Toggly Go

Local evaluation, net/http, html/template, signed defs, gRPC metrics

Install

```
go get github.com/ops-ai/Toggly.FeatureManagement/toggly-go@latest
```

Module: `github.com/ops-ai/Toggly.FeatureManagement/toggly-go`

Initialize

```
import (
  "github.com/ops-ai/Toggly.FeatureManagement/toggly-go/toggly"
  "github.com/ops-ai/Toggly.FeatureManagement/toggly-go/toggly/session"
)

c, err := toggly.NewClient(toggly.Config{
  AppKey:       os.Getenv("TOGGLY_APP_KEY"),
  Environment:  "Production",
  BaseURL:      "https://app.toggly.io/",
  SessionStore: session.NewMemoryStore(),
  SessionTTL:   30 * time.Minute,
})
if err != nil { return err }
defer c.Close()
```

Evaluate

```
evalCtx := toggly.Context{
  Identity: "user-123",
  Groups:   []string{"beta"},
  Traits:   map[string]any{"country": "US"},
}

on, err := c.IsEnabled(ctx, "MyFeature", evalCtx)

// multi-flag
on, _ = c.EvaluateGate(ctx,
  []string{"A","B"}, toggly.RequirementAny,
  evalCtx, false)
```

Variants (catalog-local)

```
// After togglyhttp / WithEvalContext (preferred):
v, err := c.GetVariant(r.Context(),
  "checkout-flow", toggly.Context{})

// CLI / worker — Config.Identity or SetIdentity:
c, err := toggly.NewClient(toggly.Config{
  AppKey: appKey, Environment: "Production",
  Identity: "user-123",
})
if err != nil { return err }
defer c.Close()
c.SetIdentity("user-123")
v, err = c.GetVariant(ctx, "checkout-flow",
  toggly.Context{})
_ = v
// Per-call Identity only to override
ambient/config.
```

html/template

```
import "github.com/ops-ai/Toggly.FeatureManagement/toggly-go/togglytemplate"

t := template.Must(template.New("base").Funcs(
  togglytemplate.FuncMap(c, nil)).
  ParseFiles("base.html"))
// Pass toggly.Context or data implementing
// TogglyContext() when executing the template.

// in template:
{{ if (feature . "MyFeature") }}…{{ end }}
{{ if (featureAny . "A" "B") }}…{{ end }}
```

Custom filters

```
type PaidPlan struct{}
func (PaidPlan) Evaluate(
  featureKey string,
  params map[string]any,
  ctx eval.Context) (bool, error) {
  plan, _ := ctx.Traits["plan"].(string)
  return plan == "pro" ||
         plan == "enterprise", nil
}
c.RegisterFilter("PaidPlan", PaidPlan{})
```

Toggly Go

Local evaluation, net/http, html/template, signed defs, gRPC metrics

net/http middleware

```
import "github.com/ops-
ai/Toggly.FeatureManagement/toggly-go/togglyhttp"

// Callbacks read your verified principal.
mux := http.NewServeMux()
mux.Handle("/admin",
  togglyhttp.FeatureGate(c, "AdminUI")(handler))
h := togglyhttp.MiddlewareWith(togglyhttp.Options{
  GetIdentity: identityFromAuth,
  GetGroups: groupsFromAuth,
  GetClaims: claimsFromAuth,
})(mux)
// Missing request fields are filled from headers.
http.ListenAndServe(":8080", h)
```

Options: `WithDenyHandler`, `WithErrorHandler`, `WithContextBuilder`.

context.Context

```
import "github.com/ops-
ai/Toggly.FeatureManagement/toggly-go/togglyctx"

ctx = togglyctx.With(ctx, toggly.Context{
  Identity: "user-123" })

on, err := c.IsEnabled(ctx, "MyFeature",
  toggly.Context{}) // Inherits ambient user
context.
// Inspect explicitly when needed:
evalCtx, present := togglyctx.From(ctx)
```

Snapshots — file / memory

```
import "github.com/ops-
ai/Toggly.FeatureManagement/toggly-
go/toggly/snapshot"

c, _ := toggly.NewClient(toggly.Config{
  AppKey: appKey, Environment: env,
  SnapshotProvider:
    snapshot.NewFileProvider("/var/lib/toggly"),
})

// or in-memory for tests
sp := snapshot.NewMemoryProvider()
```

Snapshots — Redis / Mongo

```
rdb := redis.NewClient(
  &redis.Options{Addr:"localhost:6379"})

sp := snapshot.NewRedisProvider(
  snapshot.RedisOptions{
    Client: rdb,
    Prefix: "toggly",
    TTL:    24 * time.Hour,
  })

// keys: {prefix}:definitions, {prefix}:jwks
```

Signed definitions

```
c, _ := toggly.NewClient(toggly.Config{
  AppKey: appKey, Environment: env,
  UseSignedDefinitions: true,
  AllowedKeyIDs: map[string]struct{}{
    "ABCDEF...ES256": {},
  },
})
```

ES256 / ECDSA P-256 over `<defs>|<timestamp>`; SDK keeps raw JSON to verify byte-exact.

Live updates (WebSocket)

```
c, _ := toggly.NewClient(toggly.Config{
  AppKey: appKey, Environment: env,
  EnableLiveUpdates: true,
})
```

Connects to `wss://…/{appKey}/ws?rev=`; handles `sync`, `flags-updated`, and `signing-key-updated`. Requires toggly-go v0.13.2+: `RefreshInterval` polling continues while the socket is connected but idle (and when disconnected); scheduled polls are skipped only for 20 minutes after a successful WebSocket-triggered refresh.

Toggly Go

Local evaluation, net/http, html/template, signed defs, gRPC metrics

Usage stats & metrics (gRPC)

```
c, _ := toggly.NewClient(toggly.Config{
  AppKey: appKey, Environment: env,
  EnableUsage: true,
  EnableMetrics: true,
})

c.RecordUsage("MyFeature", true, evalCtx)
```

Secure features (authz hook)

```
c, _ := toggly.NewClient(toggly.Config{
  AppKey: appKey, Environment: env,
  AuthorizationService: MyAuth{},
})
// MyAuth implements secure.AuthorizationService
```

Runs after evaluation — return false to mask a flag from this user.

Testing

```
c, _ := toggly.NewClient(toggly.Config{
  AppKey: "test", Environment: "test",
  DisableBackgroundRefresh: true,
  DisableEntityContextRegistration: true,
})
// No initial load: unknown flags stay off.
// To test definitions, use a local HTTP fixture
// with normal background loading and await a
// known result before assertions.
```

Pitfalls

- Check initialization errors before `defer c.Close()`. Close clients to release background work and connections.
- Prefer ambient / `SetIdentity` then `GetVariant(ctx, key, Context{})`; do not create a separate variants client per user.
- `EnableUndefinedOnDevelopment` allows unknown flags when set; it does not detect your deployment environment. Keep it false in production.
- When enabling signed definitions, pin `AllowedKeyIDs` after rotation to defeat downgrade attacks.
- `togglyhttp.Middleware` must wrap the router so `FeatureGate` can read the request context.

Resources

- Docs: `docs.toggly.io/sdks/go`
- pkg.go.dev: `github.com/ops-ai/...toggly-go`
- GitHub: `github.com/ops-ai/Toggly.FeatureManagement`
- Requires Go 1.24+