TestFlight Distribution Guide

This guide explains how to build and distribute your app to testers via TestFlight while minimizing your interaction with Apple's ecosystem.

Philosophy: Minimal Apple Ecosystem Involvement

This workflow is designed to:

- **Use EAS Build (cloud builds)** - No need to build locally with Xcode
- **Minimal Xcode usage** - Only for iOS Simulator testing during development
- **Keep Cursor as your IDE** - All development stays in Cursor
- **Web-based management** - Use App Store Connect website for TestFlight management
- **Command-line automation** - Use EAS CLI for builds and submissions

Region and Account Setup

Apple Developer Account Region

Important: Your Apple Developer account region determines:

- Where you can distribute the app
- Payment methods accepted
- Tax implications

Recommendation:

- **Create your Apple Developer account with UK region** if you're distributing in the UK
- Your physical location (Barbados) doesn't matter - Apple Developer accounts are location-independent
- Use your UK phone number (from your Pixel 8a) during account setup if possible
- You can access App Store Connect from anywhere in the world

Why UK region?

- Matches your target distribution market
- Easier payment processing (UK bank account/card)
- Avoids potential issues with Barbados-specific restrictions
- Your UK phone number aligns with UK account

Account Setup Steps

1. **Create Apple ID** (if you don't have one)
    - Use a professional email (not iCloud)
    - Set region to **United Kingdom** during signup
    - Use your UK phone number for verification
    - **Do NOT** use iCloud email or enable iCloud services

2. **Enroll in Apple Developer Program**
    - Go to developer.apple.com
    - Enroll with UK as your region
    - Pay $99/year (can use UK payment method)
    - Wait 24-48 hours for approval
3. **Set up App Store Connect**
    - Access via appstoreconnect.apple.com
    - Create your app listing
    - Set primary language to English (UK) if distributing in UK

Prerequisites

1. **Apple Developer Account**: Active Apple Developer Program membership ($99/year) - **set to UK region**
2. **App Store Connect Access**: Your Apple ID must have access to App Store Connect
3. **EAS CLI**: Install the Expo Application Services CLI
4. **Xcode** (minimal installation): Only needed for iOS Simulator - download from Mac App Store, install command-line tools only if needed

Setup Steps

1. Install EAS CLI

```
npm install -g eas-cli
```

2. Login to EAS

```
eas login
```

3. Minimal Xcode Installation (For Simulator Only)

You only need Xcode for iOS Simulator testing during development.

1. Download Xcode from Mac App Store (large download, ~10-15GB)
2. **Do NOT** open Xcode after installation unless you need the simulator
3. Accept license: `sudo xcodebuild -license accept`
4. Install command-line tools: `xcode-select --install` (if needed)
5. **That's it!** You won't need Xcode for building or submitting

For development testing:

- Use `expo run:ios` which will launch the simulator automatically
- Or use Cursor's terminal to run simulator commands

- **Never** open Xcode for builds - use EAS Build instead

4. Configure Your Apple Credentials

Recommended: Let EAS manage everything

EAS will automatically handle certificates and provisioning profiles. This means:

- No need to manage certificates in Keychain Access
- No need to create provisioning profiles manually
- No need to touch Xcode's signing settings

```
eas build:configure
```

When prompted:

- Choose "Let EAS handle credentials"
- Provide your Apple ID (the one with Developer Program access)
- EAS will create and manage all certificates automatically

Alternative: App Store Connect API Key (More Secure)

If you want even less Apple ecosystem involvement:

1. Create an App Store Connect API key:
   - Go to App Store Connect > Users and Access > Keys
   - Create a new key with "App Manager" or "Admin" role
   - Download the `.p8` key file (you can only download once!)
   - Note the Key ID and Issuer ID
2. Configure in EAS:

   ```
   eas credentials
   ```

   - Select iOS platform
   - Choose to use App Store Connect API key
   - Provide the key file, Key ID, and Issuer ID

Benefits of API Key approach:

- No need to provide Apple ID password to EAS
- More secure (can revoke key independently)
- Better for CI/CD if you add it later

5. Update eas.json Configuration

The `eas.json` file has been created with basic configuration. Update the `submit` section:

```
# Edit eas.json and update these values:
```

- `appleId`: Your Apple ID email (the one with Developer Program access)
- `ascAppId`: Your App Store Connect App ID (found in App Store Connect after creating the app)
- `appleTeamId`: Your Apple Team ID (found in Apple Developer portal > Membership)

Note: You can also skip this and provide these values when running `eas submit` interactively.

Building for TestFlight (Cloud Build - No Xcode Needed!)

Step 1: Create App in App Store Connect (First Time Only)

1. Go to App Store Connect
2. Click **My Apps** > **+** (plus button)
3. Fill in:
    - Platform: iOS
    - Name: Your app name
    - Primary Language: English (UK)
    - Bundle ID: `com.mc2tc.tallynative` (from your app.config.js)
    - SKU: A unique identifier (e.g., `tallynative-001`)
4. Click **Create**

Step 2: Build the iOS App (Cloud Build)

This happens entirely in the cloud - no Xcode needed!

```
eas build --platform ios --profile production
```

This will:

- Build your app in EAS's cloud infrastructure
- Generate an `.ipa` file
- Take approximately 10-20 minutes
- Handle all code signing automatically
- You can continue working in Cursor while it builds

Monitor the build:

```
# Check build status
eas build:list

# View build logs in real-time
eas build:view [BUILD_ID]
```

Step 3: Submit to App Store Connect

After the build completes, submit it to App Store Connect:

```
eas submit --platform ios --latest
```

This will:

- Upload the build to App Store Connect automatically
- Make it available in TestFlight after processing (usually 5-15 minutes)
- No need to open Xcode or App Store Connect manually

First-time setup: If you haven't configured `eas.json` submit section, EAS will prompt you for:

- Apple ID
- App Store Connect App ID
- Apple Team ID

Step 4: Verify Build in App Store Connect

1. Go to App Store Connect
2. Navigate to **My Apps** > Your App > **TestFlight**
3. Wait for build to finish processing (5-15 minutes)
4. Build will appear in the "Builds" section when ready

That's it! No Xcode needed for any of this.

Version and Build Number Management

EAS automatically manages build numbers, but you should update version numbers in `app.config.js`:

```
// app.config.js
module.exports = () => ({
  expo: {
    // ...
    version: '1.0.0',  // Update this for new releases
    ios: {
      buildNumber: '1',  // EAS will auto-increment, but you can set it
      // ...
    }
  }
})
```

Best Practice:

- Update `version` when you release new features
- Let EAS auto-increment `buildNumber` (or set it manually if needed)
- Each TestFlight build needs a unique build number

Adding Testers in TestFlight

Internal Testers (Up to 100) - Recommended for Your Tester

Internal testers don't require Beta App Review - they get builds immediately after processing.

1. Go to App Store Connect
2. Navigate to **My Apps** > Your App > **TestFlight**
3. Click **Internal Testing** tab
4. Click **+** next to "Testers" or create a new group
5. Add your tester's email address
6. Select the build you want them to test
7. Click **Add** - they'll receive an email invitation

Your tester will:

1. Receive an email invitation
2. Need to install TestFlight app from App Store (free)
3. Accept the invitation in TestFlight app
4. Download and install your app

External Testers (Up to 10,000)

External testers require Beta App Review (24-48 hours) before they can test.

1. In TestFlight, go to **External Testing** tab
2. Create a new group or use the default
3. Add the build you want to test
4. Add testers' email addresses
5. Submit for Beta App Review (required for external testers)
6. Once approved (24-48 hours), testers receive invitations

Use external testing for:

- Public beta testing
- Testing with users outside your organization
- When you need more than 100 testers

Important Notes

- **Build Number**: EAS automatically increments build numbers. You can also manage them in `app.config.js` under `ios.buildNumber`
- **Version Number**: Update `version` in `app.config.js` for new app versions
- **Processing Time**: After upload, Apple processes builds (5-15 minutes typically)
- **Beta Review**: External testers require Beta App Review (24-48 hours typically). Internal testers don't need review.
- **No Xcode Required**: Everything can be done via EAS CLI and App Store Connect website
- **Region Independence**: You can manage your UK-region Apple Developer account from anywhere (including Barbados)

Region-Specific Considerations

Barbados Location, UK Distribution

Q: Will being in Barbados cause issues? A: No. Apple Developer accounts are location-independent. You can:

- Access App Store Connect from anywhere
- Build and submit from anywhere
- Manage your UK-region account from Barbados
- Your physical location doesn't affect distribution

Q: Should I use VPN? A: Not necessary. Apple doesn't restrict access based on location. However, if you encounter any issues accessing App Store Connect, a UK VPN might help (though unlikely to be needed).

Q: Payment and Billing? A: Your Apple Developer Program billing is tied to your account region (UK), not your physical location. Use a UK payment method if possible, but international cards usually work.

Q: Tax Implications? A: Consult a tax professional. Generally:

- Your account region (UK) determines tax jurisdiction
- Your physical location (Barbados) may have different tax implications
- App Store revenue is typically taxed based on where you're selling (UK)

UK Phone Number

Your UK phone number (from Pixel 8a) is perfect for:

- Apple ID verification
- Two-factor authentication
- Account recovery
- Aligns with UK-region account

Troubleshooting

Build Fails

- Check that your Apple Developer account is active
- Verify bundle identifier matches App Store Connect (`com.mc2tc.tallynative`)
- Ensure certificates are valid (EAS manages these automatically)
- Check build logs: `eas build:view [BUILD_ID]`

Submission Fails

- Verify App Store Connect API key permissions (if using API key method)
- Check that the app exists in App Store Connect
- Ensure build number is higher than previous submissions
- Verify your Apple ID has proper permissions in App Store Connect

Testers Can't Install

- Verify they accepted the TestFlight invitation (check email)
- Check that the build has finished processing (5-15 minutes after upload)

- Ensure their device iOS version meets requirements
- Make sure they have TestFlight app installed
- For external testers: ensure Beta App Review is approved

Region/Access Issues

- If App Store Connect is inaccessible, try UK VPN (rarely needed)
- Clear browser cache/cookies for App Store Connect
- Use incognito/private browsing mode
- Check Apple System Status page for outages

Xcode Not Needed Errors

- If you see Xcode-related errors, ensure you're using `eas build` (cloud) not local builds
- For simulator: ensure Xcode is installed but you don't need to open it
- Command-line tools: `xcode-select --install` if needed

Complete Workflow Summary

Initial Setup (One-Time)

1. Create Apple Developer account (UK region)
2. Install EAS CLI: `npm install -g eas-cli`
3. Login to EAS: `eas login`
4. Configure credentials: `eas build:configure`
5. Create app in App Store Connect (web)
6. Update `eas.json` with your Apple IDs (optional)

Regular TestFlight Workflow (Every Build)

1. **Build**: `eas build --platform ios --profile production`
2. **Wait**: 10-20 minutes for cloud build
3. **Submit**: `eas submit --platform ios --latest`
4. **Wait**: 5-15 minutes for Apple processing
5. **Add Testers**: Via App Store Connect website (if new testers)
6. **Test**: Your tester receives invitation and can install

Development Testing (Local)

- Use iOS Simulator: `expo run:ios` (requires Xcode, but no interaction with Xcode UI)
- Use Android: `expo run:android` (no Apple involvement)

Quick Reference Commands

```
# Build for TestFlight (cloud build, no Xcode)
eas build --platform ios --profile production

# Submit latest build to App Store Connect
eas submit --platform ios --latest
```

```
# Check build status
eas build:list

# View build logs
eas build:view [BUILD_ID]

# Run iOS simulator (local development only)
expo run:ios

# Configure credentials (first time setup)
eas build:configure

# Manage credentials manually
eas credentials
```

Data Privacy and Apple Ecosystem Minimization

What Apple sees/knows:

- Your Apple ID email (use a dedicated email if preferred)
- App metadata (name, description, screenshots)
- Build artifacts (your compiled app)
- TestFlight tester emails

What stays local/private:

- Your source code (never uploaded to Apple)
- Your development environment
- Your local files and data
- Your Cursor IDE and workflow

Minimal Apple Services Used:

- Apple Developer Program (required for App Store)
- App Store Connect (web interface only)
- TestFlight (for distribution)
- iOS Simulator (local, no data sent to Apple)

You're NOT using:

- iCloud (unless you explicitly enable it)
- Apple ID for personal services
- Xcode for building/submitting
- Apple's development tools beyond simulator
- Any Apple ecosystem features beyond what's required