Toggly Astro

SSR / SSG · Server .astro components · Island hydration

Install

```
npm i @ops-ai/astro-feature-flags-toggly
```

Integration

ASTRO.CONFIG.MJS

```
import { defineConfig } from 'astro/config';
import togglyIntegration from
  '@ops-ai/astro-feature-flags-toggly/integration';

export default defineConfig({
  integrations: [
    togglyIntegration({
      appKey: process.env.TOGGLY_APP_KEY,
      environment: 'Production',
      flagDefaults: { 'my-flag': false },
      featureFlagsRefreshInterval: 180000,
      onError: (msg, err) => console.warn(msg, err),
    }),
  ],
});
```

Middleware

SRC/MIDDLEWARE.TS

```
import { sequence } from 'astro:middleware';
import { createTogglyMiddleware } from
  '@ops-ai/astro-feature-flags-toggly';

const toggly = createTogglyMiddleware({
  appKey: import.meta.env.TOGGLY_APP_KEY,
  environment: 'Production',
});

export const onRequest = sequence(toggly);
```

Server Feature.astro

```
---
import Feature from
  '@ops-ai/astro-feature-flags-toggly/components/Featu
---
<Feature flag="new-dashboard"><NewDashboard /></Featur
<Feature flag="new-dashboard" negate={true}>
  <OldDashboard />
</Feature>

<Feature flags={['a','b']} requirement="any">
  …
</Feature>
```

Programmatic (server)

```
---
const toggly = Astro.locals.toggly;
const on = await toggly.getFlag('feature-key');
const allFlags = await toggly.getFlags();
const hasAccess = await toggly.evaluateGate(
  ['a','b'], 'any', false);
---
```

Page-level gating

```
---
// src/pages/beta.astro
x-feature: beta-feature
title: Beta Feature Page
---
<!— entire page gated by flag —>
<!— builds toggly-page-features.json —>
```

SSR: flags fetched per request. SSG: flags fetched at build time — use `FeatureClient` for runtime updates.

Entity context

```
<Feature flag="OrderBadge"
  context={{ kind: 'Order', key: String(order.id),
    attributes: { Vip: order.vip } }}>
  <span class="badge">Featured</span>
</Feature>
```

Pass entity per check — not via global identity. Gates fail closed without context.

Toggly Astro

SSR / SSG · Server .astro components · Island hydration

FeatureClient (islands)

```
---
import FeatureClient from
  '@ops-ai/astro-feature-flags-toggly/components/Featu
---
<FeatureClient flag="widget" client="visible">
  <InteractiveWidget />
</FeatureClient>

<!— client="load" | "idle" | "visible" —>
```

React / Vue / Svelte islands

REACT

```
import { Feature, useFeatureFlag }
  from '@ops-ai/astro-feature-flags-toggly/react';

const { enabled, isReady } =
  useFeatureFlag('premium');
```

VUE

```
import Feature from
  '@ops-ai/astro-feature-flags-toggly/vue/Feature.vue
import { useFeatureFlag } from
  '@ops-ai/astro-feature-flags-toggly/vue';
```

SVELTE

```
import Feature from
  '@ops-ai/astro-feature-flags-toggly/svelte/Feature.s
import { featureFlag } from
  '@ops-ai/astro-feature-flags-toggly/svelte';

const on = featureFlag('new-dashboard');
```

Identity (client)

```
import { setIdentity, clearIdentity,
  refreshFlags } from
  '@ops-ai/astro-feature-flags-toggly';

setIdentity('user-123'); // login: starts refresh
clearIdentity();         // logout: starts refresh
// Independent manual-refresh action:
await refreshFlags();
```

SSR vs SSG

- **SSR** (`output: 'server'`): fresh flags per request — use `Feature.astro` .
- **SSG** (`output: 'static'`): flags frozen at build — use `FeatureClient` for dynamic updates.
- Hybrid: server for access control, client islands for UI enhancements.

Pitfalls

- Prefer `Feature.astro` (server) over client islands when possible — zero JS shipped.
- Always set `flagDefaults` — refresh failures preserve last-known-good flags after first load.
- Islands need a hydration directive: `client:load` , `client:idle` , or `client:visible` .
- Page gating via `x-feature` frontmatter requires edge worker for true enforcement (404).
- Enable Client Side API in Toggly dashboard for your App Key.

Resources

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