Toggly Node.js

Server-side evaluation · Signed defs · Entity context · WebSocket

Install

```
npm i @ops-ai/toggly-node-core
```

Backend App Key only — never a frontend or management credential. Zero browser dependencies. Full server-side evaluation with signed definitions, entity context, and WebSocket live updates.

Initialize

```
import { createTogglyClient } from
  '@ops-ai/toggly-node-core';

const client = createTogglyClient({
  appKey: process.env.TOGGLY_APP_KEY!, // backend
  environment: 'Production',
  identity: 'user-123',           // optional default
  featureDefaults: { 'my-flag': false },
  refreshInterval: 180000,        // 3 min
  verifySignatures: true,
  enableFileCache: true,
  fileCachePath: '.toggly-cache',
  onError: (err, ctx) ⇒
    console.error('Toggly:', ctx, err),
});

await client.init();
```

Evaluate

```
// single flag
if (await client.isFeatureOn('MyFeature')) { … }

// with user context
await client.isFeatureOn('Beta', {
  identity: 'user-123',
  groups: ['beta'],
  request: { country: 'US' },
});

// multi-flag gate
await client.evaluateFeatureGate(
  ['a','b'], 'any', false);

await client.isFeatureOff('Maintenance');
```

Entity context

Register entity kinds at startup — Node PUTs schemas to the dashboard. Entity gates fail closed without context.

```
client.registerContext(
  'Order',
  (order) ⇒ ({
    kind: 'Order',
    key: String(order.id),
    attributes: { Status: order.status },
  }),
  { keyProperty: 'id',
    properties: [{ name: 'Status', type: 'string' }] }
);

await client.isFeatureOn(
  'ExpressCheckout', undefined, order, 'Order');
```

Identity & refresh

```
await client.setIdentity('user-456');
await client.refresh();
await client.clearCache();
client.close();
```

State & hooks

```
const { features, error, initialized }
  = client.state;

client.addHook({
  getMetadata: () ⇒ ({ name: 'GA4', version: '1' }),
  afterEvaluation: async (key, ctx, data, result) ⇒
    // analytics…
  },
});
```

Toggly Node.js

Server-side evaluation · Signed defs · Entity context · WebSocket

Reliability contract

- `verifySignatures` — verify exact raw `defs` JSON bytes (never re-serialize).
- `onError` — surface fetch, cache, signature, JWKS failures.
- After first load, refresh failures preserve last-known-good flags.
- WebSocket `signing-key-updated` clears JWKS and refreshes definitions.
- `ETag` / `X-Definitions-Revision` for conditional fetches.

HTTP adapters

EXPRESS

```
npm i @ops-ai/toggly-express
app.use(togglyMiddleware({
  appKey: process.env.TOGGLY_APP_KEY!,
  getIdentity: (req) ⇒ req.user?.id,
}))
if (await req.toggly!.isFeatureOn('new-dashboard')) {
const v = await req.toggly!.getVariant('checkout-flow
process.on('SIGTERM', () ⇒ closeExpressToggly())
```

FASTIFY / HONO / KOA

```
npm i @ops-ai/toggly-fastify   # plugin
await app.register(togglyPlugin, { appKey, … })

npm i @ops-ai/toggly-hono
app.use('*', togglyMiddleware({ appKey, … }))
await c.get('toggly').isFeatureOn('new-dashboard')

npm i @ops-ai/toggly-koa
app.use(togglyMiddleware({ appKey, … }))
await ctx.state.toggly.isFeatureOn('new-dashboard')
```

Bind identity from your trusted session — not caller headers. NestJS has its own cheat sheet: `/cheatsheets/nestjs`.

Config reference

- `appKey` — backend App Key, required for API mode
- `environment` — default `Production`
- `baseUrl` — default `definitions.toggly.io`
- `refreshInterval` — ms (0 = disable)
- `enableStreaming` — WebSocket (default true)
- `registerContextsOnStartup` — default true
- `allowedKeyIds` — signing key allow-list

Pitfalls

- Use `@ops-ai/toggly-node-core` on the server — the browser SDK won't run in Node.js.
- Pass entity per check — `identity` is for user rollouts, not page entities.
- Call `clearCache()` once after upgrading if you see invalid signature errors on cold start.
- Always call `client.close()` on graceful shutdown.
- Set `featureDefaults` for offline resilience.

Resources

- Docs: `docs.toggly.io/sdks/nodejs`
- Dashboard: `app.toggly.io`