Core app quality

Last updated: May 17, 2021

A useful way to evaluate app excellence (/quality) is to walk through the workflows in your app and evaluate the smoothness and safety of the app experience.

This checklist defines a set of core quality criteria and associated tests to help you assess the quality of your app. Some of these criteria might be easy to miss, and the tests help you remember to include them in your test plans.

The checklist highlights the minimum quality that all apps should meet. Your testing will likely go well beyond what's described here.

Each item in the quality checklist has a unique ID which you might find helpful to use when you communicate with your team. You can also view the previous version of these guidelines (/docs/quality-guidelines/2021/02).

Visual experience

Your app should provide standard Android visual design and interaction patterns where appropriate, for a consistent and intuitive user experience.

We recommend using Material Design Components (https://material.io/components?platform=android) for creating a user interface in place of Android platform components where possible. This enables the modern Android look and feel, and it helps provide UI consistency across Android versions.

Area	ID	Tests	Description
Navigation	VX-N1	CR-3 (#CR-3)	The app supports standard Back button navigation (/design/patterns/navigation) and does not make use of any custom, on-screen "Back button" prompts.
	VX-N2	CR-3 (#CR-3)	The app supports gesture navigation (/training/gestures/gesturenav) for going back / going to the home screen.
	VX-N3	CR-1 (#CR-1) CR-3 (#CR-3) CR-5 (#CR-5)	The app correctly preserves and restores user or app state. The app preserves user or app state when leaving the foreground and prevents accidental data loss due to back-navigation and other state changes. When returning to the foreground, the app should restore the preserved state and any significant stateful transaction that was pending. Examples include: changes to editable fields, game progress, menus, videos, and other sections of the app or game. 1. When the app is resumed from the Recents app switcher, the app returns the user to the exact state in which it was last used. 2. When the app is resumed after the device wakes from the sleep (locked) state, the app returns the user to the exact state in which it was last used. 3. When the app is relaunched from Home or All Apps, it should do one of the following, depending on how much time has passed since it was last used: • If the app was last used a short time ago (minutes), restore the app state as close as possible to its previous state. • If more time has passed since the app was last used, try to restore the app as close as possible to its previous state; or start it from its home screen or some other default state.

Notifications	VX-S1	CR-9 (#CR-9)	Notifications follow Material Design guidelines (https://material.io/design/platform-guidance/android-notifications.html). In particular: 1. Notifications are not used for cross-promotion or advertising another product, as this is strictly prohibited by the Play Store. 2. Notification channels (/training/notify-user/channels) are defined according to best practices, rather than serving all notifications from one channel. 3. Selecting the correct notification priority (https://android-developers.googleblog.com/2018/12/notifications-from-twitter-app.html). 4. Multiple notifications are stacked into a single notification group (/training/notify-user/group), where possible. 5. Set timeouts (/reference/androidx/core/app/NotificationCompat.Builder#setTimeoutAfter(long)) for notifications where appropriate. 6. Notifications are persistent only if related to ongoing events, such as music playback or a phone call. For more information, see the Functionality section (/docs/quality-guidelines/core-app-quality#fn).
	VX-S2	CR-9 (#CR-9)	For messaging apps, social apps and conversations: 1. Use the `MessagingStyle` (/reference/androidx/core/app/NotificationCompat.MessagingStyle) notifications for conversations. 2. Support the direct reply action (/training/notify-user/build-notification#reply-action). 3. Support conversation shortcuts (/guide/topics/ui/conversations#shortcuts), and implement best practices for getting the best direct share ranking (/training/sharing/receive#get-best-ranking). 4. Support bubbles (/guide/topics/ui/bubbles).
UI and Graphics	VX-U1	CR-5 (#CR-5)	The app supports both landscape and portrait orientations (if possible) and folding / unfolding. Orientations expose largely the same features and actions and preserve functional parity. Minor changes in content or views are acceptable.
	VX-U2	CR-5 (#CR-5)	The app uses the whole screen in both orientations and does not letterbox to account for orientation changes, including folding and unfolding. Minor letterboxing to compensate for small variations in screen geometry is acceptable.
	VX-U3	CR-5 (#CR-5)	The app correctly handles rapid transitions between display orientations and device folding / unfolding without rendering problems or losing state.
Visual quality	VX-V1	CR-all (#core)	The app displays graphics, text, images, and other UI elements without noticeable distortion, blurring, or pixelation. 1. The app should use vector drawables (/guide/topics/graphics/vector-drawable-resources) where possible. 2. The app provides high-quality graphics for all targeted screen sizes and form factors. 3. No aliasing at the edges of menus, buttons, and other UI elements is visible.
	VX-V2	CR-all (#core)	The app displays text and text blocks in an acceptable manner for each of the app's supported languages. 1. Composition is acceptable in all supported form factors. 2. No cut-off letters or words are visible. 3. No improper word wraps within buttons or icons are visible. 4. There is sufficient spacing between text and surrounding elements.
	VX-V3	CR-all (#core)	The app's content, and all web contents referred to by the app, support dark theme (/guide/topics/ui/look-and-feel/darktheme).
Accessibility	VX-A1	CR-all (#core)	Touch targets should be at least 48dp in size. Learn more (/guide/topics/ui/accessibility/apps#large-controls).

	VX-A2	CR-all (#core)	The app's text and foreground content should maintain a high enough color contrast ratio with its background: • 3.0:1 for large text / graphics • 4.5:1 for small text (text smaller than 18pt, or if the text is bold and smaller than 14pt) Learn more about color and contrast (https://material.io/design/usability/accessibility.html#color-and-contrast).
	VX-A3	CR-all (#core)	Describe each UI element (/guide/topics/ui/accessibility/apps#describe-ui-element), except for `TextView`, using `contentDescription`.

Functionality

Your app should implement the expected functional behavior.

Area	ID	Tests	Description
Audio	FN-A1	CR-1 (#CR-1) CR-8 (#CR-8)	Audio resumes when the app returns to the foreground, or indicates to the user that playback is in a paused state.
	FN-A2	CR-1 (#CR-1) CR-2 (#CR-2) CR-8 (#CR-8)	If audio playback is a core feature, the app should support background playback (/guide/topics/media/mediaplayer#mpandservices).
	FN-A3	CR-0 (#CR-0)	When the user initiates audio playback, the app should do one of the following within one second: 1. Start playing the audio. 2. Provide a visual indicator that the audio data is being prepared.
	FN-A4	CR-0 (#CR-0)	The app should request audio focus (/guide/topics/media-apps/audio-focus) when audio starts playing and abandon audio focus when playback stops.
	FN-A5	CR-0 (#CR-0)	The app should handle other apps' requests for audio focus (/guide/topics/media-apps/audio-focus#audio-focus-change). For example, an app might reduce playback volume when another app plays speech.
Media	FN-M1	CR-0 (#CR-0) CR-6 (#CR-6) CR-8 (#CR-8)	If the app plays audio in the background, it must create a Notification styled with MediaStyle (https://android-developers.googleblog.com/2020/08/playing-nicely-with-media-controls.html).

	FN-M2	CR-0 (#CR-0)	If the app plays video, it should support picture-in-picture (/guide/topics/ui/picture-in-picture) playback.
	FN-M3	CR-0 (#CR-0)	If the app encodes video, it should do so using the HEVC video compression standard.
Sharing	FN-S1	CR-0 (#CR-0)	The app should use the Android Sharesheet (/training/sharing/send) when sharing content. It can suggest targets that are unavailable to custom solutions.
Background Service	FN-B1	CR-6 (#CR-6)	The app avoids running unnecessarily long services in the background. To ensure the smooth running of the user's device, the system applies various restrictions on background services (/about/versions/oreo/background#services). These are not considered good uses of background services: • Maintaining a network connection for notifications • Maintaining a Bluetooth connection • Keeping the GPS powered-on Learn how to choose the right solution for your work (/guide/background#categories_of_background_tasks).

Performance and stability

Your app should provide the performance, stability, compatibility, and responsiveness expected by users.

Area	ID	Tests	Description
Stability	PS-S1	CR-all (#core) SD-1 (#SD-1)	The app does not crash (/topic/performance/vitals/crash) or block the UI thread (/topic/performance/vitals/anr) causing ANR (Android Not Responding") errors. Utilize Google Play's pre-launch report (https://support.google.com/googleplay/android-developer/answer/9842757?hl=en-GB) to identify potential stability issues. After deployment, pay attention to the Android Vitals (/topic/performance/vitals) page in the Google Play developer console.
Performance	PS-P1	CR-all (#core) SD-1 (#SD-1)	The app loads quickly (https://medium.com/androiddevelopers/testing-app-startup-performance-36169c27ee55) or provides onscreen feedback to the user (a progress indicator or similar cue) if the app takes longer than two seconds to load.
	PS-P2	CR-all (#CR-all) SD-1 (#SD-1)	Apps should render frames every 16ms to achieve 60 frames per second. Developers can use the Profile HWUI rendering (/topic/performance/rendering/inspect-gpu-rendering) option in testing. If there are issues, tools are available to help diagnose slow rendering (/topic/performance/vitals/render).
	PS-P3	PM-1 (#PM-1)	With StrictMode enabled (see StrictMode Testing (#strictmode), below), no red flashes (performance warnings from StrictMode) are visible when testing the app. Any red flashes indicate bad behaviors regarding storage, network access, or memory leaks.
SDK	PS-T1	CR-0 (#CR-0)	The app runs on the latest public version of the Android platform without crashing or severely impacting core functionality.

	PS-T2	SP-1 (#SP-1)	The app targets the latest Android SDK (/distribute/best-practices/develop/target-sdk) needed to align with Google Play requirements by setting the `targetSdk` value.
	PS-T3	SP-1 (#SP-1)	The app is built with the latest Android SDK by setting the `compileSdk` value.
	PS-T4	SP-2 (#SP-2) SP-3 (#SP-3)	Any Google or third-party SDKs used are up-to-date (/topic/security/best-practices#services-dependencies-updated). Any improvements to these SDKs, such as stability, compatibility, or security, should be available to users in a timely manner. For Google SDKs, consider using SDKs powered by Google Play services (http://developers.google.com/android), when available. These SDKs are backward compatible, receive automatic updates, reduce your app package size, and make efficient use of on-device resources. The developer is accountable for the entire app's codebase, inclusive of any third-party SDKs used.
	PS-T5	SP-3 (#SP-3)	The app does not use non-SDK interfaces (/distribute/best-practices/develop/restrictions-non-sdk-interfaces).
	PS-T6	SP-2 (#SP-2)	No debug libraries are included in the production app. This can cause performance as well as security issues.
Battery	PS-B1	BA-1 (#BA-1)	The app properly supports the power management features that were introduced in Android 6.0 (Doze and App Standby). In the case where core functionality is disrupted by power management, only qualified apps may request an exemption. See Support for other use cases (/training/monitoring-device-state/doze-standby#other_use_cases) in Doze and App Standby. During development, developers can test app standby and doze behavior using these ADB commands (/training/monitoring-device-state/doze-standby#testing_doze_and_app_standby). In terms of battery usage, developers can use the Android Studio energy profiler (/studio/profile/energy-profiler) or the Battery Historian (/topic/performance/power/battery-historian) tool, combined with planned background work, to diagnose unexpected battery use.

Privacy & security

Your app should handle user data and personal information safely, with the appropriate level of permission.

In addition to this checklist, applications published on the Google Play Store must also follow the User Data policies (https://play.google.com/about/privacy-security/user-data/) to protect users' privacy.

Area	ID	Tests	Description
Permissions	SC-P1	SC-4 (#SC-4)	The app requests only the *absolute minimum* number of permissions that it needs to support its use case at hand. For some permissions such as location, use coarse location in place of fine location if possible.
	SC-P2		The app requests permission to access sensitive data (such as SMS, Call Log (https://support.google.com/googleplay/android-developer/answer/10208820), or Location (https://developer.android.com/privacy/best-practices#location)) or services that cost money (such as Dialer or SMS) only when directly related to the core use cases of the apps. Implications related to these permissions should be prominently disclosed to the user. Depending on how you are using the permissions, there might be an alternative way (https://developer.android.com/training/permissions/evaluating) to fulfill your app's use case without relying on access to sensitive information. For example, instead of requesting permissions related to a

			user's contacts, it may be more appropriate to request access by using an implicit intent (https://developer.android.com/guide/components/intents-common#Contacts).
	SC-P3	CR-0 (#CR-0)	The app requests runtime permissions in context, when the functionality is requested, rather than upfront during app startup.
	SC-P4	CR-0 (#CR-0)	The app clearly conveys why certain permissions are needed or follow the recommended flow to explain why it needs a permission (/training/permissions/requesting#explain).
	SC-P5	CR-0 (#CR-0)	The app should gracefully degrade (/training/permissions/requesting#handle-denial) when users deny or revoke a permission. The app should not prevent the user from accessing the app altogether.
Data & Files	SC-DF1	SC-1 (#SC-1)	All sensitive data is stored in the app's internal storage (/topic/security/best-practices#internal-storage).
	SC-DF2	SC-10 (#SC-10)	No personal or sensitive user data (/training/articles/security-tips#UserData) is logged to the system log or an app-specific log.
	SC-DF3		The app does not use any non-resettable hardware IDs (/training/articles/user-data-ids), such as the IMEI, for identification purposes.
Identity	SC-ID1	CR-0 (#CR-0)	The app provides hints to autofill (/guide/topics/text/autofill-optimize#hints) account credentials and other sensitive information, such as credit card info, physical address, and phone number.
	SC-ID2	CR-0 (#CR-0)	Integrate One Tap for Android (https://developers.google.com/identity/one-tap/android) for a seamless sign in experience.
	SC-ID3	CR-0 (#CR-0)	The app supports biometric authentication (/training/sign-in/biometric-auth) to protect financial transactions or sensitive information, such as important user documents.
App Components	SC-AC1	SC-5 (#SC-5)	The app sets the `android:exported` attribute explicitly for all activities (/guide/topics/manifest/activity-element#exported), services (/guide/topics/manifest/service-element#exported), broadcast receivers, (/guide/topics/manifest/receiver-element#exported) and especially content providers (/guide/topics/manifest/provider-element#exported). Only application components that *share data with other apps*, or components that *should be invoked by other apps*, are exported (/guide/topics/manifest/service-element#exported).
	SC-AC2	CR-0 (#CR-0) SC-4 (#SC-4)	All intents and broadcasts follow best practices: 1. Use explicit intents (/guide/components/intents-filters#Types) if the destination application is well defined. 2. Use Intents to defer permissions to a different app that already has the permission. (/topic/security/best-practices#permissions-intents) 3. Share data securely across apps (/topic/security/best-practices#permissions-share-data). 4. Intents that contain a payload are verified before use (/training/articles/security-tips#InputValidation) . 5. If you need to pass an Intent to another app, so that the receiving app can invoke and expect a callback in the calling app, do not include a nested intent in the extras. Use a PendingIntent. 6. When setting up your PendingIntents, explicitly set the immutable flag (/reference/android/app/PendingIntent#FLAG_IMMUTABLE), where applicable.

	SC-AC3	SC-3 (#SC-3)	All components that *share content between your apps* use `android:protectionLevel="signature"` (/guide/topics/manifest/permission-element#plevel) for custom permissions (/guide/topics/permissions/defining). This includes activities (/guide/topics/manifest/activity-element#prmsn), services (/guide/topics/manifest/service-element#prmsn), broadcast receivers (/guide/topics/manifest/receiver-element#prmsn), and especially content providers (/guide/topics/manifest/provider-element#prmsn). Apps should not rely on accessing a list of installed packages. The access has been restricted beginning in Android 11.
Networking	SC-N1	SC-9 (#SC-9)	All network traffic is sent over SSL (/training/articles/security-ssl).
	SC-N2	SC-6 (#SC-6)	The application declares a network security configuration (/training/articles/security-config).
	SC-N3		If the application uses Google Play services, the security provider is initialized at application startup (/training/articles/security-gms-provider).
WebViews	SC-W1	SC-6 (#SC-6)	Do not use setAllowUniversalAccessFromFileURLs() (/reference/android/webkit/WebSettings#setAllowUniversalAccessFromFileURLs(boolean)) for accessing local content. Instead, use WebViewAssetLoader (/reference/androidx/webkit/WebViewAssetLoader).
	SC-W2	SC-7 (#SC-7)	WebViews should not use addJavaScriptInterface() (/training/articles/security-tips#WebView) with untrusted content. On Android 6.0 and above, use HTML message channels (/topic/security/best-practices#message-channels) instead.
Execution	SC-E1		The app does not dynamically load (/training/articles/security-tips#DynamicCode) code from outside the app's APK. Developers should use Android App Bundles (/guide/app-bundle), which includes Play Feature Delivery (/guide/app-bundle/play-feature-delivery) and Play Asset Delivery (/guide/app-bundle/asset-delivery). Starting August 2021, the use of Android App Bundles will become mandatory for all new apps in the Google Play store.
Cryptography	SC-C1		The app uses strong, platform-provided cryptographic algorithms and a random number generator (/training/articles/security-tips#Crypto). Also, the app does not implement custom algorithms.

Google Play

Be sure that your apps can be published on Google Play.

Area	ID	Tests	Description
Policies	GP-P1	GP-all (#gp)	The app strictly adheres to the terms of the Google Play Developer Content Policy (http://play.google.com/about/developer-content-policy.html) and does not offer inappropriate content, does not use the intellectual property or brand of others, and so on.
	GP-P2	GP-1 (#GP-1)	The app maturity level is set appropriately, based on the Content Rating Guidelines (http://support.google.com/googleplay/android-developer/bin/answer.py?hl=en&answer=188189).

App Details Page	GP-D1	GP-1 (#GP-1) GP-2 (#GP-2)	The app's feature graphic follows the guidelines outlined in this support article (https://support.google.com/googleplay/android-developer/answer/9866151?visit_id=637408759199927231-2278064029&rd=1#feature_graphic_requirements). Make sure that: 1. The app listing includes a high-quality feature graphic. 2. The feature graphic does not contain device images, screenshots, or small text that will be illegible when scaled down and displayed on the smallest screen size that your app is targeting. 3. The feature graphic does not resemble an advertisement.
	GP-D2	GP-1 (#GP-1)	The app's screenshots and videos do not show or reference non-Android devices.
	GP-D3	GP-1 (#GP-1)	The app's screenshots or videos do not represent the content and experience of your app in a misleading way.
User Support	GP-X1	GP-1 (#GP-1)	Common user-reported bugs in the Reviews tab of the Google Play page are addressed if they are reproducible and occur on many different devices. If a bug occurs on only a few devices, you should still address it if those devices are particularly popular or new.

Setting up a test environment

For the purpose of setting up a test environment for this checklist, we recommend the following:

- **Focused on emulator testing** - Android Emulator is a great way to test your app under different Android versions and screen resolutions. You should set up emulated devices (AVDs) (/tools/devices) to represent the most common form factors and hardware/software combinations for your target user base. In addition to testing for phones, we also recommend you test other form factors using the following emulators *at a minimum*:
  - Foldables - 7.6" Fold-in with outer display (this is listed under phones in the AVD Manager).
  - Tablet - Pixel C 9.94" (2,560px x 1,800px).
  - For mobile app notification testing, pair a mobile device / emulator with Wear OS emulator - Wear OS Round 1.84".
- **Hardware devices** - Your test environment should include a small number of actual hardware devices that represent the key form factors and hardware/software combinations that are currently available to consumers. It's not necessary to test on *every* device that's on the market — rather, you should focus on a small number of representative devices, even using one or two devices per form factor.
- **Device test labs** - You can also use third party services, such as Firebase Test Lab (https://firebase.google.com/docs/test-lab/android/overview), to test your app on a wider variety of devices.
- **Test with the latest Android version** - In addition to testing representative Android versions for your target user base, you should always test against the latest version of Android (currently Android 11). This ensures that the latest behavior changes (/about/versions/11/behavior-changes-11) do not negatively impact your user's experience.

For more comprehensive guidance on testing including unit testing, integration testing and UI testing, check out the Android testing fundamentals (/training/testing/fundamentals).

Test procedures

These test procedures help you discover various types of quality issues in your app. You can combine the tests or integrate groups of tests together in your own test plans. See the sections above for references that associate criteria with these test procedures.

Type	Test	Description

Core Suite	CR-0	Navigate to all parts of the app — all screens, dialogs, settings, and all user flows. 1. If the application allows for editing or content creation, game play, or media playback, make sure to test those flows. 2. While testing the app, introduce interruptions from other apps, such as receiving a notification or a phone call; and apply transient changes to device attributes, such as network connectivity, battery function, GPS availability, and system load. 3. Enter and test all in-app purchase flows
	CR-1	From each app screen, press the device's Home key or swipe up in gesture navigation, then re-launch the app from the All Apps screen.
	CR-2	From each app screen, switch to another running app, and then return to the app under test using the Recents app switcher.
	CR-3	From each app screen (and dialogs), press the Back button or use the back swipe gesture.
	CR-5	From each app screen, rotate the device between landscape and portrait orientation and folding / unfolding at least three times.
	CR-6	Switch to another app to send the test app into the background. Go to Settings and check whether the test app has any services running while in the background. In Android 4.0 and higher, go to the Apps screen and find the app in the "Running" tab.
	CR-7	Press the power button to put the device to sleep, then press the power button again to wake the screen.
	CR-8	Set up a screen lock on the device. Press the power button to put the device to sleep (which locks the device). Then, press the power button again to wake the screen and unlock the device.
	CR-9	Trigger and observe in the notifications drawer all types of notifications that the app can display. Expand notifications where applicable (Android 4.1 and higher), and tap on all available actions.
	CR-10	Review Support for other use cases in Doze and App Standby.
Install on SD Card	SD-1	Repeat *Core Suite* with the app installed to a device's SD card (/guide/topics/data/install-location) (if the app supports this installation method). To move the app to SD card, you can use Settings > App Info > Move to SD Card.
Performance and Stability	SP-1	Review the Android manifest file and build configuration to ensure that the application is built against the latest available SDK (/guide/topics/manifest/uses-sdk-element#ApiLevels) (`targetSdk` and `compileSdk`).
	SP-2	Review the `build.gradle` file for any outdated dependencies.
	SP-3	Use the Android Studio lint tool (/distribute/best-practices/develop/restrictions-non-sdk-interfaces#studio-lint) to detect non-SDK interface usage. Other alternative testing methods (https://developer.android.com/distribute/best-practices/develop/restrictions-non-sdk-interfaces#test-for-non-sdk) also exist.
Performance Monitoring	PM-1	Repeat *Core Suite* with StrictMode profiling enabled (/docs/quality-guidelines/core-app-quality#strictmode). Pay close attention to garbage collection and its impact on the user experience.
Battery	BA-1	Repeat *Core Suite* across Doze and App Standby cycles. Pay close attention to alarms, timers, notifications, syncs, and so on. See Testing with Doze and App Standby (/training/monitoring-device-state/doze-standby#testing_doze_and_app_standby) for requirements and guidelines.

Security	SC-1	Review all data stored in external storage.
	SC-2	Review how the data that's loaded from external storage is handled and processed.
	SC-3	Review all content providers defined in the Android manifest file. Make sure each provider has an appropriate `protectionLevel`.
	SC-4	Review all permissions that your app requires, in the manifest file, at runtime, and in the app settings screen (Settings > App Info) on the device.
	SC-5	Review all application components (/guide/topics/manifest/application-element) defined in the Android manifest file for the appropriate export state. The exported property must be set explicitly for all components.
	SC-6	Review the app's Network Security configuration (/training/articles/security-config), ensuring that no lint checks on the configuration fail.
	SC-7	For each WebView, navigate to a page that requires JavaScript.
	SC-8	In each WebView, attempt to navigate to sites and content that aren't loaded directly by your app.
	SC-9	Declare a Network Security Configuration that disables cleartext traffic (/training/articles/security-config#CleartextTrafficPermitted), then test the app.
	SC-10	Run the application and exercise all core functionality, while observing the device log (/studio/command-line/logcat). No private user information should be logged.
Google Play	GP-1	Sign into the Google Play Developer Console (https://play.google.com/apps/publish/) to review your developer profile, app description, screenshots, feature graphic, content rating and user feedback.
	GP-2	Download your feature graphic and screenshots, and scale them down to match the display sizes on the devices and form factors that you are targeting.
	GP-3	Review all graphical assets, media, text, code libraries, and other content that's packaged in the app or expansion file download.

Testing with StrictMode

For performance testing, we recommend enabling `StrictMode` (https://developer.android.com/reference/android/os/StrictMode) in your app and using it to catch operations that could affect performance, network accesses, file reads/writes, and so on. Look for potentially problematic operations both on the main thread and on other threads.

You can set up a per-thread monitoring policy using `StrictMode.ThreadPolicy.Builder` (https://developer.android.com/reference/android/os/StrictMode.ThreadPolicy.Builder) and enable all supported monitoring in the `ThreadPolicy` using `detectAll()` (https://developer.android.com/reference/android/os/StrictMode.ThreadPolicy.Builder#detectAll()).

Make sure to enable **visual notification** of policy violations for the `ThreadPolicy` using `penaltyFlashScreen()` (https://developer.android.com/reference/android/os/StrictMode.ThreadPolicy.Builder#penaltyFlashScreen()).

Next
Large screen app quality (/docs/quality-guidelines/large-screen-app-quality) →

Last updated 2022-03-18 UTC.