Android OTA Updates in React Native

---

1. What is an OTA Update in React Native?

An OTA update allows you to update the **JavaScript bundle and assets** of a React Native app **without publishing a new version to the Play Store**.

Key idea

> **OTA updates modify how the app behaves, not what the app is.**

What actually changes

- JavaScript bundle (`index.android.bundle`)
- Assets referenced from JS (images, fonts, JSON)

What never changes

- APK / AAB
- Native Android code
- App permissions
- App identity

---

2. High-Level OTA Flow (Android)

Step-by-step runtime flow

1. User opens the app
2. Native Android app loads the React Native runtime
3. OTA client checks the update server
4. If a newer JS bundle is available:
5. Bundle is downloaded
6. Stored locally on the device
7. App reloads (or on next launch)
8. New JavaScript code executes

> No Play Store interaction happens during this process.

---

3. Types of Changes OTA Updates ACCEPT ✅

These are **safe and supported** OTA changes.

3.1 JavaScript / TypeScript Logic

- Business rules
- API integrations
- Validation logic
- Feature flags
- Bug fixes

3.2 UI & Navigation (JS only)

- Component layout changes
- Styling updates
- Adding new screens
- Conditional rendering

⚠️Avoid removing or renaming existing routes without backward compatibility.

3.3 Assets

- Images
- Fonts
- Localization files
- Static JSON

3.4 Configuration inside JS

- Environment-based behavior
- Feature toggles
- Remote config consumption

---

4. Changes OTA Does NOT Allow ❌(Play Store Required)

4.1 Native Android Code

Any change inside:

- `android/` folder
- Java / Kotlin files
- Native Modules
- C++ / NDK code

Examples:

- Adding a new camera module
- Changing Bluetooth behavior
- Modifying splash screen natively

4.2 Permissions

Permissions are fixed at install time.

❌OTA cannot:

- Add new permissions
- Remove permissions
- Change permission behavior

Examples requiring Play Store update:

- Location access
- Bluetooth
- Notifications
- Camera / Microphone

4.3 App Configuration

- App name
- App icon
- Package name
- Signing config
- Min / Target SDK
- AndroidManifest.xml

4.4 React Native / Native Dependency Changes

- React Native version upgrade
- New native dependency
- Upgrading native SDKs

---

5. What OTA Technically Allows but is RISKY ⚠️

These changes work but can break production apps if not handled carefully.

5.1 Navigation Structure Changes

Risk factors:

- Persisted navigation state
- Deep links

- Background restores

Best practices:

- Never remove routes immediately
- Deprecate screens gradually
- Use version checks

5.2 State Shape Changes

Examples:

- Redux store schema change
- AsyncStorage / MMKV structure changes

Risk:

- App crash on startup
- Infinite loading screen

Mitigation:

- State migration logic
- Versioned storage keys
- Controlled storage resets

5.3 API Contract Changes

- Backend response shape changed
- Field removed or renamed

Risk:

- OTA rolls out faster than backend

Mitigation:

- Backward-compatible APIs
- Feature flags

---

6. What Actually Breaks Apps During OTA

6.1 Missing Native Modules

JS calls a native module that does not exist in the installed app.

Result:

- Immediate runtime crash

6.2 Version Mismatch

JS expects newer native APIs than the installed version provides.

Result:

- Crash or undefined behavior

6.3 Storage / DB Migration Failures

- Old persisted data incompatible with new JS logic

Result:

- White screen
- App stuck during launch

6.4 Crash Loop

- App downloads OTA
- Crashes on startup
- Repeats endlessly

Mitigation:

- Automatic rollback
- OTA kill switch

6.5 Policy Violations

Google Play restrictions:

- Introducing sensitive behavior via OTA
- Changing app purpose
- Adding tracking or payments

Result:

- App suspension or removal

---

7. Android-Specific OTA Limitations

- No background execution during update
- OTA applies only after app restart
- OS may kill app during update
- Low storage can cause update failure

---

8. Best Practices for Android OTA

8.1 Version Gating

Always check native app version before enabling features.

8.2 Backward Compatibility

- Support at least N-1 versions
- Never assume latest native build

8.3 Gradual Rollouts

- 5% → 25% → 100%
- Monitor crashes and ANRs(App Not Responding Errors)

8.4 Kill Switch

Remote config to disable OTA instantly.

8.5 Separation of Concerns

- Native-dependent features → Play Store
- JS-only features → OTA

---

10. Summary Table

Change Type	OTA	Play Store
JS logic	✅	❌
UI changes	✅	❌
Assets	✅	❌
Navigation (safe)	⚠️	❌
Permissions	❌	✅
Native modules	❌	✅
SDK upgrades	❌	✅
