Document Management

```
import { DocumentManager } from '@apiclient.xyz/elasticsearch';

interface Product {
  name: string;
  price: number;
  category: string;
  inStock: boolean;
}

const products = new DocumentManager<Product>({
  index: 'products',
  connectionManager: connection,
  autoCreateIndex: true,
});

await products.initialize();

// CRUD operations
await products.create('prod-001', {
  name: 'Premium Widget',
  price: 99.99,
  category: 'widgets',
  inStock: true,
});
```

```
const product = await products.get('prod-001');
await products.update('prod-001', { price: 89.99 });
await products.delete('prod-001');

// Session-based batch operations with auto-cleanup
const result = await products.session({ cleanupStale: true })
  .start()
  .upsert('prod-001', { name: 'Widget A', price: 10, category: 'widgets', inStock: true })
  .upsert('prod-002', { name: 'Widget B', price: 20, category: 'widgets', inStock: false })
  .commit();

console.log(`Processed ${result.successful} documents in ${result.took}ms`);

// Iterate over all documents
for await (const doc of products.iterate()) {
  console.log(doc._source.name);
}

// Create analytics snapshot
const snapshot = await products.snapshot(async (iterator, previous) => {
  let total = 0, count = 0;
  for await (const doc of iterator) {
    total += doc._source.price;
    count++;
  }
  return { averagePrice: total / count, productCount: count };
});
```

Query Builder

```
import { createQuery } from '@apiclient.xyz/elasticsearch';

// Fluent query building
const results = await createQuery<Product>('products')
  .match('name', 'widget', { fuzziness: 'AUTO' })
  .range('price', { gte: 10, lte: 100 })
  .term('inStock', true)
  .sort('price', 'asc')
```

```
  .size(20)
  .from(0)
  .highlight({ fields: { name: {} } })
  .aggregations(agg => agg
    .terms('categories', 'category')
    .avg('avgPrice', 'price')
  )
  .execute();

console.log(`Found ${results.hits.total} products`);
console.log('Categories:', results.aggregations?.categories);
```

Bulk Operations

```
import { createBulkIndexer } from '@apiclient.xyz/elasticsearch';

const indexer = createBulkIndexer({
  flushThreshold: 1000,
  flushIntervalMs: 5000,
  maxConcurrent: 3,
  enableBackpressure: true,
  onProgress: (progress) => {
    console.log(`Progress: ${progress.processed}/${progress.total}
(${progress.successRate}%)`);
  },
});

await indexer.initialize();

// Queue operations
for (const item of largeDataset) {
  await indexer.index('products', item.id, item);
}

// Wait for completion
const stats = await indexer.flush();
console.log(`Indexed ${stats.successful} documents, ${stats.failed} failures`);
```

Key-Value Store

```
import { createKVStore } from '@apiclient.xyz/elasticsearch';

const kv = createKVStore<string>({
  index: 'app-config',
  enableCache: true,
  cacheMaxSize: 10000,
  defaultTTL: 3600, // 1 hour
  enableCompression: true,
});

await kv.initialize();

// Basic operations
await kv.set('api-key', 'sk-secret-key', { ttl: 86400 });
const key = await kv.get('api-key');

// Batch operations
await kv.mset([
  { key: 'config:a', value: 'value-a' },
  { key: 'config:b', value: 'value-b' },
]);

const values = await kv.mget(['config:a', 'config:b']);

// Scan with pattern matching
const scan = await kv.scan({ pattern: 'config:*', limit: 100 });
```

Transactions

```
import { createTransactionManager } from '@apiclient.xyz/elasticsearch';

const txManager = createTransactionManager({
  defaultIsolationLevel: 'read_committed',
  defaultLockingStrategy: 'optimistic',
  conflictResolution: 'retry',
  maxConcurrentTransactions: 100,
```

```
});

await txManager.initialize();

// ACID-like transaction
const tx = await txManager.begin({ autoRollback: true });

try {
  const account1 = await tx.read('accounts', 'acc-001');
  const account2 = await tx.read('accounts', 'acc-002');

  await tx.update('accounts', 'acc-001', { balance: account1.balance - 100 });
  await tx.update('accounts', 'acc-002', { balance: account2.balance + 100 });

  tx.savepoint('after-transfer');

  await tx.commit();
} catch (error) {
  await tx.rollback();
}
```

Logging Destination

```
import { createLogDestination, chainEnrichers, addHostInfo, addTimestamp } from
'@apiclient.xyz/elasticsearch';

const logger = createLogDestination({
  indexPrefix: 'app-logs',
  dataStream: true,
  ilmPolicy: {
    name: 'logs-policy',
    phases: {
      hot: { maxAge: '7d', maxSize: '50gb' },
      warm: { minAge: '7d' },
      delete: { minAge: '30d' },
    },
  },
  enricher: chainEnrichers(addHostInfo(), addTimestamp()),
```

```
  sampling: { strategy: 'rate', rate: 0.1 }, // 10% sampling
});

await logger.initialize();

await logger.log({
  level: 'info',
  message: 'User authenticated',
  context: { userId: '12345', service: 'auth' },
});

// Batch logging
await logger.logBatch([
  { level: 'debug', message: 'Request received' },
  { level: 'info', message: 'Processing complete' },
]);
```

Schema Management

```
import { createSchemaManager } from '@apiclient.xyz/elasticsearch';

const schema = createSchemaManager({ enableMigrationHistory: true });

await schema.initialize();

// Create index with schema
await schema.createIndex('products', {
  mappings: {
    properties: {
      name: { type: 'text', analyzer: 'standard' },
      price: { type: 'float' },
      category: { type: 'keyword' },
      createdAt: { type: 'date' },
    },
  },
  settings: {
    numberOfShards: 3,
    numberOfReplicas: 1,
```

```
  },
});

// Run migrations
await schema.migrate('products', [
  {
    version: '1.0.1',
    description: 'Add tags field',
    up: async (client, index) => {
      await client.indices.putMapping({
        index,
        properties: { tags: { type: 'keyword' } },
      });
    },
  },
]);

// Create index template
await schema.createIndexTemplate('products-template', {
  indexPatterns: ['products-*'],
  mappings: { /* ... */ },
});
```

Plugin System

```
import {
  createPluginManager,
  createRetryPlugin,
  createCachePlugin,
  createRateLimitPlugin,
} from '@apiclient.xyz/elasticsearch';

const plugins = createPluginManager({ enableMetrics: true });

// Built-in plugins
plugins.register(createRetryPlugin({ maxRetries: 3, backoffMs: 1000 }));
plugins.register(createCachePlugin({ maxSize: 1000, ttlMs: 60000 }));
plugins.register(createRateLimitPlugin({ maxRequests: 100, windowMs: 1000 }));
```

```
// Custom plugin
plugins.register({
  name: 'custom-logger',
  version: '1.0.0',
  hooks: {
    beforeRequest: async (ctx) => {
      console.log(`Request: ${ctx.operation} on ${ctx.index}`);
    },
    afterResponse: async (ctx, response) => {
      console.log(`Response: ${response.statusCode}`);
    },
  },
});
```

□□ Architecture

```
@apiclient.xyz/elasticsearch
├── core/
│   ├── config/          # Fluent configuration builder
│   ├── connection/      # Connection manager, circuit breaker, health checks
│   ├── errors/          # Typed errors and retry policies
│   ├── observability/   # Logger, metrics (Prometheus), tracing (OpenTelemetry)
│   └── plugins/         # Plugin manager and built-in plugins
└── domain/
    ├── documents/       # DocumentManager, sessions, snapshots
    ├── query/           # QueryBuilder, AggregationBuilder
    ├── bulk/            # BulkIndexer with backpressure
    ├── kv/              # Distributed KV store
    ├── transactions/    # ACID-like transaction support
    ├── logging/         # Log destination with ILM
    └── schema/          # Schema management and migrations
```

□□ Observability

Prometheus Metrics

```
import { defaultMetricsCollector } from '@apiclient.xyz/elasticsearch';

// Export metrics in Prometheus format
const metrics = defaultMetricsCollector.export();

// Available metrics:
// - elasticsearch_requests_total{operation, index, status}
// - elasticsearch_request_duration_seconds{operation, index}
// - elasticsearch_errors_total{operation, index, error_type}
// - elasticsearch_circuit_breaker_state{state}
// - elasticsearch_connection_pool_size
// - elasticsearch_bulk_operations_total{type, status}
```

Distributed Tracing

```
import { defaultTracingProvider } from '@apiclient.xyz/elasticsearch';

// OpenTelemetry-compatible tracing
const span = defaultTracingProvider.startSpan('custom-operation');
span.setAttributes({ 'custom.attribute': 'value' });
// ... operation
span.end();
```

□□ Authentication Methods

```
// Basic auth
createConfig().basicAuth('username', 'password')

// API key
createConfig().apiKeyAuth('api-key-value')

// Bearer token
createConfig().bearerAuth('jwt-token')
```

```
// Elastic Cloud
createConfig().cloudAuth('cloud-id', { apiKey: 'key' })

// Certificate
createConfig().auth({
  type: 'certificate',
  certPath: '/path/to/cert.pem',
  keyPath: '/path/to/key.pem',
})
```

⚡ Performance Tips

1. **Use bulk operations** for batch inserts — `BulkIndexer` handles batching, retries, and backpressure
2. **Enable compression** for large payloads — reduces network overhead
3. **Configure connection pooling** — `poolSize(max, min)` for optimal concurrency
4. **Leverage caching** — `KVStore` has built-in LRU/LFU caching
5. **Use sessions** for document sync — automatic cleanup of stale documents
6. **Enable circuit breaker** — prevents cascade failures during outages

□□ Compatibility

- **Elasticsearch**: 8.x, 9.x
- **Kibana**: 8.x, 9.x (for log visualization)
- **Node.js**: 18+
- **TypeScript**: 5.0+

License and Legal Information

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH or third parties, and are not included within the scope of the MIT license granted herein.

Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines or the guidelines of the respective third-party owners, and any usage must be approved in writing. Third-party trademarks used herein are the property of their respective owners and used only in a descriptive manner, e.g. for an implementation of an API or similar.

Company Information

Task Venture Capital GmbH Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

changelog.md for @apiclient.xyz/elasticsearch

2025-11-30 - 3.1.4 - fix(readme)

Clarify license, trademark and company information in README

- Update MIT license link to point to ./LICENSE and simplify wording
- Rework trademark section to mention third-party trademarks and clarify they are not covered by the MIT license
- Add guidance that trademark usage must comply with Task Venture Capital GmbH or respective third-party guidelines and require written approval
- Add explicit note that third-party trademarks are used descriptively (e.g. for API implementations)
- Normalize company registration wording (District Court Bremen) and adjust contact/email phrasing

2025-11-29 - 3.1.0 - feat(schema-manager)

Add Schema Management module and expose it in public API; update README to mark Phase 3 complete and move priorities to Phase 4

- Add ts/domain/schema/index.ts to export SchemaManager and related types
- Re-export SchemaManager and schema types from top-level ts/index.ts so schema APIs are part of the public surface
- Update README hints: mark Phase 3 Advanced Features complete (Transaction Support and Schema Management), add schema/transaction example references, and update Next Priorities to Phase 4 (Comprehensive Test Suite, Migration Guide, README Update)

2025-11-29 - 3.0.0 - BREAKING CHANGE(core)

Refactor to v3: introduce modular core/domain architecture, plugin system, observability and strict TypeScript configuration; remove legacy classes

- Major refactor to a modular v3 layout: new ts/core and ts/domain directories with clear public index exports
- Added core subsystems: configuration (ConfigurationBuilder), connection management (ElasticsearchConnectionManager) with health checks and circuit breaker, typed error hierarchy and retry policy, observability (Logger, MetricsCollector, TracingProvider)
- Introduced plugin system (PluginManager) and built-in plugins: logging, metrics, cache, retry and rate-limit
- New domain APIs: DocumentManager (with DocumentSession and snapshot/iterate support), QueryBuilder (type-safe query + aggregations), BulkIndexer, KV store and Logging domain (enrichers and destinations)
- Switched to strict TypeScript settings in tsconfig.json (many strict flags enabled) and added QUICK_FIXES.md describing import/type fixes needed
- Removed legacy files and consolidated exports (deleted old els.classes.* files, els.plugins.ts and autocreated commitinfo file)
- Public API changed: index exports now re-export core and domain modules (breaking changes for consumers — update imports and initialization flow)

2025-11-29 - 2.0.17 - fix(ci)

Update CI workflows and build config; bump dependencies; code style and TS config fixes

- Gitea workflows updated: swapped CI image to code.foss.global, adjusted NPMCI_COMPUTED_REPOURL and replaced @shipzone/npmci with @ship.zone/npmci; tsdoc package path updated.
- Removed legacy .gitlab-ci.yml (migrated CI to .gitea workflows).
- Bumped dependencies and devDependencies (e.g. @elastic/elasticsearch -> ^9.2.0, @git.zone/_ packages, @push.rocks/_ packages) and added repository/bugs/homepage/pnpm/packageManager metadata to package.json.
- Tests updated: import path change to @git.zone/tstest/tapbundle and test runner export changed to default export (export default tap.start()).
- TypeScript config changes: module and moduleResolution set to NodeNext and added exclude for dist_/**/.d.ts.
- Code cleanups and formatting: normalized object/argument formatting, trailing commas, safer ElasticClient call shapes (explicit option objects), and minor refactors across

ElasticDoc, FastPush, KVStore, ElasticIndex, ElasticScheduler and smartlog destination.
- Added .gitignore entries for local AI tool directories and added readme.hints.md and npmextra.json.

2023-08-30 - 2.0.2..2.0.16 - core

Series of maintenance releases and small bugfixes on the 2.0.x line.

- Multiple "fix(core): update" commits across 2.0.2 → 2.0.16 addressing small bugs and stability/maintenance issues.
- No single large feature added in these patch releases; recommended to consult individual release diffs if you need a precise change per patch.

2023-08-25 - 2.0.0 - core

Major 2.0.0 release containing core updates and the transition from the 1.x line.

- Bumped major version to 2.0.0 with core updates.
- This release follows a breaking-change update introduced on the 1.x line (see 1.0.56 below). Review breaking changes before upgrading.

2023-08-25 - 1.0.56 - core (BREAKING CHANGE)

Breaking change introduced on the 1.x line.

- BREAKING CHANGE: core updated. Consumers should review the change and adapt integration code before upgrading from 1.0.55 → 1.0.56 (or migrating to 2.0.x).

2023-08-18 - 1.0.40..1.0.55 - maintenance

Maintenance and fixes across many 1.0.x releases (mid 2023).

- Numerous "fix(core): update" commits across 1.0.40 → 1.0.55 addressing stability and minor bug fixes.
- Includes smaller testing updates (e.g., fix(test): update in the 1.0.x series).

2023-07-05 - 1.0.32..1.0.44 - maintenance

Maintenance sweep in the 1.0.x line (July 2023).

- Multiple small core fixes and updates across these patch releases.
- No large feature additions; stability and incremental improvements only.

2019-11-02 - 1.0.26..1.0.30 - maintenance

Patch-level fixes and cleanup in late 2019.

- Several "fix(core): update" releases to address minor issues and keep dependencies up to date.

2018-11-10 - 1.0.20 - core

Cleanup related to indices.

- fix(clean up old indices): update — housekeeping and cleanup of old indices.

2018-11-03 - 1.0.13 - core

Security/tooling update.

- fix(core): add snyk — added Snyk related changes (security/scan tooling integration).

2018-09-15 - 1.0.11 - core

Dependency and compatibility updates.

- fix(core): update dependencies and bonsai.io compatibility — updated dependencies and ensured compatibility with bonsai.io.

2018-08-12 - 1.0.9 - test

Testing improvements.

- fix(test): update — improvements/adjustments to test suite.

2018-03-03 - 1.0.7 - system

System-level change.

- "system change" — internal/system modification (no public API feature).

2018-01-27 - 1.0.4 - quality/style

Coverage and style updates.

- adjust coverageTreshold — adjusted test coverage threshold.
- update style / update — code style and minor cleanup.

2018-01-27 - 1.0.3 - core (feat)

Winston logging integration (added, later removed in a subsequent release).

- feat(core): implement winston support — initial addition of Winston logging support.

2018-01-27 - 1.0.6 - winston (fix)

Removal of previously added logging integration.

- fix(winston): remove winston — removed Winston integration introduced earlier.

2018-01-26 - 1.0.2 - core (feat)

Index generation improvement.

- feat(core): update index generation — improvements to index generation logic.

2018-01-24 - 1.0.1 - core (initial)

Project initial commit and initial cleanup.

- feat(core): initial commit — project bootstrap.
- fix(core): cleanup — initial cleanup and adjustments after the first commit.

Note: Versions that only contain bare version-tag commits (commit messages identical to the version string) have been summarized as ranges above. For detailed per-patch changes consult individual release diffs.