Toggly React Native

Expo & bare RN · Hooks · Components · Offline cache

Cheat Sheet
`@ops-ai/react-native-toggly`
RN 0.70+

Install

MAIN SDK

```
npm i @ops-ai/react-native-toggly
```

ASYNCSTORAGE (EXPO)

```
npm i @ops-ai/react-native-toggly-storage-async \
  @react-native-async-storage/async-storage
```

MMKV (BARE RN, FASTER)

```
npm i @ops-ai/react-native-toggly-storage-mmkv \
  react-native-mmkv
cd ios && pod install
```

NETWORK STATE (OPTIONAL)

```
npm i @react-native-community/netinfo
```

Provider setup

```
import { TogglyProvider, Feature,
  useFeatureFlag } from '@ops-ai/react-native-toggly';
import { createAsyncStorageAdapter } from
  '@ops-ai/react-native-toggly-storage-async';

const storage = createAsyncStorageAdapter();

<TogglyProvider
  appKey="your-app-key"
  environment="production"
  identity="user-123"
  storage={storage}
  featureDefaults={{ 'my-flag': false }}
  refreshInterval={30000}
  onReady={() => console.log('Ready')}
  onError={(err) => crashlytics().recordError(err)}
  onFlagsChanged={(flags) => …}>
  <App />
</TogglyProvider>
```

Offline / local-only

```
<TogglyProvider
  featureDefaults={{
    'new-dashboard': true,
    'beta-feature': false,
  }}>
  <App />
</TogglyProvider>
```

Hooks

USEFEATUREFLAG

```
const { isEnabled, isLoading, error }
  = useFeatureFlag('feature-key', {
    defaultValue: false,
  });
```

USEFEATUREGATE

```
const { isEnabled } = useFeatureGate(
  ['a','b'],
  { requirement: 'any', negate: false });
```

USETOGGLY

```
const { flags, isReady, error,
  refresh, setIdentity } = useToggly();
```

USEVARIANT

```
const variant = useVariant('pricing-test');
// { name, configurationValue } | null
```

Identity

```
// on login
await setIdentity(user.id);
await refresh();

// on logout
await setIdentity('');
await refresh();
```

Toggly React Native

Expo & bare RN · Hooks · Components · Offline cache

Feature component

```
<Feature featureKey="welcome-banner">
  <WelcomeBanner />
</Feature>

<Feature featureKeys={['a','b']}
  requirement="any"
  loading={<Skeleton />}>
  <PremiumContent />
</Feature>
<Feature featureKeys={['a','b']}
  requirement="any"
  negate
  loading={<Skeleton />}>
  <UpgradePrompt />
</Feature>

<Feature featureKey="maintenance" negate>
  <NormalContent />
</Feature>
```

Entity context

```
import { Feature } from
  '@ops-ai/react-native-toggly';

// Canonical entity context needs no mapper.
const orderContext = {
  kind: 'Order', key: String(order.id),
  attributes: { Status: order.status },
};

<Feature featureKey="OrderBadge"
  context={orderContext}>
  <Badge />
</Feature>
```

Gates fail closed without entity context.

Variants

```
<TogglyProvider enableVariants appKey="…">
  <App />
</TogglyProvider>

const variant = useVariant('pricing-test');
// { name, configurationValue } | null
```

Requires `enableVariants` — fetches `/evaluated-variants-signed`.

Reliability

- `onError` — fetch, storage, signature, JWK failures.
- After first load, refresh failures preserve last-known-good effective flags.
- `Feature`, `useFeatureFlag`, and `useFeatureGate` subscribe to `effectiveFlagsChanged` automatically.

Pitfalls

- Install a storage adapter for persistent offline cache — without one, flags reset on app restart.
- Always check `isLoading` before rendering flag-dependent UI.
- Set `featureDefaults` so offline mode doesn't break the app.
- Call `refresh()` after `setIdentity` for user-specific flags.
- Use MMKV on bare RN for faster reads; AsyncStorage for Expo.

Resources

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