

Android 16KBページサイズ対応を

はじめからていねいに

Flutter × Android 15

2025年7月9日

自己紹介

西峰 綾汰

にしみね りょうた

所属

- CyberAgent
- SGEマンガ事業部
- 2024年入社

趣味・興味

-  F1
-  ポーカー
-  フジロック
今年参戦します！

本日の内容

2025年11月以降、必須対応となる16KBページサイズ

- 1. なぜ必要？ - パフォーマンス向上の仕組み**
- 2. 何が起きる？ - Flutterアプリへの影響**
- 3. どう対応する？ - 具体的な導入手順**
- 4. デモで確認 - build.gradleとネイティブライブラリの設定**

今日のゴール

明日から16KB対応を始められる状態に！



まずは基本から

16KBページサイズって何？

そもそもページサイズとは

メモリ管理の基本単位

ページ = OSがメモリを管理する最小の単位

- 本の1ページのようなもの
- データの読み書きはページ単位
- これまで**4KB**が標準

身近な例で理解



図書館の本棚

- 4KB = 薄い本（管理が大変）
- 16KB = 厚い本（管理が楽）

同じ内容なら、厚い本の方が
探しやすい！

4KB → 16KBで何が変わる？

ページ数が1/4に

100万ページ → 25万ページ

管理コストが大幅削減

検索が高速化

ページテーブルが小さくなる

= メモリアクセスが速い

CPU負荷軽減

ページ切り替えが減る

= バッテリー消費削減

キャッシュ効率UP

TLBヒット率が向上

= アプリが軽快に動作

Google Playの新要件

2025年 11月1日

対応期限

API 35+

Android 15以降

重要なポイント

- ✓ 対象: 新規アプリ・アップデート両方
- ✓ 条件: Android 15 (API 35) をターゲット
- ✗ 未対応: Google Play審査に通らない

注意: 2025年9月から段階的適用の可能性もあり



実際の効果は？

パフォーマンス測定結果

Googleの公式テスト結果

🚀 アプリ起動

最大30%高速化

速度向上

⚡ 電力消費

4.56%削減

省電力

📸 カメラ起動

4.5~6.6%高速化

UX向上

⚡ システム起動

8%短縮

全体改善

出典: Android Developers Blog

"Improving Android memory efficiency - 16 KB page size" (2024)

Flutter実アプリでの測定

31.7 %増加

初回起動時間

測定データ詳細

4KB: 3,815.65 μs

16KB: 5,027.84 μs

なぜ遅くなる？ 🤔

- 初回のメモリアライメント
- ページテーブルの初期化
- でも心配無用！



長期的にはメリットが大きい



Flutterアプリへの影響

何が問題になるの？

よくあるエラー

こんなエラーに遭遇します

```
Failure [INSTALL_FAILED_INVALID_APK:  
Failed to extract native libraries, res=-2]
```

原因は？

ネイティブライブラリ (.so) のアライメント不良

- Flutterアプリには多くの.soファイルが含まれる
- これらが4KB前提でビルドされている
- 16KB環境では動作しない！

ネイティブライブラリ (.so) とは？

Flutterアプリの構成要素

Dartコード

- `lib/` 以下のコード
- Flutter Frameworkが処理
- 影響なし ✓

ネイティブライブラリ

- C/C++で書かれたコード
- プラグインに含まれる
- 要対応 !

主な.soファイル

- `libflutter.so` - Flutterエンジン
- `libapp.so` - Dartコードのコンパイル結果
- `libOO.so` - 各種プラグイン

影響を受けるパッケージ

✖ Rive

アニメーションライブラリ
librive_text.so が非対応

要対応

⚠ local_notifications

通知機能
Desugar制約（Java 11）

要設定

✓ datadog_flutter

監視・分析
v2.11.0+で対応済み

対応済

⚠ Gradle警告

memory_info, update_available
jcenter使用の警告

注意



対応方法

步步バイ・バイ・步步

STEP1: 現状を確認する

まずはAPKをチェック

```
# 1. APKをビルド  
flutter build apk --release  
  
# 2. Googleのチェックスクリプトをダウンロード  
curl -O https://android.googlesource.com/\  
.../check_elf_alignment.sh  
  
# 3. チェック実行  
./check_elf_alignment.sh build/app/outputs/flutter-apk/app-release.apk
```

結果の見方

```
./libflutter.so: ALIGNED (2^16) ✓      # OK!  
./libapp.so: ALIGNED (2^14) ✓          # OK!  
./libplugin.so: UNALIGNED (2^12) ✗    # 要修正！
```

STEP2: 開発環境を更新

必要なバージョン

| ツール | Android 14 | Android 15 | 確認方法 |

| SDK | 34 | **35** | `compileSdk` |

| NDK | r25.1 | **r27.2 → r28.1** | `ndkVersion` |

| AGP | 8.3.2 | **8.10.0** | `build.gradle` |

| Gradle | 8.9 | **8.11.1** | `gradle-wrapper` |

| core-ktx | 1.13.1 | **1.16.0** | `build.gradle` |

アップデート方法

```
# Flutterを最新に  
flutter upgrade
```

```
# Android Studioでツールを更新  
Android Studio > Settings > SDK Tools
```



build.gradleの設定

Flutterエンジニア必見！

build.gradleとは？

Androidアプリのビルド設定ファイル



場所

```
android/  
└── build.gradle      # プロジェクト全体  
  └── app/  
    └── build.gradle  # アプリ本体 ←こっち！
```



役割

- ビルドツールのバージョン指定
- 依存関係の管理
- コンパイル設定
- 今回はここを修正！

必須の修正内容

android/app/build.gradle

```
android {  
    namespace "com.example.myapp"  
    compileSdk 35 // ← Android 15に変更  
  
    defaultConfig {  
        applicationId "com.example.myapp"  
        minSdk 21  
        targetSdk 35 // ← Android 15に変更  
  
        // ↓ この設定を追加！  
        externalNativeBuild {  
            cmake {  
                arguments += [  
                    "-DANDROID_SUPPORT_FLEXIBLE_PAGE_SIZES=ON"  
                ]  
            }  
        }  
    }  
}
```

NDKバージョンの指定

実際のプロジェクトでの変遷

段階的な更新履歴

- 1. Android 14:** NDK r25.1.8937393
- 2. Android 15初期:** NDK r27.2.12479018 (PR #4049)
- 3. 16KB完全対応:** NDK r28.1.13356709 (PR #4198)

NDK r27の場合（追加設定が必要）

```
android {  
    ndkVersion "27.2.12479018" // PR #4049で使用  
  
    // CMakeLists.txtがある場合  
    externalNativeBuild {  
        cmake {  
            arguments += [ "-DANDROID_SUPPORT_FLEXIBLE_PAGE_SIZES=ON"]  
        }  
    }  
}
```

NDK r28+の場合（推奨）

依存関係の更新

androidx.coreの更新が必須

```
dependencies {  
    implementation 'androidx.core:core-ktx:1.16.0' // ← 1.16.0以上必須  
    // その他の依存関係...  
}
```

AGPバージョンも確認

android/build.gradle (プロジェクトレベル)

```
buildscript {  
    ext.kotlin_version = '1.9.0'  
    dependencies {  
        classpath 'com.android.tools.build:gradle:8.6.0' // ← 8.6.0以上  
    }  
}
```



実際にやってみよう

デモンストレーション

デモ: APKのチェック

1. 既存APKの状態確認

```
# APKをビルド  
$ flutter build apk --release  
✓ Built build/app/outputs/flutter-apk/app-release.apk  
  
# チェックスクリプト実行  
$ ./check_elf_alignment.sh app-release.apk  
  
Checking app-release.apk...  
libflutter.so: ALIGNED (2^14) ⚠  
libapp.so: ALIGNED (2^14) ⚠  
librive_text.so: UNALIGNED (2^12) ✘
```

結果: 16KB対応が必要！

デモ: build.gradle修正

2. 設定ファイルの修正

Before

```
android {  
    compileSdk 34  
  
    defaultConfig {  
        targetSdk 34  
    }  
}
```

After

```
android {  
    compileSdk 35  
    ndkVersion "27.0.12077973"  
  
    defaultConfig {  
        targetSdk 35  
  
        externalNativeBuild {  
            ...  
        }  
    }  
}
```

デモ: 再ビルドと確認

3. クリーンビルドと検証

```
# クリーンビルド  
$ flutter clean  
$ flutter pub get  
$ flutter build apk --release  
  
# 再度チェック  
$ ./check_elf_alignment.sh app-release.apk  
  
Checking app-release.apk...  
libflutter.so: ALIGNED (2^16) ✓  
libapp.so: ALIGNED (2^16) ✓  
librive_text.so: UNALIGNED (2^12) ✗ # プラグイン側の対応待ち
```

Flutter本体は対応完了！



トラブルシューティング

よくある問題と解決方法

エラー別対処法

1. INSTALL_FAILED_INVALID_APK

```
# Gradleキャッシュをクリア  
./gradlew clean  
rm -rf ~/.gradle/caches/  
  
# 再ビルト  
flutter build apk --release
```

2. AGP/Gradle互換性エラー

```
Dependency 'androidx.core:core-ktx:1.16.0' requires AGP 8.6.0+
```

解決: android/gradle/wrapper/gradle-wrapper.properties

プラグイン対応状況の確認

対応待ちプラグインの場合

1. 代替ライブラリを検討

- Rive → Lottie
- 独自実装 → Flutter標準機能

2. Issue/PRを確認

```
# GitHubで確認  
https://github.com/[plugin-name]/issues  
→ "16KB" や "page size" で検索
```

3. 一時的な回避策

```
# pubspec.yaml
```

Riveライブラリの16KB対応

対応状況と解決策

現状（2025年1月時点）

- **librive_text.so** が16KB未対応
- Flutter本体は対応済みだが、Riveプラグインがボトルネック

対応方法

1. NDK r28.1を使用

```
android {  
    ndkVersion "28.1.13356709" // PR #4198で検証済み  
}
```

2. check_elf_alignment.shで確認

テスト環境の構築

16KBエミュレータのセットアップ

1. Android Studio → AVD Manager

2. Create Virtual Device

3. System Image選択画面で:

項目	選択内容
Release Name	Android 15
API Level	35
ABI	x86_64



まとめ

今日から始める16KB対応

対応チェックリスト 1/4

現状確認

APKチェック

```
./check_elf_alignment.sh app-release.apk
```

プラグイン対応状況

- Riveライブラリのステータス確認
- ネイティブライブラリを含むプラグインのリストアップ

対応チェックリスト 2/4

環境更新

必須バージョン

- **Flutter:** 3.24.1以上
- **NDK:** r27以上 (推奨: r28.1)
- **AGP:** 8.6.0以上

更新コマンド

```
flutter upgrade
```

対応チェックリスト 3/4

設定変更

build.gradleの修正

```
android {  
    compileSdk 35  
    targetSdk 35  
    ndkVersion "28.1.13356709"  
}
```

依存関係

```
implementation 'androidx.core:core-ktx:1.16.0'
```

対応チェックリスト 4/4

テスト

16KBエミュレータで確認

- AVD Managerで「16k Page Size (Experimental)」を選択
- アプリの起動テスト
- パフォーマンス測定

最終確認

```
./check_elf_alignment.sh app-release.apk  
# すべてALIGNED (2^16)になること
```

なぜ今から準備が必要か

2025年11月まで「まだ時間がある」？

✗ 実はそうでもない理由

1. プラグインの対応待ち時間

- 人気プラグインほど対応に時間がかかる
- 代替ライブラリの検討・実装期間が必要

2. 段階的なテスト期間

- 社内テスト → ベータ版 → 本番リリース
- 各段階で問題が見つかる可能性

3. 予期せぬ問題への対処

- ネイティブコードの修正が必要な場合
- パフォーマンスチューニング

実装時の注意点 1/3

段階的な対応が必要

実際のPRから学ぶ

- PR #4049: Android 15基本対応
 - NDK r27.2.12479018を使用
 - 基本的なSDK/AGP更新
- PR #4198: 16KB完全対応
 - NDK r28.1.13356709へ更新
 - Riveライブラリ問題解決

ポイント: プラグインの対応状況により複数回の更新が必要

実装時の注意点 2/3

技術的な制約

Desugar制約

- flutter_local_notificationsの制約でJava 11に制限
- 最新のJava機能が使用不可

Gradle警告

- memory_infoがjcenter使用
- update_availableがjcenter使用
- 将来的な移行が必要

buildDir非推奨

- 影響範囲が大きいため移行見送り

実装時の注意点 3/3

検証の重要性

複数の検証手法を併用

```
# 1. スクリプトでの確認  
./check_elf_alignment.sh app-release.apk  
  
# 2. Android Studioでの確認  
# Build > Analyze APK  
  
# 3. 実機での動作確認  
# 16KBエミュレータでのテスト
```

検証ポイント

- すべてのネイティブライブラリがALIGNED
- アプリの起動と基本動作
- パフォーマンスの変化

参考リソース

公式ドキュメント

- [Android 16KB対応ガイド](#)
- [Flutter Android 15 Wiki](#)

トラブルシューティング

- [Flutter Issue #150168](#)
- [Stack Overflow](#)

サンプルコード

- [サンプルアプリ](#)

ご清聴ありがとうございました

質問・ディスカッション

一緒に16KB対応を進めましょう！

Twitter: @your_twitter

GitHub: @your_github