Toggly Dart / Flutter

Flutter widget, lifecycle-aware refresh, signed defs (ES256), offline cache

Install

```
flutter pub add feature_flags_toggly

# pubspec.yaml
dependencies:
  feature_flags_toggly: ^1.11.1
```

```
import 'package:feature_flags_toggly/
       feature_flags_toggly.dart';
```

Initialize

```
@override
void initState() {
  super.initState();
  _initToggly();
}

Future<void> _initToggly() async {
  final res = await Toggly.init(
    appKey: const String.fromEnvironment(
      'TOGGLY_APP_KEY'),
    environment: 'Production',
    useSignedDefinitions: true,
    flagDefaults: {
      'NewCheckout': false,
      'BetaUI':      false,
    },
```

Init response

```
switch (res.status) {
  case TogglyLoadFeatureFlagsResponse.fetched:
    // network OK
    break;
  case TogglyLoadFeatureFlagsResponse.cached:
    // 304 Not Modified
    break;
  case TogglyLoadFeatureFlagsResponse.defaults:
    // offline → flagDefaults
    break;
  case TogglyLoadFeatureFlagsResponse.error:
    // log + fallback
    break;
}
```

Feature widget

```
Feature(
  featureKeys: const ['NewCheckout'],
  child: const CheckoutV2(),
)

// any-of / all-of
Feature(
  featureKeys: const ['A','B'],
  requirement: FeatureRequirement.any,
  child: const Banner(),
)

// negate
Feature(
  featureKeys: const ['LegacyMode'],
  negate: true,
```

FeatureGateBuilder (conditional UI)

```
// Show/hide → Feature
// Styling / taps / behavior → FeatureGateBuilder

FeatureGateBuilder(
  featureKeys: const ['PremiumCheckout'],
  builder: (context, enabled) {
    return Text(
      'USDA ID',
      style: TextStyle(
        color: enabled ? Colors.blue : Colors.grey,
        decoration: enabled ? TextDecoration.underline
      ),
    );
  },
)
```

Programmatic check

```
final on = await Toggly.evaluateFeatureGate(
  ['A','B'],
  requirement: FeatureRequirement.all,
  negate: false,
);

if (on) doNewThing();
```

Identity & rollouts

```
// at init
await Toggly.init(
  appKey: KEY, environment: ENV,
  identity: 'user-123',
);

// at runtime — login / logout
await Toggly.setIdentity('user-456');
await Toggly.setIdentity(null); // anon device id
```

Auto-generates a stable device id when `identity` is null. Changing identity clears in-memory state and refreshes; persisted cache for other users is kept for switch-back.

Manual refresh

```
final res = await Toggly.refresh();
// res.status → fetched | cached | defaults | error
```

The SDK auto-refreshes on app foreground and on the configured interval; calling `refresh()` is rarely needed.

Offline mode (no API)

```
await Toggly.init(
  flagDefaults: {
    'NewCheckout': true,
    'BetaUI':      false,
  },
);
```

Persist across restarts (offline)

```
import 'package:feature_flags_toggly_secure_storage/
       feature_flags_toggly_secure_storage.dart';

await Toggly.init(
  appKey: KEY, environment: ENV,
  identity: userId,
  config: TogglyConfig(
    cacheProvider: SecureStorageCacheProvider(),
  ),
);
```

Packages: `feature_flags_toggly_secure_storage`, `_disk`, `_sqlite`, `_isar` — or implement `TogglyCacheProvider`.

Signed definitions

```
await Toggly.init(
  appKey: KEY, environment: ENV,
  useSignedDefinitions: true,
  config: TogglyConfig(
    trustedKeyIds: ['KID-2025-Q1'],
  ),
);
```

ECDSA P-256 / ES256 over `data|timestamp`; JWKS is cached up to 30 days, refreshed on unknown `kid`.

Provider / Riverpod

PROVIDER

```
ChangeNotifierProvider(
  create: (_) => TogglyNotifier()..load(),
  child: MaterialApp(/* ... */),
);

class TogglyNotifier extends ChangeNotifier {
  Future<void> load() async {
    await Toggly.init(appKey:KEY,environment:ENV);
    notifyListeners();
  }
}
```

Toggly Dart / Flutter

Flutter widget, lifecycle-aware refresh, signed defs (ES256), offline cache

Async checks (FutureBuilder)

```
FutureBuilder<bool>(
  future: Toggly.evaluateFeatureGate(['NewUI']),
  builder: (ctx, snap) {
    if (!snap.hasData) {
      return const CircularProgressIndicator();
    }
    return snap.data!
      ? const NewUI()
      : const ClassicUI();
  },
)
```

Debug snapshot

```
final d = Toggly.debug();
debugPrint('user: ${d['user']}');
debugPrint('lastSynced: ${d['lastSynced']}');
debugPrint('lastError:  ${d['lastError']}');
```

Cleanup

```
@override
void dispose() {
  Toggly.dispose();   // cancels timers, streams
  super.dispose();
}
```

Lifecycle behavior

- App foreground: flags auto-refresh.
- App background: refresh paused to save battery.
- Definitions revision cached per evaluation identity for HTTP 304 and WS `?rev=` sync; persist via `TogglyRevisionCacheProvider` for cold starts.
- Memory-only by default; add a cache provider + stable identity to survive restarts.

Pitfalls

- Initialize Toggly in your **root** widget's `initState` — initializing inside transient screens recreates the client.
- Always set `flagDefaults` — mobile users hit airplane mode, captive portals, and cold-start scenarios constantly.
- In tests, pass only `flagDefaults` to skip network and keep evaluations deterministic.
- Call `Toggly.dispose()` in your top widget's `dispose()` to avoid timer leaks during hot restart.
- Don't put the App Key in a publicly-readable config file; use `String.fromEnvironment` with `--dart-define` at build time.
- Web Flutter is supported — but evaluate using App Key only, never an API key.

Resources

- Docs: `docs.toggly.io/sdks/flutter`
- pub.dev: `feature_flags_toggly`
- GitHub: `github.com/ops-ai/Toggly.FeatureManagement`
- Requires Dart 2.18+ / Flutter 1.17+